**Uke Project Database**



**Drew Smith & John Maddux**

**Project Topic: Searchable Ukulele Chord Database**

**Table of Contents**

Executive Overview......................................................................................................3

Business Rules..............................................................................................................3

Database Model...........................................................................................................4

Relational Schema........................................................................................................5

Functional Dependencies.............................................................................................5

Normalized Tables........................................................................................................6

CRUD Matrix.................................................................................................................6

Data Dictionary.............................................................................................................7

Queries..........................................................................................................................7

Forms & Reports...........................................................................................................21

Reflections.....................................................................................................................27

**Executive Overview**

The ukulele is a fun instrument with a wide range of musical potential. While it only has four strings there are over a hundred chords to learn. The goal is to create a searchable database allowing our users access to song chords based on genre, artist, level of difficulty, and more.

Our system will also give users the ability to add their own collection of chords. This can range from different versions of songs to original songs.

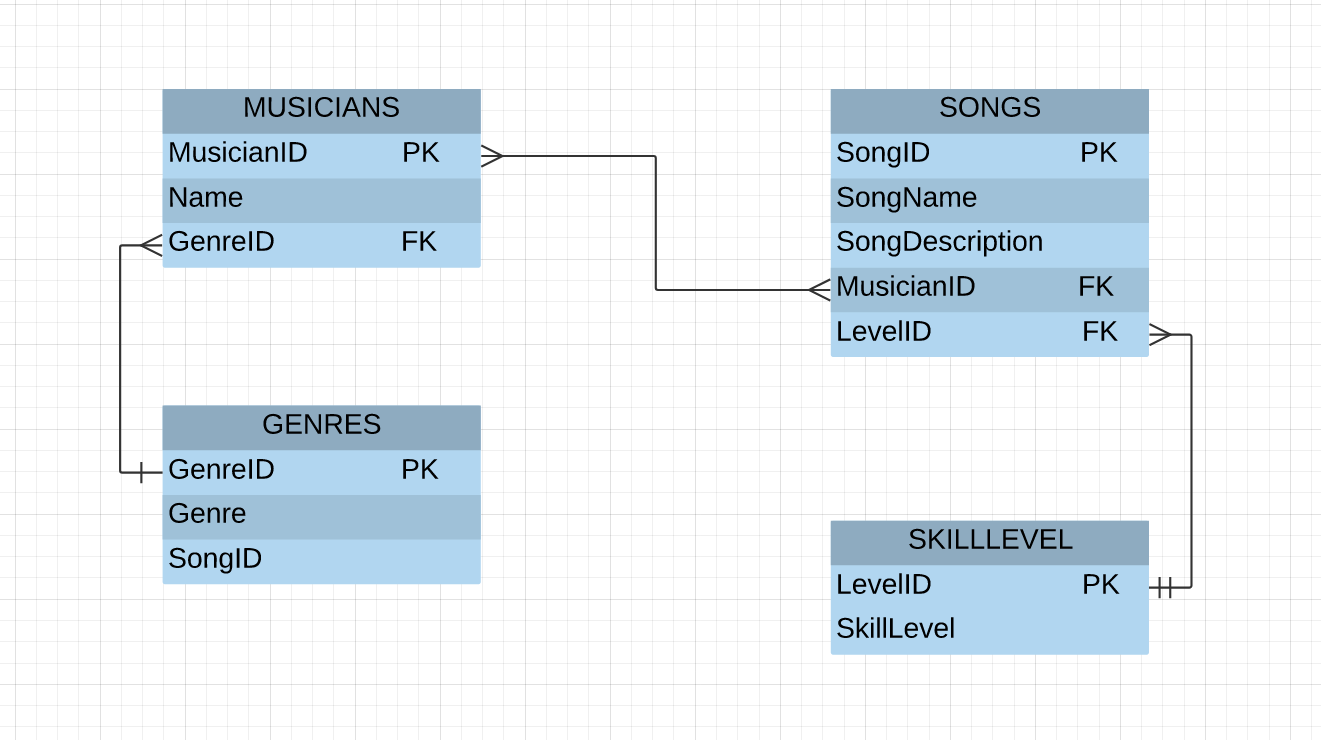
Our Project will be completed using Microsoft Access

**Business Rules**

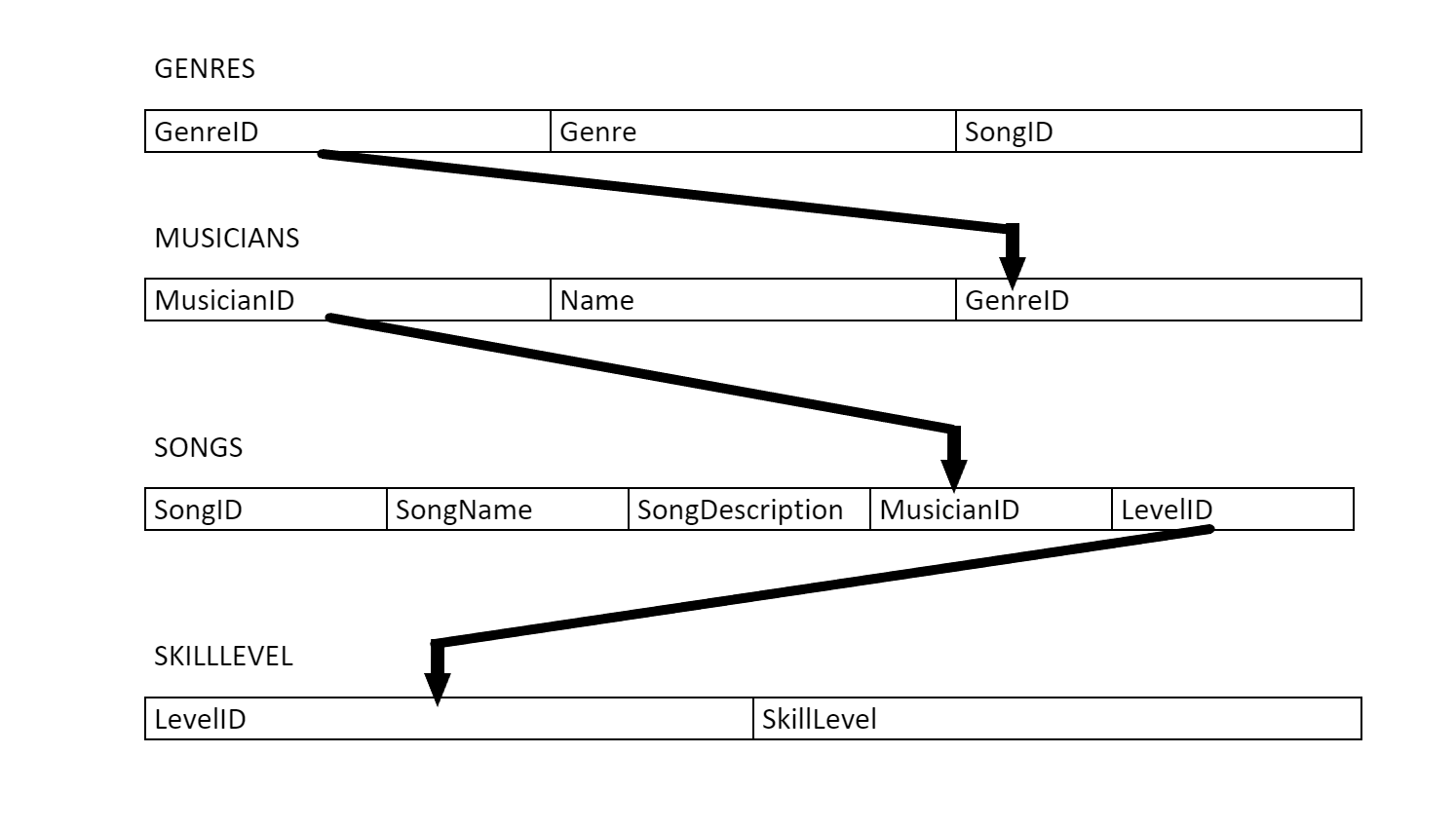
**UKE: A searchable database that will help users with their ukulele**

* UKE retains general information about artists for songs that we use and their studio companies. This information should be public, and for fair use. This information includes:
  + A unique id that is attached to each musician and each genre
  + Basic musician information
  + Basic song and genre information
    - Song title and description will be retained
* Skill level will be provided for each song for the user to be able to match their own skill to our skill levels.
* A user may sort between genre and skill level to best match their personal preferences when searching the database. This will narrow down to different musicians within the search field and the user can then narrow down their search to their favorite artists and songs.
* Data is collected from the user’s searches for review. This data will provide feedback for future database revisions

**Database Model**



**Relationship Schema**



**Functional Dependencies**

All Attributes:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *GenreID* | *Genre* | *MusicianID* | *Name* | *SongID* | *SongName* | *SongDescription* | *LevelID* | *SkillLevel* |

**Rule:** A musician ID is assigned to each artist to identify the artist's name

**MusicianID** **→ Name**

**Rule:** A song ID is assigned to each song to identify all the basic information

**SongID** **→ SongName, SongDescription**

**Rule:** A skill level will be assigned to each song in order to give user an idea of what they should be searching for.

**LevelID→ SkillLevel**

**Rule:** Songs are given genres based on their artist

**GenreID → Genre, MusicianID**

**Normalized Tables**

GENRES(**GenreID**,Genre,**SongID**)

MUSICIANS(**MusicianID**,Name,GenreID)

SONGS(**SongID**,SongName,SongDescription,**MusicianID,LevelID**)

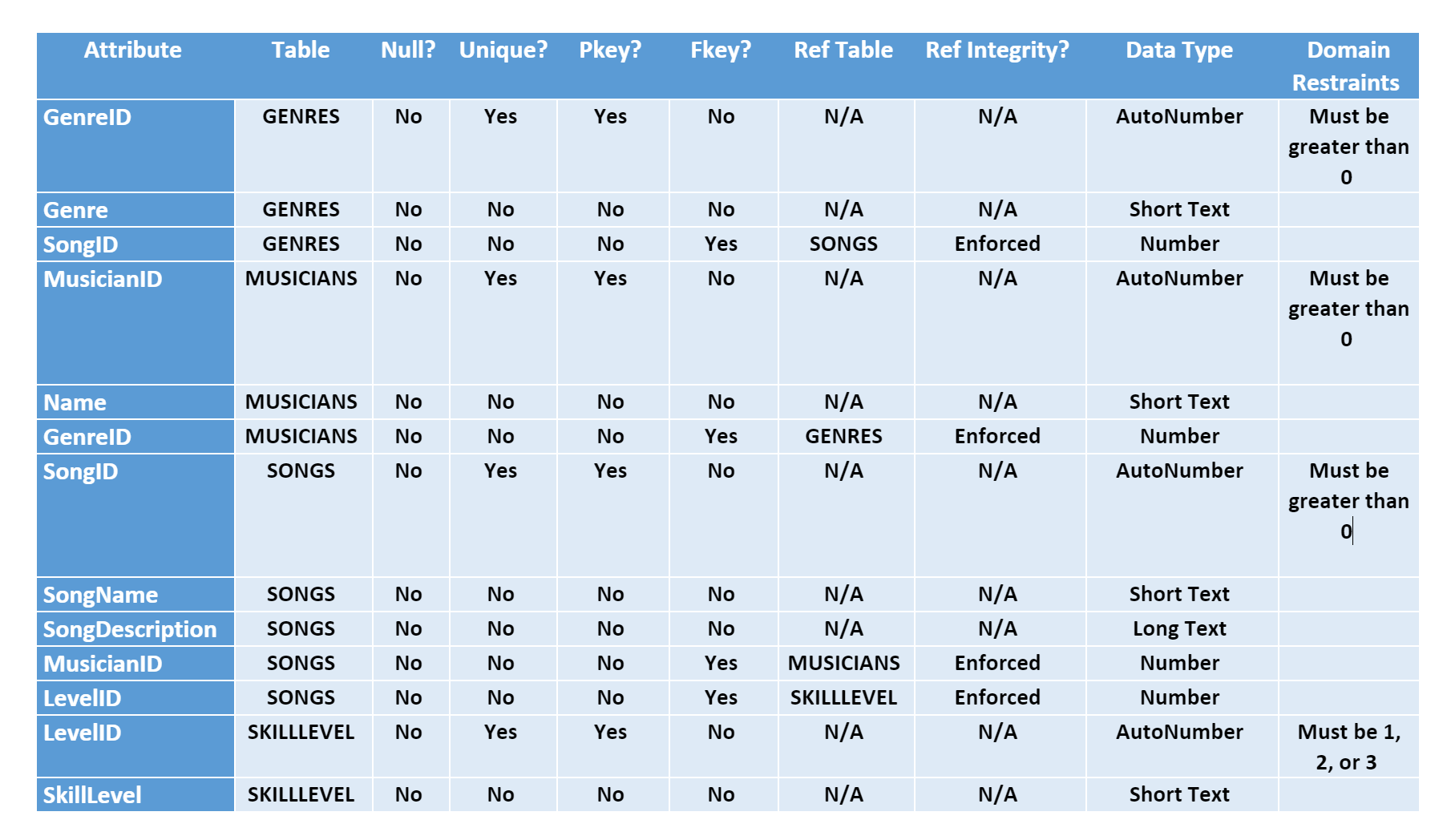
SKILLLEVEL(**LevelID**,SkillLevel)

**As shown in the relational schema, the data model is in 3NF. Each table is a relation and each table have an identified unique primary key.**

**Crud Matrix**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User Action** | **Create** | **Read** | **Update** | **Delete** |
| **Enter Website** |  | Search for Song/Musician |  |  |
| **Add Song** | Enter Song Information |  | SONG Table |  |
| **Add Artist** | Enter Musician Information |  | MUSICIANS Table |  |
| **Select Musician** |  | Pick a Musician |  |  |
| **Select Song** |  | Pick a Song |  |  |

**Data Dictionary**

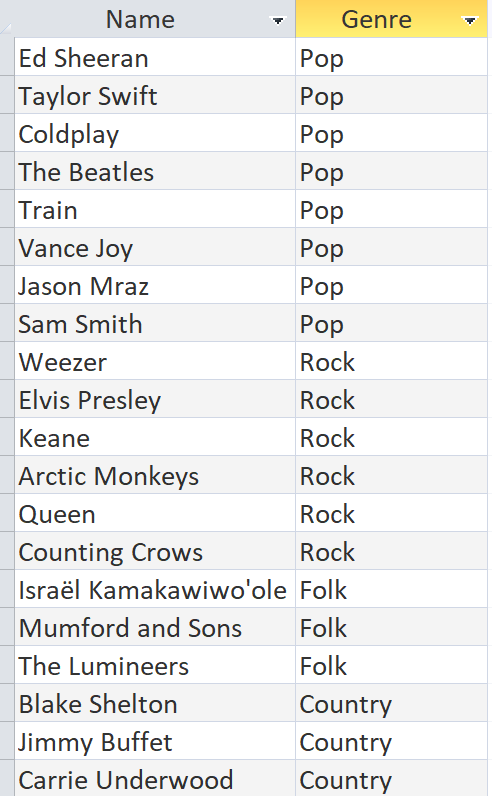


**Queries**

1. List all Musicians in the database with Genre type

**SELECT Musicians.Name, Genres.Genre**

**FROM Genres INNER JOIN Musicians ON Genres.[GenreID] = Musicians.[GenreID];**



1. List Song name and difficulty level

**SELECT Songs.[Song Name], SkillLevel.SkillLevel**

**FROM SkillLevel INNER JOIN Songs ON SkillLevel.[LevelID] = Songs.[LevelID];**



1. List all the song names and their genres

**SELECT Songs.[Song Name], Genres.Genre**

**FROM (Genres INNER JOIN Musicians ON Genres.[GenreID] = Musicians.[GenreID]) INNER JOIN Songs ON Musicians.[MusicianID] = Songs.[MusicianID];**



1. List Musician’s names with their MusicianID

**SELECT Musicians.[MusicianID], Musicians.[Name]**

**FROM Musicians;**



1. List all Songs and their Musicians and Genres

**SELECT Genres.Genre, Songs.[Song Name], Musicians.Name**

**FROM (Genres INNER JOIN Musicians ON Genres.[GenreID] = Musicians.[GenreID]) INNER JOIN Songs ON Musicians.[MusicianID] = Songs.[MusicianID];**



1. List Song Name, Description, and Musician

**SELECT Songs.[Song Name], Songs.SongDescription, Musicians.Name**

**FROM Musicians INNER JOIN Songs ON Musicians.[MusicianID] = Songs.[MusicianID];**



1. List Difficulty Level of Songs with Description

**SELECT SkillLevel.SkillLevel, Songs.[Song Name], Songs.SongDescription**

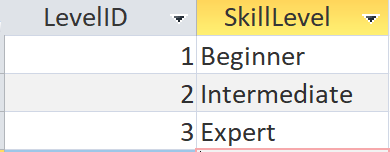
**FROM SkillLevel INNER JOIN Songs ON SkillLevel.[LevelID] = Songs.[LevelID];**



1. List LevelID and corresponding SkillLevel

**SELECT SkillLevel.[LevelID], SkillLevel.[SkillLevel]**

**FROM SkillLevel;**



1. List all Song Names with their corresponding SongID’s

**SELECT Songs.[Song Name], Songs.[SongID]**

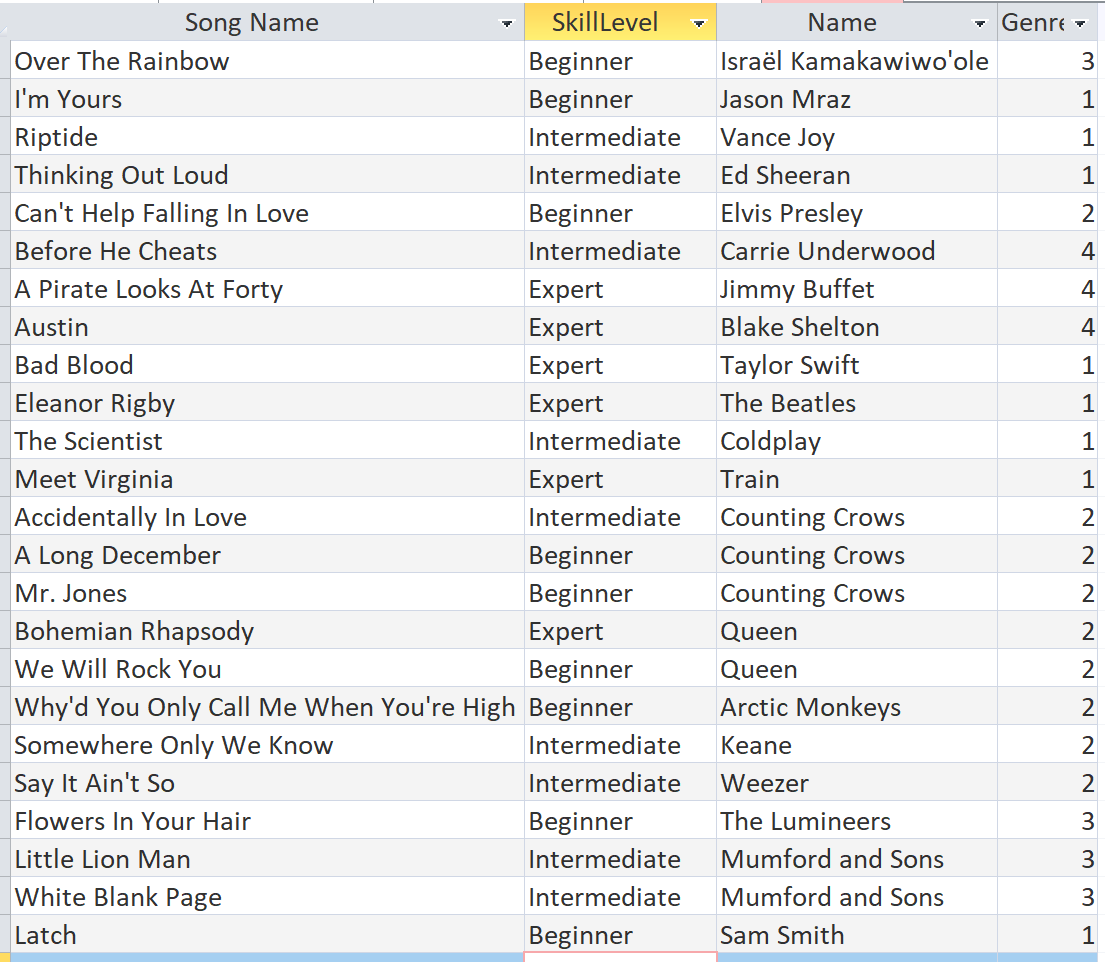
**FROM Songs;**



1. List all Song Names with Genre, Musician Name, and Genre ID

**SELECT Songs.[Song Name], SkillLevel.SkillLevel, Musicians.Name, Genres.GenreID**

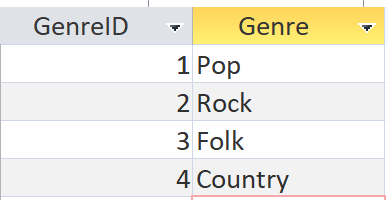
**FROM SkillLevel INNER JOIN ((Genres INNER JOIN Musicians ON Genres.[GenreID] = Musicians.[GenreID]) INNER JOIN Songs ON Musicians.[MusicianID] = Songs.[MusicianID]) ON SkillLevel.[LevelID] = Songs.[LevelID];**



1. List Genre ID and corresponding Genre

**SELECT Genres.[GenreID], Genres.[Genre]**

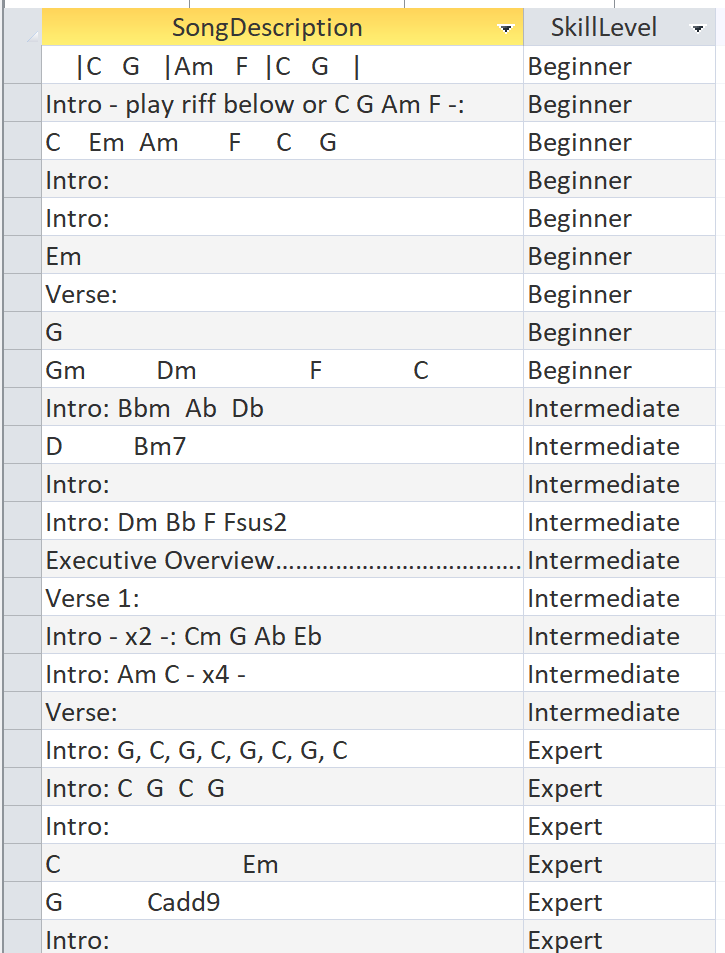
**FROM Genres;**



1. List Song Description with difficulty level

**SELECT Songs.SongDescription, SkillLevel.SkillLevel**

**FROM SkillLevel INNER JOIN Songs ON SkillLevel.[LevelID] = Songs.[LevelID];**

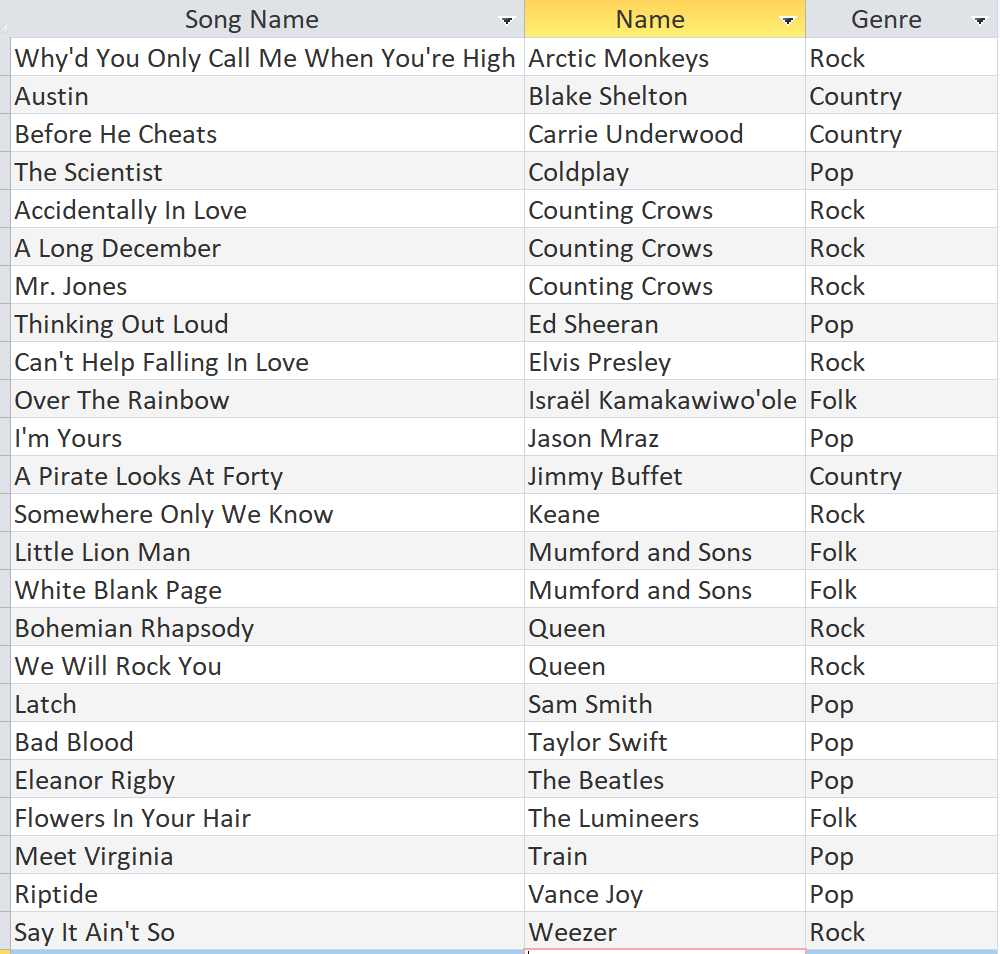


1. List Song Name with their Musicians and Genres. Sorted by Musician’s Name

**SELECT Songs.[Song Name], Musicians.Name, Genres.Genre**

**FROM (Genres INNER JOIN Musicians ON Genres.[GenreID] = Musicians.[GenreID]) INNER JOIN Songs ON Musicians.[MusicianID] = Songs.[MusicianID]**

**ORDER BY Musicians.Name;**

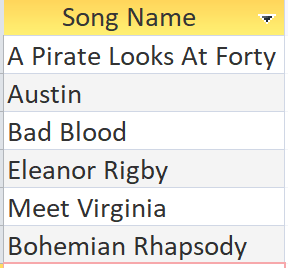


1. List Song Names with an Expert Skill Level

**SELECT Songs.[Song Name]**

**FROM SkillLevel INNER JOIN Songs ON SkillLevel.[LevelID] = Songs.[LevelID]**

**WHERE (((SkillLevel.[SkillLevel])="Expert"));**

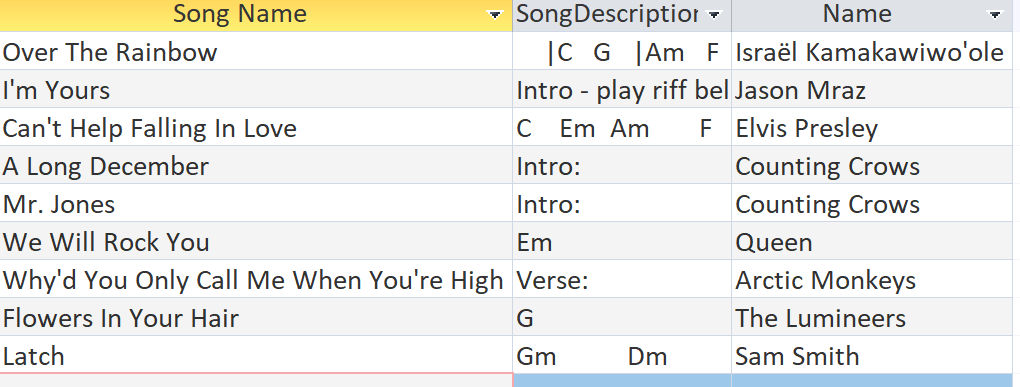


1. List all Beginner level songs with Musician’s name and Song Description

**SELECT Songs.[Song Name], Songs.SongDescription, Musicians.Name**

**FROM SkillLevel INNER JOIN (Musicians INNER JOIN Songs ON Musicians.[MusicianID] = Songs.[MusicianID]) ON SkillLevel.[LevelID] = Songs.[LevelID]**

**WHERE (((SkillLevel.[SkillLevel])="Beginner"));**



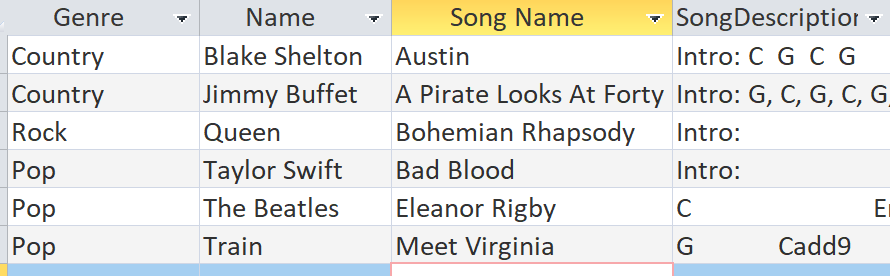
1. List all Expert level songs with Song name, Genre, Song description, and Musician

**SELECT Genres.Genre, Musicians.Name, Songs.[Song Name], Songs.SongDescription**

**FROM SkillLevel INNER JOIN ((Genres INNER JOIN Musicians ON Genres.[GenreID] = Musicians.[GenreID]) INNER JOIN Songs ON Musicians.[MusicianID] = Songs.[MusicianID]) ON SkillLevel.[LevelID] = Songs.[LevelID]**

**WHERE SkillLevel = "Expert"**

**ORDER BY Musicians.Name;**

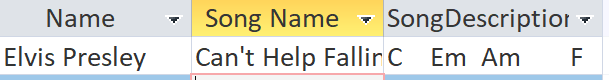


1. List only Elvis Presley’s Songs

**SELECT Musicians.Name, Songs.[Song Name], Songs.SongDescription**

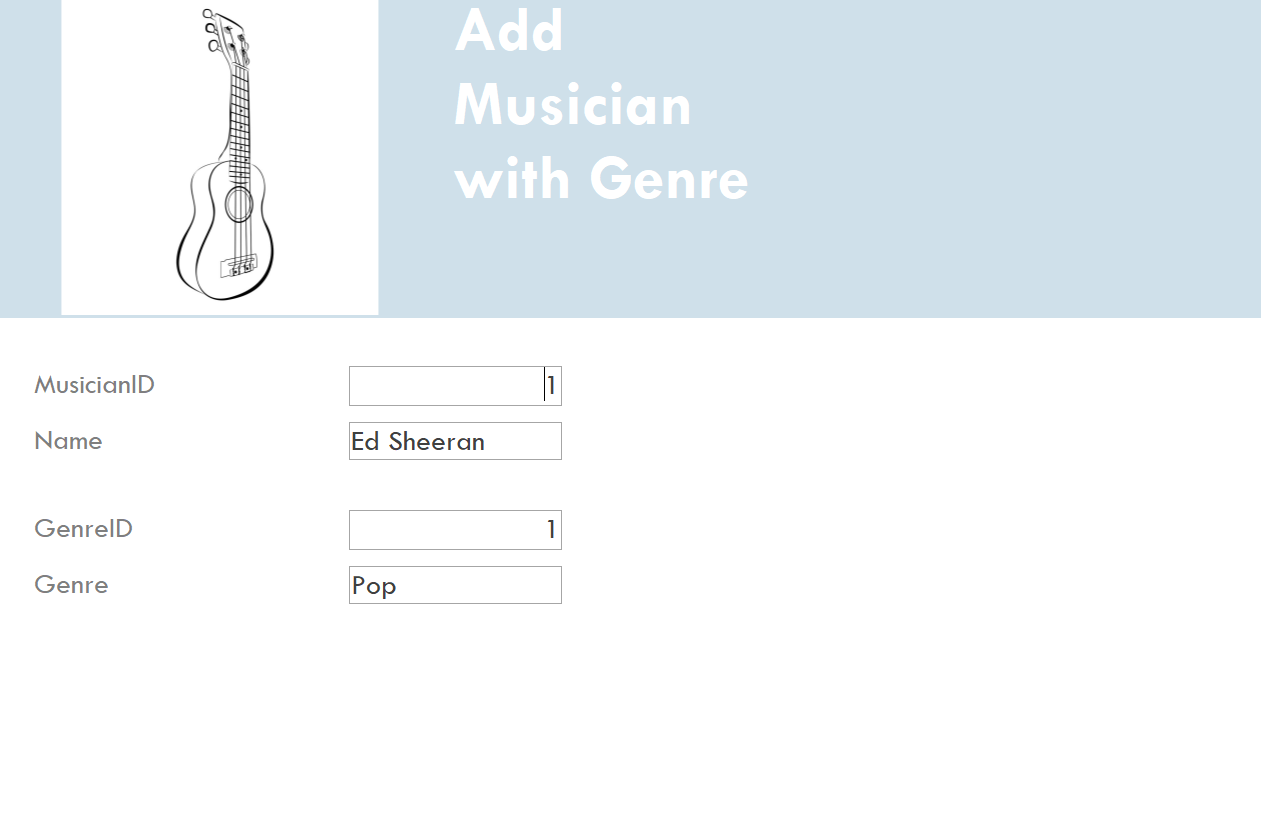
**FROM Musicians INNER JOIN Songs ON Musicians.[MusicianID] = Songs.[MusicianID]**

**WHERE (((Musicians.Name)="Elvis Presley"));**

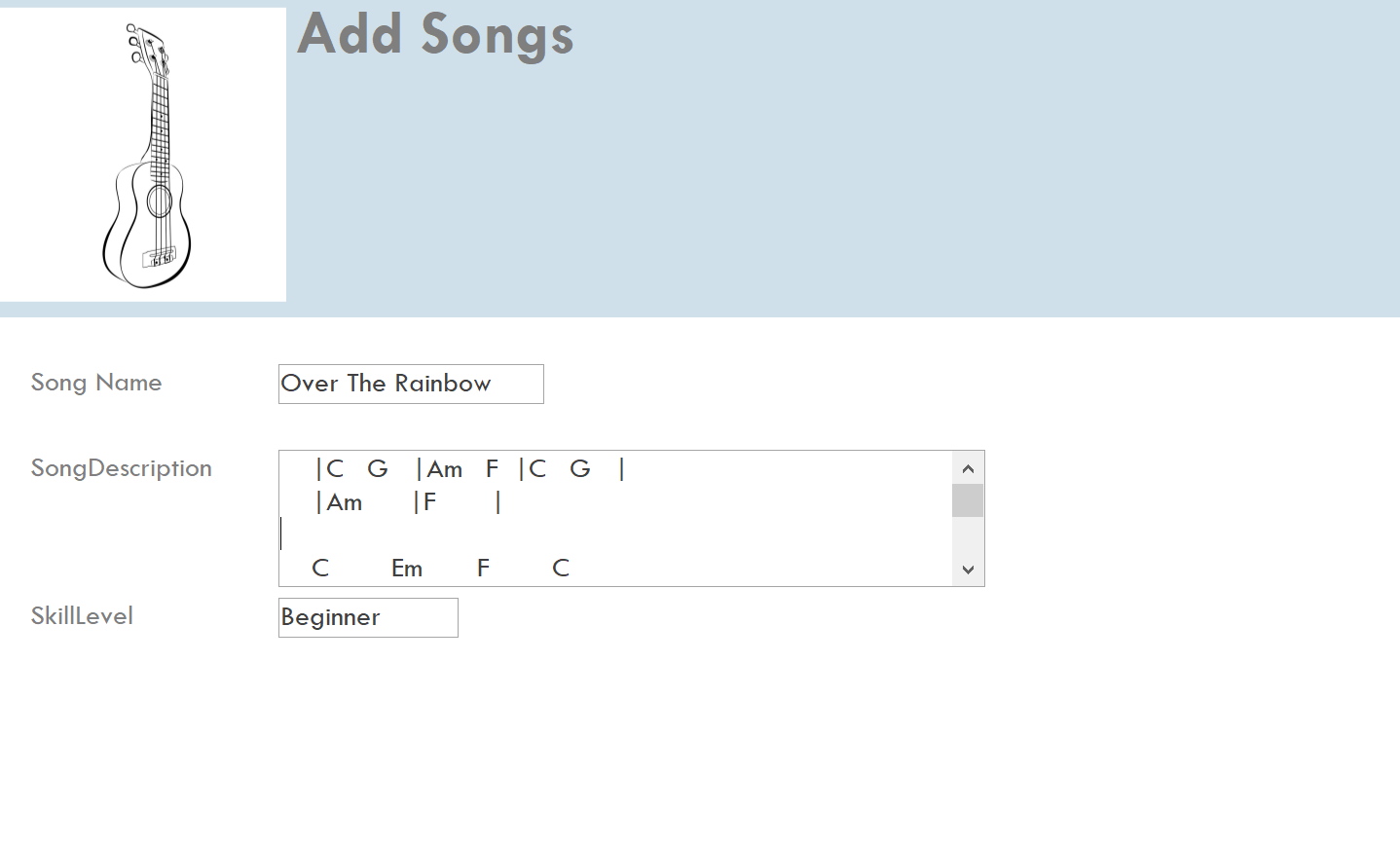


**Forms & Reports**

**Form for adding/editing Musicians with their corresponding Genre**



**Form for Adding Song**



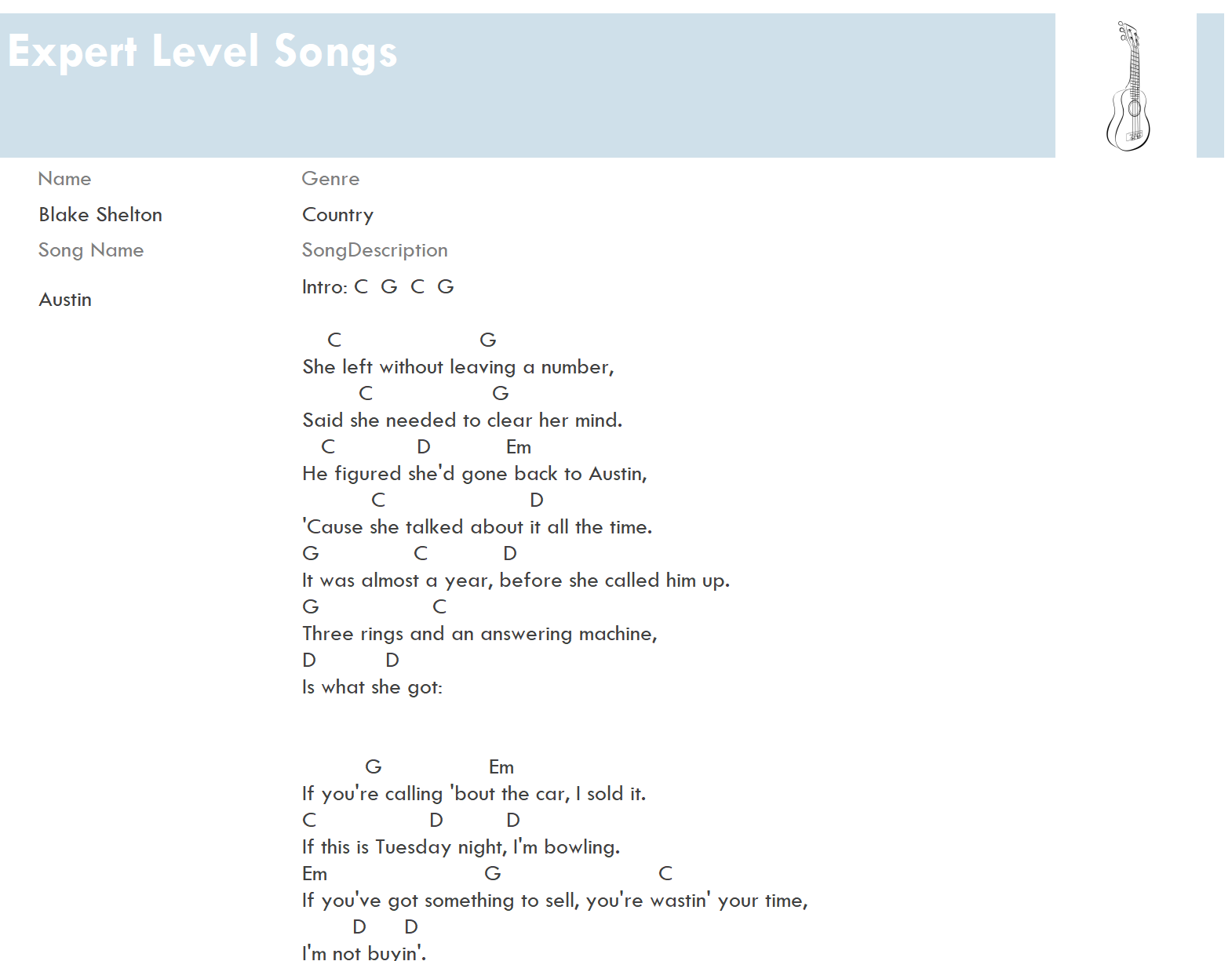
**Form for adding/editing beginner level songs. (query #15)**



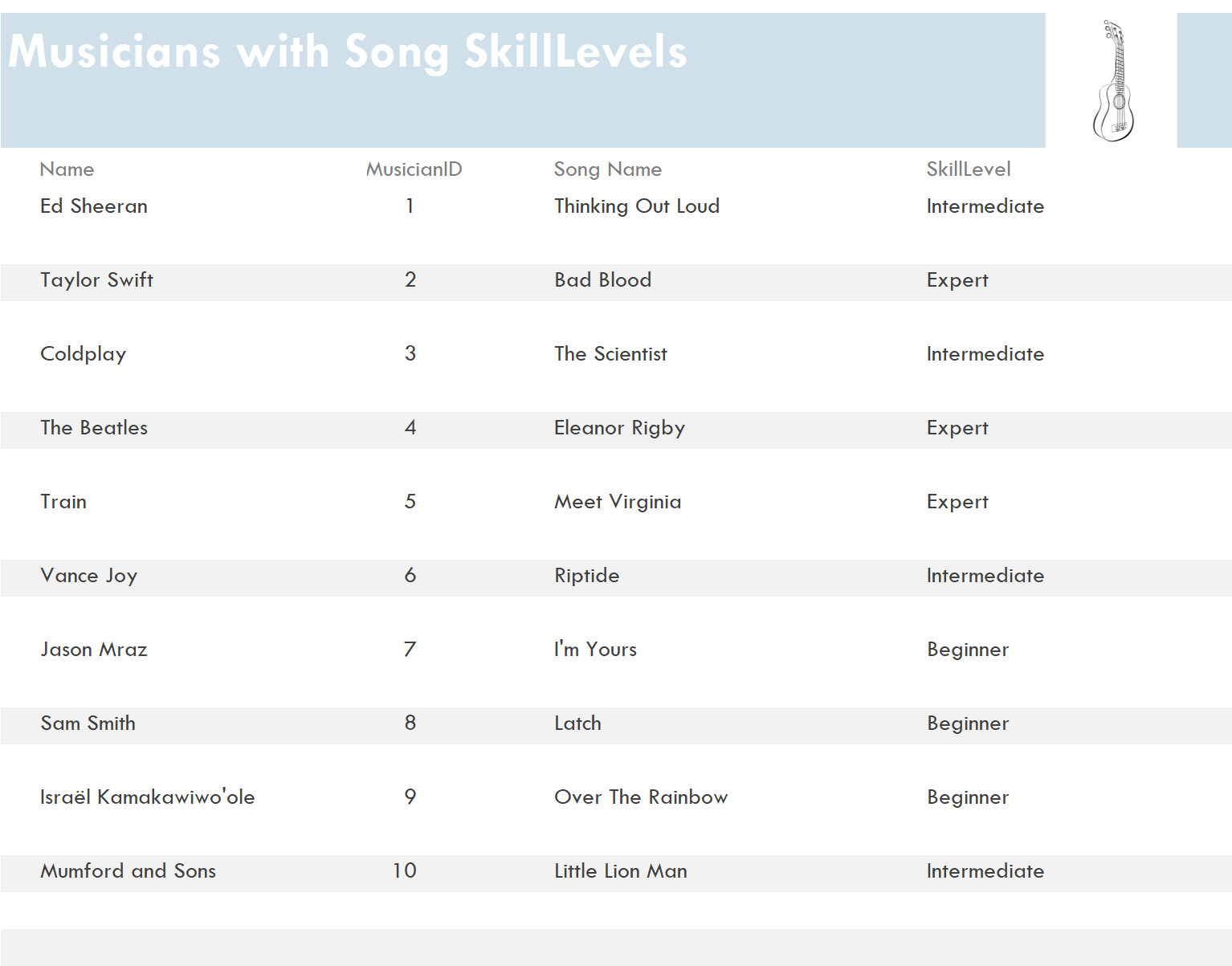
**Report showing all the beginner level songs with Musician’s name and song description. (query #15)**



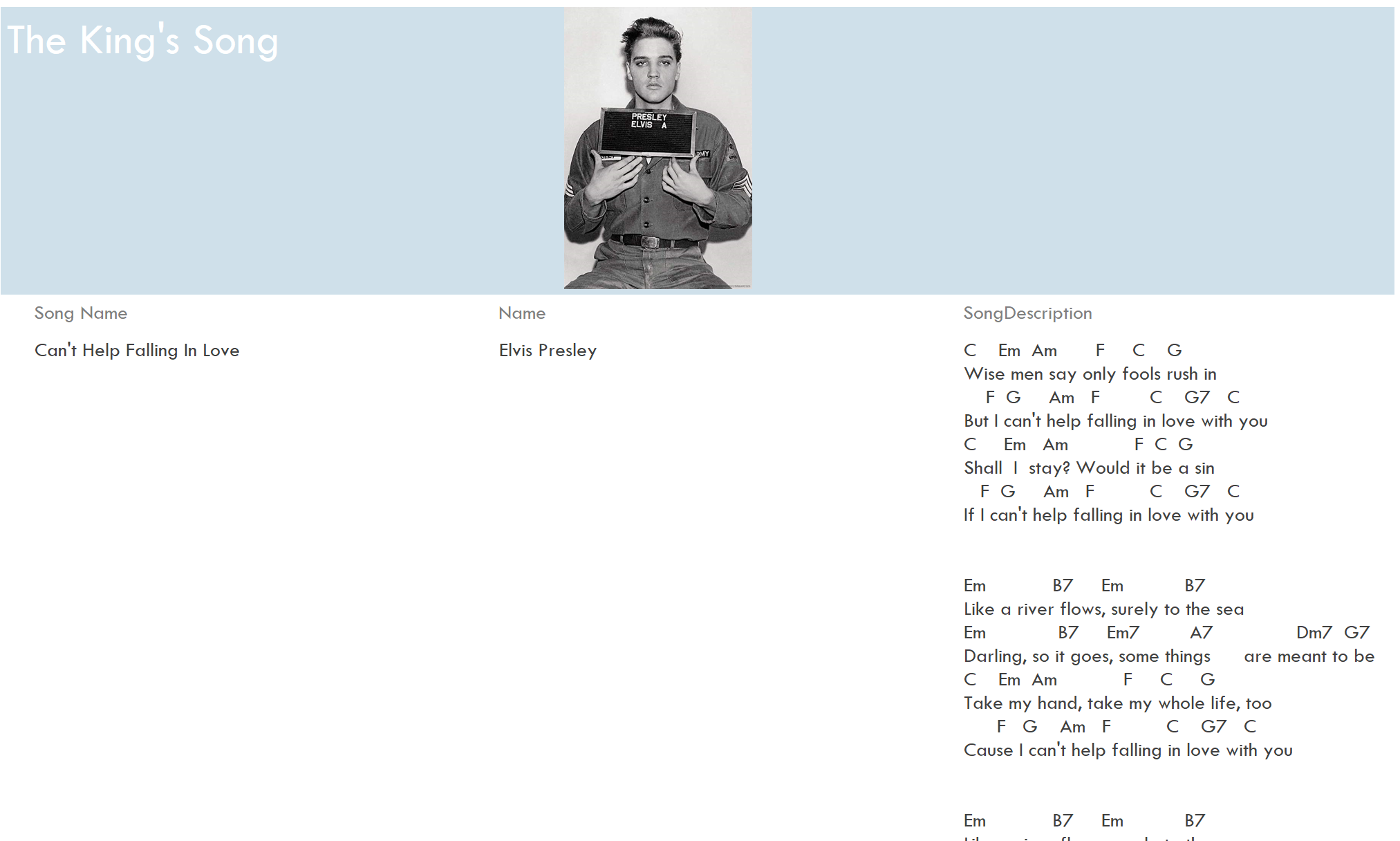
**Report showing all Expert Level songs with Genre, Artist Name, Song Description, and Song Name. (query 16)**



**Report showing Musician’s ID, Name and their Songs with Skill Level**



**Report showing only Elvis Presley’s Songs. (query # 17)**



**Reflection Drew Smith**

Creating this database was a great hands-on interaction for this class. When my partner and I sat down to begin working on this project, we knew this was going to be a challenge. We picked the topic of Ukulele because we both are into instruments and I am currently learning how to play one. This was a hard decision for both of us because we were trying to come up with a topic and nothing stood out for us quickly.

I think this project expanded on a lot that we explored in this course. Using Access allowed for us to really see our content learned this semester come alive. We were able to see real world problems and implementing solutions right before our eyes. I personally have learned how challenging it can be to create a database. A lot of our troubleshooting with issues were resolved by just trying new things until we resolved the specific issue. For example, creating a Musician primary key and it not linking it to the correct data entry for musicians, was difficult to troubleshoot. Even though we were faced with different types of issues throughout this database, we always found a solution to solve these problems.

By having the experience with creating a database, our future employment could greatly benefit ourselves and the company. The experience of this project has further developed my skills that will be used throughout my career. Thank you for allowing us to experience databases firsthand.

**Reflection John Maddux**

Our database project was a tough but rewarding achievement to complete. This project has taught our group on how to create a database and to resolve problems as they arrive. I specifically have learned more how all the parts of a database realistically rely on each other. For example, a user that enters in a specific song title that they may know, our database will need to reflect the exact naming of that song or the user will not receive the data they are searching for. This was going through the back of our minds while creating this database to help prevent many troubleshooting errors that may arise in the future.

But as we moved forward, we did encounter some issues that needed some troubleshooting attention. We used different resources that were available to us to help solve these issues. The sample databases provided in the course materials were an excellent resource to help us troubleshoot some of the issues that we were having, and some provided guidance on our development of our database. This project has expanded my knowledge of databases and with access specifically. I enjoy using access and hope to continue to use it as my career sees fit.