CS4416: Database Systems

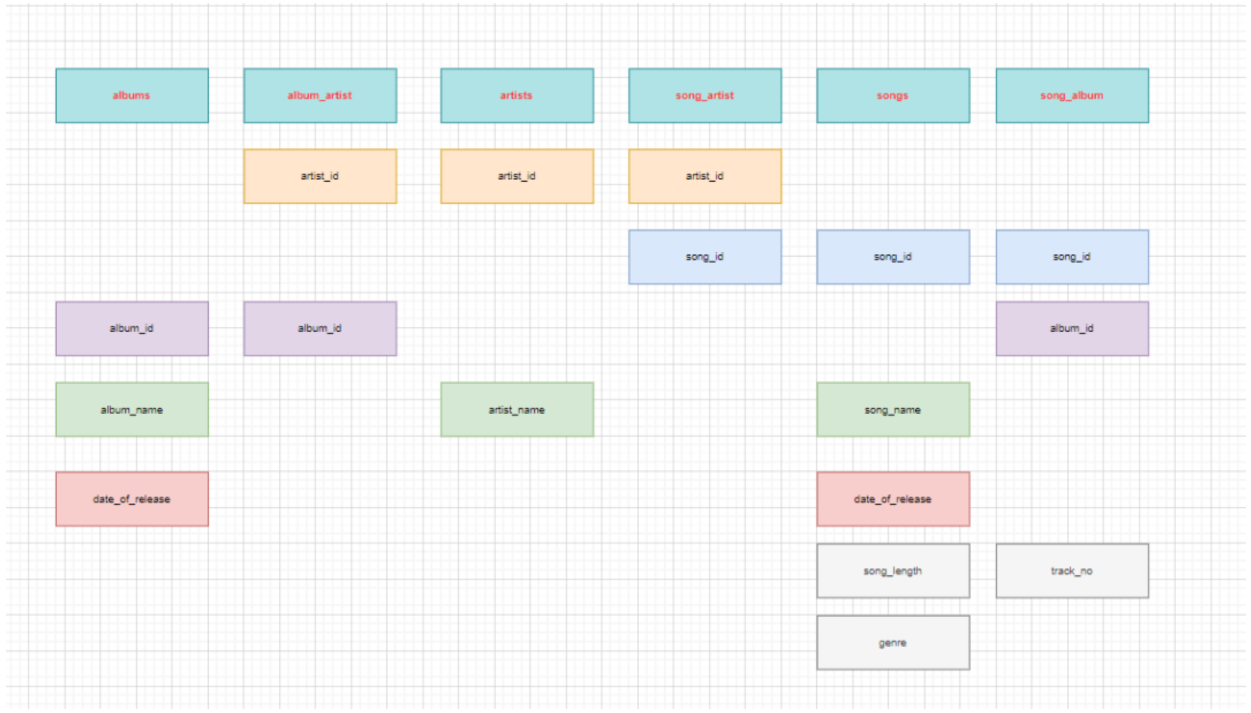
A project report
By:

- Olan Healy Student ID: 21318204
- Kevin Collins Student ID: 21344256
- ALEKSANTERI LEO HENRIK.SEPPÄ Student ID: 22216812
- Sean Capilis Student ID: 21342342

The Beginnings

Firstly, we needed a platform to test our code and setups database. We used XAMPP for windows 11 and ran phpmyadmin off it.
We followed this tutorial XAMPP setup to set up XAMPP. We then looked at the album_collection_schema.sql to make up our own database so we could test our code off it. We Inserted the schema into phpmyadmin so as the tables could be created and then we came up with album_collection_sort.sql to insert values into it. The link to our repo with this code can be found here album_collection_sort.sql. We split the questions and database into different sections so the project was split equally amongst everyone. For Q1 and Q2 we just tested them straight off the database without creating a view to see if we could get an output that was similar to what the question was asking for. As Q3 was a trigger, it was clear in the question that it would take place AFTER the insert on song_album.Q4 and Q5 were functions that needed to be made.

Q6 was an ERD diagram which just visually shows the relationships between the tables. We created an image on draw.io also which showed which attributes of the tables are used in other tables and which are just by themselves (this was just to help gather the answers to the questions)



Answers to questions:

Question1:

```
CREATE OR REPLACE VIEW Exceptions AS
SELECT artist_name, album_name
FROM artists, albums albumview
WHERE artist_id NOT IN (SELECT artist_id
FROM album_artist
WHERE album_id = albumview.album_id)
AND artist_id IN (SELECT artist_id
FROM song_artist WHERE song_id IN
(SELECT song_id FROM song_album WHERE album_id =
albumview.album_id));
```

Question2:

```
CREATE VIEW album_len AS
SELECT album_name , ROUND(SUM( songs.song_length), 2) AS
album_len

FROM song_album
JOIN songs USING (song_id)
JOIN albums USING (album_id)

GROUP BY album_name;

CREATE OR REPLACE VIEW albuminfo AS
SELECT albums.album_name, GROUP_CONCAT(DISTINCT artist_name)
AS list_of_artist, albums.date_of_release,
album_len.album_len AS total_length

FROM song_album
JOIN songs USING (song_id)
JOIN albums USING (album_id)
JOIN album_artist USING (album_id)
JOIN artists USING (artist_id)
JOIN album_len USING (album_name)
GROUP BY album_id;
```

Question3:

```
DELIMITER //
CREATE TRIGGER CheckReleaseDate
AFTER INSERT ON song_album
FOR EACH ROW BEGIN
DECLARE song_date DATE;
DECLARE album_date DATE;
SET song_date = (SELECT date_of_release FROM songs WHERE
song_id = NEW.song_id);
SET album_date = (SELECT date_of_release FROM albums WHERE
album_id = NEW.album_id);
```

```
IF (song_date > album_date)
THEN UPDATE songs SET date_of_release = album_date
WHERE song_id = NEW.song_id;
END IF;
END //
```

Question4:

```
DELIMITER //
CREATE PROCEDURE AddTrack(IN A INT, IN S INT)
    BEGIN
        DECLARE TN INT;
        SET TN = (SELECT MAX(track_no) FROM song_album WHERE
album_id = A);

        IF (SELECT COUNT(*) FROM albums WHERE album_id = A)
= 1
        AND (SELECT COUNT(*) FROM songs WHERE song_id = S) =
1
        THEN
            INSERT INTO song_album (song_id, album_id,
track_no) VALUES (S, A, TN + 1);
        END IF;
    END;
    //
```

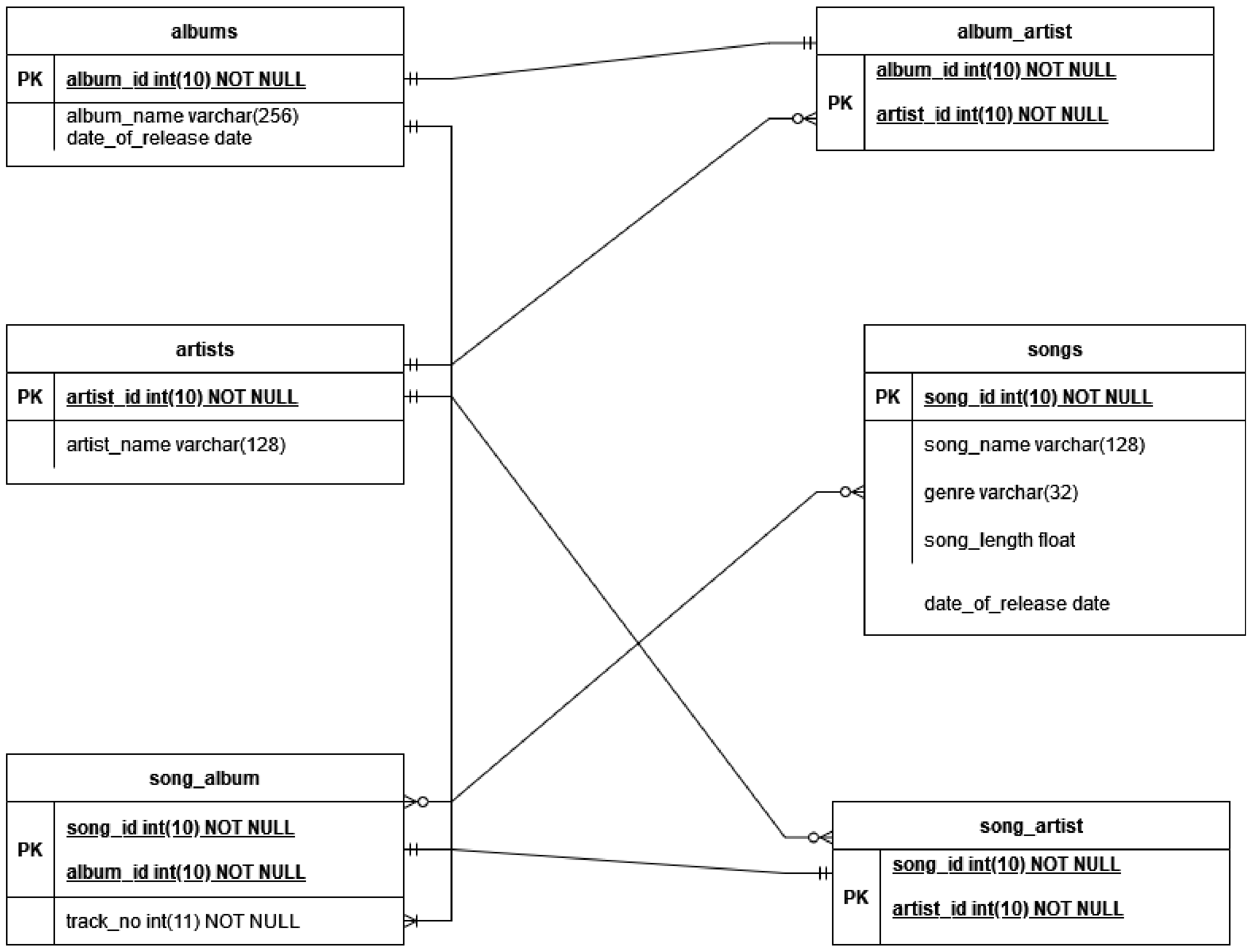
Question5:

```
DELIMITER //
CREATE FUNCTION GetTrackList(A INT(10))
RETURNS TEXT
DETERMINISTIC

BEGIN
    DECLARE B BLOB;
    SELECT GROUP_CONCAT(song_name ORDER BY track_no DESC
SEPARATOR ',') INTO B
    FROM song_album LEFT JOIN songs USING (song_id)
    WHERE album_id = A;

    RETURN B;
END;
//
```

Question6:

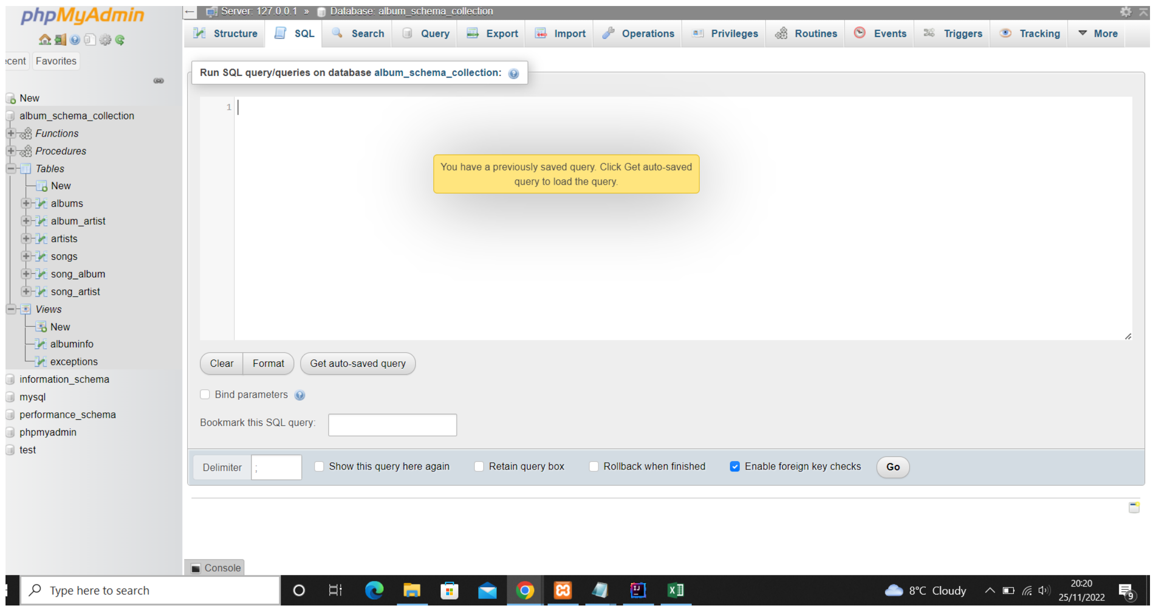


Assumptions:
- An album can have only 1 artist
- Album can have multiple songs
- A song has one artist
- an artist can have multiple songs

Instructions on how to run:

1. Download XAMPP and run phpmyadmin

Create a database called album_schema_collection and either import or paste album_collection_schema.sql in the SQL query section(follow image below). (If the link doesn't work, log on to sulis, go to module CS4416, press module material, press project specification and album_schema_collection will be present. You can also find this on our github page (github)).



Run SQL query/queries on database album_schema_collection:

You have a previously saved query. Click Get auto-saved query to load the query.

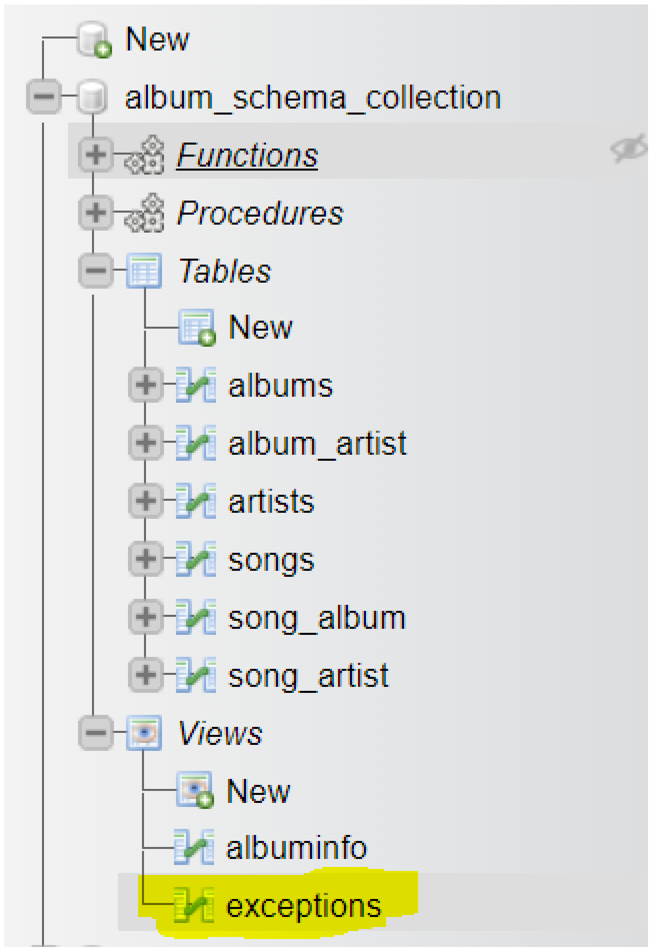
Delimiter ;

[ ] Show this query here again [ ] Retain query box [ ] Rollback when finished [✓] Enable foreign key checks

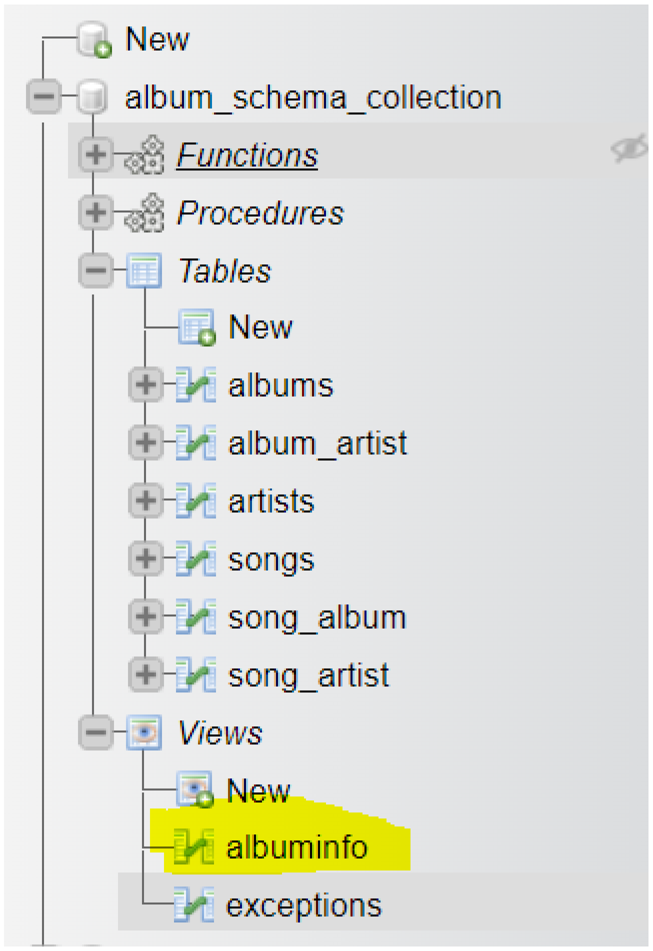
2. Import or run album_collection_code.sql (again can be found on our github and is also included in the assignment submission)
3. It is vital that album_collection_code.sql is ran before the values are inserted into the database as the trigger will not work otherwise.
4. Now, you can insert values into the database. We made one and again is on our github as album_collection_sort.sql but as the project deadline came closer, a project_test_data.sql came out along with a pdf so we could test our code to make sure the output was as expected. (Again, if the link doesn't work, this can be found on sulis under announcements in CS4416 as "Test for your project: IMPORTANT".
5. Taking that you use this testing code file, I will highlight where you can find the answers in the database below:

Where to find answers:

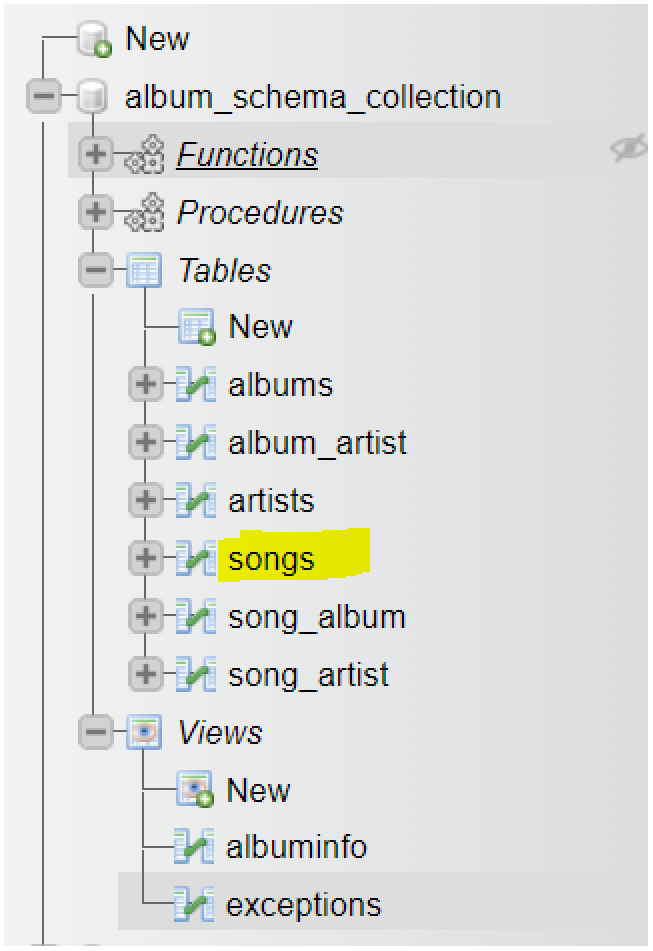
- Question1: Click on the exceptions view



- Question2: Click on albuminfo view



- Question3: You will have to click into songs and look for song : 1. song_id =21, song_name = LIGHT : 2. song_id = 22, song_name = Free



- Question 4: In the sql query section, run the following query

  CALL AddTrack(4, 18);

- Question 5: In the sql query section, run the following query

  SELECT GetTrackList(4);

  NOTE: You can change numbers for question 4 and 5 as you wish once its valid according to the data.

- Question6: This can be found in the submission file as album_collection_erd.pdf (can also be found on our github)