Python Review HW

Please try to complete the following exercises using PyCharm.

1. Using the movie-lens 1M data and `pandas.read_table` read in all three files (users, ratings, movies) into pandas DataFrames. I recommend giving columns names directly to `read_table` for each case.
   HINT: you will need to open the files in a text editor to see what the separator character. Also, the columns for each file correspond to the following:
   - users.dat: user id, gender, age, occupation code, zip
   - ratings.dat: user id, movie id, rating, timestamp
   - movies.dat: movie id, title, genre

   Use the data combining tools discussed above to combine these three objects into a single object named `movie_data`

2. Use the `movie_data` object from the previous exercise and compute the following things:
   - The 5 movies with the most number of ratings
   - A new object called `active_titles` that is made up of movies each having at least 250 ratings
   - For the subset of movies in the `active_titles` list compute the following:
     - The 3 movies with the highest average rating for females. Do the same for males.
     - The 10 movies men liked much more than women and the 10 movies women liked more than men (use the difference in average ratings and sort ascending and descending).

  - The 5 movies that had the highest standard deviation in rating.

HINT: For some of these operations it might be helpful to compute a pivot table using the `pivot_table` method of a DataFrame.
NOTE: You might also want to do some analysis on the movie genre. To do this you would have to be comfortable with using pandas vectorized string methods and probably some regular expressions. Those topics are a bit bigger than what we can cover here, but if you are looking for more practice, they are a good place to start.

3. Using the baby names data set, define a pandas object called `names_data` that includes all the years.
   HINT: You should probably do this in a for loop. Also, to get a list of all the text files in a directory (so you can loop over them), use the following code:

```
from glob import glob
files = glob('*.TXT')
```

4. Using the `names_data` object you just created, generate the following items:
   - The most popular male and female name in your birth year
   - The number of people born with your name in the same year (note, if you have a somewhat uncommon name, this may not appear. For example, someone's name is Shannen and due to its irregular spelling, wasn't reported for her birth year)
   - A plot of the number of instances of your name over time.
   - A plot of the number of the total boy names and the number of girls names each year.
   - A plot showing the fraction of male and female babies given a name similar to Lesley. By similar I mean the name starts with 'lesl' (make sure you make the name lowercase).

5. Use the built in python `json` module to load the food data into a python list. Your code should look like this

```
import json
db = json.load(open('foods-2011-10-03.json'))
```

`db` will be a list of 6636 python dictionaries, each containing nutritional information for a different food item. Each dictionary will have the following keys:

- portions
- description
- tags
- nutrients
- group
- id
- manufacture

Create a DataFrame of meta_data using the description, group, id, and manufacturer items in each dictionary.

6. Loop over `db` and construct a list of DataFrames containing the nutritional information for each record in `db`. Make sure to add a column to each of these DataFrames that contains the unique food id (`id` key in the dictionary). HINT: This food id will be a constant Finally, use the pandas combining techniques to create a nutrients DataFrame. After you have done this drop duplicate entries in this DataFrame. For example, if you had named the objects nuts you would do

   ```
   nuts = nuts.drop_duplicates()
   ```

7. Use the `rename` method to make sure that the `description` and `group` columns are un-ambiguous for both the meta_data DataFrame and the nutrients DataFrame (These column names are duplicated because every food has a description and group and each nutrient also has those identifiers). HINT: you will need to use the `columns` and `copy` arguments of the `rename` method.
   Finally, use the data combining routines to come up with a `foods` DataFrame containing all the meta_data and nutritional information. Make sure to do an `outer` style merge on the correct columns.

8. Using the `foods` DataFrame you have been building, compute the following things:
   - The food item with the highest content of each nutrient.

- A function that accepts a nutrient name and a quantile value and generates a horizontal bar plot of the amount of that nutrient in each food group. Provide a plot title. HINT: You will need to use the `quantile` and `sort` (or `order`) methods in order for this to work. Also, you can generate the horizontal bar plot using `df.plot(kind='barh')`.