**Cheng, Loi**

October 4, 2019

Harvard University Extension - Principles of Big Data Processing e88

Homework 5: **Spark Batch processing and Parquet**

* **Make sure you submit your solution document as a separate file in Canvas**
* **submit all your source code in a separate archive, named <LastName>\_<FirstName>\_HW5.zip**
* **Make sure to add full result files into that archive as well**

Please identify which problems were completed. If any were incomplete, please identify where you encountered problems.

|  |
| --- |
| *for example:*  Problem 1: 100% complete  Problem 2: 100% complete  Problem 3: 100% complete  Problem 4: 100% complete  Problem B1: Bonus: 100% complete |

**Problem 1: unique counts** [points: 40]

Paste source code of your Spark jobs into the following area [10 points]

|  |
| --- |
| Query 1  from \_\_future\_\_ import print\_function  import sys  from pyspark.sql import SparkSession  #get all the dateHour and url pairs  def getDateHourUrl(line):      columns = line.split(",")      dateHour = columns[1].split(":")[0]      url = columns[2]      return ( dateHour, url )  if \_\_name\_\_ == "\_\_main\_\_":      """          Spark      """      spark = SparkSession\          .builder\          .appName("QueryOne")\          .getOrCreate()      spark.sparkContext.setLogLevel("WARN")        #----------      #Query One      #run on s3 file no args specified, else run on local files      if len(sys.argv) > 1:          print("WARNING: reading from local filesystem")          logs = spark.sparkContext.textFile("input\_files/\*.csv")      else:          print("WARNING: reading from S3")          logs = spark.sparkContext.textFile("s3://csci.e-88.principles.of.big.data.processing/input\_files/\*.csv")      #map, group, sort and collect results      collection = logs.map(getDateHourUrl).distinct()      queryOut = collection.map(lambda x: (x[0], x[1]))\          .groupByKey()\          .mapValues(lambda vals: len(set(vals)))\          .sortByKey()\      if len(sys.argv) > 1:          queryOut.saveAsTextFile("spark\_q1")      else:          queryOut.saveAsTextFile("s3://csci.e-88.principles.of.big.data.processing/spark\_queries/spark\_emr\_q1")      print("DONE")      #-----------        spark.stop()  Query 2  from \_\_future\_\_ import print\_function  import sys  from pyspark.sql import SparkSession  #get all the dateHourUrl and user pairs  def getDateHourUrlUser(line):      columns = line.split(",")      dateHour = columns[1].split(":")[0]      url = columns[2]      user = columns[3]      return ( "::".join([dateHour, url]), user )  if \_\_name\_\_ == "\_\_main\_\_":      """          Spark      """      spark = SparkSession\          .builder\          .appName("QueryTwo")\          .getOrCreate()      spark.sparkContext.setLogLevel("WARN")        #----------      #Query Two      #run on s3 file no args specified, else run on local files      if len(sys.argv) > 1:          print("WARNING: reading from local filesystem")          logs = spark.sparkContext.textFile("input\_files/\*.csv")      else:          print("WARNING: reading from S3")          logs = spark.sparkContext.textFile("s3://csci.e-88.principles.of.big.data.processing/input\_files/\*.csv")      #map, group, sort and collect results      #mapValues with set is used to remove duplicates      collection = logs.map(getDateHourUrlUser)      queryOut = collection.map(lambda x: (x[0], x[1]))\          .groupByKey()\          .mapValues(lambda vals: len(set(vals)))\          .sortByKey()\      if len(sys.argv) > 1:          queryOut.saveAsTextFile("spark\_q2")      else:          queryOut.saveAsTextFile("s3://csci.e-88.principles.of.big.data.processing/spark\_queries/spark\_emr\_q2")      print("DONE")      #-----------        spark.stop()  Query 3  from \_\_future\_\_ import print\_function  import sys  from pyspark.sql import SparkSession  #get all the dateHourUrl and user pairs  def getDateHourUrlUser(line):      columns = line.split(",")      dateHour = columns[1].split(":")[0]      url = columns[2]      user = columns[3]      return ( "::".join([dateHour, url]), user )  if \_\_name\_\_ == "\_\_main\_\_":      """          Spark      """      spark = SparkSession\          .builder\          .appName("QueryThree")\          .getOrCreate()      spark.sparkContext.setLogLevel("WARN")        #----------      #Query Three      #run on s3 file no args specified, else run on local files      if len(sys.argv) > 1:          print("WARNING: reading from local filesystem")          logs = spark.sparkContext.textFile("input\_files/\*.csv")      else:          print("WARNING: reading from S3")          logs = spark.sparkContext.textFile("s3://csci.e-88.principles.of.big.data.processing/input\_files/\*.csv")      #map, group, sort and collect results      #nearly identical to query two, but mapValue uses a list instead of set to keep duplicates      collection = logs.map(getDateHourUrlUser)      queryOut = collection.map(lambda x: (x[0], x[1]))\          .groupByKey()\          .mapValues(lambda vals: len(list(vals)))\          .sortByKey()\      if len(sys.argv) > 1:          queryOut.saveAsTextFile("spark\_q3")      else:          queryOut.saveAsTextFile("s3://csci.e-88.principles.of.big.data.processing/spark\_queries/spark\_emr\_q3")      print("DONE")      #-----------        spark.stop() |

Show commands you used to run your jobs in the AWS EMR; include screenshots of the console output of the summary of each job (see example) [10 points]

|  |
| --- |
| Query 1      Query 2      Query 3 |

show execution stages and input/output sizes for each job [10 points]

|  |
| --- |
| Query 1          Query 2          Query 3 |

Show results of your jobs for the specified keys [10 points]

|  |
| --- |
| Query 1:  <date\_hour>, <url\_count>  2019-09-12:13, ?? 185 2019-09-12:14, ?? 186 2019-09-12:15, ?? 185 2019-09-12:16, ?? 190 2019-09-12:17, ?? 189  Query 2  <date:hour:url>, unique\_user\_count  2019-09-12:02:http://example.com/?url=003, ?? 1 2019-09-12:02:http://example.com/?url=004, ?? 3 2019-09-12:02:http://example.com/?url=005, ?? 4 2019-09-12:02:http://example.com/?url=006, ?? 10  Query 3  <date:hour:url>, event\_count  2019-09-12:02:http://example.com/?url=003, ?? 1 2019-09-12:02:http://example.com/?url=004, ?? 3  2019-09-12:02:http://example.com/?url=005, ?? 5 2019-09-12:02:http://example.com/?url=006, ?? 10 |

**Problem 2: Spark with Parquet [points: 20]**

Paste your job source code into the following area [5 points]

|  |
| --- |
| from \_\_future\_\_ import print\_function  import sys  from pyspark.sql import SparkSession  if \_\_name\_\_ == "\_\_main\_\_":      """          Spark      """      spark = SparkSession\          .builder\          .appName("WriteParquet")\          .getOrCreate()      spark.sparkContext.setLogLevel("WARN")        #----------      #      #run on s3 file no args specified, else run on local files      if len(sys.argv) > 1:          print("WARNING: reading from local filesystem")          df = spark.read.format("csv").load("input\_files/\*.csv")          df.write.parquet("hw5\_problem2.parquet")        else:          print("WARNING: reading from S3")          df = spark.read.format("csv").load("s3a://csci.e-88.principles.of.big.data.processing/input\_files/\*.csv")          df.write.parquet("s3a://csci.e-88.principles.of.big.data.processing/hw5\_problem2.parquet")      print("DONE")      #-----------        spark.stop() |

Include screenshots of your job execution and listing of the resulting Parquet files [5 points]

|  |
| --- |
|  |

Show execution and results of the parquet-tools 'meta' command. Show info about the Row groups and compression type from the meta info [10 points]

|  |
| --- |
| The parquet files have one row group, and has SNAPPY compression type  ‘meta’ Command    Part 0    Part 1    Part 2    Part 3 |

**Problem 3: Spark Dataframe - time range queries [points: 20]**

Paste your job source code into the following area [10 points]

|  |
| --- |
| from \_\_future\_\_ import print\_function  import sys  from pyspark.sql import SparkSession  from pyspark.sql.functions import \*  if \_\_name\_\_ == "\_\_main\_\_":      """          Spark      """      spark = SparkSession\          .builder\          .appName("QueryFour")\          .getOrCreate()      spark.sparkContext.setLogLevel("WARN")        #----------      #Query Four      #run on s3 file no args specified, else run on local files      if len(sys.argv) > 1:          print("WARNING: reading from local filesystem")          df = spark.read.load("hw5\_problem2.parquet")      else:          print("WARNING: reading from S3")          df = spark.read.load("s3n://csci.e-88.principles.of.big.data.processing/hw5\_problem2.parquet")      print('== read parquet ==')      df.explain()    #explain      # select datetime, country, urls columns      dfReduced = df.select(from\_unixtime(unix\_timestamp('\_c1', "yyyy-MM-dd'T'HH:mm:ss")).alias('datetime')\                              ,col('\_c4').alias('country')\                              ,col('\_c2').alias('url')\                              )        print('== select datetime, country, urls columns ==')      dfReduced.explain()     #explain      # specify datetime range      t1 = '2019-09-13 17:00:00'      t2 = '2019-09-14 09:00:00'      # filter by datetime range, add datehour column, remove duplicate rows      dfReduced = dfReduced\          .filter(dfReduced["datetime"] >= lit(t1))\          .filter(dfReduced["datetime"] <= lit(t2))\          .withColumn(  'datehour' , date\_format(col("datetime"), "yyyy-MM-dd:HH") )\          .select('datehour','country','url').drop\_duplicates()      print('== filter by datetime range, add datehour column, remove duplicate rows ==')      dfReduced.explain()     #explain        queryOut = dfReduced.groupBy("datehour", "country").agg(count("\*")).orderBy('datehour','country', ascending=True)        print('== get count of unique urls by country by hour ==')      queryOut.explain()      #explain      #run on s3 file no args specified, else run on local files      if len(sys.argv) > 1:          print("WARNING: writing to local filesystem")          queryOut.coalesce(1).write.csv('hw5\_problem3\_q4.csv')      else:          print("WARNING: writing to S3")          queryOut.coalesce(1).write.csv("s3a://csci.e-88.principles.of.big.data.processing/spark\_queries/hw5\_problem3\_q4\_emr.csv")      print("DONE")      #-----------        spark.stop() |

Include screenshots of your job execution [5 points]

|  |
| --- |
| Based on the execution plan shown in the screenshots via the explain() command, the program takes advantage of the columnar-based format. The final physical plan, which is the only one that becomes an action, shows that the program only needs to read columns \_c1 \_c2 \_c3, which is only 3 out of the total 9 columns in the data to complete the query. |

Show results of your query for the specified keys [10 points]

|  |
| --- |
| <date,hour,country>, url\_count  2019-09-13:19,IQ, ?? 1 2019-09-13:19,IR, ?? 4 2019-09-13:19,IS, ?? 9 2019-09-13:19,IT, ?? 2 2019-09-13:19,JE, ?? 4 |

**Problem 4: SparkSQL [points: 20]**

Paste your jobs source code into the following area [5 points]

|  |
| --- |
| from \_\_future\_\_ import print\_function  import sys  from pyspark.sql import SparkSession  if \_\_name\_\_ == "\_\_main\_\_":      """          Spark      """      spark = SparkSession\          .builder\          .appName("QueryThreeSQL")\          .getOrCreate()      spark.sparkContext.setLogLevel("WARN")        #----------      #Query3: get count of events per URL for a specified hour: 2019-09-12:14      #run on s3 file no args specified, else run on local files      if len(sys.argv) > 1:          print("WARNING: reading from local filesystem")          df = spark.read.load("hw5\_problem2.parquet")      else:          print("WARNING: reading from S3")          df = spark.read.load("s3a://csci.e-88.principles.of.big.data.processing/hw5\_problem2.parquet")      print('== read parquet ==')      df.explain()    #explain      # Register the DataFrame as a SQL temporary view      df.createOrReplaceTempView("allData")      # Use SQL commands to get url events for specified hour      selectedHourDF = spark.sql("""          SELECT \_c1, \_c2 FROM allData          WHERE \_c1 LIKE '%2019-09-12T14%'          """)      print('== select datetime, url columns and filter by specified hour ==')      selectedHourDF.explain()    #explain      # Register the DataFrame as a SQL temporary view      selectedHourDF.createOrReplaceTempView("selectedHourData")      # Use SQL commands to get event counts for each url      queryOut = spark.sql("""          SELECT \_c2, COUNT(\_c2)          FROM selectedHourData          GROUP BY \_c2          """).orderBy('\_c2', ascending=True)      print('== get event counts for each url ==')      queryOut.explain()    #explain      #run on s3 file no args specified, else run on local files      if len(sys.argv) > 1:          print("WARNING: writing to local filesystem")          queryOut.coalesce(1).write.csv('hw5\_problem4\_q3sql.csv')      else:          print("WARNING: writing to S3")          queryOut.coalesce(1).write.csv("s3a://csci.e-88.principles.of.big.data.processing/spark\_queries/hw5\_problem4\_q3sql\_emr.csv")      print("DONE")      #-----------        spark.stop() |

Include screenshots of your job execution and results for hour: 2019-09-12:14 [5 points]

|  |
| --- |
| http://example.com/?url=001,1  http://example.com/?url=002,1  http://example.com/?url=003,3  http://example.com/?url=004,3  http://example.com/?url=005,7  http://example.com/?url=006,3  http://example.com/?url=007,2  http://example.com/?url=008,2  http://example.com/?url=009,5  http://example.com/?url=010,2  http://example.com/?url=011,1  http://example.com/?url=012,2  http://example.com/?url=013,2  http://example.com/?url=014,3  http://example.com/?url=015,3  http://example.com/?url=016,3  http://example.com/?url=017,3  http://example.com/?url=018,3  http://example.com/?url=019,1  http://example.com/?url=020,1  http://example.com/?url=021,5  http://example.com/?url=022,4  http://example.com/?url=023,3  http://example.com/?url=025,2  http://example.com/?url=026,6  http://example.com/?url=027,3  http://example.com/?url=028,2  http://example.com/?url=029,1  http://example.com/?url=030,2  http://example.com/?url=031,1  http://example.com/?url=032,3  http://example.com/?url=034,2  http://example.com/?url=035,4  http://example.com/?url=036,5  http://example.com/?url=037,5  http://example.com/?url=038,1  http://example.com/?url=039,1  http://example.com/?url=040,2  http://example.com/?url=041,3  http://example.com/?url=042,3  http://example.com/?url=044,2  http://example.com/?url=045,5  http://example.com/?url=046,4  http://example.com/?url=047,3  http://example.com/?url=048,7  http://example.com/?url=049,2  http://example.com/?url=050,4  http://example.com/?url=051,5  http://example.com/?url=052,2  http://example.com/?url=053,3  http://example.com/?url=054,2  http://example.com/?url=055,5  http://example.com/?url=056,5  http://example.com/?url=057,5  http://example.com/?url=058,3  http://example.com/?url=060,1  http://example.com/?url=061,1  http://example.com/?url=062,2  http://example.com/?url=063,3  http://example.com/?url=064,1  http://example.com/?url=065,2  http://example.com/?url=066,4  http://example.com/?url=067,1  http://example.com/?url=068,7  http://example.com/?url=069,2  http://example.com/?url=070,2  http://example.com/?url=071,2  http://example.com/?url=072,4  http://example.com/?url=073,2  http://example.com/?url=074,2  http://example.com/?url=075,1  http://example.com/?url=076,5  http://example.com/?url=077,2  http://example.com/?url=078,3  http://example.com/?url=079,5  http://example.com/?url=081,2  http://example.com/?url=082,2  http://example.com/?url=083,5  http://example.com/?url=084,1  http://example.com/?url=085,6  http://example.com/?url=086,2  http://example.com/?url=087,5  http://example.com/?url=088,5  http://example.com/?url=089,6  http://example.com/?url=090,2  http://example.com/?url=091,4  http://example.com/?url=092,6  http://example.com/?url=093,1  http://example.com/?url=094,2  http://example.com/?url=095,5  http://example.com/?url=096,3  http://example.com/?url=097,1  http://example.com/?url=098,5  http://example.com/?url=099,4  http://example.com/?url=100,6  http://example.com/?url=101,1  http://example.com/?url=102,3  http://example.com/?url=103,2  http://example.com/?url=105,3  http://example.com/?url=106,3  http://example.com/?url=107,3  http://example.com/?url=108,9  http://example.com/?url=109,2  http://example.com/?url=110,2  http://example.com/?url=111,5  http://example.com/?url=112,2  http://example.com/?url=113,1  http://example.com/?url=114,2  http://example.com/?url=115,2  http://example.com/?url=116,2  http://example.com/?url=117,6  http://example.com/?url=118,4  http://example.com/?url=120,2  http://example.com/?url=121,1  http://example.com/?url=122,4  http://example.com/?url=124,4  http://example.com/?url=125,3  http://example.com/?url=126,2  http://example.com/?url=127,4  http://example.com/?url=128,1  http://example.com/?url=129,4  http://example.com/?url=130,2  http://example.com/?url=131,1  http://example.com/?url=132,2  http://example.com/?url=133,3  http://example.com/?url=134,3  http://example.com/?url=135,3  http://example.com/?url=137,2  http://example.com/?url=138,2  http://example.com/?url=139,3  http://example.com/?url=140,2  http://example.com/?url=141,5  http://example.com/?url=142,2  http://example.com/?url=143,2  http://example.com/?url=144,5  http://example.com/?url=145,4  http://example.com/?url=146,1  http://example.com/?url=147,2  http://example.com/?url=148,3  http://example.com/?url=149,1  http://example.com/?url=150,3  http://example.com/?url=151,2  http://example.com/?url=152,4  http://example.com/?url=153,5  http://example.com/?url=154,2  http://example.com/?url=155,6  http://example.com/?url=156,3  http://example.com/?url=157,3  http://example.com/?url=158,3  http://example.com/?url=159,2  http://example.com/?url=160,1  http://example.com/?url=162,8  http://example.com/?url=163,2  http://example.com/?url=164,1  http://example.com/?url=165,3  http://example.com/?url=167,5  http://example.com/?url=168,5  http://example.com/?url=169,2  http://example.com/?url=170,6  http://example.com/?url=171,2  http://example.com/?url=172,2  http://example.com/?url=173,2  http://example.com/?url=174,3  http://example.com/?url=175,1  http://example.com/?url=176,2  http://example.com/?url=177,4  http://example.com/?url=178,2  http://example.com/?url=180,7  http://example.com/?url=181,3  http://example.com/?url=183,3  http://example.com/?url=184,2  http://example.com/?url=185,5  http://example.com/?url=186,2  http://example.com/?url=187,4  http://example.com/?url=188,2  http://example.com/?url=189,4  http://example.com/?url=190,2  http://example.com/?url=191,3  http://example.com/?url=192,2  http://example.com/?url=193,1  http://example.com/?url=194,3  http://example.com/?url=195,3  http://example.com/?url=196,2  http://example.com/?url=197,1  http://example.com/?url=198,6  http://example.com/?url=199,1 |

Show execution plan of the query and demo/explain whether it took advantage of the Parquet columnar format (or not) [10 points]

|  |
| --- |
| Based on the execution plan shown in the screenshots via the explain() command, the program takes advantage of the columnar-based format. The final physical plan, which is the only one that becomes an action, shows that the program only needs to read columns \_c1 and \_c2, which is only 2 out of the total 9 columns in the data to complete the query. |

**Problem B1: Bonus: Top N queries** [15 points]

Paste your job source code into the following area [5 points]

|  |
| --- |
| from \_\_future\_\_ import print\_function  import sys  from pyspark.sql import SparkSession  if \_\_name\_\_ == "\_\_main\_\_":      """          Spark      """      spark = SparkSession\          .builder\          .appName("QueryFive")\          .getOrCreate()      spark.sparkContext.setLogLevel("WARN")        #----------      #Query5      #run on s3 file no args specified, else run on local files      if len(sys.argv) > 1:          print("WARNING: reading from local filesystem")          df = spark.read.load("hw5\_problem2.parquet")      else:          print("WARNING: reading from S3")          df = spark.read.load("s3a://csci.e-88.principles.of.big.data.processing/hw5\_problem2.parquet")      # Register the DataFrame as a SQL temporary view      df.createOrReplaceTempView("allData")      # Downselect data      avgTtfbDF = spark.sql("""          WITH cte AS              ( SELECT SUBSTRING(\_c1,1,10) AS day, \_c2 AS url, \_c8 AS ttfb              FROM allData )          SELECT day,url,AVG(ttfb) as avg\_ttfb FROM cte          GROUP BY day,url          ORDER BY day,url,avg\_ttfb          """)      # Register the DataFrame as a SQL temporary view      avgTtfbDF.createOrReplaceTempView("avgData")      # Downselect data      queryOut = spark.sql("""          WITH cte AS              ( SELECT day, url, avg\_ttfb, ROW\_NUMBER() OVER (PARTITION BY day ORDER BY avg\_ttfb ASC) AS ttfb\_rank              FROM avgData )          SELECT day,url,avg\_ttfb FROM cte          WHERE ttfb\_rank <= 5          ORDER BY day,avg\_ttfb          """)      #run on s3 file no args specified, else run on local files      if len(sys.argv) > 1:          print("WARNING: writing to local filesystem")          queryOut.coalesce(1).write.csv('hw5\_problem5.csv')      else:          print("WARNING: writing to S3")          queryOut.coalesce(1).write.csv("s3a://csci.e-88.principles.of.big.data.processing/spark\_queries/hw5\_problem5\_emr.csv")      print("DONE")      #-----------        spark.stop() |

Explain your solution approach [5 points]

|  |
| --- |
| The solution approach is to use SQL commands in Spark. First, we get the day, url and ttfb columns from the master data set. Then we group the rows by day+url and aggregate the average ttfb for each day+url group. Finally, we sort by the average ttfb for each day, and keep lowest 5 average ttfb and corresponding url for each day. |

Show results of your query [5 points]

|  |
| --- |
| 2019-09-12,http://example.com/?url=114,0.3931014084507042  2019-09-12,http://example.com/?url=101,0.40254500000000004  2019-09-12,http://example.com/?url=133,0.4133171875  2019-09-12,http://example.com/?url=033,0.41886785714285724  2019-09-12,http://example.com/?url=157,0.419289393939394  2019-09-13,http://example.com/?url=039,0.40281428571428574  2019-09-13,http://example.com/?url=110,0.4294693548387098  2019-09-13,http://example.com/?url=005,0.4306358490566038  2019-09-13,http://example.com/?url=006,0.4313811594202898  2019-09-13,http://example.com/?url=028,0.43190645161290314  2019-09-14,http://example.com/?url=123,0.40559807692307687  2019-09-14,http://example.com/?url=070,0.4185893333333334  2019-09-14,http://example.com/?url=124,0.42331587301587303  2019-09-14,http://example.com/?url=107,0.42657058823529415  2019-09-14,http://example.com/?url=101,0.4294369230769231  2019-09-15,http://example.com/?url=185,0.2863  2019-09-15,http://example.com/?url=175,0.6621  2019-09-15,http://example.com/?url=167,0.7849  2019-09-15,http://example.com/?url=008,0.8516 |