

Thomas Munoz Vasquez

Big Data Programming

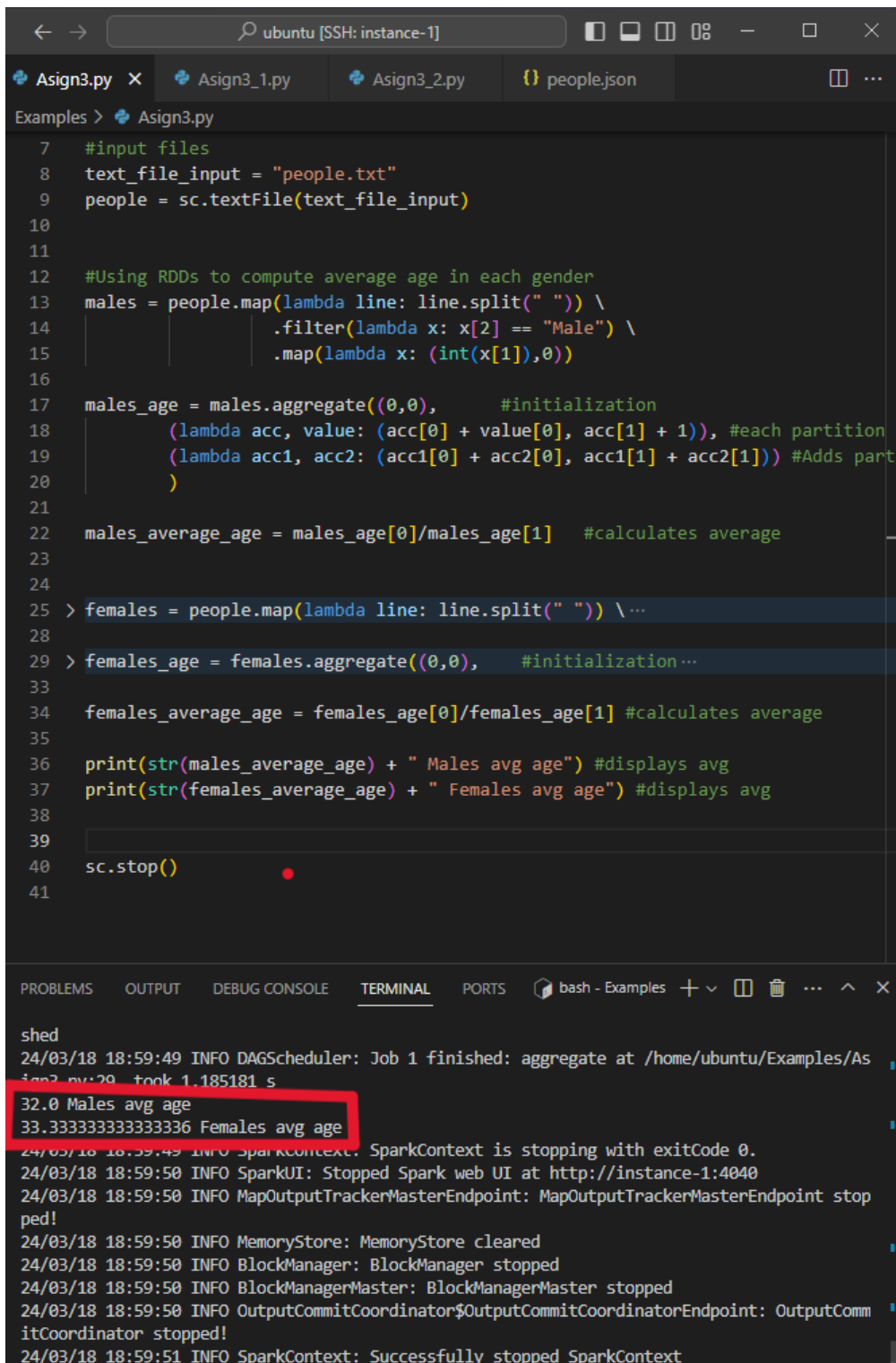
Due March 18, 2024

### Assignment 3

The goal is to implement RDD functions, Dataframe functions, and SparkSQL functions to compute the average age for males and females. The programs will read each file and perform simple operations to add and divide to calculate the average. We will use two different file formats: .txt and .json. .txt files will be used for implementing RDD functions, while .json files will be used for implementing dataframes and SparkSQL.

Using RDDS functions to calculate average.

The configuration runs the app locally on the instance-1 machine, where it reads a .txt file. The map function is used to remove spaces between lines in the file. Following this, the filter function extracts the third element from the transformation and matches it with the gender. Next, a second map function converts the age from a string to an integer to enable operations. The aggregate function then adds and counts the age, accumulating each RDD element into an accumulator that combines all partitions into one. The tuple created from the aggregate function is utilized to calculate the average age for males, and the same process is repeated for females. As you can see in Figure 1



```
7 #input files
8 text_file_input = "people.txt"
9 people = sc.textFile(text_file_input)
10
11
12 #Using RDDs to compute average age in each gender
13 males = people.map(lambda line: line.split(" ")) \
14                 .filter(lambda x: x[2] == "Male") \
15                 .map(lambda x: (int(x[1]),0))
16
17 males_age = males.aggregate((0,0),      #initialization
18                             (lambda acc, value: (acc[0] + value[0], acc[1] + 1)), #each partition
19                             (lambda acc1, acc2: (acc1[0] + acc2[0], acc1[1] + acc2[1])) #Adds part
20                             )
21
22 males_average_age = males_age[0]/males_age[1] #calculates average
23
24
25 > females = people.map(lambda line: line.split(" ")) \...
28
29 > females_age = females.aggregate((0,0),      #initialization...
33
34 females_average_age = females_age[0]/females_age[1] #calculates average
35
36 print(str(males_average_age) + " Males avg age") #displays avg
37 print(str(females_average_age) + " Females avg age") #displays avg
38
39
40 sc.stop()
41
```

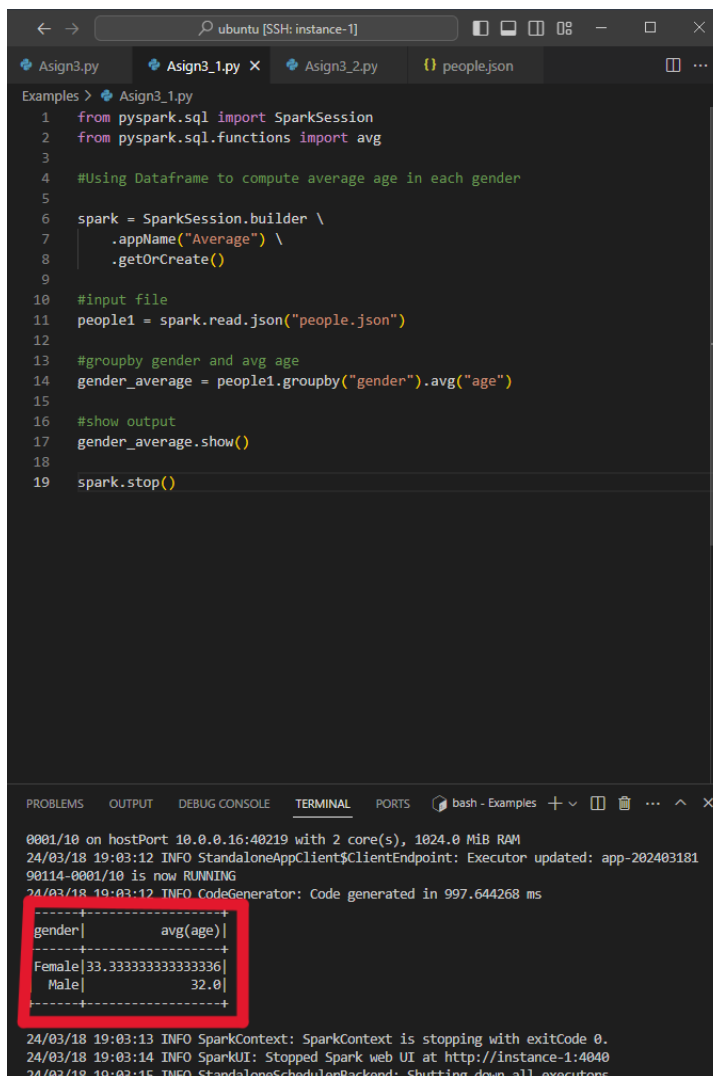
PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS bash - Examples + -

```
shed
24/03/18 18:59:49 INFO DAGScheduler: Job 1 finished: aggregate at /home/ubuntu/Examples/As
ign3.py:29, took 1.185181 s
32.0 Males avg age
33.333333333333336 Females avg age
24/03/18 18:59:49 INFO SparkContext: SparkContext is stopping with exitCode 0.
24/03/18 18:59:50 INFO SparkUI: Stopped Spark web UI at http://instance-1:4040
24/03/18 18:59:50 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stop
ped!
24/03/18 18:59:50 INFO MemoryStore: MemoryStore cleared
24/03/18 18:59:50 INFO BlockManager: BlockManager stopped
24/03/18 18:59:50 INFO BlockManagerMaster: BlockManagerMaster stopped
24/03/18 18:59:50 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputComm
itCoordinator stopped!
24/03/18 18:59:51 INFO SparkContext: Successfully stopped SparkContext
```

Figure 1. Code to calculate average age for males and females. Output from file people.txt.

Using Dataframe to calculate average.

The configuration is utilized to establish a session, either locally or in a cluster, and it's primarily recommended for use with DataFrames and Datasets APIs. It reads a .json file containing a dataset and employs the groupBy and avg functions, given that we import sql.functions at the top. This approach produces the same output but in a more structured way.



The screenshot shows a code editor with a file named `Assign3_1.py` open. The code imports `SparkSession` and `avg` from `pyspark.sql` and `pyspark.sql.functions`. It then builds a `SparkSession` with the name "Average", reads a `people.json` file, groups the data by gender, calculates the average age, and displays the result. The terminal output shows the execution details and the final result as a table with two rows: Female with an average age of 33.33333333333336 and Male with an average age of 32.0.

```
1 from pyspark.sql import SparkSession
2 from pyspark.sql.functions import avg
3
4 #Using Dataframe to compute average age in each gender
5
6 spark = SparkSession.builder \
7     .appName("Average") \
8     .getOrCreate()
9
10 #input file
11 people1 = spark.read.json("people.json")
12
13 #groupby gender and avg age
14 gender_average = people1.groupby("gender").avg("age")
15
16 #show output
17 gender_average.show()
18
19 spark.stop()
```

0001/10 on hostPort 10.0.0.16:40219 with 2 core(s), 1024.0 MiB RAM  
24/03/18 19:03:12 INFO StandaloneAppClient\$ClientEndpoint: Executor updated: app-20240318190114-0001/10 is now RUNNING  
24/03/18 19:03:12 INFO CodeGenerator: Code generated in 997.644268 ms

gender	avg(age)
Female	33.33333333333336
Male	32.0

24/03/18 19:03:13 INFO SparkContext: SparkContext is stopping with exitCode 0.  
24/03/18 19:03:14 INFO SparkUI: Stopped Spark web UI at http://instance-1:4040  
24/03/18 19:03:15 INFO StandaloneSchedulerBackend: Shutting down all executors

Figure 2. Code to calculate average age using Dataframes APIs and output.

▼ Running Applications (1)								
Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20240318190114-0001 (kill)	Average	4	1024.0 MiB		2024/03/18 19:01:14	ubuntu	RUNNING	1.5 min

Figure 3. Running in cluster.

▼ Completed Applications (2)								
Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20240318190114-0001	Average	4	1024.0 MiB		2024/03/18 19:01:14	ubuntu	FINISHED	2.1 min

Figure 4. Completed in cluster.

Figure 5. Code to calculate average age using Dataframes APIs and output.

## Using SparkSQL

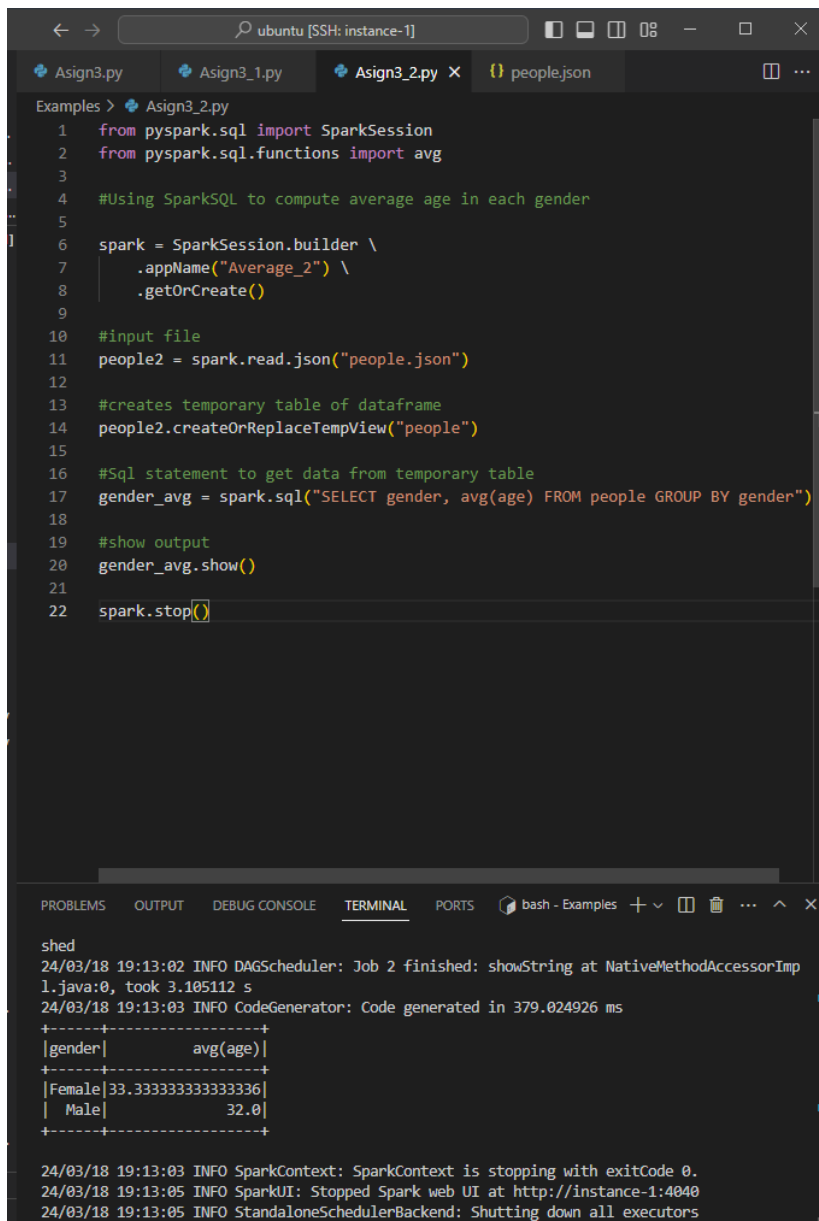
It utilizes the same configuration and file as before. This time, it creates a temporary table named people for the dataframe in the file. We retrieve the same output using an SQL query that calculates the age average for each gender.

▼ Running Applications (1)								
Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20240318191114-0003 (kill)	Average_2	4	1024.0 MiB		2024/03/18 19:11:14	ubuntu	RUNNING	13 s

Figure 6. Running in cluster.

▼ Completed Applications (4)								
Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20240318191114-0003	Average_2	4	1024.0 MiB		2024/03/18 19:11:14	ubuntu	FINISHED	1.9 min

Figure 7. Completed in cluster.



The image shows a PySpark code editor window with a file named `Assign3_2.py` and a JSON file named `people.json`. The code in `Assign3_2.py` is as follows:

```
1 from pyspark.sql import SparkSession
2 from pyspark.sql.functions import avg
3
4 #Using SparkSQL to compute average age in each gender
5
6 spark = SparkSession.builder \
7     .appName("Average_2") \
8     .getOrCreate()
9
10 #input file
11 people2 = spark.read.json("people.json")
12
13 #creates temporary table of dataframe
14 people2.createOrReplaceTempView("people")
15
16 #Sql statement to get data from temporary table
17 gender_avg = spark.sql("SELECT gender, avg(age) FROM people GROUP BY gender")
18
19 #show output
20 gender_avg.show()
21
22 spark.stop()
```

The terminal output shows the execution of the code, including the SparkSession creation, the loading of the JSON file, the creation of the temporary table, the execution of the SQL query, and the resulting output table:

```
24/03/18 19:13:02 INFO DAGScheduler: Job 2 finished: showString at NativeMethodAccessorImp
l.java:0, took 3.105112 s
24/03/18 19:13:03 INFO CodeGenerator: Code generated in 379.024926 ms
+-----+-----+
|gender|    avg(age)|
+-----+-----+
|Female|33.33333333333336|
|  Male|          32.0|
+-----+-----+
24/03/18 19:13:03 INFO SparkContext: SparkContext is stopping with exitCode 0.
24/03/18 19:13:05 INFO SparkUI: Stopped Spark web UI at http://instance-1:4040
24/03/18 19:13:05 INFO StandaloneSchedulerBackend: Shutting down all executors
```

Figure 8. Code to calculate average age using SparkSQL with output.