Universität Trier

Modellierung einer Musikdatenbank:

Entwurf und Implementierung mit SQLite

Master Digital Humanities

Datenbanken in den Digital Humanities

Wintersemester 2023

Dr. Thomas Burch

Marina Spielberg

Matrikelnummer:1656491

Abgabe: 21.03.20

Erklärung zum Portfolio

Hiermit erkläre ich, dass ich das Portfolio selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die aus fremden Quellen direkt oder indirekt übernommenen Gedanken als solche kenntlich gemacht habe.
Die Arbeit habe ich bisher keinem anderen Prüfungsamt in gleicher oder vergleichbarer Form vorgelegt. Sie wurde bisher nicht veröffentlicht.

Datum: 20.04.2024 Unterschrift:

Inhaltsverzeichnis

1. Einleitung

Welche Musikgenres sind am populärsten, welche Musikkünstler ähneln sich und welche Alben werden mit der höchsten Punktzahl in Rezensionen bewertet? Das sind Fragen, die sich mit Hilfe von SQL-Anfragen an eine relationale Datenbank mit Musikmetadaten beantworten lassen. Das Ziel dieser Arbeit ist es, ein Modell von zeitgenössischen Musikstücken und ihrer Rezeption in der Gesellschaft durch eine relationale Datenbank darzustellen und über Anfragen und Visualisierungen die Zusammenhänge der Daten zu entdecken[1]. Dafür dokumentiert der Bericht erst die zur Verfügung stehenden Datensätze, die aus Metadaten zu einer Million Liedern, sowie von Albenrezensionen des Portals Pitchfork bestehen. Anschließend wird der Prozess der Datenmodellierung vom ER-Modell bis zur Konvertierung ins rationale Modell dargestellt. Mit dieser Vorarbeit lässt sich anschließend die Datenbank erstellen und mit Daten aus den Quelldatensätzen füllen. Es wird aufgezeigt, wie durch SQL-Anfragen relationale Beziehungen der Daten sichtbar gemacht werden können. Zur Visualisierung der Daten werden die häufigsten Wörter in Rezensionen in einer Wortwolke und das Verhältnis zwischen Liedlänge und Erscheinungsjahr in einem Streudiagramm dargestellt. Abgerundet wird der Bericht durch den Music-Matcher, einer Anwendung der Datenbank für die reale Welt, die basierend auf Benutzerinteraktionen Künstlervorschläge generiert. Jedes Kapitel beginnt mit einer Übersicht der Dateien, die im jeweiligen Kapitel thematisiert werden.

2. Datenmodellierung

In diesem Abschnitt soll der Prozess der Datenmodellierung, angefangen bei der Idee für ein Datenmodell, über die verwendeten Datensätze und einem ER-Modell bis zu einem relationalen Modell, beschrieben werden.

2.1 Sachverhalt und genutzte Datensätze

Besprochene Dateien: artist_similarity.db, artist_term.db, pitchfork_reviews.sqlite, track_metadata.db

Die Musikdatenbank soll folgenden relationalen Teilaspekt der realen Welt modellieren: „Musikkünstler bringen Musik in Form von Liedern und Alben raus. Die Musik der Künstler wird von Kritikern rezensiert und bewertet.“ Sowohl Musikkünstler als auch

[1] Alle Skripte und Anhänge dieses Projektes befinden sich auch im GitHub Repository Datenbanken_DH_Portfolio.

Lieder, Alben, Kritiker und Rezensionen besitzen zusätzlich verschiedene Eigenschaften, die einen tieferen Einblick in ihre Beziehungen untereinander erlauben. Um diesen Sachverhalt in der Datenbank abzubilden, wurden Daten aus zwei Quellen verwendet, dem Datensatz „18,393 Pitchfork Reviews“ von Nolan Conaway (nachfolgend Pitchfork DB) und drei Datensätzen aus dem „The Million Song Dataset“ von Thierry Bertin-Mahieux et al. (nachfolgend Millionsong DB). Alle Datensätze liegen im SQLite Format vor.

Die Pitchfork DB enthält Rezensionen des im Jahr 1996 gegründeten amerikanischen online Musikmagazins Pitchfork, welches vor allem für seine hochgestochene Sprache bekannt ist (Itzkoff). Die Daten aus der Datenbank wurden von Conaway mit Beautiful Soup gescraped und beziehen sich auf Rezensionen der Jahre 1999 bis 2017 („Pitchfork Data“). Die Datenbank enthält die Tabellen `artists, content, genres, labels, reviews` und `year` von denen alle in die Musikdatenbank aufgenommen wurden (Conaway, „Pitchfork Reviews“).

Das Millionsong Dataset ist eine Sammlung von Metadaten von einer Million Liedern aus den Jahren 1922 bis 2010, die über die Echo Nest API gesammelt wurden und unteranderem in den Datenbanken track_metadata.db, artist_term.db und artist_similarity.db vorliegen (Thierry Bertin-Mahieux). Aus track_metadata.db sind die Attribute `track_id, title, song_id, release, artist_id, artist_name, duration, year, artist_familiarity, artist_hotttnesss` aus der Tabelle `song` für die relevant. Aus artist_term.db wurden die Tabellen `artist_mbtags` und `mbtags` übernommen, die den Künstlern Stichwörter zuweisen und aus artist_similarity.db die Tabelle `similarity`, die `artist_ids` von ähnlichen Künstlern gegenüberstellt.

Ich habe mich für die Nutzung dieser Datenquellen entschieden, weil eine Teilmenge der Künstler und Erscheinungsjahre sich in beiden Datensätzen befindet und somit beide Datensätze verbunden werden und die entstehenden Relationen erforscht werden können.

2.2 Konzeptionelle Ebene: Das ER-Modell

Besprochene Datei: Musikdatenbank_ER.png

Um aus den Daten der vier Quelldatenbanken das gewünschte Modell in eine sinnvolle Struktur zu bringen, bietet sich die Entity-Relationship Formalisierung an, da eine visuelle Gestaltung die Zusammenhänge und Quantitäten der Relationen übersichtlich darstellt (Kemper 24-25). Für die Erstellung des ER-Modells wurde die Anwendung draw.io genutzt. Das Resultat befindet sich in der Datei Musikdatenbank_ER und im Anhang 1.

Das ER-Modell umfasst sechs starke Entitäten `artist, artist_tags, song, album, review` und `author`, die jeweils einen individuellen Schlüssel besitzen. Die sieben Beziehungen zwischen den Entitäten können als folgende Aussagen formuliert werden: `artist creates song, artist has tags, author writes review, review includes artist` und `review reviews album` (eine ternäre Beziehung), `album contains song, artist is similar to artist`. Die Verbindungen zwischen `artist` und `review`, sowie zwischen `album` uns `song` ermöglichen die Vereinigung der Pitchfork DB mit der Millionsong DB und drücken so die gewünschte Modellierung der realen Welt „Die Musik der Künstler wird von Kritikern rezensiert." erfolgreich aus.

Bei den Attributen gibt es einige Besonderheiten zu beachten. `mb_tags` aus der Entität `artist_tags` und `genre` aus der Entität `album` wurden nicht zu einem Attribut zusammengefasst, denn das Genre eines Albums bestimmt nicht zwangsläufig die Kategorien des Künstlers, da sich die Kategorien während seiner Laufbahn ändern können und die Kategorien zusätzlich Ortsbezeichnungen wie „armenian" oder „Berlin" enthalten. Die Attribute `album_year` und `song_year` beinhalten nicht dieselben Informationen, denn es gibt Lieder, die keinem Album zugeordnet wurden und deshalb ein unabhängiges Attribut des Veröffentlichungsjahrs brauchen. Es wurden auch die neuen Attribute `tag_id` und `author_id` generiert, die nicht in den Quelldatenbanken vorlagen, aber als Schlüssel für eine eindeutige Identifizierung der Entitäten `artist_tags` und `author` dienen sollen. Zur Erweiterung der Datenbank könnte man zukünftig noch Attribute wie die Liedsprache und Liedtexte hinzugefügten. Um die Metadaten über die Personen in der Datenbank anzureichern, wäre die Aufnahme der GND ID der Künstler zu empfehlen.

2.3 Logische Ebene: Das relationale Modell

Besprochene Datei: Relationales_Modell.pdf
Das fertige ER-Modell wird jetzt in ein relationales Modell konvertiert, welches die Struktur der gespeicherten Daten festlegt und für die Erstellung der Tabellen der Datenbank genutzt werden kann (Kemper 71-72). Das relationale Modell sowie die einzelnen Schritte zur Konvertierung finden sich im Anhang 2.

Im Folgenden beschreibe ich die Konvertierungspraxis der Kardinalitäten nach der Chen Notation und gebe zu jeder Beziehungsform einige Beispiele aus der Musikdatenbank. Um eine N:M Beziehung darzustellen, muss eine zusätzliche Verknüpfungstabelle erstellt werden (Kemper 76-77). In der Relation von „artist has

artist_tags“, wird die Verknüpfungstabelle `has_tags` hinzugefügt, in der die Schlüssel beider Relationen, `artist_id` und `tag_id`, als Fremdschlüssel erscheinen.

Die N:M Beziehung von „artist is similar to artist“ ist ein Sonderfall, da sie rekursiv ist. Das bedeutet, dass die Entität `artist` in Beziehung mit sich selbst steht. In diesem Fall dient die `similarity` Tabelle (vorher „is simialr to“) als Verknüpfungstabelle (Kemper 384). Sie beinhaltet die Fremdschlüssel `target_artist` (der zu vergleichende Künstler) und `similar_arist` (der ähnliche Künstler), die nach ihren Rollen in der rekursiven Beziehung benannt, aber formell die `artist_id` sind.

Bei der 1:1 Beziehung kann man einen beliebigen Schlüssel von zwei Entitäten als Fremdschlüssel der jeweils anderen Beziehung verwenden (Kemper 80-81). In der Musikdatenbank findet sich so eine Beziehung zwischen `album` und `review`, was auf den ersten Blick merkwürdig erscheint, denn in der realen Welt kann ein Album von mehreren Rezensionen bewertet werden (1:N). Da in der Pitchfork DB je eine Rezension einem Album zugeordnet ist, habe ich mich für die Modellierung als 1:1 Beziehung entschieden. Möchte man das Modell realitätsgetreuer gestalten, so sollte diese Beziehung als 1:N definiert werden.

Weiterhin fällt auf, dass sowohl die Entität `review` als auch `album` als Schlüssel die `review_id` haben. Logisch betrachtet ist das nicht richtig, denn in der realen Welt werden nicht alle Alben rezensiert und brauchen deshalb ihren eigenen individuellen Schlüssel in Form von zum Beispiel einer `album_id`. Diese `album_id` würde man in die `review` Tabelle als Fremdschlüssel integrieren, um neue Alben hinzufügen zu können, die keine Rezensionen haben. Die Musikdatenbank ist jedoch durch eine 1:1 Beziehung modelliert, sodass jedes Album einer Rezension zugeordnet ist. Aufgrund dieser Beziehung könnte man die beiden Entitäten auch zu einer Tabelle zusammenführen. Ich wollte jedoch Attribute wie `score`, `url` (`review`) und `label` und `genre` (`album`), die inhaltlich zueinander gehören, trennen. Deshalb trenne ich `review` und `album` nicht und generiere keine `album_id`. Stattdessen benutze ich die `review_id` als Schlüssel für beide Entitäten. Die Individualität des Schlüssels ist durch die 1:1 Beziehung trotzdem garantiert.

Als Letztes wird die 1:N Beziehung umgewandelt. Zwischen `artist` und `review` herrscht eine 1:N Beziehung, weil ein Künstler in mehreren Rezensionen bewertet werden kann, aber eine Rezension bezieht sich in der Pitchfork DB auf nur einen Künstler. Daher wird aus der Tabelle aus der 1 Seite (`artist`) der Schlüssel `artist_id` genommen und in die Tabelle mit der N Seite (`review`) integriert und dann in `reviewed_artist` unbenannt (Kemper 78-80).

3. Datenbankerstellung

Besprochene Datei: create_musicdb.py

Mit Hilfe des relationalen Modells können nun die Tabellen mit SQLite erstellt werden. Das Skript create_musicdb.py ist dabei so aufgebaut, dass zuerst die Datenbank `musicdb` durch die Funktion `sqlite.connect` initialisiert und danach der Cursor definiert wird (6-8). Anschließend werden mit dem SQL Befehl `DROP TABLE IF EXISTS` bereits vorhandene Tabellen gelöscht, um auf Wunsch eine leere Datenbank anlegen zu können (11-18). Im Skript werden die Tabellen `artist, artist_tags, has_tags, similarity, song, album, review` und `author` mit der Funktion `execute`, sowie dem SQL `Befehl CREATE TABLE` über den Cursor erstellt (22-88). Jedes Attribut wird ferner einem Datentyp zugewiesen. Bei der Definition von Fremdschlüsseln ist zu beachten, dass durch den Befehl `REFERENCES` eine Verbindung zur Tabelle hergestellt werden muss, auf die sich der Fremdschlüssel bezieht. Beispielwiese beziehen sich die Fremdschlüssel `artist_id` und `reviewd_in` in der Tabelle `song` auf die Primärschlüssel `artist_id` der Tabelle `artist` und `review_id` der Tabelle `album` (58-61). Bei Verknüpfungstabellen, die durch eine N:M Beziehung entstanden sind, wie die Tabelle `has_tags` und `similarity`, ist es sinnvoll zusätzlich zu den beiden Fremdschlüsseln einen zusammengesetzten Schlüssel zu erstellen, der aus den beiden Fremdschlüsseln besteht, um Duplikate zu vermeiden und sicherzustellen, dass jede Kombination der Fremdschlüssel eindeutig ist (Hellweg). In der `has_tags` Tabelle wird dies durch den Befehl `PRIMARY KEY (artist_id, tag_id)` gelöst (41). Schlussendlich werden die Änderungen mit der Methode `commit` gespeichert und die Datenbankverbindung wird geschlossen (90-91).

4. Datenimporte

Besprochene Dateien: import_data_pitchfork.py, import_data_millionsongs.py[2]

Nach der Erstellung der Tabellen ist der nächste Schritt Daten aus den vier Quelldatenbanken in die Tabellen der Musikdatenbank zu importieren. Beide Importskripte sind so aufgebaut, dass zuerst eine Verbindung zu den notwendigen Datenbanken, aus denen die zu importierenden Tupel stammen, aufgebaut wird. Dann

[2]Zur Erleichterung der Lesbarkeit wird für die Zitation import_data_pitchfork.py als P und import_data_millionsongs.py als M abgekürzt.

wird für jede Linux Libertine MonoTabelle Tupel aus den relevanten Spalten der Quelldatenbanken ausgewählt und in einer Variable gespeichert. In einigen Fällen muss dann die Datenstruktur an die Anforderungen der Musikdatenbank angepasst werden. Schlussendlich werden die ausgewählten Daten, falls sie nicht schon in der Datenbank existieren, durch `INSERT OR IGNORE` in die Musikdatenbank hinzugefügt.

Wie der Code im Detail funktioniert, soll an ausgewählten Tabellen, die verschiedene Problemstellungen aufweisen, demonstriert werden. Es ist wichtig zu beachten, dass zuerst import_data_pitchfork.py und dann import_data_millionsong.db ausgeführt werden soll, da für den Import in die `reviewed_in` Spalte der `song` Tabelle auf Daten aus der `album` Tabelle zugegriffen wird, welche aus der Pitchfork DB kommen.

Die einfachsten Importe finden sich in den Tabellen `artist` (M 73-82), `album` (P 35-50) und `artist_similarity` (M 85-93). In der `artist` Tabelle, beispielweise, erfolgt die Integration in die Musikdatenbank direkt durch eine SQL-Anfrage an die `song` Tabelle der trackmetadata.db, sodass die Attribute `artist_id, artist_name, artist_hotttnesss` und `artist_familiarity` in der Variable `artist_data` gespeichert werden (M 71-72) und ohne weitere Transformationen mit der Methode `executemany` in die `artist` Tabelle importiert (M 75). Ich benutze `executemany` statt `execute`, um alle Datensätze auf einmal zu übernehmen und nicht nur jeweils eine Zeile. Für die Integration der Daten aus `artist_data` in der `INSERT` Anfrage habe ich mich für die die Benutzung von Placeholdern in der Form von „?“ entschieden, welche an Stelle von jedem einzusetzenden Wert benutzt werden, anstatt `artist_data` direkt in die Anfrage zu platzieren (M 74-75). Die SQLite Dokumentation warnt, dass letzteres zu „SQL injection attacks“ führen kann und empfiehlt die Placeholder Methode als Best Practise (Placeholders).

In den Tabellen `author` (P 17-32) und `artist_tags` (M 55-70) muss zusätzlich zum Import noch eine manuelle ID erstellt werden. Im Fall von der `author` Tabelle wird aus der `reviews` Tabelle der Pitchfork DB der Autorname und Autortyp für den Import in die `author_data` Variable aufgenommen (P 20-21). Darüber hinaus braucht die `author` Tabelle noch eine individuelle `author_id`, die als Schlüssel agiert. Dafür wird in einer for-Schleife über jedes Element aus `author_data` iteriert und die `enumerate` Funktion erteilt jedem Element des Tupels `(author, author_type)` einen Index (P 25). Somit berechnen wir die `auhtor_id` indem wir zur Indexnummer jedes Schleifendurchgangs eins addieren und den Wert in einen String umwandeln (P26). Anschließend wird `author_id, author`

und author_type in Form eines Tripels in die vorher definierte leere Liste insert_data hinzugefügt und anschließend in die author Tabelle eingefügt (P 27-32).

Der dritte Importfall bezieht sich auf die Tabellen has_tags (M 92-114) und review (P 65-83), bei denen der textbasierte Wert einer Spalte durch eine ID ausgetauscht werden soll. Beispielweise soll die review Tabelle unter anderem die Spalte review_author, die aus der auhtor_id besteht, welche wir im vorherigen Schritt angelegt haben, beinhalten. Dafür muss das Skript Autorennamen aus reviews.author der Pitchfork DB mit den Namen au author_name der Musikdatenbank abgleichen und bei einer Übereinstimmung die zum Autor zugehörige author_id in die review Tabelle der Musikdatenbank einfügen. Dafür wird zuerst in das Wörterbuch author_dict author_name als Schlüssel und die passende author_id als Wert eingetragen (P 67-69). Als zweiten Schritt wird nun über die Daten aus der reviews Tabelle der Pitchfork DB iteriert. Bei einer Übereinstimmung eines Autornamens aus der Pitchfork DB mit einem Namen aus dem Wörterbuch wird der dazugehörige Wert dieses Schlüssels, also author_id entnommen und zusammen mit weiteren Attributen zur Liste review_data_with_author_id hinzugefügt und anschließend in die Tabelle review importiert (P 72-83).

Die programmatisch anspruchsvollste Aufgabe war es, vor dem Import die Künstler- und Albennamen aus den Pitchfork und Millionsong Datenbanken in jeweils eine Spalte zusammenzuführen, um eine Verbindung zwischen den beiden Quelldatensätzen herzustellen. Diverse Künstlernamen befinden sich sowohl in der Tabelle reviews der Pitchfork DB mit dem Namen artist als auch in der Tabelle songs der trackmetadata.db als artist_name. Um eine vollumfängliche Datenintegration und Datenkonsistenz zwischen beiden Tabellen gewährleisten zu können, müssen die Strings aus beiden Quellen fusioniert werden. Erst dann kann die zum Künstlernamen passende artist_id als der Fremdschlüssel reviewed_artist der Tabelle review der Musikdatenbank importiert werden (P 85-106). Dafür wird ähnlich wie bei dem vorherigen Import ein Wörterbuch artist_name_dict mit dem Schlüssel lower_artist_nametrack und Wert artist_id erstellt (P 94-97). Das sind die Daten aus der trackmetadata.db. Zur Vereinheitlichung der Namen werden sie durch Kleinschreibung normalisiert, um Verknüpfungen mit so vielen Namen wie möglich zu erzielen (P 95). Anschließend prüft das Skript durch eine for-Schleife, ob ein normalisierter Künstlername aus der Pitchfork DB auch im Wörterbuch vorkommt. Bei einer Übereinstimmung gibt die Methode get den Wert, also die artist_id, die mit dem Schlüssel, also dem Künstlernamen, assoziiert ist an die Variable reviewed_artist_id aus (P 102). Nun müssen diese Daten in die

`review` Tabelle der Musikdatenbank aufgenommen werden. Für eine Integration der Künstlernamen mit den Rezensionen in der review Tabelle, muss die Bedingung `SET reviewed_artist = ? WHERE review_id = ?", (reviewed_artist_id, review_id)` (P 103) gelten. Das bedeutet, dass nur die zuvor mit dem Wörterbuch übereinstimmenden Künstler IDs (`reviewed_artist_id`) in die Tabelle `review` aufgenommen werden, bei denen die `review_id` aus der Musik Datenbank mit der `review_id` der Pitchfork DB übereinstimmt (P 103). Durch eine SQL-Anfrage können wir berechnen, dass 9975 Künstler IDs in die Spalte `reviewed_artist` eingefügt wurden (P 106-109). Daraus folgt, dass aus 18389 Instanzen 9975, also ungefähr die Hälfte (54%) verknüpft wurden. Gründe dafür können sein, dass für die Verknüpfung die Normalisierung durch Kleinschreibung nicht ausreichend war und dass einige Künstler nicht in beiden Datenbanken existieren.

Eine ähnliche Vorgehensweise wurde auch angewendet, um die Albennamen aus der Spalte `release` der Tabelle `songs` der trackmetadata.db und der Spalte `album_name` der Tabelle `album` der Musikdatenbank durch Normalisierung der Textwerte zu verknüpfen und die `review_id` der übereinstimmenden Alben in die Spalte `reviewed_in` der `song` Tabelle einzufügen (M 31-47). Als Ergebnis stimmen 4936 aus 18389 (27%) Albennamen aus den beiden Datensätzen überein. Hier erhalten wir eine viel niedrigere Zahl als bei den Künstlernamen, weil Albennamen eine höhere Variation bezüglich der Zeichenlänge und von nicht alphanumerischen Symbolen aufweisen. Zur Verbesserung der Resultate beider Ergebnisse sollten noch zusätzliche Techniken der Normalisierung wie die Entfernung von Sonderzeichen und Leerstellen angewendet werden.

5. SQL-Anfragen

Besprochene Dateien: sql_queries.py, Anhang 3-6

Nach dem Datenimport kann nun im Skript sql_queries.py durch gezielte SQL-Anfragen diversen Fragestellungen über Zusammenhänge zwischen den Daten nachgegangen werden. Dabei ist das Skript folgendermaßen aufgebaut: Die Funktion `execute_sql` regelt Operationen wie die Verbindung, Ausführung der Anfragen und das Schließen des Datenbanksystems (7-20). Somit wird auf Dopplungen von Code nach jeder Anfrage verzichtet. Zusätzlich existiert ein Prüfmechanismus, der im Falle eines Auslesefehlers die Meldung „Es wurden keine Daten für die Anfrage gefunden.“ druckt (15). Die SQL-Anfragen werden nacheinander in Variablen gespeichert und der Funktion `execute_sql` übergeben (36). Für eine übersichtliche Darstellung der Ergebnisse ist eine for-Schleife eingebaut, welche die Tupelinstanzen je Zeile druckt (40-42).

`Query1` (Anhang 3) sucht nach Künstlern, deren Alben mit dem höchsten Score von 10.0 Punkten bewertet wurden und sortiert die Ergebnisse nach aufsteigenden Jahren (36-39). Insgesamt wurden von 1999 bis 2016 71 von 18389 (0,39%) Alben mit zehn Punkten bewertet. Es wird deutlich, dass ab 2009 pro Jahr deutlich mehr Alben mit der Höchstpunktzahl ausgezeichnet wurden als in den Jahren zuvor. Anhand der Spalte `album_year` erkennt man, dass viele Rezensionen Alben bewerten, die nicht im selben Jahr erschienen sind. Die Band „The Beatles“ sticht besonders hervor, denn sieben ihrer Alben erhielten die Höchstpunktzahl.

`Qquery1a` (Anhang 4) vergleicht die empfundene Berühmtheit der Künstler (`popularity`) mit dem durchschnittlichen Pitchfork Score (`AVG(r.score)`) und sortiert die Ergebnisse nach dem Jahr der Rezensionen, um zu erfahren, ob es einen Unterschied zwischen Kritiker- und Hörermeinungen gibt (45-50). Man sieht, dass Künstler mit der höchsten Popularität von 1.1 bis 0.7 viel niedriger bewertet wurden (beispielweise „Daft Punk“ `popularity` 1.02/ `score` 6.26). Zusätzlich liefert `query1a` die den populärsten Künstlern zugeschrieben Tags, indem die Tabellen `artist` und `has_tags`, sowie `has_tags` und `artist_tags` verknüpft werden. Daraus folgt, dass die populärsten Künstler den Tags "hip hop" und "rnb" zugeordnet werden.

`Query2` (Anhang 5) erstellt ein Autorenprofil der Personen, die Rezensionen für Pitchfork geschrieben haben, um zu erfahren, welches Punktevergabeverhalten sie pflegten (56-61). Es werden die zehn Autoren mit ihrem Autorentyp dargestellt (`LIMIT 10`), die die meisten Rezensionen geschrieben haben (`COUNT(*) AS num_reviews`). Mit den Aggregatfunktionen `AVG(r.score)`, `MIN(r.score)` und `MAX(r.score)` lassen sich jeweils die durchschnittliche, mindeste und maximale Punktzahl der von einem Autor geschrieben Rezensionen berechnen. Man kann sehen, dass der Autor Joe Tangari in der Zeitspanne des Datensatzes (1999-2016) 815 Rezensionen geschrieben hat, also 48 pro Jahr. Dabei hat er fast die gesamte Bannbreite der Punkte vergeben (2.1 bis 10.0). Interessanterweise ist über den Autorentyp der ersten fünf Autoren nichts bekannt. Was die Punktevergabe Gewohnheiten angeht, hat Marc Masters von allen zehn Autoren die höchste Mindestpunktzahl vergeben (4.3), aber auch nie mit voller Punktzahl bewertet. Insgesamt liegt die durchschnittliche Bewertung aller zehn Autoren bei etwa sieben Punkten, was für eine ausgeglichene Punktevergabe spricht.

Die letzte SQL-Anfrage `query3` (Anhang 6) bildet die top zehn Album Genres mit der höchsten Anzahl der Reviews ab, um nachzuvollziehen, über welche Musikrichtungen Pitchfork gerne Rezensionen schrieb (67-72). Dazu müssen die Anzahl der Reviews mit

`COUNT` gezählt und eine Verknüpfung zwischen den Tabellen `album` und `review` über den Schlüssel `review_id` hergestellt werden. Das Ergebnis zeigt, dass bei 2365 der rezensierten Alben kein Genre hinterlegt wurde und dass Alben mit den Genres Rock und Electronic am häufigsten bewertet wurden.

6. Visualisierungen

In diesem Kapitel werden mit der Wortwolke und dem Streudiagramm zwei Visualisierungsmöglichkeiten der Metadaten vorgestellt, mit denen der Wortschatz von Rezensionen und die Liedlänge analysiert und dargestellt wird.

6.1 Wortwolke: Wortschatz der Albumrezensionen

Besprochene Dateien: word_frequency_wordcloud.py, Wortwolke_album_reviews.png

Um den Wortschatz von Musikkritikern zu erforschen, kann man sich als ersten Anhaltspunkt die Art und Häufigkeit der Wörter in den Rezensionen anschauen. Zur Visualisierung dieses Sachverhalts eignet sich eine Wortwolke. Das Skript word_frequency_wordcloud.py benutzt die NLTK, die Wordcloud und die Matplotlib Bibliotheken, um dies umzusetzen. Für die Berechnung der Worthäufigkeit wird zuerst mit einer SQL-Anfrage auf den Inhalt der Rezensionen zugegriffen, der in der Variable `reviews` abgespeichert wird (17-18). Nachdem die englischen Stopwords ausgewählt wurden, wird die leere Liste `relevant_words` erstellt (21- 25). Eine for-Schleife normalisiert und tokenisiert erst alle Wörter ab der Position [0] und trägt dann alle alphanumerischen Wörter, die keine Stopwords sind, in die Liste ein (28-32). Somit enthält die Liste nur Wörter, die relevant für die Häufigkeitsberechnung sind. Mit dem NLTK Ausdruck `FreqDist(relevant_words.most_common(10)`lassen sich die zehn häufigsten Wörter zusammen mit der Anzahl ihres Vorkommens aus der Liste anzeigen (36, Bird). [3] Wie vielleicht intuitiv erwartet finden sich hier viele Wörter aus dem Bereich der Musik wie "album, music, band, songs, sound, record" wieder. Das häufigste Wort ist "like", welches darauf schließen lässt, dass in den Rezensionen Vergleiche mit anderen Entitäten wie Künstlern, Musikrichtungen oder Alben hergestellt werden.

Um ein umfangreicheres Bild dieses Zusammenhangs zu erhalten, wird eine Wortwolke mit Methode `generate` aus der Bibliothek von Andreas Mueller erstellt (40,

[3] Als Resultat erhalten wir die Tupel: [('like', 63638), ('album', 44048), ('music', 35074), ('one', 34920), ('band', 31856), ('songs', 27576), ('even', 23948), ('song', 23716), ('sound', 23513), ('record', 20171)].

„Little Word Cloud Generator“). Die Parameter der Wortwolke wie Breite, Höhe und Farbe können frei gewählt werden. Da `wordcloud` einen String zum Auslesen benötigt, müssen die Tupel aus der `relevant_words` Liste mit dem join-Operator in einen String mit Leerzeichen umgewandelt werden (ebd.). Anschließend kann die Wortwolke mit Matplotlib visualisiert werden (43-47, Mueller, „Masked wordcloud“). Anhang 7 zeigt die Wortwolke. Interessanterweise kommt hier zum Beispiel das zuvor am häufigsten vorkommende Wort "like" nicht vor. Die Häufigkeiten scheinen durch die Wordcloud Bibliothek anders berechnet zu sein als durch die NLTK Bibliothek.

6.2 Streudiagramm: Liedlänge im Jahrestrend?

Besprochene Dateien: scatterplot.py, scatterpot.png

Ich habe mich gefragt, wie sich die Länge von Liedern im Verlauf der Zeit entwickelt. Zur Visualisierung einzelner Lieder in Abhängigkeit von ihrer Länge und ihrem Erscheinungsjahr eignet sich ein Streudiagramm, welches im Skript scatterplot.py mit Matpoltlib gestaltet wurde. Zur Visualisierung braucht man das Jahr und die Liedlänge aus der Tabelle `song`, welche in der Variable `data` gespeichert werden (9-10). Da es viele Instanzen mit einer Liedlänge von „0“ gibt, schließen wir sie mit `song_year > 1` aus (9). Bei ersten Tests der Anfrage wurde deutlich, dass einige Liedlängen als deutlich unter einer Minute eingetragen wurden und dies nicht den eigentlichen Liedlängen entspricht. Deshalb modelliert die Restriktion `duration >= 60` ein Lied erst ab einer Dauer von 60 Sekunden (9).

Das Streudiagramm von Matplotlib benötigt separate Listen mit Werten für die x und die y-Achse („Matplotlib.Pyplot.Scatter“). Deshalb entpacken wir die Tupel aus `data` mit `zip(*data)` und erhalten zwei Listen mit den Jahren für die x- und den Längen für die y-Achse (12). Anschließend kann das Streudiagramm über `plt.scatter` angefertigt werden (20, „Matplotlib Scatter“). Für eine leichtere Lesbarkeit der Achsen ist die Beschriftung der Jahresangaben im Intervall von fünf Jahren gegliedert und um 46°C rotiert (22). Die Liedlänge in Sekunden ist im Abstand von jeweils zwei Minuten angesetzt (23). Eine Legende informiert über die Werte der Punkte im Streudiagramms (20-21,27).

Das erstellte Streudiagramm stellt jedoch wider Erwarten eine verzerrte Interpretationsgrundlage dar, denn es entsteht der Anschein, dass die Liedlängen mit zunehmenden Jahren von etwa zwei auf etwa 22 Minuten gestiegen sind (Anhang 8). Aus einer Million Liedern der `song` Tabelle gibt es zwar viele Ausreißer, jedoch pendeln sich die Durchschnittslängen der Lieder pro Jahr auf etwa vier Minuten ein. Um beim Betrachter

Missverständnisse zu vermeiden, enthält die Grafik gelbe Punkte, die über die Durchschnittslängen der Lieder pro Jahr informieren (15-17, 21). Demnach ist das Abbilden aller Lieder im Streudiagramm nicht geeignet, um aussagekräftige Trends zwischen der Liedlänge und den Jahren zu bestimmen. Meine Visualisierung eignet sich eher als Überblick über die Position und Spannbreite der Lieder im zweidimensionalen Raum von Liedlänge und Jahren. Um die Grafik anzureichern, könnte man die Liedgenres verschiedenfarbig markieren, um herauszufinden, ob bestimmte Genres eher zu kurzen oder langen Liedern neigen.

7. Implementierung der Datenbank: Künstlerempfehlung

Benutzte Datei: user_recommendation.py

Als letztes Projekt habe ich einen interaktiven Künstlerempfehlungsalgorithmus geschrieben, der auf kleinem Niveau die Vorschläge von Streamingdiensten imitieren soll. Für eine ansprechende Interaktion mit dem Benutzer gibt das Programm je nach Stadium der Anfrage verschiedene Printstatements aus. Anfangs werden eine Begrüßung und die Funktionsweise des Music-Matchers eingeblendet (8-14). Falls es einen Treffer in der Datenbank gibt, wird dem Benutzer eine Liste mit zu dem eingegebenen Künstler ähnlichen Künstlern ausgegeben (17-27). Auf Wunsch kann darauf der nächste Name eingegeben werden (28). Ansonsten kann die while-Schleife jederzeit mit dem string "ENDE" verlassen werden (14-16). Falls kein Künstler gefunden wurde, wird dies dem Benutzer mitgeteilt (29-30).

Die Informationen über die Künstlerähnlichkeit werden in der Variable `sqlcmd` gespeichert und anschließend ausgeführt (18-21). Da die `artist_id` aus der `artist` Tabelle sowohl mit `target_artist` als auch `similar_artist` aus der `similarity` Tabelle verknüpft werden muss, werden die Präfixe „a1“ und „a2“ zur Unterscheidung verwendet. Anstelle des Placeholders wird die Benutzereingabe eingesetzt. Aufpassen muss man bei der Schreibweise in Zeile 22: Setzt man hinter `search_user` kein Komma, erscheint die Fehlermeldung: "sqlite3.ProgrammingError: Incorrect number of bindings supplied." In der SQLite Dokumentation steht, dass `execute` bei unbenannten Placeholdern den Parameter `sequence`, der ein Tupel sein kann, brauchen (Cursor.execute). Deshalb muss aus dem String `search_user` durch ein Komma am Ende ein Tupel erstellt werden. Eine Beispielantwort des Music-Matchers über ähnliche Künstler zu der Band Muse befindet sich im Anhang 9. Für ein umfangreicheres Benutzererlebnis könnte die SQL-Anfrage mit

zusätzlichen Verknüpfungen erweitert werden, um noch das höchst bewertete Album des ähnlichen Künstlers mit dem Text der Rezension anzuzeigen.

8. Fazit

Dieser Bericht hat demonstriert, wie durch Datenmodellierung und Datenbankentwurf Musikmetadaten aus dem Zeitraum 1922 bis 2010 und Rezensionsmetadaten aus den Jahren 1999-2016 in eine relationale Beziehung gesetzt werden können. Dabei ist es besonders wichtig vor der SQLite Implementierung die Struktur der verfügbaren Datensätze zu verstehen und anhand der gewünschten Modellierung eines Sachverhalts der realen Welt die Entitäten, Attribute und Beziehungen sowie ihre Kardinalitäten im ER-Modell visuell darzustellen und anschließend in das relationale Modell zu übertragen. Dies stellt eine anschließende erfolgreiche Tabellenanfertigung mit den dazugehörigen Werten, sowie Primär- und Sekundärschlüsseln sicher. Die Datenimporte haben sich durch die gegebene Komplexität von vier Quelldatenbanken und den manuell zu erstellenden Schlüsseln als am anspruchsvollsten erwiesen.

Durch die SQL-Anfragen habe ich herausgefunden, dass Pitchfork innerhalb von 17 Jahren nur wenigen Alben die Höchstpunktzahl verliehen hat und dass die Autoren aus der vollen Breite der Punkte schöpfen. Außerdem wurde deutlich, dass sich die in der Gesellschaft populärsten Künstler von den Künstlern mit den höchstbewerteten Pitchfork Rezensionen unterscheiden. Über die Wortwolke erhält der Betrachter eine ansprechende Übersicht über den verwendeten Wortschatz der Pitchfork Rezensionen. Das Kapitel über das Streudiagramm hat demonstriert, wie wichtig die Wahl der Darstellungsform für die Visualisierung eines Sachverhaltes ist. Ich wollte die Verteilung der einzelnen Liedlängen innerhalb der Zeit visualisieren. Das Streudiagramm suggeriert dem Betrachter jedoch einen Trend zur Verlängerung der Lieder im Verlauf der Zeit, obwohl die durchschnittliche Liedlänge konstant bleibt. Dies führt zur Verzerrung des Sachverhaltes und spiegelt nicht den gewünschten Sachverhalt wider. Der Versuch einer interaktiven Künstlerempfehlung mit dem Music-Matcher zeigt, dass Datenbanken nicht nur für archivarische oder analytische Zwecke, sondern auch für Use Cases aus der realen Welt verwendet werden können. Durch das Projekt der Modellierung und Implementierung einer Musikdatenbank habe ich das Potential von Datenbanken für verschiedenste Anwendungen entdeckt und hoffe dieses Wissen in weiteren Projekten der Digital Humanities anwenden und erweitern zu können.

9. Quellen

Bird, Steven, Edward Loper und Ewan Klein, *Natural Language Processing with Python*. O'Reilly Media Inc., 2009, tedboy.github.io/nlps/generated/generated/nltk.FreqDist.most_common.html.

Conaway, Nolan. „18,393 Pitchfork Reviews." *Kaggle*, 13.01.2017, www.kaggle.com/datasets/nolanbconaway/pitchfork-data.

Conaway, Nolan. „Pitchfork Data." *GitHub*, 08.01.2016, github.com/nolanbconaway/pitchfork-data/tree/master/scrape.

„Cursor.execute." *SQLITE3 - Documentation*, Version 3.12.2, docs.python.org/3/library/sqlite3.html#sqlite3.Cursor.execute.

„Draw.Io." *Flowchart Maker & Online Diagram Software*, app.diagrams.net.

Hellweg, Dennis. „Composite Primary Keys in Sqlite - DB Pilot." *DB Pilot Icon*, DB Pilot, 15.07.2023, www.dbpilot.io/sql-guides/sqlite/composite-primary-keys-in-sqlite.

Hunter, John D. „Matplotlib: A 2D graphics environment." *Computing in Science & Engineering*, Bd. 9, Nr. 3, 2007, 90–95, https://doi.org/10.5281/zenodo.10661079.

Itzkoff, Dave. „Inside Pitchfork, the Site That Shook up Music Journalism." *Wired*, Conde Nast, 13.10.2015, www.wired.com/2015/10/the-pitchfork-effect.

Kemper, Alfons und Eickler, André. *Datenbanksysteme: Eine Einführung*, De Gruyter Oldenbourg, 2009.

„Matplotlib Scatter." *W3Schools*, www.w3schools.com/python/matplotlib_scatter.asp.

„Matplotlib.Pyplot.Scatter."*Matplotlib 3.8.3 Documentation*, In matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.scatter.html.

Mueller, Andreas C. „A Little Word Cloud Generator in Python." *GitHub*, Version 1.9.1, 27.04.2023, github.com/amueller/word_cloud.

Mueller, Andreas C. „Masked wordcloud.” *GitHub*, 170.06.2018, github.com/amueller/word_cloud/blob/main/examples/masked.py.

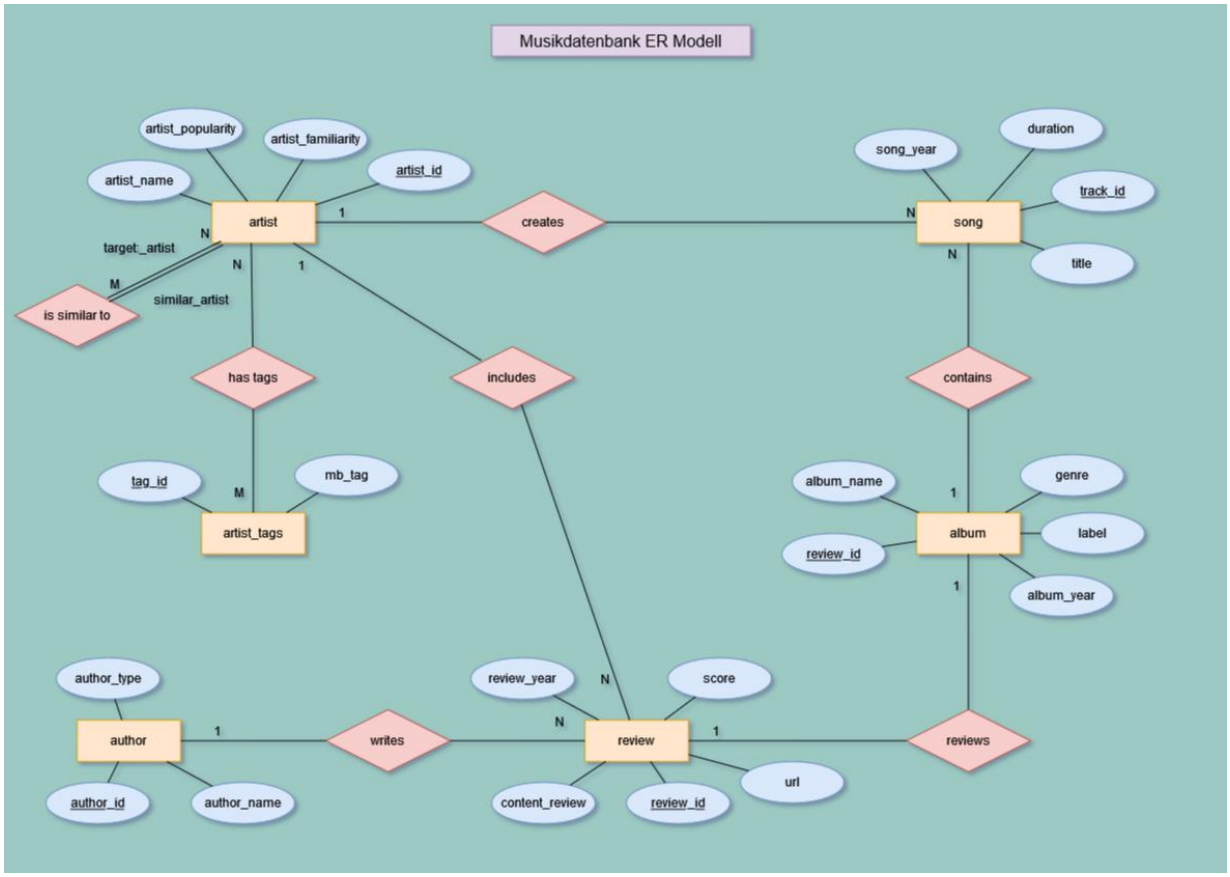
„Removing Stop Words with NLTK in Python.” *GeeksforGeeks*, 3.01.2024, www.geeksforgeeks.org/removing-stop-words-nltk-python.

„Placeholders.“ *SQLITE3 - Documentation*, Version 3.12.2, docs.python.org/3/library/sqlite3.html#sqlite3-placeholders.

Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman und Paul Lamere. „The Million Song Dataset”. Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011, millionsongdataset.com/faq.

10. Anhang

Anhang 1: ER-Modell



Anhang 1: Musikdatenbank_ER.png

Anhang 2: Relationales Modell

Konvertierung des ER-Musikdatenbankmodells zum relationalen Modell

MUSIKDATENBANK = {artist, artist_tags, has_tags, similarity, song, album, review, author}
artist: {artist_id:varchar, artist_name: varchar, artist_popularity:float, artist_familiarity:float}
artist_tags: {tag_id: int, tag: text}
has_tags: {↑ artist_id:varchar, ↑tag_id:int }
~~is_similar_to~~ similarity: {~~ artist_id~~ ↑ target_artist:varchar, ~~artist_id~~ ↑ similar_artist:varchar}
song { track_id: varchar, title: text, song_year:int, duration:float, ↑ artist_ID:varchar, ~~review_id~~ ↑ reviewed_in:int}
album: { review_id:int, album_name: text, genre:text, album_year:int, label:text}
review: { review_id:int, content_review:text, score:float, review_year:int, url:text, ~~artist_id~~ ↑reviewed_artist:varchar, ~~author_id~~ ↑review_author:varchar}
author: { author_id:varchar, author_name:text, author_type:text}

einzelne Schritte zur Nachverfolgung:
1. Konvertierung starker Entity-Typen

artist: {artist_id, artist_name, artist_popularity, artist_familiarity}
artist_tags: { artist_id, mb_tag }
song: { track_id, title, song_year, duration }
album: { review_id , album_name , genre , album_year , label}
review: { review_id, content_review, score, review_year, url }
author: { author_id, author_name, author_type }

2. N:M

artist: {artist_id, artist_name, artist_popularity, artist_familiarity, ~~track_id~~ song_id}

~~is_similar_to~~ similarity: {~~artist_id~~ target_artist~~, artist_id~~ similar_artist}

artist: {artist_id, artist_name}

tags: {tag_id, tag}

has_tags: {{ artist_id, tag_id }

3. 1:N

artist: {artist_id, artist_name, artist_popularity, artist_familiarity}

song: { track_id, title, song_year, duration, artist_ID}

~~creates: { artist_id, track_id }~~

artist: {artist_id, artist_name, artist_popularity, artist_familiarity, ~~track_id~~ song_id}

review: { review_id, content_review, score, review_year, url, ~~artist_id~~ reviewed_artist}

~~includes: {artist_id, review_id}~~

song { track_id, title, song_year, duration, ~~review_id~~ reviewed_in }

album: album: { review_id , album_name , genre , album_year , label}

~~contains: { track_id, { review_id}~~

author: { author_id, author_name, author_type }

review: { review_id, content_review, score, review_year, url, ~~artist_id~~ reviewed_artist, ~~author_id~~ review_author}

~~writes { author_id , review_id }~~

4. 1:1

review: { review_id, content_review, score, review_year, url, ~~artist_id~~ reviewed_artist, ~~author_id~~ review_author }

album: { review_id , album_name , genre , album_year , label}

~~reviews: { review_id, review_id }~~

Anhang 3: query1

score	artist_name	review_year	album_name	album_year
10.0	Bonnie Prince Billy	1999	i see a darkness	1999
10.0	Radiohead	2000	kid a	2000
10.0	Pink Floyd	2000	animals	
10.0	JOHN COLTRANE	2001	the olatunji concert: the last live recording	2001
10.0	Elvis Costello & The Attractions	2002	this year's model	1979
10.0	Wilco	2002	yankee hotel foxtrot	2002
10.0	...And You Will Know Us By The Trail Of Dead	2002	source tags and codes	2002
10.0	Television	2003	marquee moon	1977
10.0	Glenn Branca	2003	the ascension	2003
10.0	Pavement	2004	crooked rain, crooked rain: la's desert origins	1994
10.0	The Clash	2004	london calling: 25th anniversary legacy edition	2004
10.0	Boards of Canada	2004	music has the right to children	1998
10.0	James Brown	2004	live at the apollo [expanded edition]	1963
10.0	Bruce Springsteen	2005	born to run: 30th anniversary edition	2005
10.0	Neutral Milk Hotel	2005	in the aeroplane over the sea	1998
10.0	DJ Shadow	2005	endtroducing... [deluxe edition]	1996
10.0	Wire	2006	pink flag	1977
10.0	Joy Division	2007	unknown pleasures	1979
10.0	Sonic Youth	2007	daydream nation: deluxe edition	1988
10.0	R.E.M.	2008	murmur [deluxe edition]	2008
10.0	Otis Redding	2008	otis blue: otis redding sings soul [collector's edition]	1965
10.0	The Stone Roses	2009	the stone roses	2009
10.0	The Beatles	2009	the beatles	1968

10.0	The Beatles	2009	abbey road	1969
10.0	The Beatles	2009	rubber soul	1965
10.0	The Beatles	2009	revolver	1966
10.0	The Beatles	2009	sgt. pepper's lonely hearts club band	1967
10.0	The Beatles	2009	magical mystery tour	1967
10.0	The Beatles	2009	stereo box	2009
10.0	Radiohead	2009	kid a: special collectors edition	2009
10.0	R.E.M.	2009	reckoning [deluxe edition]	2009
10.0	Michel Colombier	2009	histoire de melody nelson	1971
10.0	Beastie Boys	2009	paul's boutique	1989
10.0	Kanye West / Mos Def	2010	my beautiful dark twisted fantasy	2010
10.0	The Cure	2010	disintegration [deluxe edition]	2010
10.0	The Rolling Stones	2010	exile on main st. [deluxe edition]	2010
10.0	Pavement	2010	quarantine the past	2010
10.0	Spiritualized	2010	ladies and gentlemen we are floating in space [collector's editon]	2009
10.0	Can	2011	tago mago [40th anniversary edition]	1971
10.0	The Beach Boys	2011	the smile sessions	2011
10.0	Talk Talk	2011	laughing stock	2011
10.0	Nirvana	2011	nevermind [20th anniversary edition]	2011
10.0	The Dismemberment Plan	2011	emergency & i [vinyl reissue]	2011
10.0	William Basinski	2012	the disintegration loops	2012
10.0	My Bloody Valentine	2012	isn't anything	1988
10.0	Nirvana	2013	in utero: 20th anniversary edition	1993
10.0	Peter Green	2013	rumours	1977
10.0	Nas	2013	illmatic	1994
10.0	J Dilla	2013	donuts (45 box set)	2006
10.0	Public Enemy	2014	it takes a nation of millions to hold us back	1988
10.0	The Velvet Underground	2014	the velvet underground 45th anniversary super deluxe edition	1969
10.0	Mobb Deep	2014	the infamous	1995
10.0	The Velvet Underground	2014	white light/white heat	1968

10.0	JOHN COLTRANE	2015	a love supreme: the complete masters	2015
10.0	A Tribe Called Quest	2015	people's instinctive travels and the paths of rhythm	1990
10.0	Van Morrison	2015	astral weeks	1968
10.0	The Velvet Underground	2015	loaded: re-loaded 45th anniversary edition	1970
10.0	The Rolling Stones	2015	sticky fingers	1971
10.0	Public Image Ltd	2016	metal box	1979
10.0	Bob Dylan	2016	blood on the tracks	1975
10.0	Brian Eno	2016	another green world	1975
10.0	Stevie Wonder	2016	songs in the key of life	1976
10.0	Nina Simone	2016	in concert	1964
10.0	Neil Young	2016	tonight's the night	1975
10.0	Kate Bush	2016	hounds of love	1985
10.0	Prince	2016	sign "o" the times	1987
10.0	Prince	2016	1999	1982
10.0	Prince	2016	dirty mind	1980
10.0	Michael Jackson	2016	off the wall	1979
10.0	David Bowie	2016	"heroes"	1977
10.0	David Bowie	2016	low	1977

Anhang 4: query1a

artist_name	artist_popularity	average_score	tag
Kanye West / Mos Def	1.08250255673	8.425	hip-hop
Daft Punk	1.02125558749	6.25714285714286	french
T.I. and Jim Jones	0.947858226052	6.65555555555556	hip hop rnb and dance hall
Coldplay	0.916053228284	5.72	british
Rihanna	0.908202619208	6.55	barbadian
Trey Songz	0.8863227203	6.4	hip hop rnb and dance hall
Eminem	0.879236744738	5.8	hip-hop
Usher	0.851233891127	7.1	hip hop rnb and dance hall
The Beatles	0.840462688027	8.89047619047618	british
Cee-Lo	0.829597258356	6.75	hip hop rnb and dance hall
Lady GaGa	0.823266685206	7.35	pop
Alicia Keys	0.821648466961	6.8	rnb
The Killers	0.819588282229	5.78	rock and indie
Weezer	0.816312599454	4.46666666666666	american
Green Day	0.812307796139	5.72	punk rock
Bruce Springsteen	0.807587096405	7.61818181818182	rock
Ne-Yo	0.797990081888	6.4	hip hop rnb and dance hall
Kings Of Leon	0.788805935162	4.78571428571429	american
Mariah Carey	0.787005468689	8.1	whistle register
R. Kelly	0.783843023046	6.73333333333333	rnb
Metallica	0.780334119573	5.2	metal
Wiz Khalifa	0.778657268782	6.03333333333333	hip hop rnb and dance hall
Adele	0.768886618761	7.3	pop and chart
Michael Jackson	0.766545450757	7.76666666666668	pop
Jimmy Eat World	0.757673698624	4.6	rock and indie
Deadmau5 Feat. MC Flipside	0.755817702881	5.9	dance and electronica
U2	0.750311503248	7.11818181818182	irish
Johnny Cash	0.747251065679	7.65	country

The Black Keys	0.7417239014	7.175	blues rock
B.o.B	0.738869426689	4.2	united states
Dr. Dre	0.736085314699	8.65	hip hop
The Rolling Stones	0.735765692951	8.075	british
Miranda Lambert	0.724708598132	7.8	country
Guns N' Roses	0.71366981958	5.8	hard rock
Miley Cyrus	0.71112820829	3.0	miley-cyrus
Vampire Weekend	0.708889028041	8.675	american
Ghostface Killah	0.702102406983	7.025	hip hop rnb and dance hall
Elvis Presley	0.697315675002	8.0	rock
Ryan Adams & The Cardinals	0.690201297029	6.6	country
MGMT	0.686812712165	6.9	américain
Radiohead	0.675887011418	7.92307692307692	rock
John Lennon	0.67240252398	7.1	british
Snow Patrol	0.667294595643	6.375	northern irish
Sufjan Stevens	0.66577089971	7.89166666666667	folk rock
Ace Karaoke Productions	0.665576619159	4.9	pop
Queen	0.664676434343	6.7	rock
Regina Spektor	0.663526023988	6.4	américain
Arcade Fire	0.661957638324	8.54	canadian
Nirvana	0.661382924338	8.2	soft rock
The Temper Trap	0.659898510193	5.15	australia
Ciara	0.657663310479	5.98	pop and chart
Crystal Castles	0.657558086441	7.775	electronica
Jazmine Sullivan	0.654075995756	8.1	hip hop rnb and dance hall
The White Stripes	0.653897217428	7.66666666666667	indie rock
Robert Plant	0.653543774817	7.0	uk
The Decemberists	0.651676012516	7.08571428571429	indie pop
The Smashing Pumpkins	0.64722472378	6.96363636363636	alternative rock
Kylie Minogue	0.645488743476	6.23333333333333	australian
Etta James	0.641347950577	9.0	rnb
Rufus Wainwright	0.64041969134	6.7	popera
Pink Floyd	0.638906376718	7.58	progressive rock

Anhang 5: query2

	author_name	num_reviews	author_type	average_score	lowest_score	highest_score
1	joe tangari	815	NULL	7.37533742331288	2.1	10.0
2	stephen m. deusner	725	NULL	6.97241379310345	0.2	10.0
3	ian cohen	699	NULL	6.35393419170243	0.2	10.0
4	brian howe	500	NULL	7.05760000000001	2.8	9.6
5	mark richardson	476	NULL	7.57857142857143	1.0	10.0
6	stuart berman	445	associate editor	7.08921348314606	1.0	10.0
7	marc hogan	439	senior staff writer	6.60410022779044	0.9	9.2
8	nate patrin	347	contributor	7.04149855907781	2.8	10.0
9	marc masters	312	contributor	7.46891025641026	4.3	9.8
10	jayson greene	299	associate editor	7.10635451505017	1.0	10.0

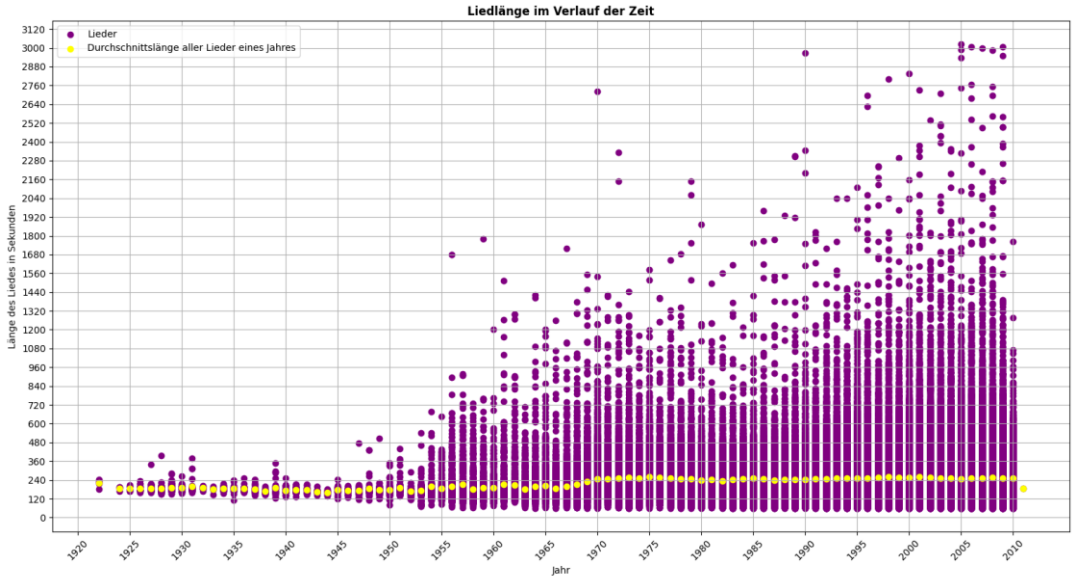
Anhang 6: query3

	genre	number_reviews
1	rock	6428
2	electronic	3874
3	*NULL*	2365
4	experimental	1617
5	rap	1308
6	pop/r&b	1048
7	metal	730
8	folk/country	646
9	jazz	220
10	global	153

Anhang 7: Wortwolke

Anhang 7: Wortwolke_album_reviews.png

Anhang 8: Streudiagramm



Anhang 8: scatterpot.png

Anhang 9: Music-Matcher

```
Willkommen beim Music-Matcher! Entdecke, welche Künstler ähnlich zueinander sind.
Deine Künstlerempfehlungem warten schon auf dich!
Gib einen Künslter ein, den du magst (z.B. Muse, Depeche Mode, Nirvana):
Um das Programm zu beenden gib ENDE ein.
Muse
Hier ist eine Liste von Künstlern, die ähnlich sind wie Muse:
Radiohead
Placebo
The Smashing Pumpkins
Kaiser Chiefs
Arctic Monkeys
Coldplay
Yeah Yeah Yeahs
Rage Against The Machine
Kasabian
Franz Ferdinand
The Cooper Temple Clause
Jeremy Enigk
Jeff Buckley
We Are Scientists
Them Crooked Vultures
Bright Eyes
The Last Shadow Puppets
My Chemical Romance
Skunk Anansie
Longpigs / Hugh Jones
People In Planes
The Killers
The Mars Volta
Our Lady Peace
U2
Blur
Coldplay Tribute
Sleeping At Last
Ash
Queens Of The Stone Age
Keane
Snow Patrol
Panic! At The Disco
Oasis
Ross Copperman
The White Stripes
Stereophonics
The Temper Trap
Starsailor
Rufus Wainwright
Suede
Sick Puppies
```

```
The Bravery
The Crash Motive
Remy Zero
Razorlight
The Strokes
Phantom Planet
The Fratellis
Weezer
The Kooks
VAST
Wishbone Ash
Pulsedriver
The Raconteurs
Supergrass
Wolfmother
Travis
Ours
Dragon Ash
Mustafa Ozkent
Queen
The Vines
Buffalo Killers
Nirvana
The Honeymoon Killers
David Bowie
Pendulum
Placebo Effect
Teddy Thompson / Rufus Wainwright
The Automatic
Ash Grunwald
The Little Killers
OneRepublic
The Beach Boys
Höre gerne rein! Oder versuche es noch einmal
Gib einen Künslter ein, den du magst (z.B. Muse, Depeche Mode, Nirvana):
Um das Programm zu beenden gib ENDE ein.
```