Grouping Lab

Lab 1: GroupBy() Method

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

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

// Declare variables and fill data
List<Song> songs = SongRepository.GetAll();
List<IGrouping<int?, Song>> list = new();

// TODO: Write Your Query Here


// Loop through each Genre
foreach (var group in list) {
  // Display the GenreId and the Count of Songs
  Console.WriteLine($"GenreId: {group.Key}\tCount:
{group.Count()}");
}

// Pause for Results
Console.ReadKey();
```

Write a query to group all songs by ***GenreId***.

Try it Out

Run the application and the results should look like the following:

```
D:\Training\DotNet6\LINQ\05-Group\03-AggregateGroupBy\Labs-...
GenreId: 38      Count: 83
GenreId: 2       Count: 1830
GenreId: 17      Count: 280
GenreId: 9       Count: 47
GenreId: 43      Count: 156
GenreId: 7       Count: 195
GenreId: 33      Count: 48
GenreId: 14      Count: 12
GenreId: 22      Count: 87
GenreId: 16      Count: 57
GenreId: 4       Count: 28
GenreId: 1       Count: 10
GenreId: 36      Count: 72
GenreId: 19      Count: 14
GenreId: 18      Count: 21
GenreId: 20      Count: 24
GenreId: 15      Count: 3
GenreId: 8       Count: 47
GenreId: 41      Count: 13
GenreId: 24      Count: 49
GenreId: 13      Count: 2
GenreId: 31      Count: 1
GenreId: 28      Count: 1
GenreId: 21      Count: 4
GenreId: 6       Count: 272
GenreId: 37      Count: 1
GenreId: 26      Count: 1
GenreId: 10      Count: 25
GenreId: 25      Count: 37
GenreId: 40      Count: 3
GenreId: 29      Count: 2
GenreId: 12      Count: 3
GenreId: 27      Count: 6
GenreId: 42      Count: 2
GenreId: 11      Count: 1
GenreId: 34      Count: 1
GenreId: 32      Count: 29
GenreId: 23      Count: 1
GenreId: 39      Count: 1
GenreId: 5       Count: 34
```

Lab 2: GroupBy() Method Using Where

Write a query to group all songs by ***GenreId***, but only include those groups where the Count() is greater than or equal to 10.

Order the groups by the Count().

Replace the existing loop with a new loop to iterate over the group and display each ***GenreId*** and the count of all songs within the group.

When displaying the ***GenreId*** and the count, pad the ***GenreId*** two spaces on the left.

```
// Loop through each Genre
foreach (var group in list) {
  // Display the GenreId and the Count of Songs
  Console.WriteLine($"GenreId:
{group.Key?.ToString().PadLeft(2)}\tCount: {group.Count()}");
}
```

Try it Out

Run the application and the results should look like the following:

```
D:\Training\DotNet6\LINQ\05-Group\03-AggregateGroupBy\La...
GenreId:  1      Count: 10
GenreId: 14      Count: 12
GenreId: 41      Count: 13
GenreId: 19      Count: 14
GenreId: 18      Count: 21
GenreId: 20      Count: 24
GenreId: 10      Count: 25
GenreId:  4      Count: 28
GenreId: 32      Count: 29
GenreId:  5      Count: 34
GenreId: 25      Count: 37
GenreId:  9      Count: 47
GenreId:  8      Count: 47
GenreId: 33      Count: 48
GenreId: 24      Count: 49
GenreId: 16      Count: 57
GenreId:  3      Count: 72
GenreId: 36      Count: 72
GenreId: 38      Count: 83
GenreId: 22      Count: 87
GenreId: 43      Count: 156
GenreId:  7      Count: 195
GenreId:  6      Count: 272
GenreId: 17      Count: 280
GenreId: 30      Count: 749
GenreId:  2      Count: 1830
GenreId: 35      Count: 1965
```

Lab 3: Group and Aggregate

Right mouse-click on the **EntityClasses** folder.

Create a new class named **SongStats**

Add code that looks like the following to gather the total songs, minimum and maximum ratings, and the average rating for each group.

```
#nullable disable

using System.Text;

namespace LINQLab.EntityClasses;

public class SongStats {
  #region Constructor
  public SongStats() {
    TotalSongs = 0;
    MinRating = int.MaxValue;
    MaxRating = int.MinValue;
    AverageRating = double.MaxValue;
  }
  #endregion

  public int? GenreId { get; set; }
  public int? TotalSongs { get; set; }
  public int? MinRating { get; set; }
  public int? MaxRating { get; set; }
  public double? AverageRating { get; set; }

  #region ToString Override
  public override string ToString() {
    StringBuilder sb = new(1024);

    sb.AppendLine($"Genre: {GenreId}");
    sb.AppendLine($"  Total Songs: {TotalSongs}");
    sb.AppendLine($"  Minimum Rating: {MinRating}");
    sb.AppendLine($"  Maximum Rating: {MaxRating}");
    sb.AppendLine($"  Average Rating: {AverageRating}");

    return sb.ToString();
  }
  #endregion
}
```

Open the **Program.cs** file and replace all the code to look like the following:

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

// Declare variables and fill data
List<Song> songs = SongRepository.GetAll();
List<SongStats> list = new();

// TODO: Write Your Query Here


// Loop through each SongStat object
foreach (var stat in list) {
  // Display the statistics in Debug window
  // because of the buffer size on console window
  System.Diagnostics.Debug.WriteLine(stat);
}
```

NOTE: You must use the Debug.WriteLine() method because the results that come out are too much for the console window buffer.

Write a query to group all songs by ***GenreId***.

Create a new SongStats object each time through the iteration gathering the ***GenreId***, and setting the ***TotalSongs***, ***MinRating***, ***MaxRating***, and ***AverageRating*** properties.

Try it Out

Run the application

Select **View | Output** from the Visual Studio menu and view the results in the Debug console as shown in the following screenshot.

```
Output
Show output from: Debug

Genre: 32
  Total Songs: 29
  Minimum Rating: 40
  Maximum Rating: 80
  Average Rating: 51.724137931034484

Genre: 23
  Total Songs: 1
  Minimum Rating: 60
  Maximum Rating: 60
  Average Rating: 60

Genre: 39
  Total Songs: 1
  Minimum Rating: 80
  Maximum Rating: 80
  Average Rating: 80

Genre: 5
  Total Songs: 34
  Minimum Rating: 40
  Maximum Rating: 40
  Average Rating: 40

The program '[12200] LINQLab.exe' has exited
```