Exam 3 (MW) - Results

Attempt 1 of 1

Written Dec 3, 2025 9:59 AM - Dec 3, 2025 11:29 AM

Attempt Score **56 / 100 - 56 %**
Overall Grade (Highest Attempt) **56 / 100 - 56 %**

Multiple Choices, 40 points

Question 1 — 2 / 2 points

Which of the following is NOT among the supported parameters in REST API of Firebase realtime database?

- ○ orderBy
- ○ limitToFirst
- ○ startAt
- ✓ ◉ groupBy

Question 2 — 0 / 2 points

Which of the following data formats is most structured?

- ○ JSON
- ✗ ◉ XML
- ➡ ○ Relation
- ○ Text

Question 3 — 0 / 2 points

Which of the following MongoDB operators is used to join two MongoDB collections?

- ○ $group
- ○ $match
- ➡ ○ $lookup
- ✗ ◉ $unwind

Question 4 — 0 / 2 points

Which of the following is most likely to be the page size of SSD?

- ○ 4B
- ➡ ○ 4KB
- ✗ ◉ 4MB
- ○ 4GB

Question 5 — 0 / 2 points

Which operations in SSD are conducted page by page?

- ○ Read only
- ➡ ○ Read and write
- ○ Write only
- ✗ ◉ Erase only

Question 6 — 3 / 3 points

Which of the following is the correct order of tasks being executed in Hadoop MapReduce?

- ✓ ◉ Map, shuffle, reduce
- ○ Map, reduce, shuffle
- ○ Shuffle, map, reduce
- ○ Reduce, map, shuffle

Question 7 — 3 / 3 points

Consider joining two relations R and S with the amount of memory M (pages). Sizes (i.e., the number of blocks) of R and S are denoted as B(R) and B(S) respectively. Suppose both B(R) and B(S) is larger than M. Which of the following statements about choices of join algorithms is correct?

- ○ Nested-loop join algorithm always has higher cost than sort-merge join algorithm.
- ○ Nested-loop join algorithm always has higher cost than partitioned-hash join algorithm.
- ○ Sort-merge join algorithm always has the same cost as the partitioned hash join algorithm.
- ✓ ◉ If B(R) < B(S), then placing R in the outer loop of the nested-loop join will incur a lower cost.

Question 8 — 0 / 3 points

Which of the following transformations in Spark involves shuffling?

- ○ map
- ✗ ◉ filter
- ○ flatMap
- ➡ ○ reduceByKey

Question 9 — 3 / 3 points

Suppose there are some empty author elements in a book XML document. Which of the following XPath expressions correctly finds such elements?

- ○ //author[node()]
- ○ //author[*]
- ✓ ◉ //author[not(node())]
- ○ //author[not(text())]

Question 10 — 3 / 3 points

Consider an XML element about book: <book isbn=1234> <author>John</author> <author>123</author> <isbn>5678</isbn> <title>Database</title> </book>What syntax error does the above book element have?

- ○ It has isbn as both attribute and sub-element.
- ✓ ◉ The value of the Isbn attribute (1234) needs to be quoted.
- ○ It can not have multiple author subelements.
- ○ None of above.

Question 11 — 0 / 3 points

Suppose the actual bandwidth of reading 100 and 200 random blocks of data from a hard drive is B1 and B2 respectively. Which of the following statements about the relationship between B1 and B2 is most accurate?

- ✗ ◉ B1 < B2
- ○ B1 > B2
- ➡ ○ B1 = B2
- ○ B2 = 2*B1

Question 12 — 3 / 3 points

How many HDFS blocks are needed to store 1GB of data?

- ○ 2
- ○ 4
- ✓ ◉ 8
- ○ 10

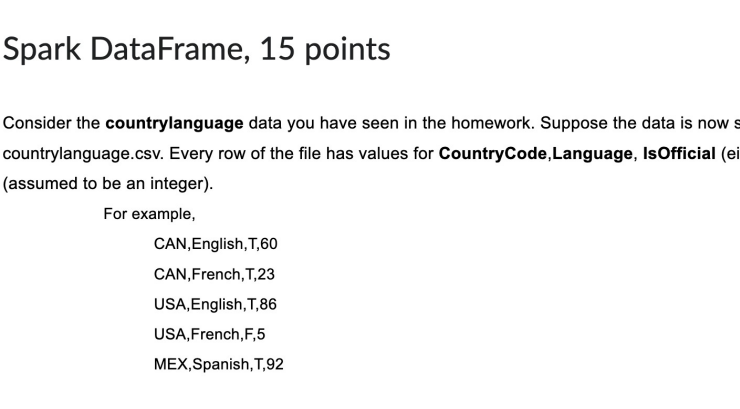
Question 13 — 3 / 3 points

Consider two relations: Product(id int primary key, title varchar(20)); Sales(id int primary key, qty float, foreign key(id) references Product(id)); Suppose Product has 100 rows and Sales has 1000 rows. How many rows will the following query produce: Select * from Product natural join Sales;

- ○ 100
- ✓ ◉ 1000
- ○ 900
- ○ 1100

Question 14 — 3 / 3 points

The following diagram you have seen in class illustrates the shuffling process in Hadoop MapReduce. Recall that flowcharts in boxes marked as map and reduce tasks describe the steps involved in the tasks. How many map tasks and reduce tasks does the MapReduce job illustrated by the diagram have?



- ○ 3 map and 4 reduce tasks
- ✓ ◉ 4 map and 3 reduce tasks
- ○ 3 map and 3 reduce tasks
- ○ 4 map and 4 reduce tasks

Question 15 — 3 / 3 points

Continue from the above question. Which tasks will be merging runs?

- ○ Map task only
- ○ Reduce task only
- ✓ ◉ Both map and reduce tasks
- ○ Neither map nor reduce tasks

Spark DataFrame, 15 points

Consider the **countrylanguage** data you have seen in the homework. Suppose the data is now stored in a CSV file, countrylanguage.csv. Every row of the file has values for **CountryCode**,**Language**, **IsOfficial** (either 'T' or 'F'), and **Percentage** (assumed to be an integer).

For example,

CAN,English,T,60
CAN,French,T,23
USA,English,T,86
USA,French,F,5
MEX,Spanish,T,92

Further suppose the **country** data is stored in country.csv and it has only two columns: **Code** and **Name**. For example,

CAN,Canda
USA,United States
MEX,Mexico

Suppose a Spark dataframe called "cl" has been created for you:

cl = spark.read.csv('countrylangauge.csv', …)
country = spark.read.csv('country.csv', …)

The dataframe also has four columns with data and format corresponding to that in the csv file, that is, the first attributes are strings and the last attribute is an integer.

Also the following import statement has been executed.

import pyspark.sql.functions as fc

Using the cl (and fc), write a Spark Dataframe script for each of the following SQL queries. Note your script should not return extra information that SQL queries are not asking for, and should return the attributes with names stated in the SQL queries.

Question 16 — 3 / 6 points

Select Language, max(Percentage) as max_p
From CountryLanguage
Where IsOfficial = 'T'
Group by Language
Having count(*) >= 10;

```
result = (
 cl.filter(fc.col("IsOfficial") == "T")
  .groupBy(fc.col("Language"))
  .filter(fc.count("*") >= 10)
  .select(
      "Language",
      fc.max(fc.col("Percentage")).alias("max_p")
  )
).show()
```

The correct answer is not displayed for Written Response type questions.

▼ Hide question 16 feedback

Feedback

-1 incorrect usage of groupby
-1 missing aggregation
-1 missing count in aggregation

Question 17 — 4 / 5 points

Select country.Name, Language
From country natural join countrylanguage
Where IsOfficial = 'T';

```
joined = country.join(cl, country["CountryCode"]==cl["Code"], "inner")
result = joined.filter(fc.col("IsOfficial") == "T")
               .select(fc.col("country.Name"), fc.col("cl.Language").alias("Language"))
result.show()
```

The correct answer is not displayed for Written Response type questions.

▼ Hide question 17 feedback

Feedback

-1 incorrect join, wrong column names for both dataframes

Question 18 — 3 / 4 points

(select Language
from countrylanguage
where IsOfficial = 'F' and CountryCode = "CAN")
except
(select Language
from countrylanguage
where IsOfficial = 'F' and CountryCode = "USA");

```
nonofficial_lang_in_can = \
  cl.filter((fc.col("IsOfficial")=="F") & (fc.col("CountryCode")=="CAN"))  \
    .select("Language")

nonofficial_lang_in_usa = \
   cl.filter((fc.col("IsOfficial")=="F") & (fc.col("CountryCode")=="USA"))  \
     .select("Language")
```

The correct answer is not displayed for Written Response type questions.

▼ Hide question 18 feedback

Feedback

-1 missing subtraction

Query execution, 15 points

Now suppose the country and countrylangauage data are stored in an RDBMS server. Consider the following SQL query:

Select *
From country join countrylanguage
on country.Code = countrylanguage.CountryCode

Suppose we have the following situation:

- B(country) = 5,000, that is, country table occupies 5000 blocks on storage device.
- B(countrylanguage) = 10,000.
- M = 102, that is, there are 102 pages in main memory that can be used for the algorithm.

Question 19 — 5 / 5 points

Suppose the server decides to use the (block-based) nested-loop join algorithm, with the country table placed in the outer loop.

Describe the algorithm with pseudo code.

```
# batch 5000 blocks into 100 pages, where each page contains 5000/100=50 blocks
country_batches = batch(country_blocks, M-2)

result = []
for country_batch in country_batches:  # outer loop: 50 times
    for cl_block in countrylanguage_blocks:  # inner loop: 10,000 times
        result.extend(JOIN(country_batch, cl_block, on="country.Code==countrylanguage.CountryCode"))
  # join the batch (50 blocks of country) with cl_block (1 block)

# We can further implement a simple JOIN using nested-loop by entries:
def JOIN(block1, block2, condition):
    joined = []
    for e1 in block1:
        for e2 in block2:
            if condition(e1, e2):
                joined.append(e1 + e2)  # here "+" means concatination
    return joined
```

The correct answer is not displayed for Written Response type questions.

Question 20 — 2 / 2 points

Suppose the server decides to use the (block-based) nested-loop join algorithm, with the country table placed in the outer loop.

What is the cost (block I/O's) of the algorithm? Show your derivation.

Pages used to batch the blocks = M - pages used for I/O buffers = 102 - 2 = 100
#iter in outer loop = B(country) / 100 pages = 50
#iter in inner loop = B(countrylanguage) = 10,000
Cost in loading country table = B(country) = 5,000
Cost in loops = #iter in outer loop * #iter in inner loop = 50 * 10,000 = 500,000
Total Cost = 5,000 + 500,000 = 505,000

The correct answer is not displayed for Written Response type questions.

Question 21 — 1 / 6 points

Now suppose the server uses the sort-merge join algorithm instead. Assume that 100 blocks of memory is used for sorting and 101 blocks for merging.

Describe the steps of executing the algorithm on the above tables. Be sure to indicate the number of runs and the size of runs at each step.

Sorting pass:
Sort(country) -> 2*B(country) = 2*5,000 = 10,000
(Because we need another B(country) to perform the sorting besides B(country) blocks storage)
Sort(countrylanguage) -> 2*B(countrylanguage) = 2*10000 = 20,000

Merging pass:
In outer loop, iterate sorted country table.
For each line in the block, join its key value with countrylanguage table, if the key (Code) equal to countrylanguage table's key (CountryCode)
since the B(country)+B(countrylanguage)=10,500 > (M-1)^2=10,000
we need 2*B(country)=10,000 runs to fit into the buffer
Then we need B(country)+B(countrylanguage) for merging

The correct answer is not displayed for Written Response type questions.

▼ Hide question 21 feedback

Feedback

+1 partial attempt

Question 22 — 1 / 2 points

Now suppose the server uses the sort-merge join algorithm instead. Assume that 100 blocks of memory is used for sorting and 101 blocks for merging.

What is the total cost (block I/O's) of the algorithm on joining the two tables? Show your derivation.

Sort(country) -> 2*B(country) = 2*5,000 = 10,000
Sort(countrylanguage) -> 2*B(countrylanguage) = 2*10000 = 20,000
we need 2*B(country)=10,000 runs to fit into the buffer
Then we need B(country)+B(countrylanguage) for merging

Total cost: 5*B(country)+3*B(countrylanguage)=55,000 I/O's

The correct answer is not displayed for Written Response type questions.

▼ Hide question 22 feedback

Feedback

+1 correct merging cost

Hadoop MapReduce, 15 points

Consider the same countrylangauge.csv data that you have seen in the Spark DataFrame question above. Schema and sample data are reproduced here:

===
Every row of the file has values for **CountryCode**, **Language**, **IsOfficial** (either 'T' or 'F'), and **Percentage**.
For example,
CAN,English,T,60
CAN,French,T,23
USA,English,T,86
USA,French,F,5
MEX,Spanish,T,92
===

Consider an SQL query:

Select Language, max(Percentage)
From CountryLanguage
Where IsOfficial = 'T'
Group by Language
Having count(*) > 1

Consider finding the answer to the above SQL query using Hadoop MapReduce.

Question 23 — 4 / 5 points

Write a pseudocode for the map function of map task.

```
def Map(row):
    if val == "T":
        output(row[1], (row[2], 1))
        # Key: Language; Value: Percentage, and a constant 1 for counting
    else:
         output(row[1], (None, 0))
         # Since each row should be mapped into a value, even rows failed to be filtered need to "hold a place" here
```

The correct answer is not displayed for Written Response type questions.

▼ Hide question 23 feedback

Feedback

-1 inputs for map function should be key value pair

Question 24 — 0 / 2 points

State the output from the map task for the above sample data.

We input with an entire row (i.e. the Key is the row, and the Value is none), and map the row into (row[1], (row[2], 1)), where:
1. the key row[1] is the Language column that we need to group by
2. the value is a tuple
2.1. the first element row[2] is the Percentage column we use to aggregate (MAX)
2.2. the second element is a constant 1, also used to aggregate (COUNT)

The correct answer is not displayed for Written Response type questions.

▼ Hide question 24 feedback

Feedback

-2 incorrect

Question 25 — 1 / 5 points

Write a pseudocode for the reduce function of reduce task.

```
def ReduceCount(reduced, key, val):
     output(reduced + val[1])  # this is a simple SUM reduce, but in our situation, each val[1] is 1, so this becomes a COUNT(*)


def ReduceMax(reduced, key, val):
    output(max(reduced, val)
```

The correct answer is not displayed for Written Response type questions.

▼ Hide question 25 feedback

Feedback

+1 partial attempt

Question 26 — 0 / 3 points

State the input and output of **each** call to the reduce function executed for the above sample data.

Input of ReduceCount(reduced, key, val):
key: Code, val: 1

Input of ReduceMax(reduced, key, val):
key: Code, val: Percentage

The correct answer is not displayed for Written Response type questions.

▼ Hide question 26 feedback

Feedback

-3 incorrect

Spark RDD, 15 points

Note in this question, "sc" refers to a SparkContext object.

Question 27 — 1 / 5 points

Consider the following Spark RDD code:

lst = [1, 2, 3, 4, 5, 6] # note lst may take other values
data = sc.parallelize(lst, 2)

Write a single RDD aggregate function to compute both sum and count of values in data.

```
sum_rdd = data.map(lambda x: (x, x)).reduceByKey(lambda x: x[1][0]+x[1][1])
sum_rdd.collect()

count_rdd = data.map(lambda x: (x, 1)).reduceByKey(lambda x: x[1][0]+x[1][1])
count_rdd.collect()
```

The correct answer is not displayed for Written Response type questions.

▼ Hide question 27 feedback

Feedback

+1 partial attempt

Question 28 — 1 / 2 points

Consider the following Spark RDD code:

lst = [1, 2, 3, 4, 5, 6] # note lst may take other values
data = sc.parallelize(lst, 2)

Explain how the aggregation function computes both sum and count in parallel.

Spark RDD uses HDFS which allows parallel computing of many maps.
If we use data.parallelize(lst, 2), then two parallel threads will perform maps concurrently, which may result into shuffling the data before reducing.

The correct answer is not displayed for Written Response type questions.

▼ Hide question 28 feedback

Feedback

-1 no stating local sums are than merged to get final result

Question 29 — 4 / 8 points

Consider the same countrylangauge.csv data that you have seen in the Spark DataFrame question above. Schema and sample data are reproduced here:

===
Every row of the file has values for **CountryCode**, **Language**, **IsOfficial** (either 'T' or 'F'), and **Percentage**.
For example,
CAN,English,T,60
CAN,French,T,23
USA,English,T,86
USA,French,F,5
MEX,Spanish,T,92
===

Consider an SQL query:

Select Language, count(*)
From CountryLanguage
Where IsOfficial = 'T'
Group by Language

Suppose the following codes have been executed for you.

lines = sc.textFile('CountryLanguage.csv', 2)

Using the lines RDD, write a Spark RDD to find out the answer to the above SQL query.

```
results = (
  lines.filter(lambda x: x[2]=="T")
       .map(lambda x: x[1], (x[1], 1))
       .reduceByKey(lambda x: (x[0][0], x[1][0]+x[1][1])   # language, 1+1
).collect()
```

The correct answer is not displayed for Written Response type questions.

▼ Hide question 29 feedback

Feedback

+2 correct filtering function
+2 correct map function before reduce

Done