Introduction

This is a complete `MySQL database` with a `Python3 CLI`. The database `MUS3RHQ Label` is designed for a recording label agency and encompasses details about artists, albums, tours, employees, departments, investors, and more.

Instructions to Run the Program

Functional Requirements

MUS3RHQ Label supports the following functional requirements:

Insert

1. **Adding New Song/Album Data**
    - Function: `add_album_data()`
    - Description: Adds new album data including songs, collaborations, and teasers.

```
INSERT INTO Album (AlbumID, ReleaseDate) VALUES (<Album_ID>, '<Release_Date>');
INSERT INTO ChartPosition (AlbumID, ChartPositionNumber) VALUES (<Album_ID>, <Chart_Position>);
INSERT INTO Genre (AlbumID, GenreType) VALUES (<Album_ID>, '<Genre_Type>');
INSERT INTO Album_Tracklist (AlbumID, TrackList, Duration) VALUES (<Album_ID>, '<Tracklist>', <Duration>);
```

2. **Tour Schedule with Date Validity**
    - Function: `add_tour_dates()`
    - Description: Adds tour dates to the database and checks for date overlaps.

```
INSERT INTO tourdates (tourid, start_date, end_date) VALUES (<Tour_ID>, '<Start_Date>', '<End_Date>');
```

Update

1. **Chart Position Updates**
    - Function: `update_chart_position(album_name, new_chart_position)`
    - Description: Updates chart positions for albums.

```
UPDATE ChartPosition SET ChartPositionNumber = <New_Chart_Position> WHERE AlbumID = <Album_ID>;
```

2. **Marketing Campaign Reallocation**
    - Function: `assign_budget_to_campaign(user_budget)`
    - Description: Reallocates campaign budgets based on album performance.

```
UPDATE marketing_campaigns SET Budget = <User_Budget> WHERE campaign_name = '<Campaign_Name>';
```

Delete

1. **Managing Artist Transitions: Deleting Departing Artists**
    - Function: `delete_artist(artist_id)`
    - Description: Removes artist records when they leave the label.

```
DELETE FROM Artists WHERE artist_id = <Artist_ID>;
```

2. **Employee Deletion**
    - Function: `delete_employee(employee_id)`
    - Description: Deletes employee records for departed individuals.

```
DELETE FROM Employee WHERE EmployeeID = <Employee_ID>;
```

Selection

1. **Recent Album Releases by Artists**

- Function: `artists_released_past_year()`
- Description: Retrieves artists who released albums in the past year.

```
SELECT DISTINCT a.artist_name FROM Artists a INNER JOIN Album al ON a.album_id = al.AlbumID WHERE al.ReleaseDate >= DATE_SUB(N
```

Projection

1. **Searching by Genre**
   - Function: `get_album_names_by_genre(genre)`
   - Description: Retrieves album names by a specific genre.

```
SELECT an.AlbumName FROM Genre g INNER JOIN Album a ON g.AlbumID = a.AlbumID INNER JOIN AlbumName an ON a.CoverID = an.CoverID
```

2. **Searching tours in a certain Location**
   - Function: `get_tours_by_location(city, state)`
   - Description: Retrieves tours planned in a specific location.

```
SELECT t.tour_name FROM Tours t INNER JOIN TourLocations tl ON t.tour_id = tl.tour_id WHERE tl.city = '<City>' AND tl.state =
```

Aggregate

1. **Exploring Album Prolificacy**
   - Function: `total_released_albums()`
   - Description: Calculates the total count of released albums.

```
SELECT SUM(num_released_albums) AS TotalReleasedAlbums FROM Artists;
```

2. **Spotlighting the Minimalist Artist**
   - Function: `artist_with_min_albums()`
   - Description: Finds the artist with the least number of albums released.

```
SELECT artist_name, MIN(num_released_albums) AS MinReleasedAlbums FROM Artists;
```

Search

1. **Artist Name Lookup**
   - Function: `check_artist_name(user_input)`
   - Description: Searches for an artist name in the database.

```
SELECT artist_name FROM Artists WHERE artist_name LIKE '%<User_Input>%';
```

Functional Analysis

1. **Get High Grossing Collaboration**
   - Function: `get_high_grossing_collaboration()`
   - Description: Finds the highest grossing album's artist and associated producer.

```
SELECT ar.artist_name AS Artist, pr.producer_id, pr.employee_id, pr.total_revenue AS Producer_Revenue FROM Album a JOIN Artist
```

2. **Average Chart Position**
   - Function: `average_chart_position()`
   - Description: Calculates the average chart position per artist.

```
SELECT ar.artist_id, ar.artist_name, AVG(cp.ChartPositionNumber) AS AverageChartPosition FROM Artists ar JOIN Album al ON ar.a
```
