iTunes Playlist Comparer

Brandon Carroll
ISys 540
4/13/2010

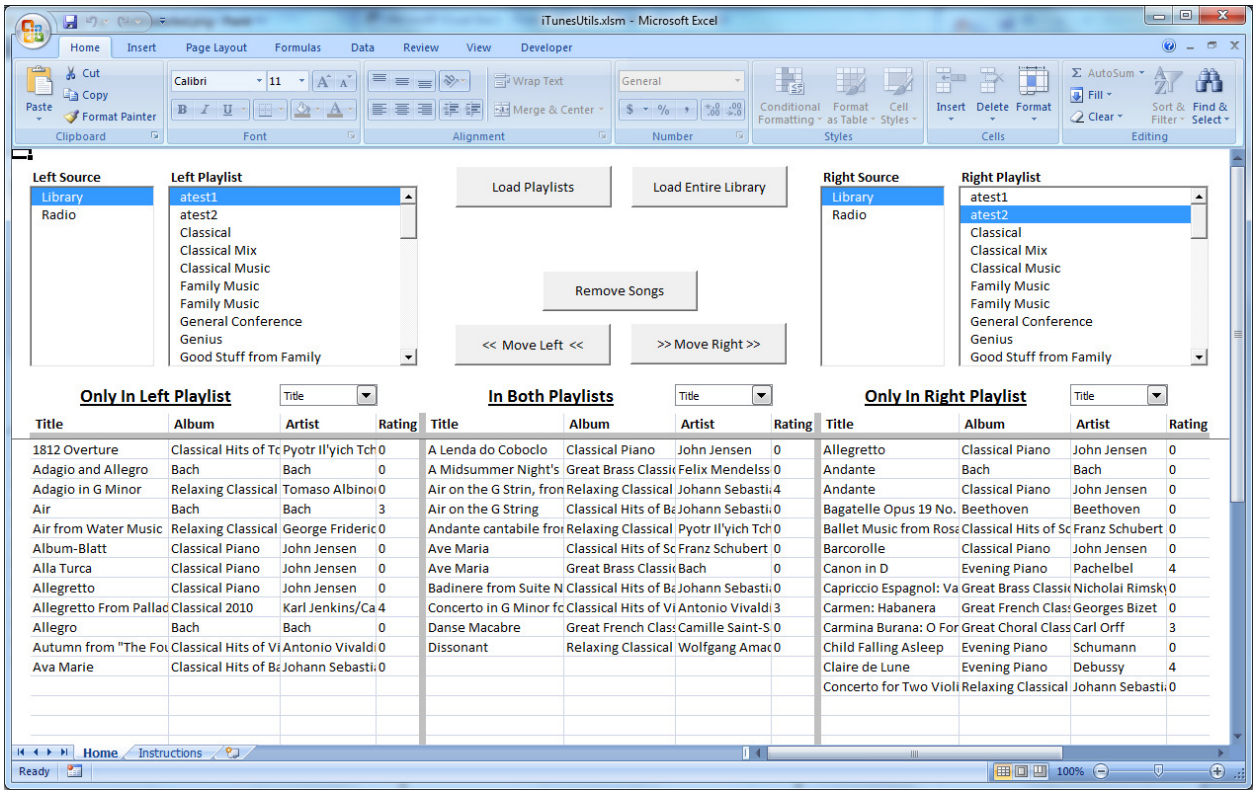
Executive Summary

Motivation

For the past several years, I have wished that iTunes had the capability to compare (or perform a "diff" on) two playlists. There are many situations I have encountered where the ability to see which songs are common between two playlists and which songs are different would be useful. For instance, I have often wanted to verify that all the songs I wanted in a playlist actually got added to it. To do this, it would be a lot easier for me to scroll through a list of songs that are *not* in the playlist because I would notice the ones that were out of place. However, iTunes only lets you view the songs that *are* in the playlist, and it is a lot harder to notice songs that aren't there but should be because nothing is out of place. This has become even more of a problem for me with the introduction of the newer iPod Touches and iPhones because their solid-state drives have smaller capacities than older iPods and are too small to hold my entire library. Thus, I have to pick and choose which songs get put on my iPod and would often like to see the list of songs that are *not* in the playlists that get synced to it as I make my decisions.

Solution

For my project, I used Excel and VBA to write a utility that allows the user to compare and edit playlists in iTunes. When the user selects playlists on the left and right sides, the worksheet sorts the songs from the playlists into three columns. The left and right columns contain the songs that are only in the left or right playlist, respectively. The middle column lists the songs that are in both playlists. The user can sort the three columns by any of the four fields displayed in the list. Furthermore, the user can select and move songs from one column to another, or remove them from the playlists entirely. The worksheet uses the COM interface exposed by iTunes to propagate these changes into the actual playlists in the iTunes library. The user interface is shown in the image below.



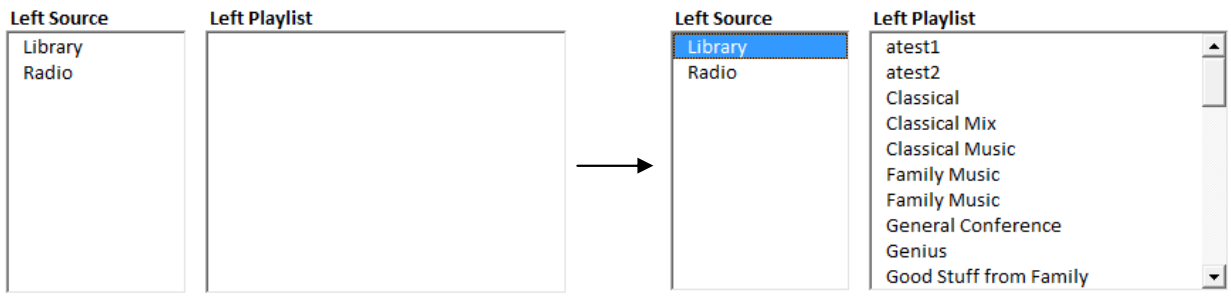
User Interface

Loading the iTunes library

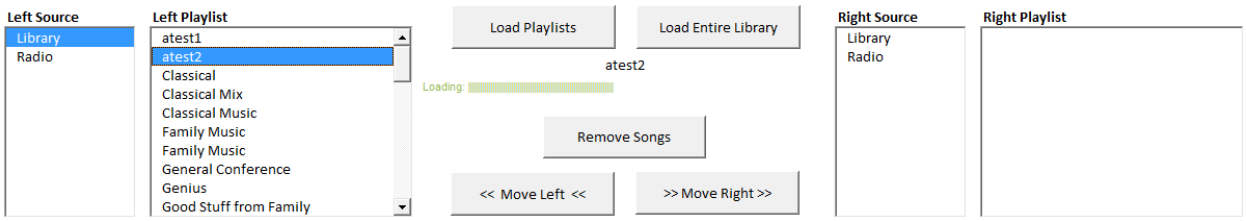
There are two different methods to load data from the iTunes library represented by the "Load Playlists" and "Load Entire Library" buttons at the top of the UI. The "Load Playlists" button loads information about all the available sources and playlists in iTunes, but not information about the tracks in the playlists. Because of this, it executes very quickly. The tradeoff is that the information about the songs in a playlist has to be loaded on the fly when that playlist is selected for the first time, introducing periodic delays depending on the size of the playlist (a playlist with about 2400 songs took 17 seconds to load on my computer). The "Load Entire Library" button loads all of the data for the tracks up front in exchange for not having delays later when you click on a playlist for the first time. Depending on the size of the library, it can take several minutes to load. Since it is fast, the worksheet automatically loads the source and playlist data when opened.

Selecting and comparing playlists

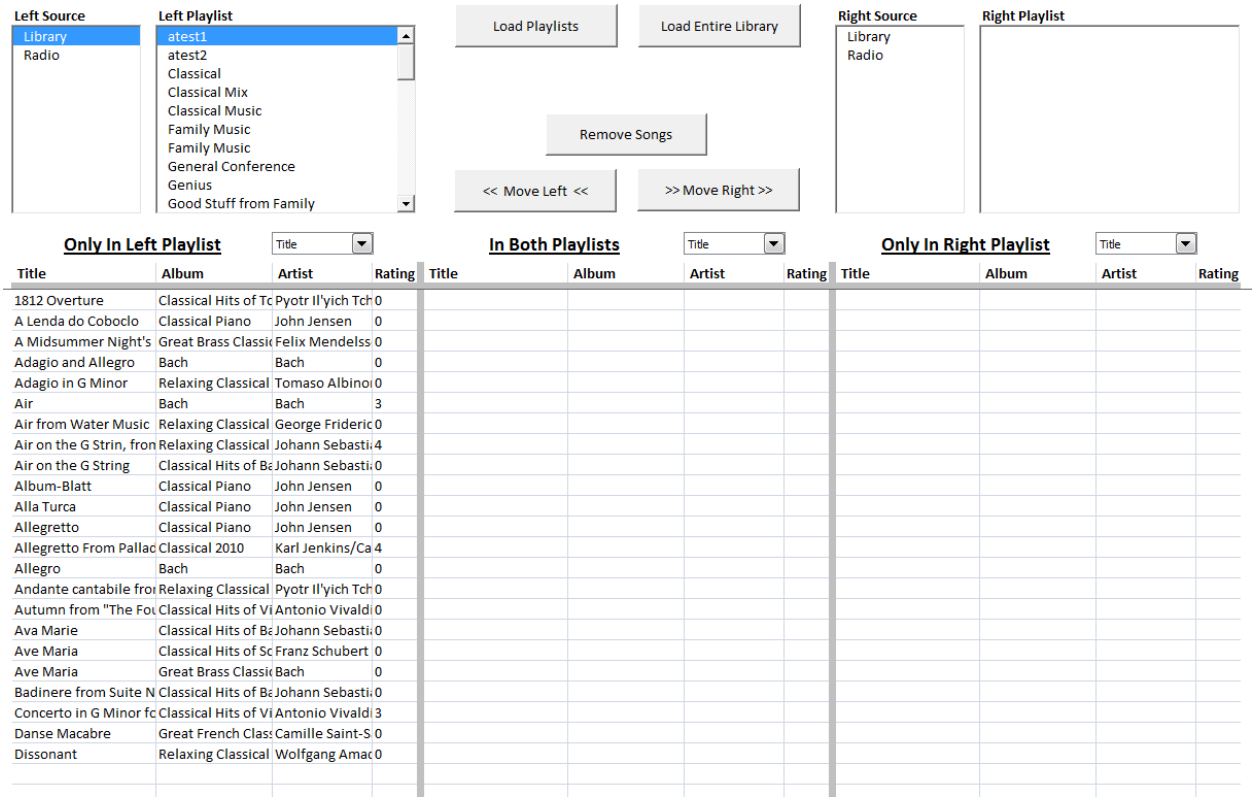
The playlists are selected in the "Left Source", "Left Playlist", "Right Source", and "Right Playlist" boxes at the top left and top right of the screen. Typically, the user will want to select the "Library" source for the main iTunes library. However, the iTunes radio as well as any iPods that are currently plugged into the computer will also show up as different sources (each with their own set of playlists). Once a source has been selected, the corresponding playlist box is populated with all the playlists available from that source, as shown below:



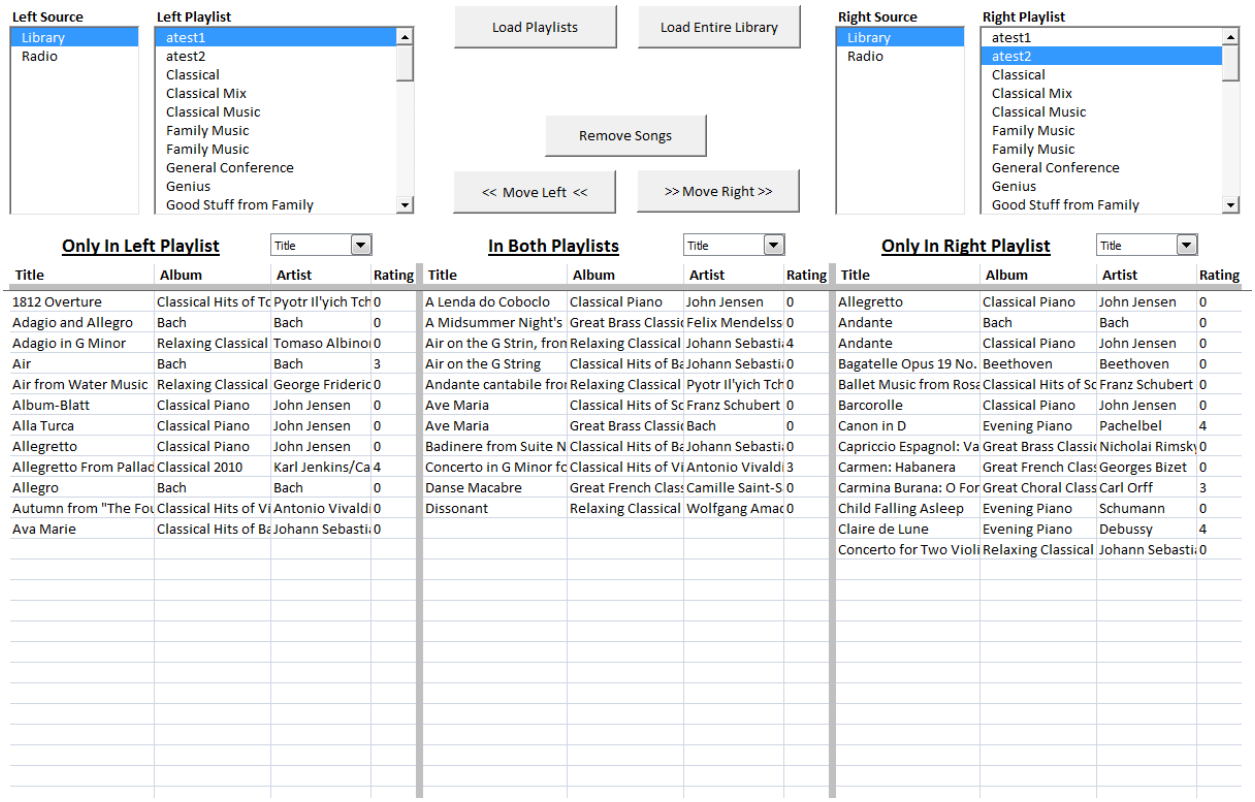
A playlist can then be selected from the playlist box. If the data for that playlist has not yet been loaded, there may be a delay (depending on the length of the playlist) while the data is loaded. During the delay, the name of the playlist being loaded and a green progress bar will appear in the middle of the top pane, indicating how long it will take to load. These can be seen in the screenshot below:



Once a playlist has been loaded, its data is cached in a hidden sheet so that there will no longer be any more loading delays when that playlist is selected (unless the user flushes the cache by clicking the "Load Playlists" button again). Once selected, the playlist's songs will show up in the pane below:

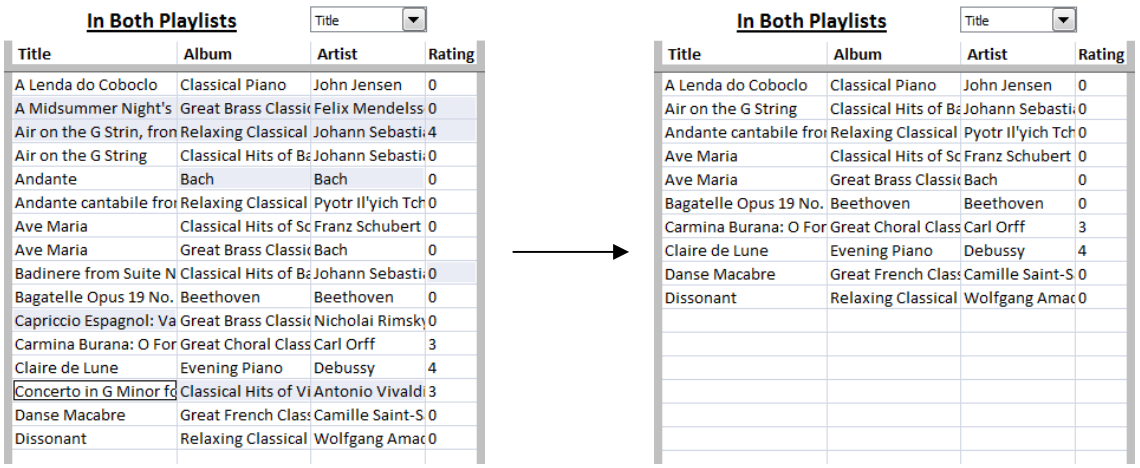


To compare the playlist against another, simply select the other playlist on the right side and the songs will be sorted into the appropriate columns:

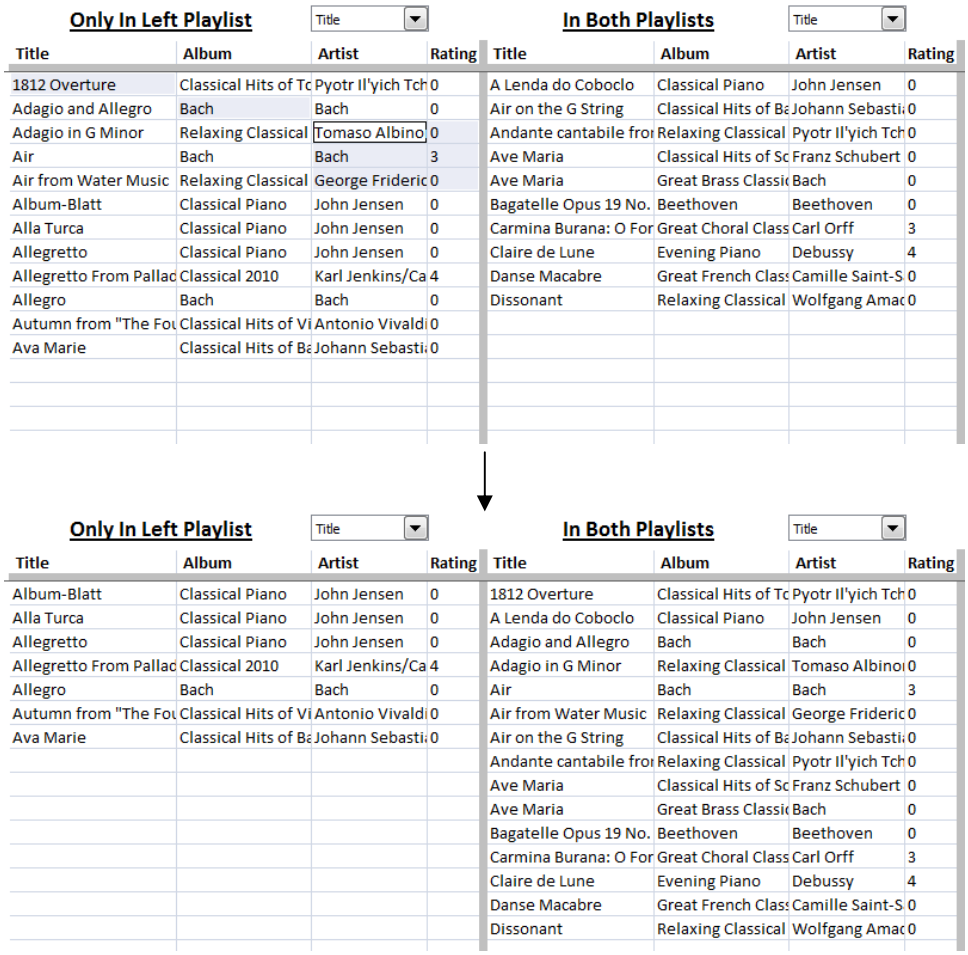


Editing playlists

The “Remove Songs”, “Move Left”, and “Move Right” buttons are provided to edit the playlists that have been selected. Before pressing any of these buttons, the user should select one or more songs to perform the action on in one of the three panes below. To select a song, simply select at least one of the cells containing information for that song. Multiple songs can be selected by clicking and dragging, or by holding the ctrl key while clicking to make a disjoint selection. The images below show a selection made before pressing the “Remove Songs” button and the result afterward.



Note that all 6 rows in which at least one cell was selected were removed from the playlists. Selections are made in the exact same manner prior to using the “Move Left” or “Move Right” buttons. If songs are moved from the right column to the middle column, they will get added to the left playlist (which they were not a part of before). If moved from the middle column to the left column, the songs will get removed from the right playlist so that they are only in the left one. The images below demonstrate selecting moving songs from the left column to the middle by clicking the “Move Right” button:



Only In Left Playlist | Title | **In Both Playlists** | Title

Title	Album	Artist	Rating
1812 Overture	Classical Hits of Tc	Pyotr Il'yich Tch	0
Adagio and Allegro	Bach	Bach	0
Adagio in G Minor	Relaxing Classical	Tomaso Albino	0
Air	Bach	Bach	3
Air from Water Music	Relaxing Classical	George Frideric	0
Album-Blatt	Classical Piano	John Jensen	0
Alla Turca	Classical Piano	John Jensen	0
Allegretto	Classical Piano	John Jensen	0
Allegretto From Pallad	Classical 2010	Karl Jenkins/Ca	4
Allegro	Bach	Bach	0
Autumn from "The Fou	Classical Hits of Vi	Antonio Vivaldi	0
Ava Marie	Classical Hits of Ba	Johann Sebasti	0

Title	Album	Artist	Rating
A Lenda do Coboclo	Classical Piano	John Jensen	0
Air on the G String	Classical Hits of Ba	Johann Sebasti	0
Andante cantabile fro	Relaxing Classical	Pyotr Il'yich Tch	0
Ave Maria	Classical Hits of Sc	Franz Schubert	0
Ave Maria	Great Brass Classic	Bach	0
Bagatelle Opus 19 No.	Beethoven	Beethoven	0
Carmina Burana: O For	Great Choral Class	Carl Orff	3
Claire de Lune	Evening Piano	Debussy	4
Danse Macabre	Great French Class	Camille Saint-S	0
Dissonant	Relaxing Classical	Wolfgang Amac	0

Only In Left Playlist | Title | **In Both Playlists** | Title

Title	Album	Artist	Rating
Album-Blatt	Classical Piano	John Jensen	0
Alla Turca	Classical Piano	John Jensen	0
Allegretto	Classical Piano	John Jensen	0
Allegretto From Pallad	Classical 2010	Karl Jenkins/Ca	4
Allegro	Bach	Bach	0
Autumn from "The Fou	Classical Hits of Vi	Antonio Vivaldi	0
Ava Marie	Classical Hits of Ba	Johann Sebasti	0

Title	Album	Artist	Rating
1812 Overture	Classical Hits of Tc	Pyotr Il'yich Tch	0
A Lenda do Coboclo	Classical Piano	John Jensen	0
Adagio and Allegro	Bach	Bach	0
Adagio in G Minor	Relaxing Classical	Tomaso Albino	0
Air	Bach	Bach	3
Air from Water Music	Relaxing Classical	George Frideric	0
Air on the G String	Classical Hits of Ba	Johann Sebasti	0
Andante cantabile fro	Relaxing Classical	Pyotr Il'yich Tch	0
Ave Maria	Classical Hits of Sc	Franz Schubert	0
Ave Maria	Great Brass Classic	Bach	0
Bagatelle Opus 19 No.	Beethoven	Beethoven	0
Carmina Burana: O For	Great Choral Class	Carl Orff	3
Claire de Lune	Evening Piano	Debussy	4
Danse Macabre	Great French Class	Camille Saint-S	0
Dissonant	Relaxing Classical	Wolfgang Amac	0

Sorting the columns

The three columns displaying the result of the comparison can be sorted by any of the 4 fields displayed by selecting it from the dropdown box next to the column title. The three columns can be sorted independent of each other and operations to add or remove songs can still be performed regardless of the sorting order. The images below show the middle column sorted by album name and by artist name:

In Both Playlists Album

Title	Album	Artist	Rating
Adagio and Allegro	Bach	Bach	0
Air	Bach	Bach	3
Bagatelle Opus 19 No.	Beethoven	Beethoven	0
Air on the G String	Classical Hits of Ba	Johann Sebasti	0
Ave Maria	Classical Hits of Sc	Franz Schubert	0
1812 Overture	Classical Hits of Tc	Pyotr Il'yich Tch	0
A Lenda do Coboclo	Classical Piano	John Jensen	0
Claire de Lune	Evening Piano	Debussy	4
Ave Maria	Great Brass Classic	Bach	0
Carmina Burana: O For	Great Choral Class	Carl Orff	3
Danse Macabre	Great French Class	Camille Saint-S	0
Adagio in G Minor	Relaxing Classical	Tomaso Albino	0
Air from Water Music	Relaxing Classical	George Frideric	0
Andante cantabile fro	Relaxing Classical	Pyotr Il'yich Tch	0
Dissonant	Relaxing Classical	Wolfgang Amac	0

In Both Playlists Artist

Title	Album	Artist	Rating
Adagio and Allegro	Bach	Bach	0
Air	Bach	Bach	3
Ave Maria	Great Brass Classic	Bach	0
Bagatelle Opus 19 No.	Beethoven	Beethoven	0
Danse Macabre	Great French Class	Camille Saint-S	0
Carmina Burana: O For	Great Choral Class	Carl Orff	3
Claire de Lune	Evening Piano	Debussy	4
Ave Maria	Classical Hits of Sc	Franz Schubert	0
Air from Water Music	Relaxing Classical	George Frideric	0
Air on the G String	Classical Hits of Ba	Johann Sebasti	0
A Lenda do Coboclo	Classical Piano	John Jensen	0
1812 Overture	Classical Hits of Tc	Pyotr Il'yich Tch	0
Andante cantabile fro	Relaxing Classical	Pyotr Il'yich Tch	0
Adagio in G Minor	Relaxing Classical	Tomaso Albino	0
Dissonant	Relaxing Classical	Wolfgang Amac	0

Implementation Documentation

My project ended up having nearly 1000 lines of VBA code and probably took around 40 hours (I didn't keep track).

Interacting with iTunes

I queried and edited playlist data in iTunes via the COM interface that it exposes. Below are sample lines of code used to open iTunes and to query a track's name:

```
Set iTunes = CreateObject("ITunes.Application")
trackName = iTunes.sources(sourceIndex).playlists(playlistIndex).tracks(i).name
```

Caching library data

Since it takes a long time to iterate through lots of tracks through the iTunes COM interface, I cached data locally in the workbook to speed things up (other than the first time a playlist is loaded). I originally wanted to store it all in arrays. However, I needed to be able to sort the data as well and Excel does not have any built-in functions for sorting arrays. Although I could have easily written an implementation of the quicksort algorithm, I did not want the performance hit of using non-built-in functions. Thus, I stored the cached data on a hidden worksheet where I could use the built-in functionality to sort ranges of cells. Another hidden sheet contained the data for the sources and playlists. Since I was sorting the data for my own purposes, I had to store the corresponding indices into the actual iTunes library and sort them along with the names and other data.

The progress bar

I used a fairly simple solution for creating a progress bar in a cell that I found online. I simply used the REPT function to repeatedly output a pipe character and set the font to one where the pipe characters fit closely enough together to look like a bar. The formula for the progress bar cell is given below:

```
=IF(PlaylistSheet!A1 = "", "", "Loading: " & REPT("|", 170 * PlaylistSheet!A1 / PlaylistSheet!A2))
```

The playlist comparison algorithm

In preparation for comparing playlists, I sorted my cached data by all 4 fields that I looked at (Title, Album, Artist, Rating). To compare, I simply started at the beginning of the sorted list and asked if all 4 fields were the same for the first item in each list. If they were, that one went into the middle column and I incremented the index for both lists. If they weren't, the one that would come first in sorting order went into the column for that playlist and I *only* incremented the index for that list. I would then compare the two items at the indices again and repeat the same process. When one of the indices reached the end of its playlist, I would add all the remaining songs in the other playlist to the column for that playlist.

Allowing the user to sort the comparison columns by hiding data in cells

To allow the user to sort the columns, I needed some way to keep track of the indices into my cached data for each song where those indices would get sorted with the data being displayed. I did not, however, want the user to be able to see these indices (which would be meaningless to him anyway). My solution was to hide the indices in the formulas for the cells. I wrote a function that could take up to three parameters and did nothing but return the leftmost one:

```
Function HideData(visibleString As String, hiddenData As Variant, Optional hiddenData2 As Variant) As String
    HideData = visibleString
End Function
```

I could then set the indices as parameters to the function and parse them out later. For instance, if I wanted to display the song title "Ave Maria" with index 3 for the left playlist and index 5 for the right one, I would set the cells formula to '=HideData("Ave Maria",3,5)'. That way, the cell simply displayed "Ave Maria" but still had the indices hidden within its formula. When the user later selected the cell and performed an operation on it, I could parse out the 3 and 5 so that I would know what song it was in my cached data and obtain the indices into the actual iTunes library.

Protecting the formatting and hidden data on the main page

Since I hide data in the columns on the home page of my workbook, I did not want the user inadvertently typing things over the contents of the cells. I also did not want anyone messing up the formatting of that page or the instructions page. For instance, I have the top pane (containing all the buttons, controls, and column headings) frozen in place so that the controls are always accessible even when the user is scrolling through the listed songs. To prevent the user from inadvertently messing things like that up, I protected both the home sheet and the instructions sheet with the password, "password". I noted this password on the instructions page of the workbook so that anyone who later wants to intentionally modify my work can. I thought about protecting the VBA code as well, but decided against it since normal people don't inadvertently stumble upon it and I wanted those who know how to be able to modify it.

Difficulties Encountered

User Interface Issues

One of the hard parts of this project was simply trying to account for all the different things a user could do and handle the errors so that my workbook didn't crash. For most of these cases, I simply popped up a message box explaining the problem. For instance, when the user selected the exact same playlist on left and right, the "Move Left" and "Move Right" operations no longer make sense. They also don't make sense when no playlists or only one playlist has been selected. The "Remove Songs" operation makes no sense when a playlist has not been selected. I also had to check to make sure that all of the songs a user selected for an operation were in the same column and pop up appropriate messages for that.

Playlists that cannot be edited

ITunes does not allow the user to make changes to certain playlists, like auto-generated playlists. Attempts to change these playlists would throw an error that I had to catch so that I could display an appropriate message to the user and cancel the operation.

ITunes also does not allow songs to be added to or from playlists on an iPod through the COM interface. However, it does not throw any errors when attempts are made to do so. It just silently ignores them. I have not yet been able to find a solution to this problem and put a warning about it on the instructions sheet in my workbook. It is particularly nasty because my code thinks the changes have been made and updates its cached data as if they had been, thus causing discrepancies between my cached data and the actual iTunes library. Such discrepancies could later cause *different* songs to be moved around in iTunes instead of the ones the user selected in the workbook because the indices could be off. There may be a way to determine if a source is an iPod or not and thus prevent this from happening, but I have not had time to find it yet.

Removing songs from a playlist shifts all subsequent song indices

When I first wrote the code to remove songs from a playlist, I noticed that some different songs would get removed than the ones I had selected on my worksheet. When I tracked the bug down, it turned out that I was not shifting my cached indices in the same way iTunes does when a song gets removed. Thus, when the user selected a group of songs to remove, I would have to iterate through them one at a time removing them and then shifting all my stored indices that came after that song up one to keep them in sync with iTunes.

Workbook size inflation

Since my workbook imports lots of data from iTunes and stores it in potentially huge ranges in its sheets, its size became an issue. At one point, it the size of the workbook file got up to about 5 MB. When I noticed this, I wrote an event that would clear out all the cached data prior to saving the workbook. I also prompted the user to make sure that was OK first. However, the size of the workbook did not change. After researching some online, I realized that Excel still thought the UsedRange for the sheets was huge. To shrink it back down, I ended up having to delete (not just clear) the rows and columns where I had formerly stored data before saving the workbook. Once I got this working well, the

size of my workbook shrunk from 5 MB to about 90 KB (over 55 times smaller). There is no point to having old cached data in the workbook anyway since it all just gets refreshed when it is opened again.