

Big Data Hadoop and Spark Developer

Retail Business Analytics

Final Project – An Nguyen

Date: 12/14/2022

1. Explore the customer records saved in the "customers-tab-delimited" directory on HDFS • Show the client information for those who live in California • Save the results in the result/scenario1/solution folder
 - Only the customer's entire name should be included in the output

```
# final code for submitting spark job (scenario 1)
from pyspark.sql import SparkSession

#create spark session
spark = SparkSession.builder.appName('spark').getOrCreate()

#input data
df = spark.read.option('delimiter','\t').csv(r'data-files/customers-tab-delimited/part-

#apply schema on data, making it a dataframe
df = df.toDF('index','first_name','last_name','x1','x2','street','city','state','zip_co

#using filter function, select only rows with state == CA
ca_df = df.filter('state == "CA"')

#now we combine first and last name into full name
from pyspark.sql import functions as F
final_df = ca_df.select(F.concat(F.col('first_name'),\
                                F.lit(' '),\
                                |                                F.col('last_name')).alias('full_name'))

#save file to the correct directory
final_df.write.format('csv').option('header',True).option('sep','\t').save(r'results/sc

#close out app to save resource
spark.stop()
```

submit job via

```
spark3-submit --conf spark.ui.port=6065 --deploy-mode client ca.py
```

```
/ user/ antnguyen72gmail/ results/ sc1/  
task1/  
part-00000-edf51616-21d3-42ca-ac49-  
89622e92b907-c000.csv
```

```
full_name  
Mary Jones  
Katherine Smith  
Jane Luna  
Robert Smith  
Margaret Wright  
Mary Smith  
Howard Smith  
Mary Kim  
Douglas James  
Mary Simmons  
Frank Gillespie  
Joseph Young  
Sean Smith  
Lauren Freeman  
Alice Warner  
Mary Smith  
Mary Gallagher
```

2. Explore the order records saved in the “orders parquet” directory on HDFS • Show all orders with the order status value "COMPLETE" • Save the data in the "result/scenario2/solution" directory on HDFS •
The "order date" column should be in the "YYYY-MM-DD" format
Use GZIP compression to compress the output
Include order number, order date, and current situation in the output

```
# final code for submitting spark job (scenario 2)
from pyspark.sql import SparkSession

#create spark session
spark = SparkSession.builder.appName('spark').getOrCreate()

#input data
df = spark.read.parquet(r'data-files/orders_parquet/shortName.parquet',inferSchema=True)

#apply schema on data is not needed because header is included

#using filter function, select only rows with order_status == COMPLETE
complete_df = df.filter('order_status == "COMPLETE"')

# after applying the first filter, convert time stamp into standard date format
from pyspark.sql.functions import from_utc_timestamp, from_unixtime, date_format, col
# convert timestamp
normalized_date_df = complete_df.withColumn('new_date',from_utc_timestamp(from_unixtime(df['order_date']/1000),0))
# extract date
final_df = normalized_date_df.withColumn('date',date_format(col('new_date'),'yyyy-MM-dd').cast('date'))
# select columns we need
final_df = final_df['order_id','order_customer_id','date','order_status']

# save file to the correct directory using gzip compression
final_df.write.format('csv').option('header',True).option('sep','\t').option('compression','gzip').save(r'results/sc1/task2/part-0000-a5a6d978-2296-4933-906c-d8241075db2c-c000.csv.gz')

#close out app to save resource
spark.stop()
```

/ user/ antnguyen72gmail/ results/ sc1/
task2/
part-0000-a5a6d978-2296-4933-906c-
d8241075db2c-c000.csv.gz

Output rendered from compressed gzip file.

	order_id	order_customer_id	date	order_status
	3	12111	2013-07-24	COMPLETE
	5	11318	2013-07-24	COMPLETE
	6	7130	2013-07-24	COMPLETE
1	7	4530	2013-07-24	COMPLETE
	15	2568	2013-07-24	COMPLETE
	17	2667	2013-07-24	COMPLETE
	22	333	2013-07-24	COMPLETE
	26	7562	2013-07-24	COMPLETE
22	28	656	2013-07-24	COMPLETE
	32	3960	2013-07-24	COMPLETE
	35	4840	2013-07-24	COMPLETE
	45	2636	2013-07-24	COMPLETE
172	56	10519	2013-07-24	COMPLETE
	63	1148	2013-07-24	COMPLETE
	65	5903	2013-07-24	COMPLETE
	67	1406	2013-07-24	COMPLETE
	71	8646	2013-07-24	COMPLETE
	72	4349	2013-07-24	COMPLETE

- Explore the customer records saved in the "customers-tab-delimited" directory on HDFS • Produce a list of all consumers who live in the city of "Caguas" • Save the results in the result/scenario3/solution folder • The result should only contain records with the value "Caguas" for the customer city

Use snappy compression to compress the output
Save the file in the orc format

```
# final code for submitting spark job (scenario 3)
from pyspark.sql import SparkSession

#create spark session
spark = SparkSession.builder.appName('spark').getOrCreate()

#input data
df = spark.read.option('delimiter','\t').csv(r'data-files/customers-tab-delimited/part-m-00000',inferSchema)

#apply schema on data, making it a dataframe
df = df.toDF('index','first_name','last_name','x1','x2','\
            'street','city','state','zip_code')

#using filter function, select only rows with city = Caguas
ca_df = df.filter('city == "Caguas"')

#save file to the correct directory
ca_df.write.format('orc').option('compression','snappy').option('header',True).save(r'results/sc1/task3/')

#close out app to save resource
spark.stop()
```

 Home

/ user/ antnguyen72gmail/ results/ sc1/
task3/
**part-00000-b8569524-d99e-4542-8fa3-
66a44b065fd7-c000.snappy.orc**

```
0000000: 4f 52 43 13 00 00 0a 07 12 05 08 e8      ORC.....#P.7
          23 50 00 37
0000010: 00 00 0a 19 0a 03 00 00 00 12 12 08      .....#..
          e8 23 12 0b
0000020: 08 06 10 a4 c2 01 18 f8 8f 8e 1b 50      .....P.I..
          00 49 00 00
0000030: 0a 22 0a 03 00 00 00 12 1b 08 e8 23      .".....#"...
          22 14 0a 05
0000040: 41 61 72 6f 6e 12 07 5a 61 63 68 61      Aaron..Zachary..
          72 79 18 f6
0000050: f4 02 50 00 4f 00 00 0a 25 0a 03 00      ..P.O...%.....
          00 00 12 1e
2000060: 08 e8 23 22 17 0a 06 41 62 62 6f 74      ..#"...Abbott..Z
          74 12 09 5a
0000070: 69 6d 6d 65 72 6d 61 6e 18 9c 98 03      immerman....P.@.
          50 00 40 00
```

4. Explore the order records saved in the “categories” directory on HDFS • Save the result files in CSV format • Save the data in the result/scenario4/solution directory on HDFS • Use lz4 compression to compress the output

```
# final code for submitting spark job (scenario 4)
from pyspark.sql import SparkSession

#create spark session
spark = SparkSession.builder.appName('spark').getOrCreate()

#input data
df = spark.read.option('delimiter', ',').csv(r'data-files/categories/part-m-00000', inferSchema=True)

#apply schema on data, making it a dataframe
df = df.toDF('index', 'type', 'category')

#save file to the correct directory
df.write.format('csv').option('header', True).option('compression', 'lz4')\
    .option('sep', ',').save(r'results/sc1/task4/')

#close out app to save resource
spark.stop()
```

submit job via

spark3-submit --conf spark.ui.port=6065 --deploy-mode client categories.py

for compressing, I could not find a way to compress file straight in the hdfs system, so I resorted to copying it to the local linux file system, compress it then copy it back. Your kernel did not have lz4 library installed

hdfs dfs -copyToLocal /user/antnguyen72gmail/results/sc1/task4/part-00000-58f13e38-67f9-44f3-9ba9-858686dc369c-c000.csv

lz4 part-00000-58f13e38-67f9-44f3-9ba9-858686dc369c-c000.csv

hdfs dfs -put part-00000-58f13e38-67f9-44f3-9ba9-858686dc369c-c000.csv.lz4

/user/antnguyen72gmail/results/sc1/task4/part-00000-58f13e38-67f9-44f3-9ba9-858686dc369c-c000.csv.lz4

... 41 more

Caused by: java.lang.RuntimeException: native lz4 library not available
at org.apache.hadoop.io.compress.Lz4Codec.getCompressorType(Lz4Cod

The screenshot shows a file explorer interface. On the left, a sidebar displays a directory tree with 'task4' selected. Below it, a list of files is shown: '_SUCCESS', 'part-00000-58f13e38-67f9-44f3-9ba9-858686dc369c-c000.csv', and 'part-00000-58f13e38-67f9-44f3-9ba9-858686dc369c-c000.csv.lz4'. The main area on the right shows the details of the selected file. It includes a path '/ user/ antnguyen72gmail/ results/ sc1/ task4/ part-00000-58f13e38-67f9-44f3-9ba9-858686dc369c-c000.csv', a 'Refresh' button, and a 'View as binary' option. Below this, a preview of the file's content is shown, listing various sports and equipment categories like 'index, type, category', '1,2, Football', '2,2, Soccer', '3,2, Baseball & Softball', '4,2, Basketball', '5,2, Lacrosse', '6,2, Tennis & Racquet', '7,2, Hockey', '8,2, More Sports', and '9,2, Cardio Equipment'. A 'Download' button is visible at the bottom of the preview area.

- Only products with a price of more than 1000.0 should be included in the output
Save the output files in parquet format
Remove data from the table if the product price is greater than 1000.0
Save the data in the result/scenario5/solution directory on HDFS
Use snappy compression to compress the output

```

from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('spark').getOrCreate()

# read all files

# get a list of all file names from the directory
import os
file_list = os.listdir(r'data-files/products_avro/')

# using loop, add each dataframe objects onto a list
df_list = []
for line in file_list:
    df = spark.read.format("avro").load(f'data-files/products_avro/{line}')
    df_list.append(df)

# use reduce functional tool to apply a lambda function to an iterable list
# in this case, we unioned all the dataframes together
from functools import reduce
combined_df = reduce(lambda x,y: x.union(y), df_list)

# output rows with product_price is greater than 1000.0
combined_df.filter('product_price > 1000.0').show()

# remove data from table if the product price is greater than 1000.0
final_df = combined_df.filter('product_price <= 1000.0')

# save file
# tried to save as avro with compression but avro is not compat with snappy
# turns out orc is the only compatible with snappy??
final_df.write.format('orc').option('compression','snappy').save(r'results/sc1/task5/')

# close out spark session
spark.stop()

```

< task5



Filter...

part-00000-c109341c-f5c0-421e-89e7-d45f20

part-00001-c109341c-f5c0-421e-89e7-d45f20

part-00002-c109341c-f5c0-421e-89e7-d45f20

part-00003-c109341c-f5c0-421e-89e7-d45f20

Search for file name

Actions

Copy Path

Open in Importer

Home

/user/antnguyen72gmail/results/sc1/task5

<input type="checkbox"/>	Name	Size	User	Group	Permi
<input type="checkbox"/>	↑		antnguyen72gmail	hadoop	drwxr
<input type="checkbox"/>	.		antnguyen72gmail	hadoop	drwxr
<input type="checkbox"/>	._SUCCESS	0 bytes	antnguyen72gmail	hadoop	-rw-rw-
<input type="checkbox"/>	part-00000-c109341c-f5c0-421e-89e7-d45f201ef668-c000.snappy.parquet	13.6 KB	antnguyen72gmail	hadoop	-rw-rw-
<input type="checkbox"/>	part-00001-	16.7 KB	antnguyen72gmail	hadoop	-rw-rw-

```

for this job
22/12/15 04:09:58 INFO cluster.YarnScheduler: Killing all running tasks in stage 1: Stage finished
22/12/15 04:09:58 INFO scheduler.DAGScheduler: Job 1 finished: showString at NativeMethodAccessorImpl.java:0, took
4.616677 s
22/12/15 04:09:58 INFO codegen.CodeGenerator: Code generated in 13.834702 ms
+-----+-----+-----+-----+-----+-----+
|product_id|product_category_id|product_name|product_description|product_price|product_image|
+-----+-----+-----+-----+-----+-----+
|66|4|SOLE F85 Treadmill|SOLE F85 Treadmill|1799.99|http://images.acm...|
|199|10|SOLE F85 Treadmill|SOLE F85 Treadmill|1799.99|http://images.acm...|
|208|10|SOLE F35 Elliptical|SOLE F35 Elliptical|1999.99|http://images.acm...|
|496|22|SOLE F85 Treadmill|SOLE F85 Treadmill|1799.99|http://images.acm...|
|1048|47|Spalding Beast 60...|Spalding Beast 60...|1099.99|http://images.acm...|
+-----+-----+-----+-----+-----+-----+

22/12/15 04:09:58 INFO datasources.FileSourceStrategy: Pushed Filters: IsNotNull(product_price),LessThanOrEqual(pre
duct_price,1000.0)
22/12/15 04:09:58 INFO datasources.FileSourceStrategy: Post-Scan Filters: isnotnull(product_price#4),(product_price
#4 <= 1000.0)
22/12/15 04:09:58 INFO datasources.FileSourceStrategy: Output Data Schema: struct<product_id: int, product_category
_id: int, product_name: string, product_description: string, product_price: float ... 1 more field>
22/12/15 04:09:58 INFO datasources.FileSourceStrategy: Pushed Filters: IsNotNull(product_price),LessThanOrEqual(pre
duct_price,1000.0)
22/12/15 04:09:58 INFO datasources.FileSourceStrategy: Post-Scan Filters: isnotnull(product_price#16),(product_pric

```

6. Explore the “products_avro” stored in product records

REQUIREMENT:

Only products with a price of more than 1000.0 should be in the output

The pattern "Treadmill" appears in the product name

Save the output files in parquet format

Save the data in the result/scenario6/solution directory on HDFS

Use GZIP compression to compress the output

```

from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('spark').getOrCreate()

# read all files

# get a list of all file names from the directory
import os
file_list = os.listdir(r'data-files/products_avro/')

# using loop, add each dataframe objects onto a list
df_list = []
for line in file_list:
    df = spark.read.format("avro").load(f'data-files/products_avro/{line}')
    df_list.append(df)

# use reduce functional tool to apply a lambda function to an iterable list
# in this case, we unioned all the dataframes together
from functools import reduce
combined_df = reduce(lambda x,y: x.union(y),df_list)

# output rows with product_price is greater than 1000.0
greater_df = combined_df.filter('product_price > 1000.0')

# create a temp view for easier querying
greater_df.createOrReplaceTempView('table')

# query rows with the pattern "Treadmill"
final_df = spark.sql("select * from table where lower(product_name) like '%treadmill%'")

# save file as parquet format using gzip compression
final_df.write.format('parquet').option('compression','gzip').save(r'results/sc1/task6/')

# close out spark session
spark.stop()

spark-submit --conf spark.ui.port=6065 --deploy-mode client avro_2.py

```

task6

Filter...

_SUCCESS

part-00000-35a2d551-7e20-43d0-851c-27d3c

part-00002-35a2d551-7e20-43d0-851c-27d3c

Back

Refresh

View as binary

Stop preview

Download

Last modified 12/15/2022 6:35 PM

Home

/ user/ antnguyen72gmail/ results/ sc1/ task6/ part-00000-35a2d551-7e20-43d0-851c-27d3cfb6ba52-c000.gz.parquet

Output rendered from compressed parquet file.

```
{
  "product_id": 66,
  "product_price": 1799.989990234375,
  "product_description": "",
  "product_image": "http://images.acmesports.sports/SOLE+F85+Treadmill",
  "product_category_id": 4,
  "product_name": "SOLE F85 Treadmill"
}, {
  "product_id": 199,
  "product_price": 1799.989990234375,
  "product_description": "",
  "product_image": "http://images.acmesports.sports/SOLE+F85+Treadmill",
  "product_category_id": 10,
  "product_name": "SOLE F85 Treadmill"
}
```

7. Explore the order records that are saved in the “orders parquet” table on HDFS

REQUIREMENT:

Output all PENDING orders in July 2013

Output files should be in JSON format

Save the data in the result/scenario7/solution directory on HDFS.

Only entries with the order status value of "PENDING" should be included in the result

Order date should be in the YYYY-MM-DD format

Use snappy compression to compress the output, which should just contain the order date and order status

Your environment was not setup in such a way that allowed me to save a json file with snappy compression. Therefore, I used my local setup to perform the same task

```
# final code for submitting spark job (scenario 7)
from pyspark.sql import SparkSession

#create spark session
spark = SparkSession.builder.appName('spark').getOrCreate()

#input data
df = spark.read.parquet('data-files/orders_parquet/shortName.parquet',inferSchema=True)

#apply schema on data is not needed because header is included

#using filter function, select only rows with order_status == PENDING
pending_df = df.filter('order_status == "PENDING"')

# after applying the first filter, convert time stamp into standard date format
from pyspark.sql.functions import from_utc_timestamp, from_unixtime, date_format, col
# convert timestamp
normalized_date_df = pending_df.withColumn('new_date',from_utc_timestamp(from_unixtime(df['order_date']/1000),'UTC-8'))
# extract date
final_df = normalized_date_df.withColumn('date',date_format(col('new_date'),'yyyy-MM-dd').cast('date'))

# select columns we need
final_df = final_df['date','order_status']

# save file
final_df.write.format('json').option('compression','snappy').save(r'results/sc1/task7/')

spark.stop()

spark3-submit --conf spark.ui.port=6065 --deploy-mode client pending.py
```


Filter files by name	
/ ... / sc1 / task7 /	
Name	Last Modified
_SUCCESS	2 minutes ago
part-00000-36f0b974-f4f4-4546-bea8-cec2f20d1ce4-c000.json.snappy	2 minutes ago