Assignment2_SamLogsdon

September 25, 2019

1 Setup (Run this block first)

```
[ ]: DATA_PATH = 'datascience.stackexchange.com'
import preprocessing
from pyspark import SparkContext
import os
os.environ['JAVA_HOME'] = '/Library/Java/JavaVirtualMachines/adoptopenjdk-8.jdk/
 ↪Contents/Home'
users = preprocessing.user_xml(f'{DATA_PATH}/Users.xml')
posts = preprocessing.post_xml(f'{DATA_PATH}/PostHistory.xml')

sc: SparkContext = SparkContext.getOrCreate()
user_rdd = sc.parallelize(users)
posts_rdd = sc.parallelize(posts)
```

2 Question 1

From the Users.xml file, find all users which are from Georgia and output to screen their DisplayName only.

```
[ ]: ga_users = user_rdd.filter(lambda s: (" GA" in s.Location or "Atlanta, Georgia"␣
 ↪in s.Location))
for user in ga_users.collect():
    print(user.DisplayName)
```

3 Question 2

Using the Users.xml file, provide the count for all users which joined (CreationDate) in 2017. (30 points). Output this to the screen.

```
[ ]: users_2017 = user_rdd.filter(lambda s: "2017" in s.CreationDate)
print(f'{users_2017.count()} accounts created in 2017.')
```

4 Question 3

Using the PostHistory file, count the number of Posts that feature the words "Spark" and "Scala". Output this to the screen.

[ ]:
```
filtered_posts = posts_rdd.filter(lambda s: "scala" in s.Text.lower() and␣
 ↪"spark" in s.Text.lower())
print(f'{filtered_posts.count()} posts')
```

5 Question 4

Using the PostHistory file, provide a total count of the words used by each distinct user. In other words, count all words in all posts for each user and display this to screen. You can only identify users by the UserID (30 points). You get 15 bonus points if you get the actual DisplayName of the user.

[ ]:
```
from operator import add
display_names = user_rdd.map(lambda s: (s.Id, s.DisplayName))
post_counts = posts_rdd.map(lambda s: (s.UserId, len(s.Text.split())))
grouped_post_counts = post_counts.reduceByKey(add)
joined_rdd = grouped_post_counts.join(display_names).sortBy(lambda s: s[1][0],␣
 ↪False)
print(f'{"UserId":<10}  {"DisplayName":<35}  WordCount')
for x in joined_rdd.collect():
    print(f'{x[0]:<10}  {x[1][1]:<35}  {x[1][0]}')
```

[ ]:
```

```

6 Question 5

Using the users.xml, comments.xml and PostHistory.xml files, produce a single file that includes the following information: DisplayName, Number of Comments, total Score and Number of posts. This file should have the users (DisplayName) sorted by score, descending from higher to lower.

[ ]:
```
from operator import add
comments = preprocessing.comments_xml(f'{DATA_PATH}/Comments.xml')
comments_rdd = sc.parallelize(comments)
mapped_rdd = comments_rdd.map(lambda s: (s.UserId, int(s.Score)))

comment_counts = mapped_rdd.countByKey()
cc_rdd = sc.parallelize([(k, v) for k, v in comment_counts.items()])
score_rdd = mapped_rdd.foldByKey(0, add)

post_counts = posts_rdd.map(lambda s: (s.UserId, s.Id)).countByKey()
pc_rdd = sc.parallelize([(k, v) for k, v in post_counts.items()])
```

```
final_rdd = pc_rdd.join(cc_rdd).join(score_rdd).join(display_names)

final_rdd = final_rdd.mapValues(lambda v: (v[0][0][0], v[0][0][1], v[0][1],␣
 ↪v[1])).sortBy(lambda s: s[1][2], False)
with open('question5.csv', 'w') as fp:
    fp.write('UserId, DisplayName, PostCount, CommentCount, TotalScore\n ')
    for row in final_rdd.collect():
        fp.write(f'{row[0]}, {row[1][3]}, {row[1][0]}, {row[1][1]},␣
 ↪{row[1][2]}\n')
```