**1. Record\_Label:**

* Primary Key: Label\_ID, an INTEGER type unique identifier for each record label, cannot be null.
* Other Attributes:
  + Label\_Name, a VARCHAR(50) type that is used to store the name of each record label, cannot be null

**2. Follows:**

* Primary Key: Follower\_Tag, Followed\_Tag, both foreign keys of type VARCHAR(50), are used to keep track of the many-to-many follows relationship. Neither primary key can be NULL. Both Follower\_Tag and Followed\_Tag’s primary key is a User.Tag.
* Constraints: If either Follower\_Tag OR Followed\_Tag is DELETED, delete this instance. If either Follower\_Tag OR Followed\_Tag is CHANGED, cascade.

**3. User:**

* Primary Key: Tag, a VARCHAR(50) type unique identifier starting with ‘@’  for each user, so that users can find each other using their unique tag. This value makes it so that if users want to have the same username, they can still be identified.
* Other Attributes:
  + Number\_Followers, an INT type that keeps track of the number of followers a user has. This number cannot be null and defaults to zero.
  + Num\_Following, an INT type that keeps track of the number of users a user is following. This number cannot be null and defaults to zero.
  + Username, a VARCHAR(50) type that keeps track of a user’s username, the name that shows up on their profile primarily. This cannot be null.

**4. Artist:**

* Primary Key: Tag, a VARCHAR(50) type unique identifier that serves as both the primary key and a foreign key referencing User(Tag). This ensures that every artist is also a registered user in the system. It cannot be null.
* Other Attributes:
  + Stage\_Name: A VARCHAR(100) type that stores the artist’s publicly displayed stage name. This cannot be NULL.
  + Monthly\_Listeners: An INT type that keeps track of the number of monthly listeners for the artist. This value cannot be NULL and defaults to 0.
  + Avg\_Ranking: A FLOAT type that keeps track of the average ranking of the artist, derived from the average of the artist’s album ranks and track ranks. This CAN be null, if the artist has no ranked tracks or albums.
* Constraints:
  + Tag must exist in the User table before an artist entry can be created.
  + ON DELETE CASCADE → If a user is deleted, their corresponding artist profile is also deleted.
  + ON UPDATE CASCADE → If a user’s Tag is changed, the change is automatically reflected in the Artist table.
  + Tag cannot be NULL → Every artist must have a valid, unique identifier that links to an existing user.

**5. Listens\_To:** (Many-to-many relationship)

* Primary Key: Listener\_Tag, Artist\_Tag — both are foreign keys of type VARCHAR(50) used to keep track of the many-to-many Listens\_To relationship between listeners and artists.
* Constraints:
  + Primary Key (Listener\_Tag, Artist\_Tag) ensures that each listener-artist pair is unique.
  + Foreign Keys:
    - Listener\_Tag references Listener(Tag), ensuring that the listener exists.
    - Artist\_Tag references Artist(Tag), ensuring that the artist exists.
  + ON DELETE CASCADE: If a listener is deleted, all records of their listening history are removed.If an artist is deleted, all records of their listens are removed.
  + ON UPDATE CASCADE: If a listener’s tag is changed, the change propagates to Listens\_To. If an artist’s tag is changed, the change propagates to Listens\_To.
  + Neither Listener\_Tag nor Artist\_Tag can be NULL.

**6. Listener:**

* Primary Key: Tag, a VARCHAR(50) type unique identifier that serves as both the primary key and a foreign key referencing User(Tag). This ensures that every listener is also a registered user in the system. Tag cannot be NULL.
* Other Attributes:
  + Top\_Artist, a VARCHAR(50) type that serves as a foreign key referencing Artist(Tag). This attribute stores the tag of the listener’s favorite artist. A listener can have at most one favorite artist at a time. This value can be NULL if a listener has not chosen a favorite artist.
* Constraints:
  + Tag must exist in the User table before a listener entry can be created.
  + Top\_Artist must reference an existing artist in the Artist table.
  + ON DELETE CASCADE → If a listener is deleted, their favorite artist reference is also deleted.
  + ON DELETE SET NULL → If an artist is deleted, any listener who had that artist as their favorite will have their Top\_Artist set to NULL.
  + ON UPDATE CASCADE → If an artist’s Tag is changed, the update is automatically reflected in the Listener table.
  + Tag cannot be NULL → Every listener must have a valid, unique identifier that links to an existing user.

**7. Playlist** (a weak entity of the User)

* Primary Key: Playlist\_ID, an INTEGER type unique identifier for each playlist. This serves as the primary key.
* Foreign Key: Tag, a VARCHAR(50) type foreign key referencing User(Tag), representing the user who created the playlist.
* Other Attributes:
  + Playlist\_Name, a VARCHAR(100) type that stores the name of the playlist. This cannot be NULL.
  + Num\_Tracks, an INT type that represents the number of tracks in the playlist. This value is derived from the Added\_To relationship and should not be manually set.
* Constraints:
  + Playlist must be linked to an existing user (Tag references User(Tag)).
  + ON DELETE CASCADE → If a user is deleted, all of their playlists are deleted.
  + ON UPDATE CASCADE → If a user’s tag is updated, the change is reflected in the Playlist table.
  + Num\_Tracks is a derived attribute, so it should be calculated dynamically from Added\_To rather than stored.

**8. Added\_To** (Many-to-many relationship)

* Primary Key:
  + Playlist\_ID, an INTEGER type foreign key referencing Playlist(Playlist\_ID). This ensures that the track is associated with a valid playlist.
  + Track\_ID, an INTEGER type foreign key referencing Track(Track\_ID). This ensures that the track being added exists in the system.
* Constraints:
  + Primary Key (Playlist\_ID, Track\_ID) → Ensures that a track cannot be added to the same playlist more than once.
  + Foreign Keys:Playlist\_ID references Playlist(Playlist\_ID), ensuring the playlist exists. Track\_ID references Track(Track\_ID), ensuring the track exists.
  + ON DELETE CASCADE: If a playlist is deleted, all records of tracks in that playlist are also deleted. If a track is deleted, all records of that track being in playlists are also deleted.
  + ON UPDATE CASCADE: If a playlist’s ID changes, the change is reflected in Added\_To. If a track’s ID changes, the change is reflected in Added\_To.

**9. Writes** (Many-to-many relationship)

* Primary Key:
  + Artist\_Tag, a VARCHAR(50) type foreign key referencing Artist(Tag). This ensures that the track is associated with a valid artist.
  + Track\_ID, an INTEGER type foreign key referencing Track(Track\_ID). This ensures that the track being written exists in the system.
* Constraints:
  + Primary Key (Artist\_Tag, Track\_ID) → Ensures that each artist can only have one writing contribution per track.
  + Foreign Keys: Artist\_Tag references Artist(Tag), ensuring the artist exists. Track\_ID references Track(Track\_ID), ensuring the track exists.
  + ON DELETE CASCADE: If an artist is deleted, all records of tracks they have written are also deleted. If a track is deleted, all records of artists who wrote it are also deleted.
  + ON UPDATE CASCADE: If an artist’s tag is updated, the change is reflected in Writes. If a track’s ID is updated, the change is reflected in Writes.

**10. Interaction** (Weak entity of User)

* Primary Key: Interaction\_ID, an INTEGER type unique identifier for each interaction. This serves as the weak entity key and ensures that each interaction has a distinct ID
* Foreign Keys:
  + Tag, a VARCHAR(50) type foreign key referencing User(Tag), representing the user who created the interaction.
  + Track\_ID, an INTEGER type foreign key referencing Track(Track\_ID), representing the track the interaction is associated with (can be NULL).
  + Album\_ID, an INTEGER type foreign key referencing Album(Album\_ID), representing the album the interaction is associated with (can be NULL).
* Constraints:
  + Primary Key (Interaction\_ID) uniquely identifies each interaction.
  + ON DELETE CASCADE
    - If a user is deleted, all their interactions are deleted.
    - If a track is deleted, any interactions linked to that track are deleted.
    - If an album is deleted, any interactions linked to that album are deleted.
  + ON UPDATE CASCADE → If a user’s, track’s, or album’s ID is updated, the change is reflected in Interaction.
  + Track\_ID and Album\_ID can be NULL, since an interaction can be associated with either a track, an album, or neither.

**11. Review:** (Subclass of Interaction)

* Primary Key: Interaction\_ID, an INTEGER type foreign key referencing Interaction(Interaction\_ID), ensuring that every review is an interaction.
* Other Attributes:
  + Review\_Txt, a TEXT type that contains the user's written review. This cannot be NULL.
* Constraints:
  + Primary Key (Interaction\_ID) ensures that each review is linked to an interaction.
  + ON DELETE CASCADE → If an interaction is deleted, its review is deleted.
  + There is no situation where an Interaction ID will UPDATE

**12. Like** (Subclass of Interaction)

* Primary Key: Interaction\_ID, an INTEGER type foreign key referencing Interaction(Interaction\_ID), ensuring that every like is an interaction.
* Other Attributes:
  + Liked, a BOOLEAN type that indicates whether the interaction was a like (TRUE) or unlike (FALSE).
* Constraints:
  + Primary Key (Interaction\_ID) ensures that each like is linked to an interaction.
  + ON DELETE CASCADE → If an interaction is deleted, its like is deleted.
  + There is no situation where an Interaction ID will UPDATE

**13. Rating** (Subclass of Interaction)

* Primary Key: Interaction\_ID, an INTEGER type foreign key referencing Interaction(Interaction\_ID), ensuring that every rating is an interaction.
* Other Attributes:
  + Rating\_Value, a FLOAT type that represents the user’s rating out of 10. This cannot be NULL and must be between 0.0 and 10.0.
* Constraints:
  + Primary Key (Interaction\_ID) ensures that each rating is linked to an interaction.
  + ON DELETE CASCADE → If an interaction is deleted, its rating is deleted.
  + There is no situation where an Interaction ID will UPDATE

**14. Album**:

* Primary Key: Album\_ID, an INTEGER type unique identifier for each album.
* Other Attributes:
  + Title, a VARCHAR(255) type that stores the album's title. This cannot be NULL.
  + Date\_Released, a DATE type that records the release date of the album. This cannot be NULL.
  + Like\_Count, an INT type derived from the Like interactions, representing the total number of likes the album has received. This should not be manually set.
  + Avg\_Rating, a FLOAT type derived from the Rating interactions, representing the average user rating of the album. This should not be manually set.
  + Num\_Tracks, an INT type derived from the Is\_On relationship (Tracks to Album), representing the total number of tracks on the album. This should not be manually set.
  + Producer, a VARCHAR(100) type that stores the title of the producer of the album. This can be NULL
* Constraints:
  + Primary Key (Album\_ID) uniquely identifies each album.
  + ON DELETE CASCADE: If an album is deleted, all interactions (likes, ratings) related to it should also be deleted.
  + ON UPDATE CASCADE: If the album’s ID is changed, it should be reflected in all relationships.
  + Like\_Count is derived by counting the number of Like interactions associated with the album.
  + Avg\_Rating is derived by computing the average of all Rating\_Value entries from the Rating interactions associated with the album.
  + Num\_Tracks is derived by counting the number of tracks associated with the album through the Is\_On relationship.

**15. Track:**

* Primary Key: Track\_ID, an INTEGER type unique identifier for each track.
* Foreign Key: Album\_ID, an INTEGER type foreign key referencing Album(Album\_ID), representing the album the track belongs to.
* Other Attributes:
  + Title, a VARCHAR(255) type that stores the track's title. This cannot be NULL.
  + Date\_Released, a DATE type that records the release date of the track. This cannot be NULL.
  + Genre, a VARCHAR(50) type that categorizes the track’s genre (e.g., Rock, Pop, Jazz). This can be NULL if the genre is unknown.
  + Length, a TIME type that represents the track’s duration (hh:mm:ss). This cannot be NULL.
  + Like\_Count, an INT type derived from the Like interactions, representing the total number of likes the track has received. This should not be manually set.
  + Avg\_Rating, a FLOAT type derived from the Rating interactions, representing the average user rating of the track. This should not be manually set.
  + Producer, a VARCHAR(100) type that stores the title of the producer of the track. This can be NULL
* Constraints:
  + Primary Key (Track\_ID) uniquely identifies each track.
  + Foreign Key (Album\_ID) references Album(Album\_ID), ensuring that the track is part of an existing album.
  + ON DELETE SET NULL: If an album is deleted, the track remains but its Album\_ID is set to NULL (to prevent orphaned tracks from being lost).
  + ON UPDATE CASCADE: If the album’s Album\_ID changes, it is reflected in the Track table.
  + Like\_Count is derived by counting the number of Like interactions associated with the track.
  + Avg\_Rating is derived by computing the average of all Rating\_Value entries from the Rating interactions associated with the track.

**16. Chart:**

* Primary Key: Chart\_ID, an INTEGER type unique identifier for each chart.
* Other Attributes:
  + Category, a VARCHAR(50) type that stores the category of the chart (e.g., "Top 100 Songs", "Top 50 Albums"). This cannot be NULL.
  + Avg\_Ranking, a FLOAT type that represents the average ranking of tracks or albums on the chart. This can be NULL if no rankings are available.
* Constraints:
  + Primary Key (Chart\_ID) uniquely identifies each chart.
  + ON DELETE CASCADE: If a chart is deleted, all associated rankings (Track\_Rank and Album\_Rank) are also deleted.
  + ON UPDATE CASCADE: If the Chart\_ID is updated, the change is reflected in all related rankings.

**17. Track\_Rank:**

* Primary Key:
  + Track\_ID, an INTEGER type foreign key referencing Track(Track\_ID). This ensures the track exists.
  + Chart\_ID, an INTEGER type foreign key referencing Chart(Chart\_ID). This ensures the chart exists.
* Other Attributes:
  + Rank, an INTEGER type that represents the position of the track on the chart. This cannot be NULL.
* Constraints:
  + Primary Key (Track\_ID, Chart\_ID) ensures that each track can only appear once on a chart.
  + Foreign Keys:
    - Track\_ID references Track(Track\_ID), ensuring the track exists.
    - Chart\_ID references Chart(Chart\_ID), ensuring the chart exists.
  + ON DELETE CASCADE: If a track is deleted, its ranking on any chart is also deleted. If a chart is deleted, all track rankings on that chart are also deleted.
  + ON UPDATE CASCADE: If a Track\_ID or Chart\_ID is updated, the change is reflected in Track\_Rank.

**18. Album\_Rank:**

* Primary Key:
  + Album\_ID, an INTEGER type foreign key referencing Album(Album\_ID). This ensures the album exists.
  + Chart\_ID, an INTEGER type foreign key referencing Chart(Chart\_ID). This ensures the chart exists.
* Other Attributes:
  + Rank, an INTEGER type that represents the position of the album on the chart. This cannot be NULL.
* Constraints:
  + Primary Key (Album\_ID, Chart\_ID) ensures that each album can only appear once on a chart.
  + Foreign Keys:
    - Album\_ID references Album(Album\_ID), ensuring the album exists.
    - Chart\_ID references Chart(Chart\_ID), ensuring the chart exists.
  + ON DELETE CASCADE: If an album is deleted, its ranking on any chart is also deleted. If a chart is deleted, all album rankings on that chart are also deleted.
  + ON UPDATE CASCADE: If an Album\_ID or Chart\_ID is updated, the change is reflected in Album\_Rank.