

This assignment is inspired by the SQL assignment from CMU.

<https://15445.courses.cs.cmu.edu/fall2020/homework1/>

Instructions

Download the data from here -

<https://15445.courses.cs.cmu.edu/fall2020/files/musicbrainz-cmdb2020.db.gz>

Step - 1: Load the Database

1. Download the data from the above mentioned link.
2. Unzip the data to retrieve **musicbrainz-cmdb2020.db** file.
3. Check the contents of the database by running the **.tables** command on the sqlite3 terminal. You should see **fifteen tables**, and the output should look like this:

```
C:\Users\Kanav\Downloads\Task _ SQL\yTask>sqlite3 musicbrainz-cmdb2020.db
SQLite version 3.33.0 2020-08-14 13:23:32
Enter ".help" for usage hints.
sqlite> .tables
area                artist_credit_name  medium              release_status
artist              artist_type         medium_format       work
artist_alias        gender              release             work_type
artist_credit        language            release_info
sqlite>
```

Step - 2: Check the schema

Get familiar with the schema (structure) of the tables (what attributes do they contain, what are the primary and foreign keys). Run the **.schema \$TABLE_NAME** command on the sqlite3 terminal for each table. The output should look like the example below for each table.

area

```
sqlite> .schema area
CREATE TABLE [area] (
  [id] INTEGER,
  [name] TEXT,
  [comment] TEXT
);
sqlite>
```

Contains details for an area. For example, this is a row from the table:

95339|Great Neck|village

For us, the important field is name (e.g., "Great Neck").

artist

```
sqlite> .schema artist
CREATE TABLE [artist] (
  [id] INTEGER,
  [name] TEXT,
  [begin_date_year] INTEGER,
  [begin_date_month] INTEGER,
  [begin_date_day] INTEGER,
  [end_date_year] INTEGER,
  [end_date_month] TEXT,
  [end_date_day] TEXT,
  [type] INTEGER,
  [area] INTEGER,
  [gender] INTEGER,
  [comment] TEXT
);
sqlite> CREATE INDEX ix_artist_name ON artist (name);
sqlite> CREATE INDEX ix_artist_area ON artist (area);
```

Contains details of an artist. For example, this is a row from the table:

519|Michael Jackson|1958|8|29|2009|6|25|1|222|1|"King of Pop"

For us, the important fields are name (e.g., "Michael Jackson") and area (e.g., 222).

artist_alias

```
sqlite> .schema artist_alias
CREATE TABLE [artist_alias] (
  [id] INTEGER,
  [artist] INTEGER,
  [name] TEXT
);
sqlite> CREATE INDEX ix_artist_alias ON artist_alias(artist);
```

Contains alternate names for the artists. For example, this is a row from the table:

125604|916107|AOA

artist_credit

```
sqlite> .schema artist_credit
CREATE TABLE [artist_credit] (
  [id] INTEGER,
  [name] TEXT,
  [artist_count] INTEGER
);
sqlite> CREATE INDEX ix_artist_credit_id ON artist_credit (id);
```

Contains lists of artists. For example, this is a row from the table:
966419|Bounty Killer feat. Beenie Man & Dennis Brown|3

artist_credit_name

```
sqlite> .schema artist_credit_name
CREATE TABLE [artist_credit_name] (
  [artist_credit] INTEGER,
  [position] INTEGER,
  [artist] INTEGER,
  [name] TEXT
);
sqlite> CREATE INDEX ix_artist_credit_name ON artist_credit_name (artist_credit);
```

Contains mappings from artist credits to artists. For example, this is a row from the table:

966419|0|39320|Bounty Killer

artist_type

```
sqlite> .schema artist_type
CREATE TABLE [artist_type] (
  [id] INTEGER,
  [name] TEXT
);
```

Contains details of an artist type. For example, this is a row from the table:
1|Person

gender

```
sqlite> .schema gender
CREATE TABLE [gender] (
  [id] INTEGER,
```

```
[name] TEXT,  
[description] TEXT  
);
```

Contains details for a gender type. For example, this is a row from the table:

1|Male|NULL

language

```
sqlite> .schema language  
CREATE TABLE [language] (  
  [id] INTEGER,  
  [name] TEXT  
);
```

Contains details for a language type. For example, this is a row from the table:

120|English

medium

```
sqlite> .schema medium  
CREATE TABLE [medium] (  
  [id] INTEGER,  
  [release] INTEGER,  
  [position] INTEGER,  
  [format] INTEGER,  
  [name] TEXT  
);  
sqlite> CREATE INDEX ix_medium_release ON medium (release);
```

Contains details for a medium. For example, this is a row from the table:

287750|287750|1|8|NULL

For us, the important fields are release (e.g., 287750) and type (e.g., 8).

medium_format

```
sqlite> .schema medium_format  
CREATE TABLE [medium_format] (  
  [id] INTEGER,  
  [name] TEXT,  
  [description] TEXT  
);  
sqlite> CREATE INDEX ix_medium_format_id on medium_format (id);
```

Contains details for a medium format. For example, this is a row from the table:

1|CD|NULL

release

```
sqlite> .schema release
```

```
CREATE TABLE [release] (
```

```
  [id] INTEGER,
```

```
  [name] TEXT,
```

```
  [artist_credit] INTEGER,
```

```
  [status] INTEGER,
```

```
  [language] INTEGER,
```

```
  [comment] TEXT
```

```
);
```

```
sqlite> CREATE INDEX ix_release_id ON release (id);
```

```
sqlite> CREATE INDEX ix_release_artist_credit ON release (artist_credit);
```

The table contains music releases. For example, this is a row from the table:

1671523|Back to the Future, Part III: Original Motion Picture Soundtrack|21443|1|120|25th anniversary edition

release_info

```
sqlite> .schema release_info
```

```
CREATE TABLE [release_info] (
```

```
  [release] INTEGER,
```

```
  [area] INTEGER,
```

```
  [date_year] INTEGER,
```

```
  [date_month] INTEGER,
```

```
  [date_day] INTEGER
```

```
);
```

```
sqlite> CREATE INDEX ix_release_info_release ON release_info (release);
```

The table contains the detailed information of a release. For example, this is a row from the table:

1966419|222|2017|1|13

release_status

```
sqlite> .schema release_status
```

```
CREATE TABLE [release_status] (  
  [id] INTEGER,  
  [name] TEXT,  
  [description] TEXT  
);
```

Contains the details of a release status. For example, this is a row from the table:

1|Official|Any release officially sanctioned by the artist and/or their record company. Most releases will fit into this category.

work

```
sqlite> .schema work  
CREATE TABLE [work] (  
  [id] INTEGER,  
  [name] TEXT,  
  [type] INTEGER,  
  [comment] TEXT  
);  
sqlite> CREATE INDEX ix_work_name ON work (name);  
sqlite> CREATE INDEX ix_work_type ON work (type);
```

Contains the details of a work. For example, this is a row from the table:

12446282|Thousand Miles Behind|17|NULL

work_type

```
sqlite> .schema work_type  
CREATE TABLE [work_type] (  
  [id] INTEGER,  
  [name] TEXT,  
  [description] TEXT  
);  
sqlite> CREATE INDEX ix_work_type_name ON work_type (name);
```

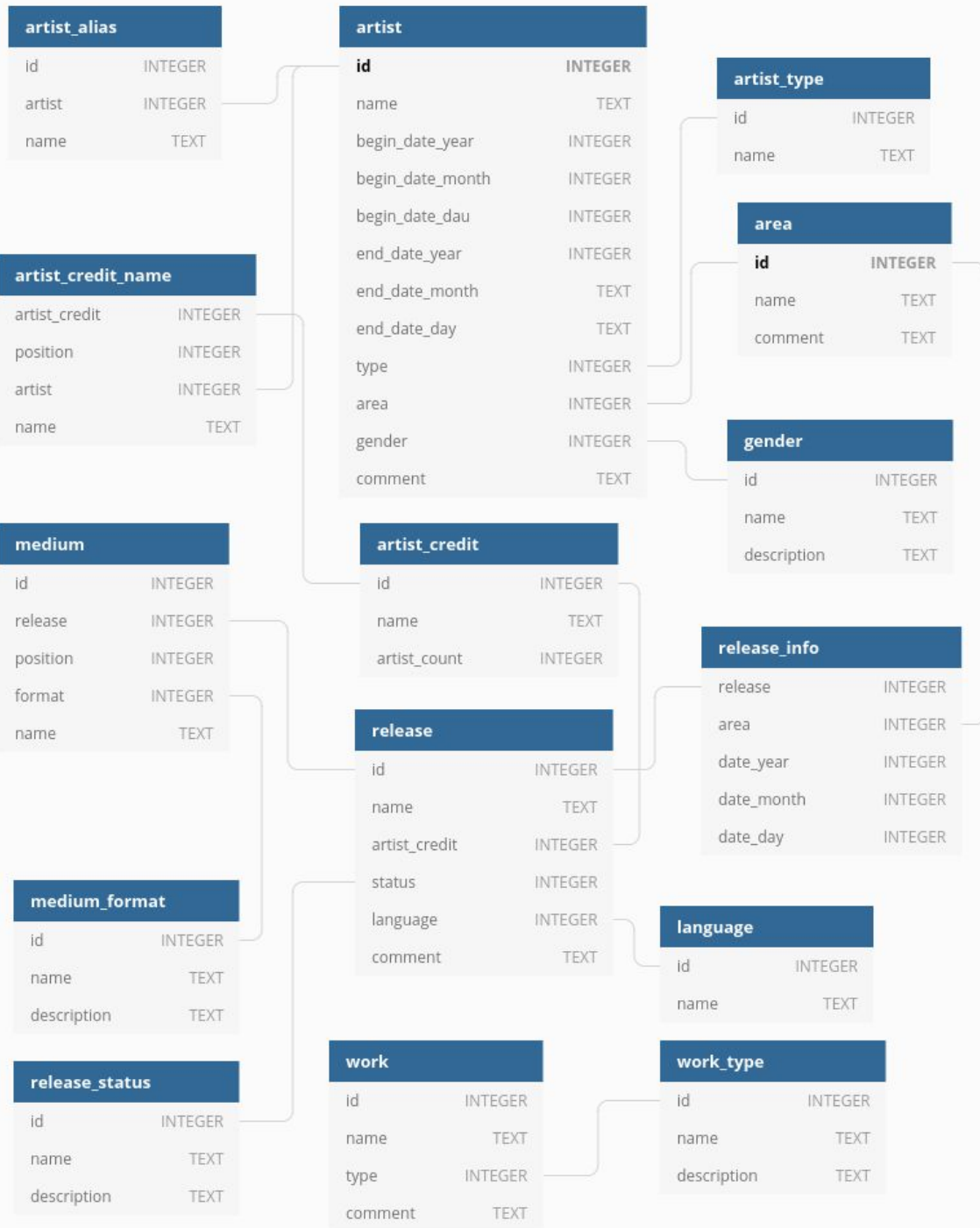
Contains the details of a work type. For example, this is a row from the table:

14|Quartet|A quartet is a musical composition scored for four voices or instruments.

Sanity Check

Count the number of rows in the table

```
sqlite> select count(*) from artist;  
1682989
```



Step - 3: Construct the SQL Queries

Now, it's time to start constructing the SQL queries and put them into the placeholder files.

Q1 (q1_sample):

The purpose of this query is to make sure that the formatting of your output matches exactly the formatting of our auto-grading script.

Details: List all types of work ordered by type ascendingly.

Answer: Here's the correct SQL query and expected output:

```
sqlite> select name from work_type order by name;
```

Answer:

Aria
Audio drama
Ballet
Beijing opera
Cantata
Concerto
Incidental music
Madrigal
Mass
Motet
Musical
Opera
Operetta
Oratorio
Overture
Partita
Play
Poem
Prose
Quartet
Sonata
Song
Song-cycle
Soundtrack
Suite
Symphonic poem
Symphony
Zarzuela
Etude

Q2 (q2_long_name):

List works with the longest name of each type.

Details: For each work type, find works that have the longest names. There might be cases where there is a tie for the longest names - in that case, return all of them. Display work names and corresponding type names, and order it according to work type (ascending) and use work name (ascending) as tie-breaker.

Q3 (q3_old_music_nations):

List top 10 countries with the most classical music artists (born or started before 1850) along with the number of associated artists.

Details: Print country and number of associated artists before 1850. For example, Russia|191. Sort by number of artists in descending order.

Q4 (q4_dubbed_smash):

List the top 10 dubbed artist names with the number of dubs.

Details: Count the number of distinct names in artist_alias for each artist in the artist table, and list only the top ten who's from the United Kingdom and started after 1950 (not included). Print the artist name in the artist table and the number of corresponding distinct dubbed artist names in the artist_alias table.

Q5 (q5_vinyl_lover):

List the distinct names of releases issued in vinyl format by the British band Coldplay.

Details: Vinyl format includes ALL vinyl dimensions excluding VinylDisc. Sort the release names by release date ascendingly.

Q6 (q6_old_is_not_gold):

Which decades saw the most number of official releases? List the number of official releases in every decade since 1900. Like 1970s|57210.

Details: Print all decades and the number of official releases. Releases with different issue dates or countries are considered different releases. Print the relevant decade in a fancier format by constructing a string that looks like this: 1970s. Sort the decades in decreasing order with respect to the number of official releases and use decade (descending) as tie-breaker. Remember to exclude releases whose dates are NULL.

Q7 (q7_release_percentage):

List the month and the percentage of all releases issued in the corresponding month all over the world in the past year. Display like 2020.01|5.95.

Details: The percentage of releases for a month is the number of releases issued in that month divided by the total releases in the past year from 07/2019 to 07/2020, both included. Releases with different issue dates or countries are considered different releases. Round the percentage to two decimal places using ROUND(). Sort by dates in ascending order.

Q8 (q8_collaborate_artist):

List the number of artists who have collaborated with Ariana Grande.

Details: Print only the total number of artists. An artist is considered a collaborator if they appear in the same artist_credit with Ariana Grande. The answer should include Ariana Grande herself.

Q9 (q9_dre_and_eminem):

List the rank, artist names, along with the number of collaborative releases of Dr. Dre and Eminem among other most productive duos (as long as they appear in the same release) both started after 1960 (not included). Display like [rank]|Dr. Dre|Eminem|[# of releases].

Details: For example, if you see a release by A, B, and C, it will contribute to three pairs of duos: A|B|1, A|C|1, and B|C|1. You will first need to calculate a rank of these duos by number of collaborated releases (release with artist_credit shared by both artists) sorted descendingly, and then find the rank of Dr. Dre and Eminem. Only releases in English are considered. Both artists should be solo artists. All pairs of names should have the alphabetically smaller one first. Use artist names (asc) as tie breaker.

Hint: Artist aliases may be used everywhere. When doing aggregation, using artist ids will ensure you get the correct results. One example entry in the rank list is 9|Benj Pasek|Justin Paul|27

Q10 (q10_around_the_world):

Concat all dubbed names of The Beatles using comma-separated values(like "Beatles, fab four").

Details: Find all dubbed names of artist "The Beatles" in artist_alias and order them by id (ascending). Print a single string containing all the dubbed names separated by commas.

Hint: You might find [CTEs](#) useful.