Join and Unions Lab

Lab 1: Inner Join

Right mouse-click on the EntityClasses folder.

Create a new class named **SongGenre** with the following code.

```
#nullable disable
using System.Text;
namespace LINQLab.EntityClasses;
public partial class SongGenre
        public int SongId { get; set; }
        public string SongName { get; set; }
        public string Artist { get; set; }
        public string Album { get; set; }
        public int? GenreId { get; set; }
        public string GenreName { get; set; }
          #region ToString Override
        public override string ToString() {
                  StringBuilder sb = new(1024);
                  sb.AppendLine($"Song: {SongName} ID: {SongId}");
                  sb.AppendLine($" Artist: {Artist}");
sb.AppendLine($" Album: {Album}");
                   sb.AppendLine($" Genre (Genre (G
                   return sb.ToString();
          #endregion
```

Open the **Program.cs** file and replace the code with the following.

```
using LINQLab.EntityClasses;
using LINQLab.RepositoryClasses;

// Declare variables and fill data
List<Song> songs = SongRepository.GetAll();
List<MusicGenre> genres = MusicGenreRepository.GetAll();
List<SongGenre> songGenres = new();

// TODO: Write Your Query Here

// Display the Results
foreach (var song in songGenres) {
   System.Diagnostics.Debug.WriteLine(song);
}
```

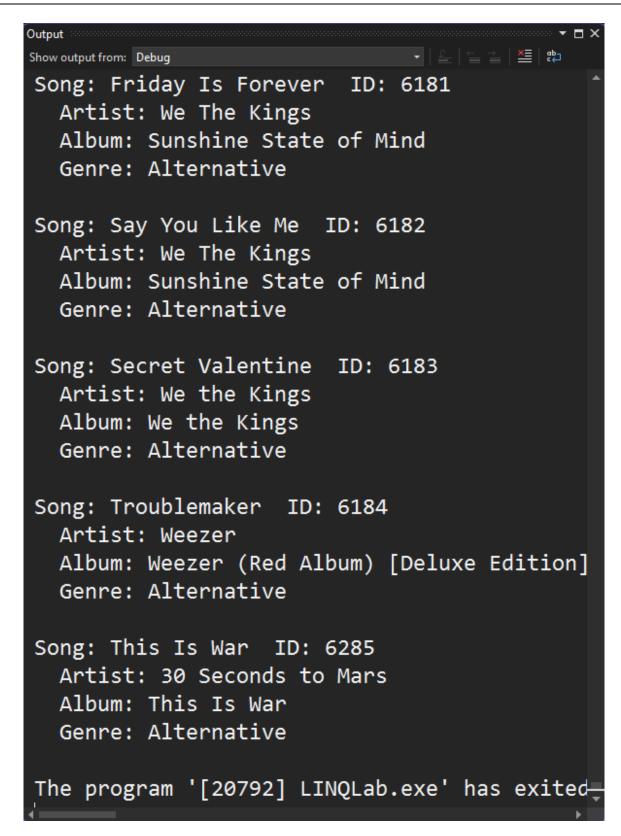
Write a query to join the Song list with the Genre list and create a new SongGenre object with the appropriate data from each of the corresponding properties of the SongGenre object.

Try it Out

Run the application

Select View | Output from the Visual Studio menu

View the results in the Debug window



Lab 2: One to Many

Right mouse-click on the EntityClasses folder.

Create a new class named **GenreSongs** with the following code.

```
#nullable disable
using System. Text;
namespace LINQLab.EntityClasses;
public partial class GenreSongs
 public int GenreId { get; set; }
 public string GenreName { get; set; }
 public List<Song> Songs { get; set; }
  #region ToString Override
  public override string ToString() {
   StringBuilder sb = new(1024);
   sb.AppendLine($"Genre: {GenreName} Genre ID: {GenreId}");
   foreach (var song in Songs) {
      sb.AppendLine($" {song.SongName} - {song.Artist}");
   return sb.ToString();
  #endregion
}
```

Open the **Program.cs** and replace the code with the following:

```
using LINQLab.EntityClasses;
using LINQLab.RepositoryClasses;

// Declare variables and fill data
List<Song> songs = SongRepository.GetAll();
List<MusicGenre> genres = MusicGenreRepository.GetAll();
List<GenreSongs> genreSongs = new();

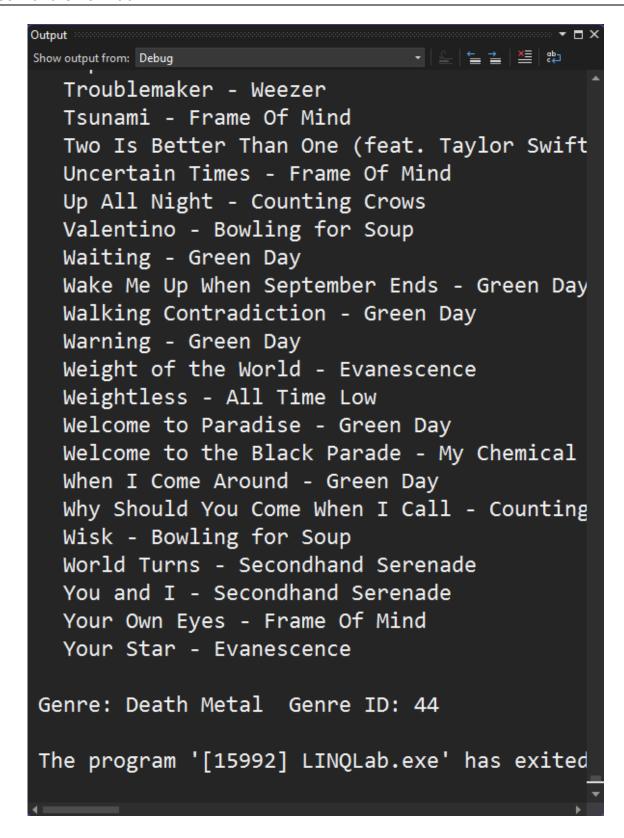
// TODO: Write Your Query Here

// Display the Results
foreach (var song in genreSongs) {
   System.Diagnostics.Debug.WriteLine(song);
}
```

Write a query to get all genres ordering by the *Genreld*. Join these to the songs collection where the *Genreld* equals the *Genreld* in the genre collection. Use the **into** keyword to get the list of songs for each genre. Create a new *GenreSongs* object and put the data from songs and genres into this new object.

Try it Out

Run the application and view the results:



Lab 3: Union using Comparer

Open the **Program.cs** file and replace the code with the following:

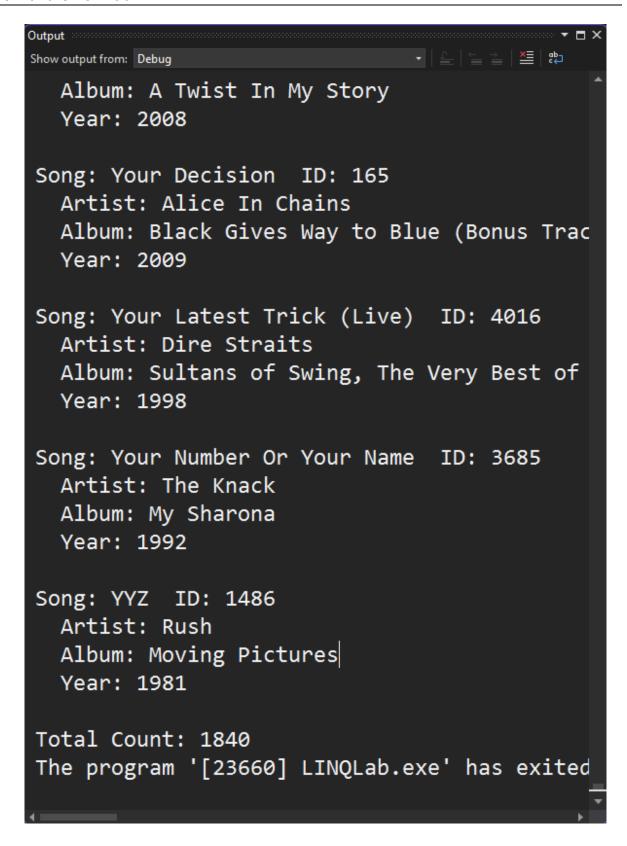
Write a query to union the songs in the two lists together.

Use the **SongComparer** class to ensure no duplicates are allowed.

Order by the song name.

Try it Out

Run the application and view the results:



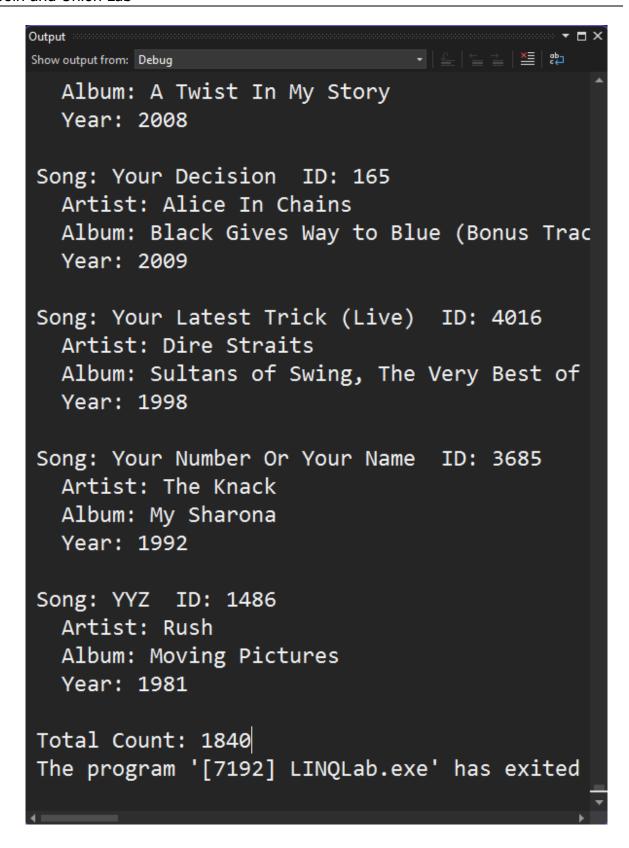
Lab 4: UnionBy

Open the **Program.cs** file and remove the declaration for the SongComparer.

Write a query to union the two lists together using the UnionBy() method and the appropriate lambda expression.

Try it Out

Run the application and view the results:



Join and Unions Lab Answers

Lab 1: Inner Join (Query Syntax)

```
songGenres = (from row in songs
    join genreRow in genres
    on row.GenreId equals genreRow.GenreId
    select new SongGenre {
        SongId = row.SongId,
        SongName = row.SongName,
        Artist = row.Artist,
        Album = row.Album,
        GenreId = row.GenreId,
        GenreName = genreRow.Genre
}).OrderBy(row => row.GenreId).ToList();
```

Lab 1: Inner Join (Method Syntax)

```
songGenres = songs.Join(genres,
    row => row.GenreId,
    genreRow => genreRow.GenreId,
    (row, genreRow) => new SongGenre {
        SongId = row.SongId,
        SongName = row.SongName,
        Artist = row.Artist,
        Album = row.Album,
        GenreId = row.GenreId,
        GenreName = genreRow.Genre
}).OrderBy(row => row.GenreId).ToList();
```

Lab 2: One to Many (Query Syntax)

Lab 2: One to Many (Method Syntax)

Lab 3: Union using Comparer (Query Syntax)

Lab 3: Union using Comparer (Method Syntax)

```
songs = songs1
    .Union(songs2, sc)
    .OrderBy(row => row.SongName).ToList();
```

Lab 4: UnionBy() Method Query Syntax)

Lab 4: UnionBy() Method (Method Syntax)

```
songs = songs1.UnionBy(songs2, row => row.SongId)
    .OrderBy(row => row.SongName).ToList();
```