Overview: Use Pandas data structures in Python to analyze data about artists, users, play counts, and tags from last.fm.

Learning objectives: Gain experience using Pandas to process and clean datasets, build data structures to support analysis, and perform data analytics to answer specific questions.

Project specification:
For this project, you will analyze the same data set from last.fm that was used in Project 2. A zip file containing the data is made available by the Grouplens research team at the University of Minnesota. To download it, go to the page: http://grouplens.org/datasets/hetrec-2011/
and download the file: hetrec2011-lastfm-2k.zip

For Project 3 you will use the following .dat files: artists.dat, user_artists.dat, and user_taggedartists.dat. Details about these files are given in the readme.txt file and in Project 2.

Reading the Data
Your program should start by reading the data contained in the .dat files described above and storing the data in pandas data structures. For this assignment, read the files directly into Pandas DataFrames:

```
artists_df = pd.read_csv('artists.dat', sep='\t')
```

Cleaning and Indexing the Data
After initially reading the data into DataFrames, you will need to filter out the "bad" records that were outlined in Project 2, and you may wish to create additional Pandas data structures (Series and other DataFrames) that include indexes on the data.

In order to receive full credit, you should use Pandas data structures and features for the core functionality of Project 3rather than building your own data structures from built-in Python data types such as dicts and lists. There may be times where you need to use a list or dict for some small purpose – that is okay – but the core functionality should be accomplished using Pandas.

For most of the problems, you are asked to group and aggregate the data in various ways (e.g. sum the play counts for artists across all users). Often there are several ways you can use PANDAS for these situations, and you will have to make decisions about using Series or DataFrames, indexes or groupby, and how to configure the data structures to support merge and join operations.

For the data in artists.dat, you may find it helpful to create an index that maps artist ids to artist names. If you use a DataFrame with an artist ids as the index, you could later use pd.merge() to combine it with result sets to easily add the artist names to be output. Or you could use a Series or DataFrame without an index as a simple lookup table. Since I want you to get practice with Pandas, either of these approaches is okay – just don't use a regular Python dict!

For the data in user_artists.dat, you could create Series or DataFrame with a two-level hierarchical index with user id (level 0) and artist id (level 1). However, you could also put all three columns of data (uid, aid, playcount) into a DataFrame with a default integer index and then use groupby to aggregate the data as needed. Again, these approaches and others that use Pandas features are okay – just don't build a list of dicts as was suggested for Project 2.

For the data in user_taggedartists.dat, after reading the data into an initial DataFrame, you should do some filtering to make sure that your "final" DataFrame contains only records where: 1) the artist id appeared in the artists.dat file, and 2) the year is 2005 or greater. Filtering for the year can be done using Boolean

indexing as we did in class examples. One approach to filtering the missing artist ids is to use pd.merge(), but there are other good solutions as well.

Data Analytics and Specific Queries

For the data analytics, you will re-answer a subset of the questions from Project 2 as shown below. For each question, clearly comment the section of your code that addresses the question, and in your comments, provide a brief description of the approach that you took. Again, it is okay to use a list or dict for some small purpose, but the core functionality should be accomplished using Pandas.

There is no "user interface" for this project. I will run your program and it should produce output all at once for the questions below.

In your output for each question, print:

- A blank line
- A line of 40 exclamation points "!!!!!!!!"
- Another blank line
- A line with the question number and the brief description of the question
- Another blank line
- Then print the output for that question.

1. *Who are the top artists?* Print a list of the 10 artists with the most song plays (across all users), sorted by number of song plays. For each of the top 10, print the artist id, the artist name, and the total number of song plays for that artist. As an example, I created the output below by generating results in a DataFrame, merging it with the artists names, and then printing it (e.g. `print results_df`). Other output formats are acceptable as long as they are easy to read.

```
                   name  playcount
aid
289      Britney Spears    2393140
72         Depeche Mode    1301308
89            Lady Gaga    1291387
292  Christina Aguilera    1058405
. . .
```

2. *Who are the top users?* Print the 10 user ids with the most song plays (across all artists), sorted by number of song plays. For each, print the user id and total number of song plays for that user.

```
      playcount
uid
757      480039
2000     468409
1418     416349
1642     388251
. . .
```

3. *What artists have the most listeners?* Print a list of the 10 artists with the highest number of users who have listened to at least one song by that artist, sorted by number of listeners. For each artist, print the artist name, artist id, and the number of distinct users who have listened to a song by that artist.

```
                   name  numlisteners
aid
89            Lady Gaga           611
289      Britney Spears           522
288             Rihanna           484
227         The Beatles           480
. . .
```

4. *What artists with at least 50 listeners have the highest average number of plays per listener?* Compute the average number of plays per listener for each artist that has at least 50 listeners. In your output, print the average with two decimal places.

```
Depeche Mode (72) 4614.57
Britney Spears (289) 4584.56
In Flames (503) 3539.52
Duran Duran (51) 3143.41
. . .
```

5. *For August and September 2005, what artists were tagged the most?* For August and September 2005, print a list of the 10 artists with the highest number of tags for that month. Show the month and year on a line by themselves, followed by lines with the artist name, artist id, and number of tags for that artist for that month+year.

```
Aug 2005
  David Bowie(599):  num tags = 11
  New Order(59):  num tags = 7
  Rod Stewart(2861):  num tags = 7
  Ultravox(1013):  num tags = 6
  Siouxsie and the Banshees(1083):  num tags = 6
  Duran Duran(51):  num tags = 5
  INXS(99):  num tags = 5
  The Beatles(227):  num tags = 5
  Heavenly(4324):  num tags = 5
  Lou Reed(4541):  num tags = 5

Sep 2005
  Blood Ruby(11892):  num tags = 13
  The Fleshtones(550):  num tags = 3
  Greg Dulli(14468):  num tags = 3
  ...
```

Reflection statement

In Project 2, you analyzed the last.fm data by building your own data structures using built-in Python data types such as list and dicts. In Project 3, you analyzed the same data using Pandas data structures and features. Write a paragraph of 100 to 200 words in which you reflect on the differences in these two approaches (build your own data structures vs. using Pandas). Were there aspects that were easier in one approach but not the other? What were the pros and cons of each? Which did you prefer and why? This question asks you to reflect on your experience, so there is no "right" answer here. Include your paragraph as a clearly labelled comment at the END of your Python code.

Grading:

Your program will be evaluated based on its functionality, programming logic, and programming style. Functionality focuses on the question, "Does your program product the correct results?" Programming logic considers whether the approach you implemented in your code is correct (or close to correct). Programming style looks at how easy it is to understand your code – is it organized well, did you use functions appropriately, did you include good comments?

As outlined earlier, part of your grade on this assignment will also depend on appropriate use of Pandas data structures and features.

How to turn in your assignment:
Your program should be contained in a single file and be entirely code that you write yourself. Name your file according to the following convention:

```
youronyen_p3.py
```

Replace *youronyen* with your actual Onyen (e.g. my assignment would be `rcapra_p3.py`). The character between *youronyen* and the "p3.py" part should be a single underscore. There should be no spaces or other characters in the filename. Files with names that do not follow this convention may receive a substantial grade deduction.

I will test your program by running it with Python 2.7, with the specified .dat files in the same directory.

Submit your file electronically through the Sakai by going to the Assignments area and finding the "P3" assignment. After you think you have submitted the assignment, I strongly recommend checking to be sure the file was uploaded correctly by clicking on it from within Sakai. Keep in mind that if I cannot access your file, I cannot grade it.

If for some reason you need to re-submit your file, you must add a version number to your filename. Sakai is configured so that it will accept up to 3 total submissions. Use the following file naming convention if you need to re-submit:

Your first submission:	`youronyen_p3.py`
Your second submission:	`youronyen_p3_v2.py`
Your third submission:	`youronyen_p3_v3.py`

Sakai is also configured with a due date and an "accept until" date. Submissions received after the due date (even just 1 minute!) may receive a 10% penalty per day. The "accept until" date is 5 days after the due date. Unless you have made arrangements with me in advance, submissions will not be accepted after the "accept until" date and will have a score of zero recorded.