**Term Project -Database**

**Music Library Database:**

**Narrative:** A local music enthusiast community has been struggling to organize their music collections and listening habits effectively. Many enthusiasts face issues like mismanaged playlists and an inability to track listening history. In an effort to improve music management, we are developing a sample database to provide a better way to organize music collections and generate listening reports.

**User Groups:**

To address the needs of different users, the music library database identifies the following user groups:

**Music Enthusiasts:** These users are passionate about music and want to efficiently manage their music collections, create playlists, rate songs and participate in marketing campaigns.

**Admins:** Administrators have full access to the database and can manage user accounts, artists, albums, genres, and perform administrative tasks.

**User Views:**

**Music Enthusiasts:**

* Passionate about music.
* Manage music collections efficiently.
* Create and manage playlists.
* Rate songs on a scale of 1 to 5.
* Access lyrics for songs, if available.
* Participate in Marketing campaigns, get premium subscriptions.

**Admins:**

* Have full access to the database.
* Manage user accounts.
* Manage artists, albums, and genres.
* Perform administrative tasks to ensure the database's smooth operation. Promote Marketings.

**Table design:**

1. Songs Table:

* SongID: Primary key, unique identifier for each song.
* Title: Text field, stores the song title.
* ArtistID: Foreign key referencing the Artists table, identifies the song's artist.
* AlbumID: Foreign key referencing the Albums table, identifies the album the song belongs to.
* GenreID: Foreign key referencing the Genres table, identifies the song's genre.
* ReleaseYear: Date field, stores the year the song was released.
* Duration: Integer field, stores the song duration in seconds.

2. Albums Table:

* AlbumID: Primary key, unique identifier for each album.
* Title: Text field, stores the album title.
* ArtistID: Foreign key referencing the Artists table, identifies the album's artist.
* ReleaseYear: Date field, stores the year the album was released.

3. Artists Table:

* ArtistID: Primary key, unique identifier for each artist.
* Name: Text field, stores the artist's name.

4. Genres Table:

* GenreID: Primary key, unique identifier for each genre.
* GenreName: Text field, stores the genre name.

5. Playlists Table:

* PlaylistID: Primary key, unique identifier for each playlist.
* UserID: Foreign key referencing the Users table, identifies the user who created the playlist.
* Title: Text field, stores the playlist title.

6. Users Table:

* UserID: Primary key, unique identifier for each user.
* Username: Text field, stores the user's username.
* Password: Text field, stores the user's password (hashed for security!).
* FullName: Text field, stores the user's full name.
* Email: Text field, stores the user's email address.
* RegistrationDate: Date field, stores the user's registration date.

7. Ratings Table:

* RatingID: Primary key, unique identifier for each rating.
* UserID: Foreign key referencing the Users table, identifies the user who submitted the rating.
* SongID: Foreign key referencing the Songs table, identifies the song being rated.
* Rating: Integer field, stores the user's rating of the song (1-5 scale).

8. Lyrics Table:

* LyricID: Primary key, unique identifier for each lyrics entry.
* SongID: Foreign key referencing the Songs table, identifies the song the lyrics belong to.

9. PlaylistSongsTable:

* PlaylistID: Foreign key referencing the Playlist table, identifies the playlist the song belongs to.
* SongID: Foreign key referencing the Songs table, identifies the song added to the playlist.

10. Marketing Table:

* MarketingID: Primary key, unique identifier for each marketing campaign.
* CampaignName: Text field, stores the campaign's name.
* CampaignType: Text field, stores the campaign's type (e.g., email marketing, social media).
* StartDate: Date field, stores the campaign's start date.
* EndDate: Date field, stores the campaign's end date.
* Description: Text field, stores a brief description of the campaign.
* UserID: Foreign key referencing the Users table, identifies the user responsible for the campaign.

**Business Questions:**

1. Which artists have the highest number of albums in the database?
2. What is the average number of playlists created by each user?
3. What are the top-rated songs in the database?
4. How many users have provided ratings, and what is the average rating given?
5. Which marketing campaigns have been the most successful in terms of user engagement?
6. What are the most popular songs in the "mood" playlist?
7. What is the most played song overall, and who are the top three users playing it?//
8. What is the total duration of songs in each playlist, and which playlist has the longest total duration?

**DQL Queries:**

* What are the most popular songs in the "mood" playlist?

SELECT Songs.Title, COUNT(PlaylistSongs.SongID) AS PlayCount

FROM Songs

LEFT JOIN PlaylistSongs ON Songs.SongID = PlaylistSongs.SongID

JOIN Playlists ON PlaylistSongs.PlaylistID = Playlists.PlaylistID

WHERE Playlists.Title = 'mood'

GROUP BY Songs.SongID

ORDER BY PlayCount DESC;

**Output:**

God is a Woman|1

* How many users have provided ratings, and what is the average rating given?

SELECT COUNT(DISTINCT UserID) AS NumberOfUsers, AVG(Rating) AS AvgRating

FROM Ratings;

**Output:**

3|3.8

* Which marketing campaigns have been the most successful in terms of user engagement?

SELECT Marketing.CampaignName, COUNT(DISTINCT Playlists.UserID) AS UserEngagement

FROM Marketing

INNER JOIN Playlists ON Marketing.UserID = Playlists.UserID

LEFT JOIN PlaylistSongs ON Playlists.PlaylistID = PlaylistSongs.PlaylistID

GROUP BY Marketing.CampaignName

ORDER BY UserEngagement DESC;

**Output:**

Summer Sale|1

New Release|1

Holiday Special|1

* What is the total duration of songs in each playlist, and which playlist has the longest total duration?

SELECT Playlists.Title, SUM(Songs.Duration) AS TotalDuration

FROM Playlists

LEFT JOIN PlaylistSongs ON Playlists.PlaylistID = PlaylistSongs.PlaylistID

JOIN Songs ON PlaylistSongs.SongID = Songs.SongID

GROUP BY Playlists.PlaylistID

ORDER BY TotalDuration DESC

LIMIT 1;

**Output:**

myplaylist|1053

* What are the top-rated songs in the database?

SELECT Songs.Title, AVG(Ratings.Rating) AS AvgRating

FROM Songs

INNER JOIN Ratings ON Songs.SongID = Ratings.SongID

GROUP BY Songs.SongID

HAVING AvgRating > 3

ORDER BY AvgRating DESC

LIMIT 5;

**Output:**

Attention|5.0

Love Story|5.0

God is a Woman|4.0

* Which artists have the highest number of albums in the database?

SELECT Artists.Name, COUNT(Albums.AlbumID) AS AlbumCount

FROM Artists

INNER JOIN Albums ON Artists.ArtistID = Albums.ArtistID

GROUP BY Artists.ArtistID

HAVING AlbumCount > 1

ORDER BY AlbumCount DESC;

**Output:**

Ariana|2

* What is the average number of Playlists created by a user?

SELECT UserID, COUNT(PlaylistID) AS PlaylistCount

FROM Playlists

GROUP BY UserID;

**Output:**

1|1

2|1

3|2

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated