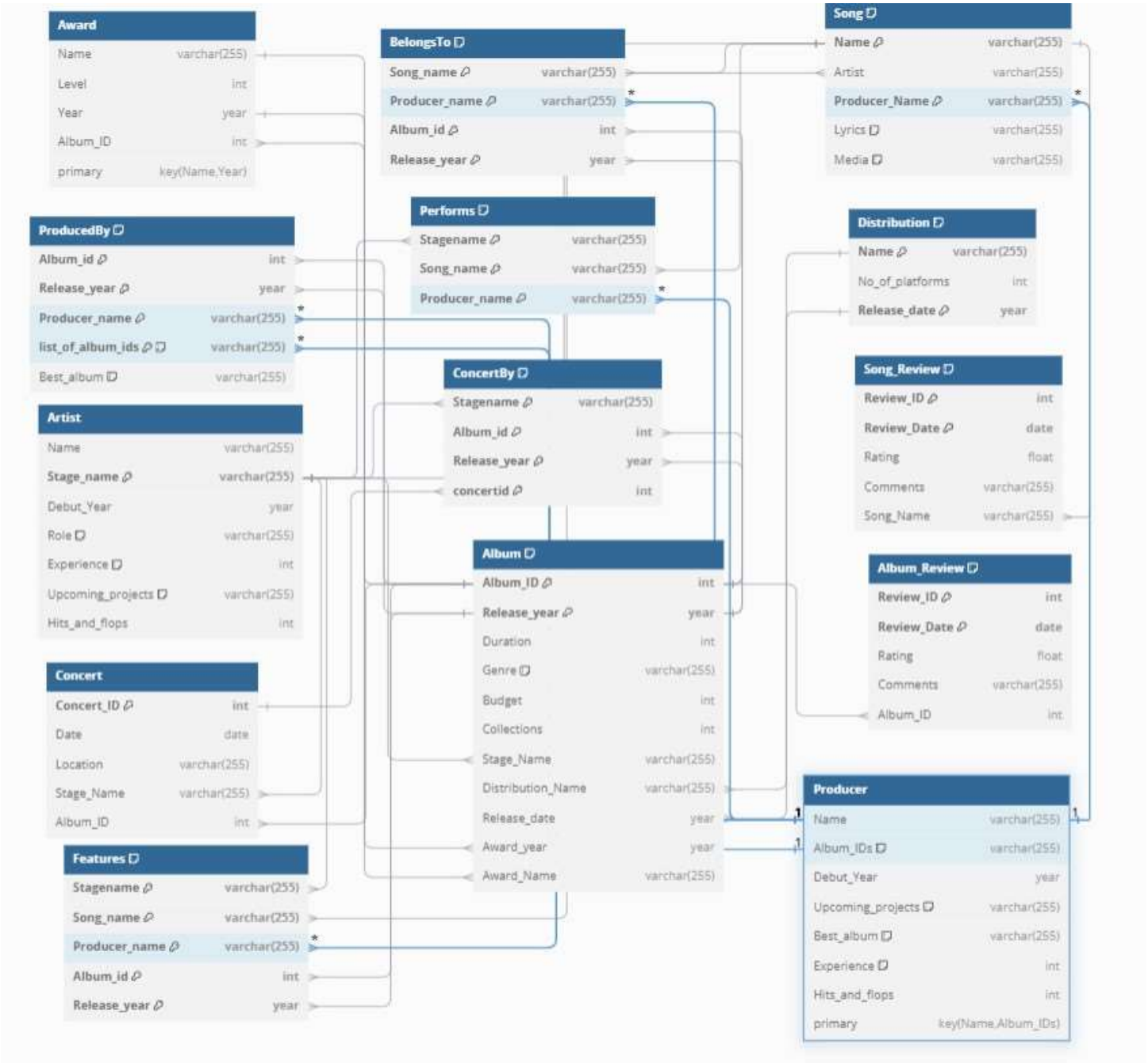
PROJECT PHASE 3
DATA AND APPLICATIONS

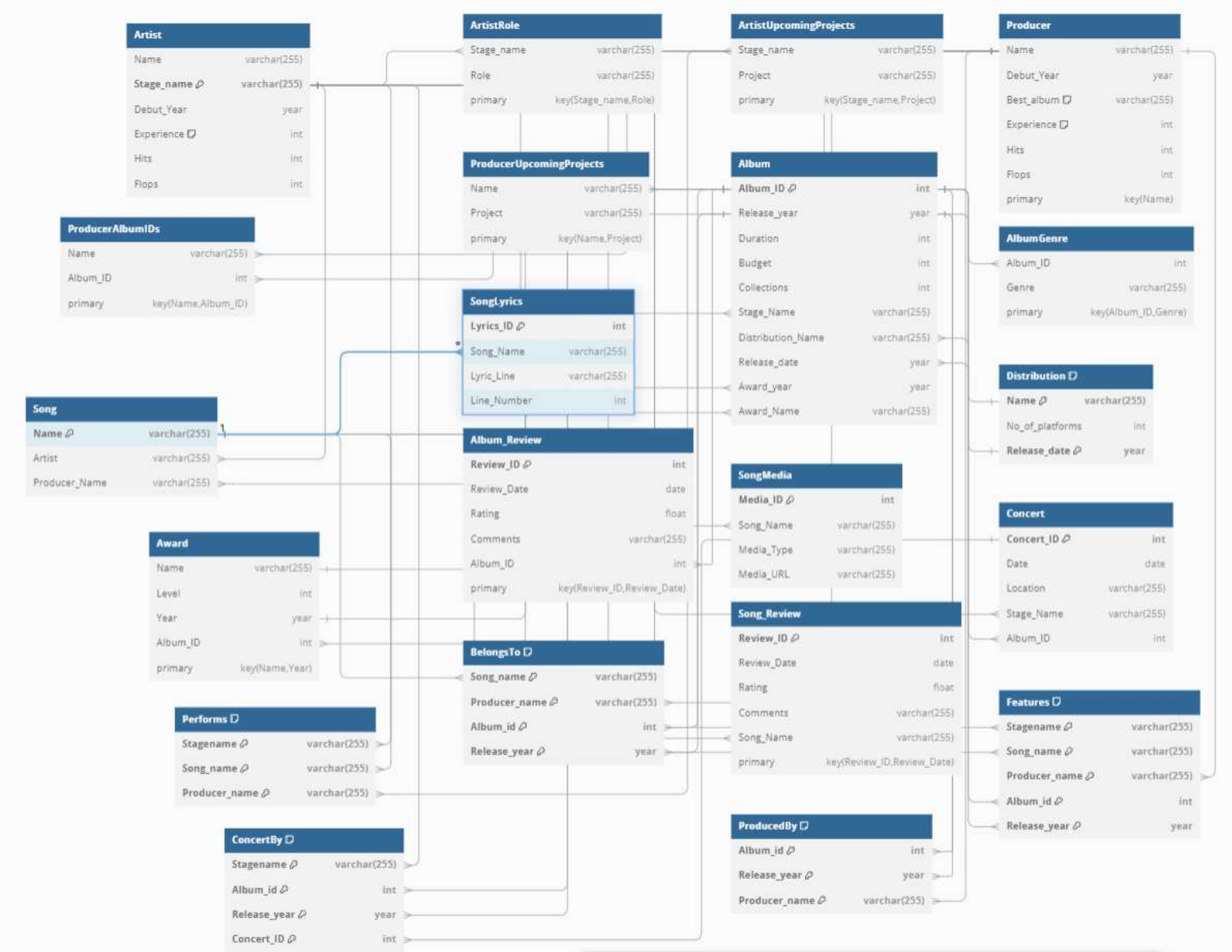
1. Relational model

https://dbdiagram.io/d/Relationalmodel-67389106e9daa85acaa9d969

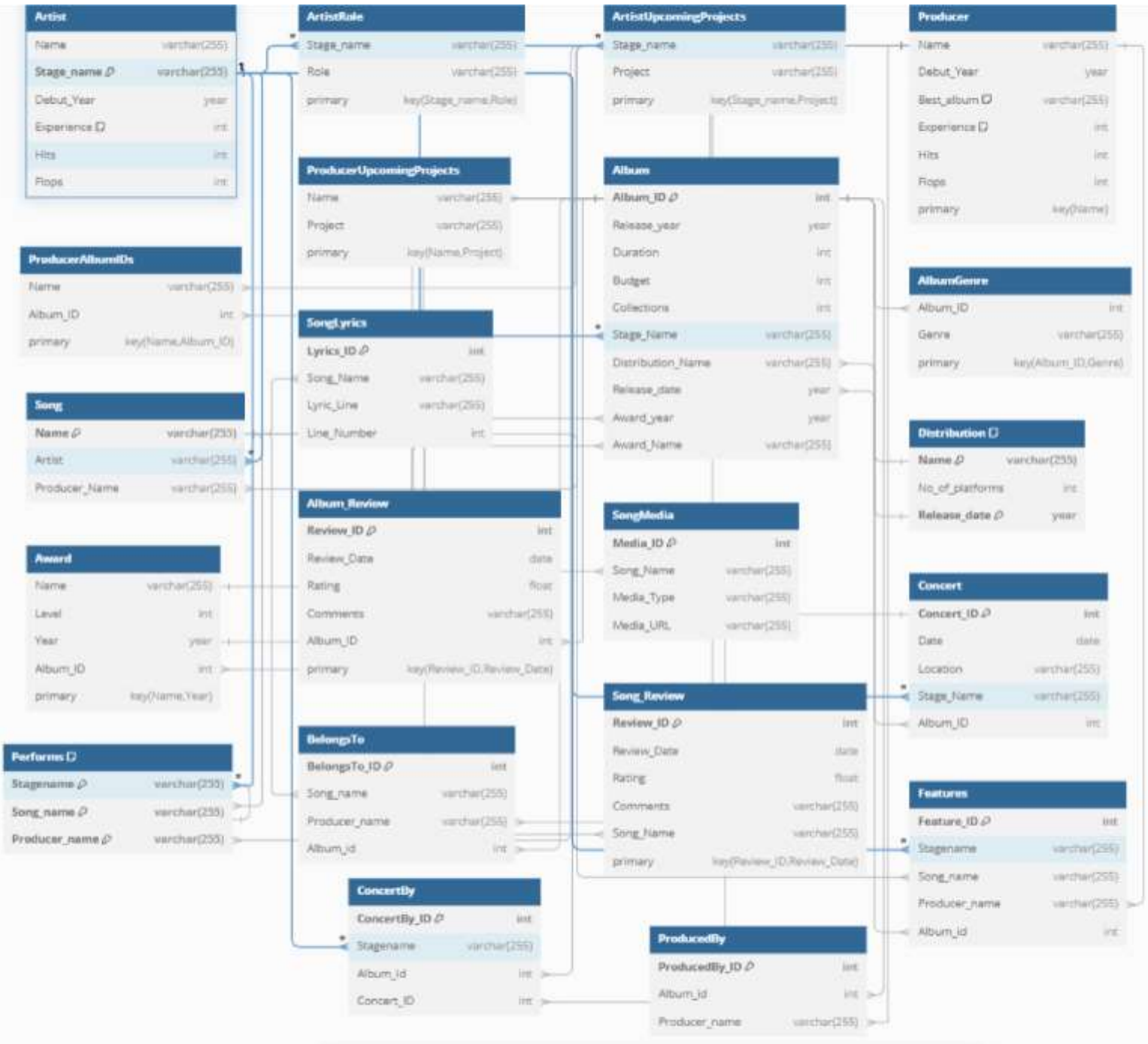


2. 1NF

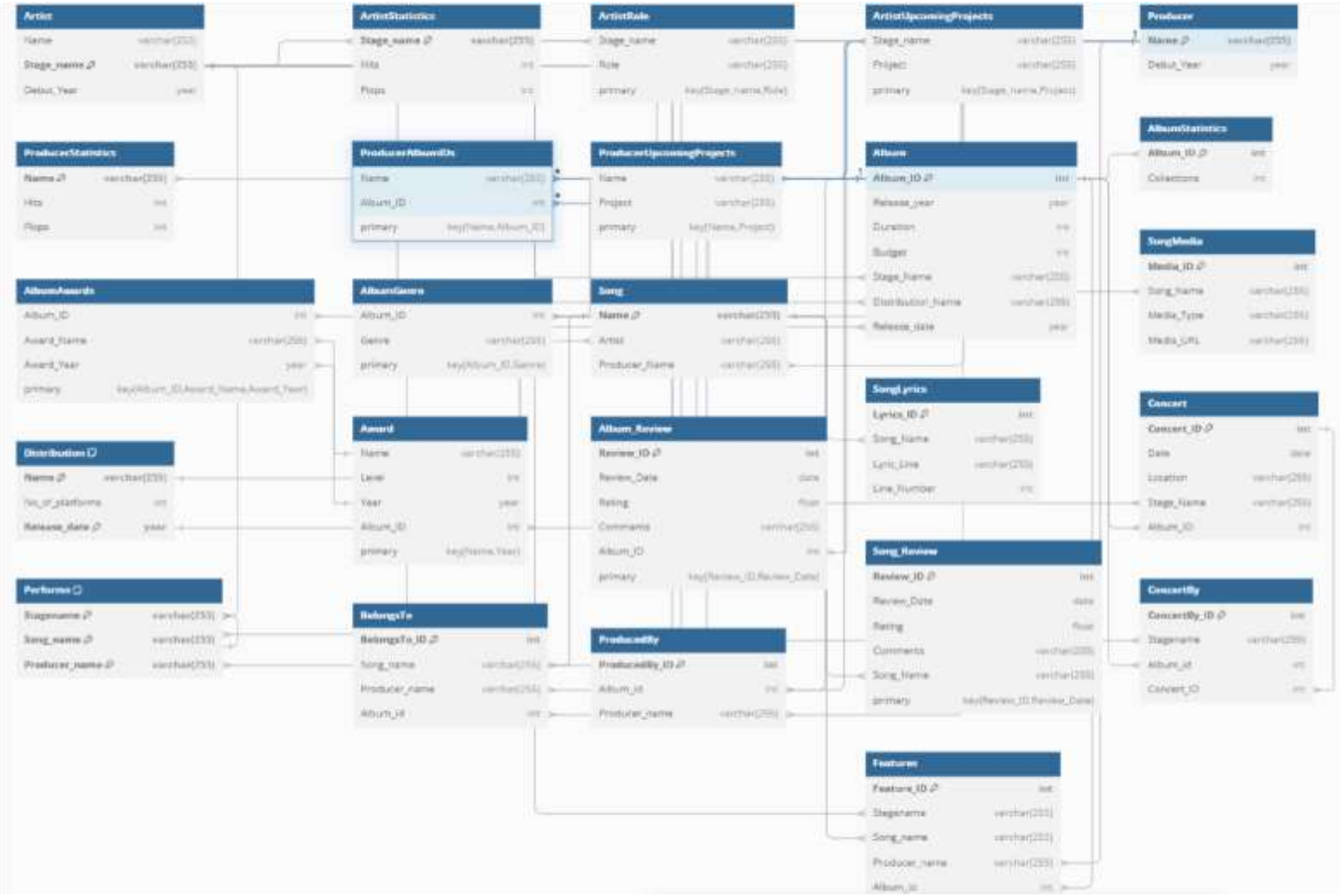
https://dbdiagram.io/d/1NF-673871f8e9daa85acaa83513



3. 2NF
https://dbdiagram.io/d/2NF-67387446e9daa85acaa852c0



4. 3NF
https://dbdiagram.io/d/3NF-67387947e9daa85acaa890ce



The details of the connections are in the link

We would like to clarify an important assumption made during the normalization process of the project, as reflected in the diagrams. Specifically, we treated certain composite attributes as atomic for the purposes of normalization. This decision was based on the dependency of these attributes' values on the tuple context, which may vary between being composite or atomic depending on the table's design.
For instance, the attribute "Comment" could either contain a single comment or multiple comments based on the table's requirements. To simplify the normalization process and maintain consistency, we assumed such attributes to be atomic within the scope of this project.

Please consider this into account during evaluation the project

1. Regular Entity Types Transformation
   - Artist: Primary key is Stage_name. Multi-valued attributes (Role, Upcoming_projects) maintained in same table
   - Producer: Composite primary key (Name, Album_IDs) due to multi-valued Album_IDs attribute
   - Album: Composite primary key (Album_ID), with foreign keys added for relationships
   - Song: Composite primary key (Name, Producer_Name) due to the relationship with Producer
2. Weak Entity Types Handling
   - Album_Review: Primary key combines Review_ID and Review_Date
   - Song_Review: Primary key combines Review_ID and Review_Date
   - Concert: Made into a regular entity with Concert_ID as primary key
3. Relationship Transformations

   M: N Relationships (separate tables):

   - Performs: Combines Artist.Stage_name, Song.Name, Producer.Name
   - BelongsTo: Links Song, Album with composite keys
   - Features: Combines all participating entities' keys
   - ConcertBy: Links Artist, Album, and Concert

   1: N Relationships (foreign keys):

   - ProducedBy: Managed through foreign keys in Album table
   - Distribution and Award relationships: Handled via foreign keys in Album

4. Inheritance Structure: The Singer subclass was merged into the Artist table since it had minimal additional attributes

5. Key Changes

   - Composite attributes were flattened into single columns
   - Multi-valued attributes were kept in the same tables but noted
   - Derived attributes were maintained but marked as derived
   - Foreign key constraints were added to maintain referential integrity

6. Notable Design Decisions

   - Maintained original composite keys where necessary for data integrity
   - Used surrogate keys (e.g., Concert_ID) where appropriate
   - Relationship tables preserve the original cardinality constraints
   - Added appropriate foreign key references to maintain relationships

Key Changes at Each Stage:

1. Relational Model to 1NF

- Multi-valued attributes were decomposed into separate relations:
  - Artist's Role and Upcoming_projects → ArtistRole and ArtistUpcomingProjects tables
  - Producer's Album_IDs and Upcoming_projects → ProducerAlbumIDs and ProducerUpcomingProjects tables
  - Song's Lyrics and Media → SongLyrics and SongMedia with surrogate keys
  - Album's Genre → AlbumGenre table
- Maintained composite keys for relationship tables (Performs, BelongsTo, Features, ConcertBy)
- Each attribute contains only atomic values

2. 1NF to 2NF

- Composite attribute Hits_and_flops split into separate Hits and Flops columns in Artist and Producer tables
- Introduced surrogate primary keys to eliminate partial dependencies:
  - BelongsTo_ID replaced composite key (Song_name, Producer_name, Album_id, Release_year)
  - ProducedBy_ID replaced (Album_id, Release_year, Producer_name)
  - Feature_ID replaced multiple attribute key
  - ConcertBy_ID replaced composite key structure
- All non-key attributes now fully dependent on their primary keys

3. 2NF to 3NF

- Removed transitive dependencies by creating new relations:
  - Artist table split into basic Artist info and ArtistStatistics
  - Producer table split into Producer info and ProducerStatistics
  - Album table split into Album, AlbumStatistics, and AlbumAwards
- Each non-key attribute is now directly dependent on its primary key
- Eliminated derived attributes into their own tables
- Maintained referential integrity through foreign key relationships

Key Improvements:

1. Better data integrity
2. Reduced data redundancy
3. More efficient storage
4. Easier maintenance
5. More flexible querying capabilities