


## What you will do

Now you are ready! We are going to do three tasks in this assignment. There are several results you need to gather along the way to enter into the quiz after this reading.

1. **Counting unique users:** The method `.unique()` can be used to select the unique elements in a column of data. In this question, you will compute the number of unique users who have listened to songs by various artists. For example, to find out the number of unique users who listened to songs by 'Kanye West', all you need to do is select the rows of the song data where the artist is 'Kanye West', and then count the number of unique entries in the 'user\_id' column. Compute the number of unique users for each of these artists: 'Kanye West', 'Foo Fighters', 'Taylor Swift' and 'Lady GaGa'. **Save these results to answer the quiz at the end.**
2. **Using groupby-aggregate to find the most popular and least popular artist:** each row of `song_data` contains the number of times a user listened to particular song by a particular artist. If we would like to know how many times any song by 'Kanye West' was listened to, we need to select all the rows where 'artist'=='Kanye West' and sum the 'listen\_count' column. If we would like to find the most popular artist, we would need to follow this procedure for each artist, which would be very slow. Instead, you will learn about a very important method:

```
1 .groupby()
```

You can read the [documentation about groupby here](#) . The `.groupby` method computes an aggregate (in our case, the sum of the 'listen\_count') for each distinct value in a column (in our case, the 'artist' column).

Follow these steps to find the most popular artist in the dataset:

- The `.groupby` method has two important parameters:
  - i. `key_columns`, which takes the column we want to group, in our case, 'artist'
  - ii. `operations`, where we define the aggregation operation we using, in our case, we want to sum over the 'listen\_count'.
- With this in mind, the following command will compute the sum `listen_count` for each artist and return an SFrame with the results:

```
1 song_data.groupby(key_columns='artist', operations={'total_count': turicreate.aggregate
```

- With this in mind, the following command will compute the sum *listen\_count* for each artist and return an SFrame with the results:

```
1 song_data.groupby(key_columns='artist', operations={'total_count': turicreate.aggregate
```

the total number of listens for each artist will be stored in *'total\_count'*.

- Sort the resulting SFrame according to the *'total\_count'*, and find the artist with the most popular and least popular artist in the dataset. **Save these results to answer the quiz at the end.**

3. **[OPTIONAL] Using groupby-aggregate to find the most recommended songs:** Now that we learned how to use *.groupby()* to compute aggregates for each value in a column, let's use to find the song that is most recommended by the *personalized\_model* model we learned in the Jupyter notebook above. Follow these steps to find the most recommended song:

- Split the data into 80% training, 20% testing, using *seed=0*, as was done in the Jupyter notebook above.
- Train an *item\_similarity\_recommender*, as done in the Jupyter notebook, using the training data.
- Next, we are going to make recommendations for the users in the test data, but there are over 200,000 users (58,628 unique users) in the test set. Computing recommendations for these many users can be slow in some computers. Thus, we will use only the first 10,000 users only in this question. Using this command to select this subset of users:

```
1 subset_test_users = test_data['user_id'].unique()[0:10000]
```

- Let's compute one recommended song for each of these test users. Use this command to compute these recommendations:

```
1 personalized_model.recommend(subset_test_users,k=1)
```

- Finally, we can use `.groupby()` to find the most recommended song! :) When we used `.groupby()` in the previous question, we summed up the total `'listen_count'` for each artist, by setting the parameter `SUM` in the aggregator:

```
1 operations={'total_count': turicreate
2 .aggregate.SUM('listen_count')}
```

For this question, we simply want to count how often each song is recommended, so we will use the `COUNT` aggregator instead of `SUM`, and store the results in a column we will call `'count'` by using:

```
1 operations={'count': turicreate.aggregate.COUNT() }
```

And, since we want to use the song titles as the key to the aggregator instead of the `'artist'`, we use:

```
1 key_columns='song'
```

- By sorting the results, you will find out the most recommended song to the first 10,000 users in the test data! **Due to randomness in train-test split, the most recommended song may come out differently for different people. This is why we chose not to assign a quiz question for this section.**

✓ Completed

Go to next item

## ✓ Congratulations! You passed!

Grade  
received 100%

Latest Submission  
Grade 100%

To pass 80% or  
higher

[Go to next item](#)

1. Which of the artists below have had the most unique users listening to their songs?

1 / 1 point

✓ Correct

2. Which of the artists below is the most popular artist, the one with highest total listen\_count, in the data set?

1 / 1 point

✓ Correct

3. Which of the artists below is the least popular artist, the one with smallest total listen\_count, in the data set?

1 / 1 point

✓ Correct