Instructions: The objective of this final is to show the ability to take requirements and stories, and translate them into a supporting database design, and then implement that design in SQL. Using the requirements below you will be expected to submit the following artifacts/files:

- An Entity Relationship Diagram (ERD or UML notation acceptable)
- A Schema Diagram (UML notation)
- SQL scripts for implementing the database described by the schema diagram. Including Foreign Key constraints. These constraints should cascade updates and restrict deletes.
- At least 4 SQL Statements (with comments) that will demonstrate one of each the record CRUD action (Create, Retrieve, Update, Delete). In other words, at least one INSERT (Create), SELECT (Retrieve), UPDATE and DELETE statement. You do not have to write a SELECT statement for all data queries listed in the user stories, but your data model must support such a query. *For extra credit, implement these actions as Stored Procedures with appropriate parameters.*

Note: No Call Level Interface or GUI code is required. You are simply designing the database, and writing queries to create and interact with it. No front end development is necessary. Also, the ERD and Schema Diagrams must be digitally drawn using a diagramming tool such as Draw.io, Visio, etc. No hand drawn diagrams will be accepted. Submit your diagrams (saved as image or document files), and SQL scripts (saved as text files) in a single zip file in the appropriate location on canvas.

Scenario: You have been contracted by a Pop Music Record Label to create a database to track and organize the music produced by artists signed to the label. Each individual artist should be stored with their given name and other identifying attributes such as date of birth, address, etc. Artists are then collected into Groups. Groups of artists, also known as 'bands' or 'acts' have their own name and demographics, such as 'date formed' and 'city of origin.' Groups can also consist of a single artist and an artist can be a part of multiple groups. For example, the artist John Michael Osbourne plays in two musical acts (1) Black Sabbath, along with several other band mates and (2) Ozzie Osbourne in his solo career. 'Groups' release 'Albums' and 'Albums' contain a list of 'Tracks.' The attributes that are added to the Album and Track entities are up to the designer, but they however must enable the behavior expected in user stories.

Based on the description above, the system should support the following user stories.

As a user, I would like...

1. ...to see which albums were released within any given time period.
2. ...to know what role each member plays in a group (e.g. Lead Singer, Drummer)
3. ...to know all of the groups a particular artist is a member of.
4. ...to know the order that the tracks appear on an album.
5. ...to know the length of each track on each album.
6. ...to delete albums recorded by a particular group in the case they sign with another label or breakup before any albums are released.
7. ...to change a group's name in the case they decide to change their name.
8. ...to add or remove group members or modify roles from a given group.
9. ...to look up a track by name and see if there are any tracks recorded with the same name recorded by a different artist.
10. ...to create a new album before the track list is known.
11. ...to know what artists perform different roles in different groups (e.g. Drummer in one group, and Singer in another),