CSC105 SQL Practice Problems: Spotify Lab 2

> Generated on 4/25/2023, 11:15:22 PM

Problem information

- Code: `spotify02`
- Name: `Spotify Dataset`
- Description: `Spotify entity information and listening activity dataset, divided into 3 levels for workshop.`
- Template DB: `undefined`
- Generator: `local/spotify01/main`
- Task count: `54`

Tasks

(#0) 001: Show all data from table 'lv1_tracks'

- Description

```
Show all rows and all columns from table 'lv1_tracks' without any condition and let MySQL use
default sorting for the result.
```

- Query:

```
SELECT * FROM lv1_tracks
```

- Tags: `LV1`, `easy`
- Hint

```
- Suggested search query: `mysql query all rows and columns of a table`
- This will query all rows and all columns from a table.
```

(#1) 002: Show first 10 rows of data from table 'lv1_tracks'

- Description

```
Show first 10 rows and all columns from table 'lv1_tracks' without any condition and let MySQL use
default sorting for the result.
```

- Query:

```
SELECT * FROM lv1_tracks LIMIT 10
```

- Tags: `LV1`, `easy`
- Hint

```
- Suggested search query: `mysql query first n rows of a table`
- This will query first 10 rows and all columns from a table.
```

(#2) 003: Show only columns of 'name', 'album' and 'artist' of all data from table 'lv1_tracks'

- Description

```
Show all rows and only columns of 'name', 'album' and 'artist' from table 'lv1_tracks' without any
condition and let MySQL use default sorting for the result.
```

- Query:

```
SELECT name, album, artist FROM lv1_tracks
```

- Tags: `LV1`, `easy`
- Hint

```
- Suggested search query: `mysql query only specific columns of a table`
- This will query all rows and specific columns from a table.
```

(#3) 004: Show only columns of 'name', 'album' and 'artist' of first 10 rows from table 'lv1_tracks'

- Description

```
Show first 10 rows and only columns of 'name', 'album' and 'artist' from table 'lv1_tracks'
without any condition and let MySQL use default sorting for the result.
```

- Query:

```
SELECT name, album, artist FROM lv1_tracks LIMIT 10
```

- Tags: `LV1`, `easy`
- Hint

```
- This will query all rows and specific columns from a table.
```

(#4) 005: Show all data from table 'lv1_tracks' sorted by 'name' in ascending order

- Description

```
Show all rows and all columns from table 'lv1_tracks' without any condition and sort the result by 'name' in ascending order.
```

- Query:

```
SELECT * FROM lv1_tracks ORDER BY name ASC
```

- Tags: `LV1`, `easy`
- Hint

```
- Suggested search query: `mysql query all rows and columns of a table and sort by one column`
- This will query all rows and all columns from a table and sort the result by 'name' in ascending order.
```

(#5) 006: Show all data from table 'lv1_tracks' sorted by 'name' in descending order

- Description

```
Show all rows and all columns from table 'lv1_tracks' without any condition and sort the result by 'name' in descending order.
```

- Query:

```
SELECT * FROM lv1_tracks ORDER BY name DESC
```

- Tags: `LV1`, `easy`
- Hint

```
- Suggested search query: `mysql query all rows and columns of a table and sort by one column`
- This will query all rows and all columns from a table and sort the result by 'name' in descending order.
```

(#6) 007: Show all data from table 'lv1_tracks' sorted by 'name' in ascending order

- Description

```
Show all rows and all columns from table 'lv1_tracks' without any condition and sort the result by
'name' in ascending order.
```

- Query:

```
SELECT * FROM lv1_tracks ORDER BY name ASC
```

- Tags: `LV1`, `easy`
- Hint

```
- Suggested search query: `mysql query all rows and columns of a table and sort by multiple
columns`
- This will query all rows and all columns from a table and sort the result by 'name' in ascending
order.
```

(#7) 008: Show all data from table 'lv1_tracks' sorted by 'name' in descending order

- Description

```
Show all rows and all columns from table 'lv1_tracks' without any condition and sort the result by
'name' in descending order.
```

- Query:

```
SELECT * FROM lv1_tracks ORDER BY name DESC
```

- Tags: `LV1`, `easy`
- Hint

```
- Suggested search query: `mysql query all rows and columns of a table and sort by multiple
columns`
- This will query all rows and all columns from a table and sort the result by 'name' in
descending order.
```

(#8) 009: Show all data from table 'lv1_tracks' of first 10 rows that sorted by 'name' in descending order

- Description

```
Show all rows and all columns from table 'lv1_tracks' without any condition and sort the result by 'name' in descending order.
```

- Query:

```
SELECT * FROM lv1_tracks ORDER BY name DESC LIMIT 10;
```

- Tags: `LV1`, `easy`
- Hint

```
- Suggested search query: `mysql query all rows and columns of a table and sort by multiple columns`
- This will query all rows and all columns of first 10 rows from a table and sort the result by 'name' in descending order.
```

(#9) 010: Show all data from table 'lv1_tracks' that duration is greater than 200000

- Description

```
Show all rows and all columns from table 'lv1_tracks' without any condition and sort the result by 'name' in descending order.
```

- Query:

```
SELECT * FROM lv1_tracks WHERE duration > 200000
```

- Tags: `LV1`, `easy`
- Hint

```
- Suggested search query: `mysql query all rows and columns of a table and sort by multiple columns`
- This will query all rows and all columns that duration is greater than 200000.
```

(#10) 011: Show all data from table 'lv1_tracks' that duration is greater than 200000 and sort by 'name' in ascending order

- Description

```
Show all rows and all columns from table 'lv1_tracks' without any condition and sort the result by 'name' in descending order.
```

- Query:

```
SELECT * FROM lv1_tracks WHERE duration > 200000 ORDER BY name ASC
```

- Tags: `LV1`, `easy`
- Hint

```
- Suggested search query: `mysql query all rows and columns of a table and sort by multiple columns`
- This will query all rows and all columns that duration is greater than 200000 and sort by 'name' in ascending order.
```

(#11) 012: Show all data from table 'lv1_tracks' that duration is greater than 200000 and duration is less than 300000.

- Description

```
Show all rows and all columns from table 'lv1_tracks' that duration is greater than 200000 and duration is less than 300000.
```

- Query:

```
SELECT * FROM lv1_tracks WHERE duration > 200000 AND duration < 300000
```

- Tags: `LV1`, `easy`
- Hint

```
- Suggested search query: `mysql query all rows and columns of a table and sort by multiple columns`
- This will query all rows and all columns that duration is greater than 200000 and duration is less than 300000.
```

(#12) 013: Show all data from table 'lv1_tracks' that duration is greater than 200000 and duration is less than 300000 and sort by 'name' in ascending order.

- Description

```
Show all rows and all columns from table 'lv1_tracks' that duration is greater than 200000 and
duration is less than 300000 and sort by 'name' in ascending order.
```

- Query:

```
SELECT * FROM lv1_tracks WHERE duration > 200000 AND duration < 300000 ORDER BY name ASC
```

- Tags: `LV1`, `easy`
- Hint

```
- Suggested search query: `mysql query all rows and columns of a table and sort by multiple
columns`
- This will query all rows and all columns that duration is greater than 200000 and duration is
less than 300000 and sort by 'name' in ascending order.
```

(#13) 014: Show all data from table 'lv1_tracks' that duration is greater than 280000 and duration is less than 300000 and sort by 'name' in descending order.

- Description

```
Show all rows and all columns from table 'lv1_tracks' that duration is greater than 280000 and
duration is less than 300000 and sort by 'name' in descending order.
```

- Query:

```
SELECT * FROM lv1_tracks WHERE duration > 280000 AND duration < 300000 ORDER BY name DESC
```

- Tags: `LV1`, `easy`
- Hint

```
- Suggested search query: `mysql query all rows and columns of a table and sort by multiple
columns`
- This will query all rows and all columns that duration is greater than 280000 and duration is
less than 300000 and sort by 'name' in descending order.
```

(#14) 015: Show all data from table 'lv1_tracks' that duration is greater than 280000 and duration is less than 300000 and sort by 'name' in descending order and limit to 10 rows.

- Description

```
Show all rows and all columns from table 'lv1_tracks' that duration is greater than 280000 and
duration is less than 300000 and sort by 'name' in descending order and limit to 10 rows.
```

- Query:

```
SELECT * FROM lv1_tracks WHERE duration > 280000 AND duration < 300000 ORDER BY name DESC LIMIT 10
```

- Tags: `LV1`, `easy`
- Hint

```
- Suggested search query: `mysql query all rows and columns of a table and sort by multiple
columns`
- This will query all rows and all columns that duration is greater than 280000 and duration is
less than 300000 and sort by 'name' in descending order and limit to 10 rows.
```

(#15) 016: Show all data from table 'lv1_tracks' that duration is greater than 200000 and duration is less than 350000 and release year is 2020 and sort by 'name' in descending order and limit to 10 rows and offset to 3 rows.

- Description

```
Show all data from table 'lv1_tracks' that duration is greater than 200000 and duration is less
than 350000 and release year is 2020 and sort by 'name' in descending order and limit to 10 rows
and offset to 3 rows.
```

- Query:

```
SELECT * FROM lv1_tracks WHERE duration > 200000 AND duration < 350000 AND year = 2020 ORDER BY
name DESC LIMIT 10 OFFSET 3
```

- Tags: `LV1`, `easy`
- Hint

```
- Suggested search query: `mysql query all rows and columns of a table and sort by multiple columns`
- This will query all rows and all columns that duration is greater than 280000 and duration is less than 300000 and release year is 2000.
```

(#16) 017: Show all data from table 'lv1_tracks' that popularity is greater than or equal 40 and explicit is 1.

- Description

```
Show all rows and all columns from table 'lv1_tracks' that popularity is greater than or equal 40 and explicit is 1
```

- Query:

```
SELECT * FROM lv1_tracks WHERE popularity >= 40 AND explicit = 1
```

- Tags: `LV1`, `easy`
- Hint

```
- Suggested search query: `mysql query all rows and columns of a table and sort by multiple columns`
- This will query all rows and all columns that popularity is greater than or equal 40 and explicit is 1.
```

(#17) 019: Show all tracks that track name start with 'A'

- Description

```
Show all tracks that track name start with 'A' and let the result be sorted by default order.
```

- Query:

```
SELECT * FROM lv1_tracks WHERE name LIKE 'A%'
```

- Tags: `LV1`, `easy`
- Hint

```
- Suggested search query: `mysql where start with`, `mysql like condition`
```

(#18) 020: Show all tracks that track name start with 'A' or 'J'

- Description

```
Show all tracks that track name start with 'A' or 'J' and let the result be sorted by default order.
```

- Query:

```
SELECT * FROM lv1_tracks WHERE name LIKE 'A%' OR name LIKE 'J%'
```

- Tags: `LV1`, `easy`
- Hint

```
- Use like condition (as the same with last task) for both 'A' and 'J' condition with OR operator
```

(#19) 021: Show all tracks that track name contains 'A' in the track name

- Description

```
Show all tracks that track name contains 'A' in the track name and let the result be sorted by default order.
```

- Query:

```
SELECT * FROM lv1_tracks WHERE name LIKE '%A%'
```

- Tags: `LV1`, `easy`
- Hint

```
- Suggested search query: `mysql where contains substring`, `mysql like condition`
```

(#20) 022: Query all rows with only 2 column, track name and track name in lowercase

- Description

```
Let's query using function LOWER() to make data be lowercase. For example, `SELECT name, LOWER(name) FROM lv1_tracks`
```

- Query:

```
SELECT name, LOWER(name) FROM lv1_tracks
```

- Tags: `LV1`, `mid`
- Hint

```
- Suggested search query: `mysql lowercase`, `mysql string manipulation`
```

(#21) 023: Show all tracks that track name contains 'A' in the track name with case insensitive (contains 'a' or 'A')

- Description

```
Try using function LOWER() to make data be lowercase. So character 'A' and 'a' will be treated as
the same and we can use LIKE condition to query data.
```

- Query:

```
SELECT * FROM lv1_tracks WHERE LOWER(name) LIKE '%a%'
```

- Tags: `LV1`, `mid`
- Hint

```
- Suggested search query: `mysql lowercase`, `mysql where case insensitive`
```

(#22) 024: Show data that contains 'THE' (case-insensitive) in the track name or album name or artist name

- Description

```
Try using function LOWER() to make data be lowercase. Then, use LIKE condition to filter data and
use OR operator to combine multiple conditions.
```

- Query:

```
SELECT * FROM lv1_tracks WHERE LOWER(name) LIKE '%the%' OR LOWER(album) LIKE '%the%' OR
LOWER(artist) LIKE '%the%'
```

- Tags: `LV1`, `mid`
- Hint

```
- Suggested search query: `mysql lowercase`, `mysql where case insensitive`
```

(#23) 025: Show data of first 10 rows from table 'lv1_tracks' with a column named 'track_name' instead of name and a column named 'track_duration' instead of duration.

- Description

```
Show name of first 10 rows from table 'lv1_tracks' with a column named 'track_name' instead of
name and a column named 'track_duration' instead of duration.
```

- Query:

```
SELECT name AS track_name, duration AS track_duration FROM lv1_tracks LIMIT 10
```

- Tags: `LV1`, `mid`
- Hint

```
- Use a comma to separate multiple columns. For example: `SELECT column1 AS another_name, column2
AS just_another_name FROM lv1_tracks LIMIT 10`
```

(#24) 026: Show all columns from table 'lv1_tracks' but change only column named 'name' to 'track_name'.

- Description

```
Show all columns from table 'lv1_tracks' but change only column named 'name' to 'track_name'.
```

- Query:

```
SELECT name AS track_name, lv1_tracks.* FROM lv1_tracks
```

- Tags: `LV1`, `mid`
- Hint

```
- Use a comma to separate multiple columns. For example: `SELECT column1 AS another_name, column2,
column3, column4 FROM lv1_tracks LIMIT 10`.
- Use `*` to select all columns. For example: `SELECT * FROM lv1_tracks`
```

(#25) 027: Find out the year of the oldest song.

- Description

```
Find out the year of the oldest song.
```

- Query:

```
SELECT MIN(year) FROM lv1_tracks;
```

- Tags: `LV1`, `mid`
- Hint

```
MySQL MIN() and MAX() functions
```

(#26) 028: Find out 2 value of the oldest year and the newest year.

- Description

```
Find out 2 value of the oldest year and the newest year.
```

- Query:

```
SELECT MIN(year), MAX(year) FROM lv1_tracks;
```

- Tags: `LV1`, `mid`
- Hint

```
- Use a comma to separate multiple columns. For example: `SELECT column1, column2, column3,
column4 FROM lv1_tracks LIMIT 10`.
```

(#27) 029: Find out the oldest year and the newest year of tracks with custom column name.

- Description

```
Find out the oldest year and the newest year of tracks but change output column name of MIN(year)
to 'oldest_year' and change output column name of MAX(year) to 'newest_year'.
```

- Query:

```
SELECT MIN(year) AS oldest_year, MAX(year) AS newest_year FROM lv1_tracks;
```

- Tags: `LV1`, `mid`
- Hint

```
- Use a comma to separate multiple columns. For example: `SELECT column1, column2, column3, column4 FROM lv1_tracks LIMIT 10`.
- Use `AS` to change output column name. For example: `SELECT column1 AS another_name, column2 AS just_another_name FROM lv1_tracks LIMIT 10`.
```

(#28) 030: Count all data from table 'lv1_tracks' without any condition.

- Description

```
Count all rows and all columns from table 'lv1_tracks' without any condition.
```

- Query:

```
SELECT COUNT(*) FROM lv1_tracks
```

- Tags: `LV1`, `mid`
- Hint

```
- Suggested search query: `mysql count all rows and columns of a table to a number`
- This will count all rows and all columns from table 'lv1_tracks' without any condition.
```

(#29) 031: Count all data from table 'lv1_tracks' and change the output column name to 'total_tracks'.

- Description

```
Count all rows and all columns from table 'lv1_tracks' and change the output column name to 'total_tracks'.
```

- Query:

```
SELECT COUNT(*) AS total_tracks FROM lv1_tracks
```

- Tags: `LV1`, `mid`
- Hint

```
- Suggested search query: `mysql count all rows and columns of a table to a number`
- This will count all rows and all columns from table 'lv1_tracks' and change the output column name to 'total_tracks'.
```

(#30) 032: Count all data from table 'lv1_tracks' released in each year.

- Description

```
Count all rows and all columns from table 'lv1_tracks' that released in each year into 2 columns named 'year' and 'total_tracks'.
```

- Query:

```
SELECT COUNT(*) FROM lv1_tracks GROUP BY year
```

- Tags: `LV1`, `mid`
- Hint

```
- Suggested search query: `mysql group by`, `mysql count data by each group`, `mysql count data that have same value in a column`
```

(#31) 033: Count all data from table 'lv1_tracks' released in each year and change the output column name to 'total_tracks'.

- Description

```
Count all rows and all columns from table 'lv1_tracks' that released in each year into 2 columns named 'year' and 'total_tracks'.
```

- Query:

```
SELECT COUNT(*) AS total_tracks FROM lv1_tracks GROUP BY year
```

- Tags: `LV1`, `mid`
- Hint

```
- Use `AS` to change the output column name. For example: `SELECT COUNT(*) AS total_tracks FROM lv1_tracks GROUP BY year`
```

(#32) 034: Sum the total duration of all tracks in table 'lv1_tracks' released in each year.

- Description

```
Sum the total duration of all tracks in table 'lv1_tracks' released in each year into 2 columns named 'year' and 'total_duration'.
```

- Query:

```
SELECT SUM(duration) AS total_duration FROM lv1_tracks GROUP BY year
```

- Tags: `LV1`, `mid`
- Hint

```
- Suggested search query: `mysql sum and group by`, `mysql sum data by each group`, `mysql sum function`
```

(#33) 035: Sort the year that have the most number of tracks in table 'lv1_tracks' from highest to lowest.

- Description

```
Sort the year that have the most number of tracks in table 'lv1_tracks' from highest to lowest. If there are 2 years that have the same number of tracks, sort them by year from lowest to highest.
```

- Query:

```
SELECT year, COUNT(*) AS total_tracks FROM lv1_tracks GROUP BY year ORDER BY total_tracks DESC, year ASC
```

- Tags: `LV1`, `mid`
- Hint

```
Let's try this syntax: SELECT year, COUNT(*) AS total_tracks FROM lv1_tracks GROUP BY year ORDER BY total_tracks DESC
```

(#34) 036: Concat name and artist name of first 10 rows from table 'lv1_tracks' into a column named 'display_name'.

- Description

```
Concat name and artist name of first 10 rows from table 'lv1_tracks' into a column named
'display_name'.
```

- Query:

```
SELECT CONCAT(name, ' - ', artist) AS display_name FROM lv1_tracks LIMIT 10
```

- Tags: `LV1`, `mid`
- Hint

```
- Suggested search query: `mysql concat`, `mysql concat multiple columns`
- This will concat name and artist name of first 10 rows from table 'lv1_tracks' into a column
named 'display_name'.
```

(#35) 037: Sort data from both column, 'total_tracks' and 'total_duration' from highest to lowest.

- Description

```
Show all data from table 'lv1_tracks' by sort all columns from table 'lv1_tracks' by release year
from highest to lowest,
if year is the same, sort by duration from shortest to longest.
```

- Query:

```
SELECT year, COUNT(*) AS total_tracks, SUM(duration) AS total_duration FROM lv1_tracks GROUP BY
year ORDER BY total_tracks DESC, total_duration ASC
```

- Tags: `LV1`, `mid`
- Hint

```
- Suggested search query: `mysql sort multiple columns`
```

(#36) 038: Display the data information into the given format 1.

- Description

```
Display the data information into the given format (try query anything for system to show the expected output).
```

- Query:

```
SELECT artist, COUNT(*) AS total_tracks, SUM(duration) AS total_duration FROM lv1_tracks GROUP BY artist ORDER BY total_tracks
```

- Tags: `LV1`, `high`
- Hint

```
Cimorelli - Believe In You (https://www.youtube.com/watch?v=k7w9LZQlxS4)
```

(#37) 039: Display the data information into the given format 2.

- Description

```
Display the data information into the given format (try query anything for system to show the expected output).
```

- Query:

```
SELECT artist, album, COUNT(*) AS total_tracks, SUM(duration) AS total_duration FROM lv1_tracks GROUP BY artist, album ORDER BY total_tracks
```

- Tags: `LV1`, `high`
- Hint

```
Try Try Try (https://www.youtube.com/watch?v=TDP5ktmQFto), keep fighting!
```

(#38) 040: Display the data information into the given format 3.

- Description

```
Display the data information into the given format (try query anything for system to show the expected output).
```

- Query:

```
SELECT album, COUNT(*) AS total_tracks, AVG(duration) AS average_duration FROM lv1_tracks GROUP BY album ORDER BY average_duration
```

- Tags: `LV1`, `high`
- Hint

```
Try Try Try (https://www.youtube.com/watch?v=TDP5ktmQFto), keep fighting!
```

(#39) 041: Display the data information into the given format 4.

- Description

```
Display the data information into the given format (try query anything for system to show the expected output) with explicit track percentage in each album.
```

- Query:

```
SELECT album, (SELECT COUNT(*) FROM lv1_tracks t1 WHERE t1.album = lv1_tracks.album AND t1.explicit = 1) AS explicit_tracks_count, (SELECT COUNT(*) FROM lv1_tracks t2 WHERE t2.album = lv1_tracks.album_name) AS total_tracks, (SELECT COUNT(*) FROM lv1_tracks t3 WHERE t3.album = lv1_tracks.album AND t3.explicit = 1) / (SELECT COUNT(*) FROM lv1_tracks t4 WHERE t4.album = lv1_tracks.album) AS explicit_percentage FROM lv1_tracks GROUP BY album ORDER BY explicit_percentage
```

- Tags: `LV1`, `high`
- Hint

```
- Suggested search query: `mysql subquery`, `mysql subquery in select`
```

(#40) 042: Display the data information into the given format 5.

- Description

```
Select artist that have most number of released tracks in each year with limit only 1 artist per year.
```

- Query:

```
SELECT year, (SELECT artist FROM lv1_tracks WHERE year = lv1_tracks.year GROUP BY artist ORDER BY COUNT(*) DESC LIMIT 1) AS top_artist_name_by_track_count FROM lv1_tracks GROUP BY year ORDER BY 'year'
```

- Tags: `LV1`, `high`
- Hint

```
- Suggested search query: `mysql subquery`
```

(#41) 101: Show all rows of data from table 'lv2_tracks'

- Description

```
Show all rows and all columns from table 'lv2_tracks' without any condition and let MySQL use
default sorting for the result.
```

- Query:

```
SELECT * FROM lv2_tracks;
```

- Tags: `LV1`, `easy`
- Hint

```
Just use basic query statement to query from table 'lv2_tracks'.
```

(#42) 102: Show first row of data from table 'lv2_tracks'

- Description

```
Show first row and all columns from table 'lv2_tracks' without any condition and let MySQL use
default sorting for the result.
```

- Query:

```
SELECT * FROM lv2_tracks LIMIT 1;
```

- Tags: `LV2`, `easy`
- Hint

```
- Suggested query: `mysql query first n rows of a table`
- This will query first 10 rows and all columns from a table.
```

(#43) 103: Show only 'album_id' column from the first row of data from table 'lv2_tracks'

- Description

```
Show only 'album_id' column from the first row of data from table 'lv2_tracks' without any
condition and let MySQL use default sorting for the result.
```

- Query:

```
SELECT album_id FROM lv2_tracks LIMIT 1;
```

- Tags: `LV2`, `easy`
- Hint

```
- Suggested query: `mysql query first n rows of a table`
- Use `SELECT` with specific column name instead of '*' statement to query only specific columns.
```

(#44) 104: Show album information from the track previously queried.

- Description

```
From the last task, you should have a query result with album_id of the first track in table
'lv2_tracks'.
Use that album_id to query album with that specific id from table 'lv2_albums'.
```

- Query:

```
SELECT * FROM lv2_albums WHERE lv2_albums.id = (SELECT album_id FROM lv2_tracks LIMIT 1);
```

- Tags: `LV2`, `easy`
- Hint

```
- Use basic query statement to query from table 'lv2_albums' with `WHERE id = ...` condition.
```

(#45) 105: Show all track information with linked album information.

- Description

```
Use MySQL JOIN statement to query the track information with linked album information by using ON lv2_tracks.album_id = lv2_albums.id to join the two tables.
```

- Query:

```
SELECT * FROM lv2_tracks WHERE album_id = (SELECT album_id FROM lv2_tracks LIMIT 1);
```

- Tags: `LV2`, `easy`
- Hint

```
Let's try `SELECT * FROM lv2_tracks WHERE album_id = (SELECT album_id FROM lv2_tracks LIMIT 1);`
```

(#46) 106: Show track name and album name of all data.

- Description

```
Use MySQL JOIN statement to query the track name and album name of all data by using ON lv2_tracks.album_id = lv2_albums.id to join the two tables.
```

- Query:

```
SELECT lv2_tracks.name, lv2_albums.name FROM lv2_tracks JOIN lv2_albums ON lv2_tracks.album_id = lv2_albums.id
```

- Tags: `LV2`, `easy`
- Hint

```
- Suggested query: `mysql query only specific columns of a table`
- Join table track and album
```

(#47) 107: Show track name and album name of all data with track name starting with 'A'.

- Description

```
Use MySQL JOIN statement to query the track name and album name of all data with track name
starting with 'A' by using ON lv2_tracks.album_id = lv2_albums.id to join the two tables.
And mix it with WHERE statement to filter the result.
```

- Query:

```
SELECT lv2_tracks.name, lv2_albums.name FROM lv2_tracks JOIN lv2_albums ON lv2_tracks.album_id =
lv2_albums.id WHERE lv2_tracks.name LIKE 'A%'
```

- Tags: `LV2`, `easy`
- Hint

```
- Suggested query: `mysql query only specific columns of a table`
- Join table track and album
- Use `WHERE` statement to filter the result.
```

(#48) 108: Show track name, album name and artist name of all data

- Description

```
Use MySQL JOIN statement to query the track name, album name and artist name of all data by using
ON lv2_tracks.album_id = lv2_albums.id to join the track and album table.
And then use another JOIN statement to join the artist table with the previous joined table by
using ON lv2_albums.artist_id = lv2_artists.id to join the two tables.
```

- Query:

```
SELECT lv2_tracks.name, lv2_albums.name, lv2_artists.name FROM lv2_tracks JOIN lv2_albums ON
lv2_tracks.album_id = lv2_albums.id JOIN lv2_artists ON lv2_albums.artist_id = lv2_artists.id
```

- Tags: `LV2`, `easy`
- Hint

```
Let's try `SELECT lv2_tracks.name, lv2_albums.name, lv2_artists.name FROM lv2_tracks JOIN
lv2_albums ON lv2_tracks.album_id = lv2_albums.id JOIN lv2_artists ON lv2_albums.artist_id =
lv2_artists.id`
```

(#49) 109: Query track and album information using base table 'lv2_albums'

- Description

```
Use the same JOIN statement as the previous task to query, but this time use the base table
'lv2_albums' to query the track and album information.
```

- Query:

```
SELECT * FROM lv2_albums JOIN lv2_tracks ON lv2_tracks.album_id = lv2_albums.id
```

- Tags: `LV2`, `easy`
- Hint

```
Let's try `SELECT * FROM lv2_albums JOIN lv2_tracks ON lv2_tracks.album_id = lv2_albums.id`
```

(#50) 110: Concat track name and album name into one column

- Description

```
Use MySQL CONCAT function to concat track name and album name into one column.
```

- Query:

```
SELECT CONCAT(lv2_tracks.name, ' - ', lv2_albums.name) FROM lv2_tracks JOIN lv2_albums ON
lv2_tracks.album_id = lv2_albums.id
```

- Tags: `LV2`, `easy`
- Hint

```
- Use CONCAT function to concat two columns form the joined result into one column.
- Suggested query: `mysql concat two columns`
- Let's try `SELECT CONCAT(lv2_tracks.name, ' - ', lv2_albums.name) FROM lv2_tracks JOIN
lv2_albums ON lv2_tracks.album_id = lv2_albums.id`
```

(#51) 111: Show all catcatinated track name in one column and album name in another column

- Description

```
Use MySQL CONCAT function to concat track name of each album into one column.
And then use AS statement to rename the column name to `tracks`.
```

- Query:

```
SELECT CONCAT(lv2_tracks.name) AS tracks FROM lv2_albums JOIN lv2_tracks ON lv2_tracks.album_id =
lv2_albums.id
```

- Tags: `LV2`, `easy`
- Hint

```
- Let's try `SELECT CONCAT(lv2_tracks.name) AS tracks FROM lv2_albums JOIN lv2_tracks ON
lv2_tracks.album_id = lv2_albums.id`
```

(#52) 112: Show sum of track duration of each album

- Description

```
Use MySQL SUM function to sum the duration of each album.
```

- Query:

```
SELECT SUM(lv2_tracks.duration) FROM lv2_albums JOIN lv2_tracks ON lv2_tracks.album_id =
lv2_albums.id
```

- Tags: `LV2`, `easy`
- Hint

```
- Let's try `SELECT SUM(lv2_tracks.duration) FROM lv2_albums JOIN lv2_tracks ON
lv2_tracks.album_id = lv2_albums.id`
```

(#53) 113: Show sum of track duration of each artist and order by the sum of duration

- Description

```
Use MySQL SUM function to sum the duration of each artist joined from artist to album to track.
And then use ORDER BY statement to order the result by the sum of duration.
```

- Query:

```
SELECT SUM(lv2_tracks.duration) FROM lv2_artists JOIN lv2_albums ON lv2_albums.artist_id =
lv2_artists.id JOIN lv2_tracks ON lv2_tracks.album_id = lv2_albums.id ORDER BY
SUM(lv2_tracks.duration)
```

- Tags: `LV2`, `easy`
- Hint

```
- Let's try `SELECT SUM(lv2_tracks.duration) FROM lv2_artists JOIN lv2_albums ON
lv2_albums.artist_id = lv2_artists.id JOIN lv2_tracks ON lv2_tracks.album_id = lv2_albums.id ORDER
BY SUM(lv2_tracks.duration)`
```