CS6401 – SQL Project

Album collection

Course: Master's in Business Analytics

Module: Database in Practice (CS6401)

Module Leader: Dr Nikola S. Nikolov

Table of Contents

Introduction

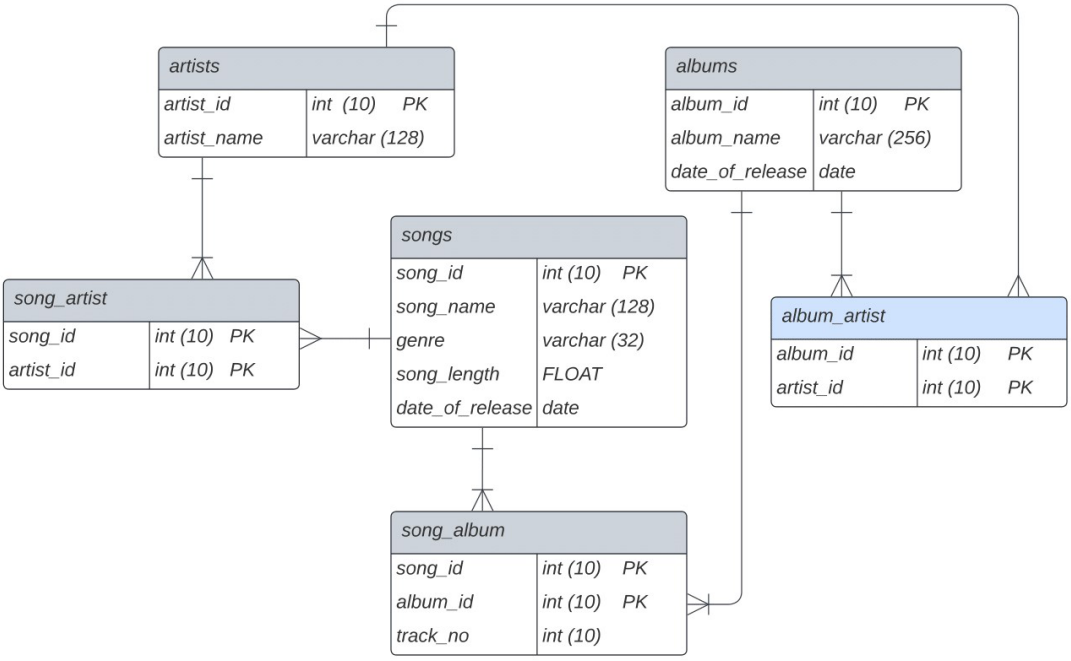
SQL is a crucial component in managing and analyzing data from relational databases. As a part of our semester-1 coursework, we had a module titled "Database in Practice," where we worked on an SQL project by considering the relational database schema “album_collection_schema” provided by the module leader. The project involved utilizing advanced SQL concepts to cover various topics, such as creating customized views to display data, implementing triggers to automate database actions, and designing stored procedures and functions to facilitate interaction with databases. We have shared code snippets & outputs of the project here. Also, we utilized MySQL Workbench for this project and are thrilled to share our work with you, hoping that you find it informative and engaging.

Note – DML statements typically provide output data, but it can be challenging to display output from DDL statements as they do not return data. We have attempted to include output snippets of relevant codes (post-creation) to demonstrate the possible outcomes.

Members of the Group:-

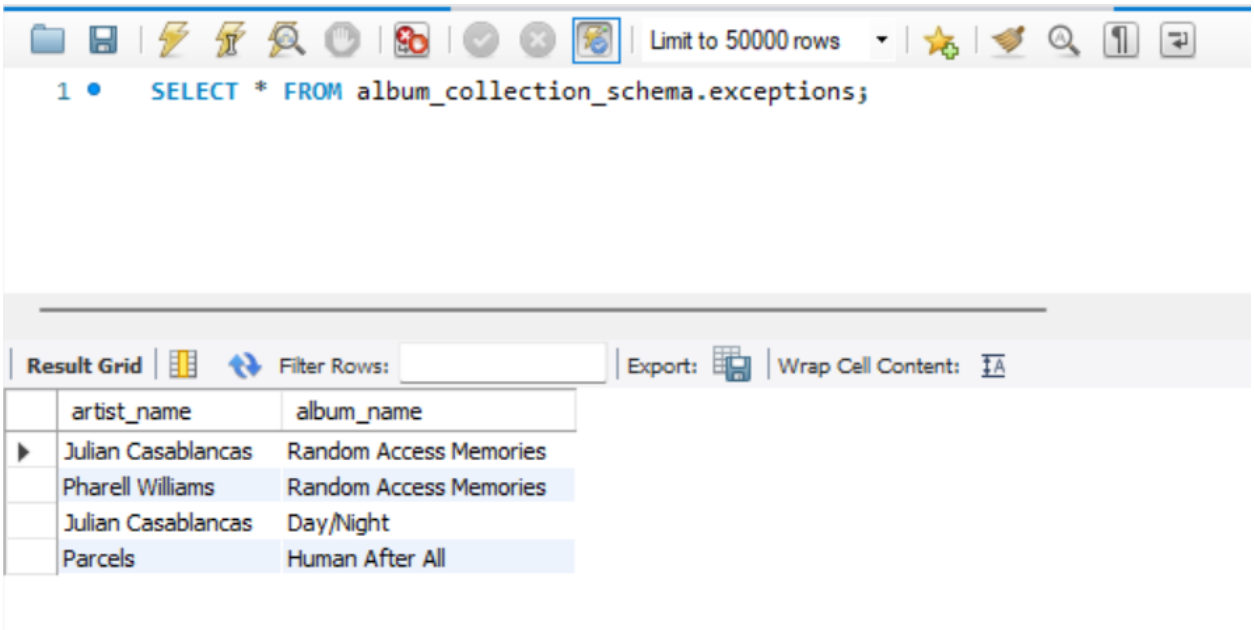
Student Names	Student ID
1) Dheeraj Shrikrishna Kulkarni	22086919
2) Rakshith Gowda T N	22118012
3) Raj Mahesh Taksale	22037071
4) Swapna Sudha Sabut	22152881

ER Diagram of ‘album_collection_schema’:



1) Creation of VIEW named ‘Exceptions’

```sql
/* 1) Create view Exceptions(artist_name, album_name).
(A, B) is a data row in this view if and only if artist A contributes to at least one song
on album B (according to table song_artist) but artist A is not listed as one of the artists
on album B in table album_artist. There should be no duplicate data rows in the view. */

CREATE VIEW Exceptions AS
SELECT DISTINCT ar.artist_name, al.album_name
FROM song_artist sar
JOIN artists ar
ON sar.artist_id = ar.artist_id
JOIN song_album sal
ON sal.song_id = sar.song_id
JOIN albums al
ON al.album_id = sal.album_id
LEFT JOIN album_artist alar
ON alar.album_id = sal.album_id AND alar.artist_id = sar.artist_id
WHERE alar.album_id IS NULL
```



2) Creation of VIEW named 'AlbumInfo'

```
/* 2) Create view AlbumInfo(album_name, list_of_artist, date_of_release, total_length).
Each album should be listed exactly once. For each album, the value in column list_of_artists
is a comma-separated list of all artists on the album according to table album_artist.
The value in column total_length is the total length of the album in minutes. */


CREATE VIEW AlbumInfo AS
SELECT al.album_name,
(   SELECT GROUP_CONCAT(ar.artist_name)
    FROM artists ar
    JOIN album_artist alar ON ar.artist_id = alar.artist_id
    WHERE al.album_id = alar.album_id) AS list_of_artist ,al.date_of_release,
    (SELECT ROUND(SUM(s.song_length),2)
    FROM songs s
    JOIN song_album sal ON s.song_id = sal.song_id
    WHERE sal.album_id = al.album_id
) total_length
FROM albums al
```

Limit to 50000 rows

```
SELECT * FROM album_collection_schema.albuminfo;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

album_name	list_of_artist	date_of_release	total_length
Random Access Memories	Daft Punk,Parcels	2013-05-17	80.16
Human After All	Daft Punk	2005-03-14	24.8
Phrazes for the Young	Julian Casablancas	2009-11-02	13.8
Day/Night	Parcels	2021-11-05	45.24

3) Creation of TRIGGER named ‘CheckReleaseDate’

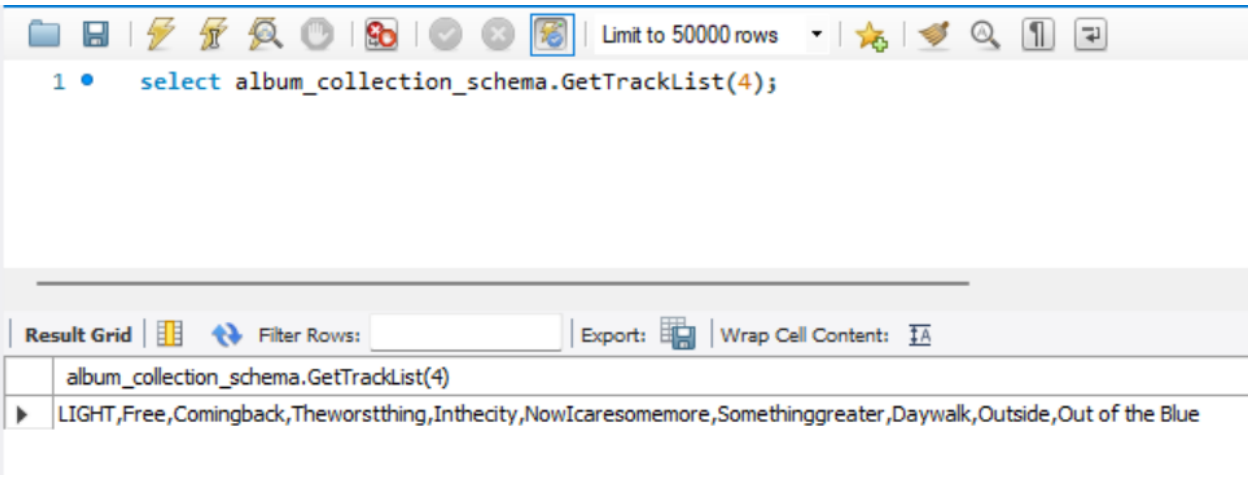
```
/* 3). Write trigger CheckReleaseDate that does the following. Assume a new row (S, A, TN) is inserted
into table song_album with song_id S, album_id A and track_no TN. Check if the release date of song S
is later than the release date of album A. If this is the case, then change the release date of song S
in table songs to be the same as the release date of album A. */

DELIMITER //
CREATE TRIGGER CheckReleaseDate
AFTER INSERT
ON song_album
FOR EACH ROW
BEGIN
    UPDATE songs,
    (SELECT song_id, album_release_date
    FROM
    (SELECT a.album_id AS album_id,
    als.song_id AS song_id,
    a.date_of_release AS album_release_date,
    als.date_of_release AS song_release_date
    FROM albums a
    JOIN
    (SELECT DISTINCT sa.album_id, s.song_id, s.date_of_release
    FROM songs s INNER JOIN song_album sa ON s.song_id = sa.song_id
    GROUP BY sa.album_id, s.song_id, s.date_of_release
    ORDER BY song_id) als
    ON a.album_id = als.album_id AND als.date_of_release > a.date_of_release
    GROUP BY a.album_id, als.song_id, a.date_of_release, als.date_of_release
    )albson) AS alb
    SET songs.date_of_release = alb.album_release_date
    WHERE songs.song_id = alb.song_id;
END //
DELIMITER ;
```

4) Creation of STORED PROCEDURE named ‘AddTrack’

```
/* 4) Write stored procedure AddTrack(A, S) where A is an album_id and S is a songs_id. The procedure
should check if A is an album_id already existing in table albums and S is a song_id already existing
in table songs. If both conditions are satisfied then the procedure should insert data row (A, S, TN+1)
into table song_album where TN is the highest track_no for album A in table song_album before inserting
the row. */

DELIMITER //
CREATE PROCEDURE AddTrack(album_id INT(10), song_id INT(10))
BEGIN

IF (SELECT EXISTS(SELECT albums.album_id FROM albums,song_album WHERE song_album.album_id =
albums.album_id))
    AND
   (SELECT EXISTS(SELECT songs.song_id FROM songs,song_album WHERE song_album.song_id = songs.song_id))
THEN
    INSERT INTO song_album(song_id, album_id, track_no)
        SELECT song_id, album_id,(SELECT Max(track_no) + 1
        FROM song_album);
END IF;
END //
DELIMITER ;
```

5) Creation of STORED FUNCTION named ‘GetTrackList’

```sql
/* 5) Write stored function GetTrackList(A) which, for a given album_id A, returns a comma-separated
list of the names of all songs on the album ordered according to their track_no. */

DELIMITER $$
CREATE FUNCTION GetTrackList(a_id INT)
RETURNS VARCHAR(250) DETERMINISTIC
BEGIN
DECLARE List_of_tracks VARCHAR(250) DEFAULT"";
SELECT GROUP_CONCAT(song_name) INTO List_of_tracks
FROM
(SELECT sa.album_id AS album_id,
 sa.song_id AS song_id,
 sa.track_no AS track_no,
 s.song_name AS song_name
FROM song_album sa
INNER JOIN songs s
ON sa.song_id = s.song_id
GROUP BY sa.album_id, sa.song_id, sa.track_no, s.song_name
ORDER BY album_id) sas
WHERE album_id = a_id
GROUP BY album_id
ORDER BY album_id;
RETURN List_of_tracks;
END $$
DELIMITER ;
```



Conclusion

In conclusion, this course has provided an in-depth understanding of SQL and its various functionalities. The SQL code presented in this project covers some advanced features of SQL, such as views, triggers, stored procedures, and functions. By working through these examples, we gained practical experience in using SQL for real-world data management tasks. With this course's knowledge, we enhanced our data management skills and understood advanced SQL concepts.

THANK YOU