Exam 3 - Requires Respondus LockDown Browser - R...

Attempt 1 of 1

Written Apr 28, 2025 2:59 PM - Apr 28, 2025 4:48 PM

Attempt Score **93.5 / 100 - 93.5 %**

Overall Grade (Highest Attempt) **93.5 / 100 - 93.5 %**

[Spark DataFrame, 20 points]

Consider the following tables storing student information and performance in courses offered in Spring 2025.

```
students(id, name, program, gpa)
        // e.g., (100, “john smith”, “applied math”, 3.8)
        // note: id is integer, gpa is real number, and other attributes are strings.
courses(number, title, instructor)
        // e.g., (“dsci351”, “databases”, “david chu”)
take(sid, cno, score)
        // where sid is student id, cno is course number,
        //     and score is student’s total score (a real number) on the course
        // e.g., (100, “dsci351”, 95.2)
```

Assume that data are stored in CSV files (students.csv, courses.csv, and take.csv). For example, students.csv might have the following rows.

```
id,name,program,gpa
100,john smith,applied math,3.8
101,mary smith,econ,3.9
…
```

Assume the following script has been executed for you.

```
import pyspark.sql.functions as fc
```

```
students = spark.read.csv('students.csv', header=True, inferSchema=True)
courses = spark.read.csv('courses.csv', header=True, inferSchema=True)
take = spark.read.csv('take.csv', header=True, inferSchema=True)
```

For each of the following SQL queries, write a Spark DataFrame script to find the answer to the query. 5 points each question.

Question 1 — 5 / 5 points

```
select sid, avg(score)
from take
group by sid
having count(*) >= 3;
```

```
take.groupBy("sid")
.agg(fc.count("*").alias("cnt"), fc.avg("score").alias("avg_score"))
.filter("cnt >= 3")
.select("sid", "avg_score")
.show()
```

Question 2 — 4 / 5 points

```
select name , cno
from students join take on students.id = take.sid
where cno = "dsci351" or cno = “dsci551”
order by name;
```

```
t_1 = take.where((take.cno == "dsci351") or (take.cno == "dsci551"))
join_1 = students.join(t_1, students.id == t_1.sid)
join_1.select("name", "cno").orderBy("name").show()
```

Question 3 — 5 / 5 points

```
(select sid
 from take
 where cno = "dsci351")
subtract
(select sid
 from take where cno = "dsci551");
```

```
t_1 = take.filter(' cno = "dsci351" ').select("sid")
t_2 = take.filter(' cno = "dsci551" ').select("sid")
t_1.subtract(t_2).show()
```

Question 4 3 / 5 points

```
with t as (select cno, count(*) cnt
           from take group by cno)
select cno from t
where cnt = (select min(cnt) from t);
```

```
t = take.groupBy('cno')
.agg(fc.count("*").alias("cnt"))
.select("cno", "cnt")

min_cnt = t.select("cnt").min()

t.where(t.cnt == min_cnt).select("cno").show()
```

▼ Hide question 4 feedback

Feedback

1. `min_cnt = t.select("cnt").min()` should be `min_cnt = t.agg(fc.min("cnt")).first()[0]`. The `min()` function on a DataFrame will not return the minimum value as you expect; you need to aggregate it and extract the result.

2. `t.where(t.cnt == min_cnt)` is incorrect. You should use `.filter(t['cnt'] == min_cnt)` instead, to compare the column with the minimum count.

[Spark RDD, 20 points]

Consider the same tables as you have seen (duplicated below).

- students(id, name, program, gpa)
- courses(number, title, instructor)

- take(sid, cno, score)

Suppose the following scripts have been executed for you on the data stored in CSV files as before:

```
students = spark.read.csv('students.csv', header=True, inferSchema=True)
courses = spark.read.csv('courses.csv', header=True, inferSchema=True)
take = spark.read.csv('take.csv', header=True, inferSchema=True)

students_rdd = students.rdd
courses_rdd = courses.rdd
take_rdd = take.rdd
```

Write a Spark RDD script to find answers for each of the following SQL queries. 5 points each question.

Question 5 — 3 / 5 points

```
select sid, avg(score)
from take
group by sid
having count(*) >= 3;
```

```
take_rdd
.map(lambda r: (r['sid'], (r['score'], 1)))
.reduceByKey(lambda u, v: u[0]+v[0], u[1]+v[1])
.filter(lambda r: r[1][1] >= 3)
.mapByValues(lambda r: r[0]/r[1])
.collect()
```

▼ Hide question 5 feedback

Feedback

incorrect reduceByKey syntax

Question 6 — 5 / 5 points

```
select name, cno
from students join take on students.id = take.sid
where cno = "dsci351" or cno = "dsci551"
order by name;
```

```
s_1 = students_rdd.map(lambda r: (r['id'], r['name']))

t_1 = take_rdd.filter(lambda r: (r['cno'] == "dsci351") or (r['cno'] ==
"dsci551")).map(lambda r: (r["sid"], r["cno"]))

join_1 = s_1.join(t_1)

join_2 = join_1.map(lambda r: (r[1][0], r[1][1]))
join_2.sortBy(lambda r: r[0]).collect()
```

Question 7 5 / 5 points

```
(select sid
 from take
 where cno = "dsci351")
except
(select sid
 from take where cno = "dsci551");
```

```
t_1 = take_rdd.filter(lambda r: r['cno'] == "dsci351").map(lambda r: r['sid'])
t_2 = take_rdd.filter(lambda r: r['cno'] == "dsci551").map(lambda r: r['sid'])

t_1.subtract(t_2).collect()
```

Question 8 5 / 5 points

```
with t as (select cno, count(*) cnt
                from take group by cno)
select cno from t
where cnt = (select min(cnt) from t);
```

```
t = take_rdd.map(lambda r: (r['cno'], 1)).reduceByKey(lambda u, v: u+v)

min_cnt = t.map(lambda r: r[1]).reduce(lambda u, v: min(u, v))

t.filter(lambda r: r[1] == min_cnt).map(lambda r: r[0]).collect()
```

[Hadoop, 20 points]

Now consider only the students table:

students(id, name, program, gpa)

Again, assume that data are stored in students.csv with header row removed. Suppose the following are sample data:

```
100,john smith,math,3
101,mary smith,econ,4
102,david johnson,math,4
103,bill chen,cs,4
104,jennifer small,cs,3
105,john biden,math,3
```

Consider the following SQL query:

```
Select program, avg(gpa), count(*)
From students
Group by program
Having count(*) >= 2;
```

Write a Hadoop MapReduce program (in pseudo code) to find the answer to the above query, taking students.csv as the input. Assume there is only one map and one reduce task.

Question 9 5 / 5 points

Write the pseudo code for the map function.

```
for each key, values in students do:
 #key: line offset, values: line content
 split values with(",") and we got an array of token
 program = token[2]
 gpa = token[3]
 output( (program, (gpa, 1)))
```

▼ Hide question 9 feedback

Feedback

allowing missing function signature/definition because you stated inputs to the function clearly

Question 10 — 3 / 3 points

State the output key-value pairs of the map task on the above sample data.

the output key-value pairs of the map task will be like this:

(math, (3, 1))
(econ, (4, 1))
(math, (4, 1))
(cs, (4, 1))
(cs, (3, 1))
(math, (3, 1))

Question 11 — 6.5 / 7 points

Write the pseudo code for the reduce function. Note that you should NOT assume there are pre-implemented functions like len, count, sum, avg, etc. Instead, you are supposed to write these functions on your own

output of map function are sorted by key at shuffle phase

```
for each key in sorted keys do:
 float cnt = 0
 float sum = 0

 for each pair (k, v) assciated with key do
   # k: program, v: a pair of (gpa, 1)
   sum += v[0]
   cnt += v[1]
   avg = sum/cnt

 if cnt >= 2:
   output(key, avg, cnt)
```

Hide question 11 feedback

Feedback

-0.5: missing function signature/definition (not mentioned inputs to the functions)

Question 12 4 / 5 points

For each execution of the reduce function (for the sample data), state its input and output.

input:
(math, ((3, 1), (4, 1), (3, 1)))
(cs, ((4, 1),(3, 1)))

output:
(math, 3.33, 3)
(cs, 3.5, 2)

Hide question 12 feedback

Feedback

-1: missing econ input

[Query execution, 20 points]

Now suppose the same tables are stored in MySQL server. Recall the tables are:

- students(id, name, program, gpa)
- courses(number, title, instructor)
- take(sid, cno, score)

Consider the following join query:

```
select *
from take, students
where take.sid = students.id
```

Suppose the database server implements the join using the partitioned-hash join algorithm. Suppose that 101 blocks are used for partitioning steps and 102 blocks are used for the join steps.

Question 13 — 8 / 8 points

Suppose B(take) = 5, 000 (that is, the take table occupies 5,000 blocks), B(students) = 2,000.

Describe the steps in executing the algorithm for joining these two tables. Be sure to indicate the number of buckets, the size of each bucket, and how the buckets are further processed.

1.
hash take on attribute sid
100 buckets, size: 50 blocks/bucket with hash function h
e.g., t1, t2, ... t100

send all buckets to the disk

2.
hash students on attribute id with hash function h
100 buckets, size: 20 blocks/bucket
e.g., s1, s2, ... s100

send all buckets to the disk

3. since min(bucket of take, bucket of students) < (M-2) , we can directly join them in this way:

join(t1, s1)
join(t2, s2)
...
join(t100, s100)

Question 14 — 2 / 2 points

Suppose B(take) = 5, 000 (that is, the take table occupies 5,000 blocks), B(students) = 2,000.

What is the I/O cost of the algorithm? Show your derivation.

1. hash take on attribute sid

100 buckets, size: 50 blocks/bucket
Cost: 2B(take)

2. hash students on attribute id

100 buckets, size: 20 blocks/bucket
Cost: 2B(students)

3. join take and students:
Cost: B(take) + B(students)

Total cost: 3(B(take) + B(students)) = 3(5,000+2,000) = 21,000 blocks (I/O)

Question 15 8 / 8 points

Now suppose B(take) = 50,000, and B(students) = 20,000.

Describe the steps in executing the algorithm for joining these two tables. Be sure to indicate the number of buckets, the size of each bucket, and how the buckets are further processed.

1.
hash take on attribute sid with hash function h
100 buckets, size: 500 blocks/bucket
e.g., t1, t2, ... t100

send all buckets to the disk

2.
hash students on attribute id with hash function h
100 buckets, size: 200 blocks/bucket
e.g., s1, s2, ... s100

send all buckets to the disk

3. join take and students:
since the buckets of both tables exceed the number of available blocks in memory, we need to hash them again.

For each t1, t2, ... t100
we hash each take bucket on attribute sid again with a different hash function h'
100 buckets, size: 5 blocks/bucket
e.g., t1 1, t1 2, ... t100 100

send all buckets to the disk

For each s1, s2, ... s100
we hash each students bucket on attribute id again with h'
100 buckets, size: 2 blocks/bucket
e.g., s1 1, s1 2, ... s100 100

send all buckets to the disk

then we can join them now
join(t1 1, s1 1)
join(t1 2, s1 2)
...
join(t100 100, s100 100)

Question 16 2 / 2 points

Now suppose B(take) = 50,000, and B(students) = 20,000.

What is the I/O cost of the algorithm? Show your derivation.

1. hash take on attribute sid

100 buckets, size: 500 blocks/bucket
Cost: 2B(take)

2. hash students on attribute id

100 buckets, size: 200 blocks/bucket
Cost: 2B(students)

3. join take and students:
since the buckets of both tables are too big, we need to hash them again.

we hash each take bucket on attribute sid again with a different hash function h'
100 buckets, size: 5 blocks/bucket
Cost: 2B(take)

we hash each students bucket on attribute id again with h'
100 buckets, size: 2 blocks/bucket
Cost: 2B(students)

then we can join them now
Cost: B(take) + B(students)

Total cost: 5(B(take) + B(students)) = 5(50,000+20,000) = 350,000 blocks (I/O)

[Miscellaneous, 20 points]

For each of the following questions, select a **single** choice that best answers the question.

Question 17 2 / 2 points

Which of the following is NOT a valid JSON value?

- ○ {"name": 25}
- ○ ["name", "john"]
- ○ {"name": []}
- ◉ {25: "name"}

Question 18 2 / 2 points

Which of the following statements about ER model is NOT correct?

- (●) A relationship cannot have an attribute.
- ( ) An entity set can have multiple keys.
- ( ) A key can have multiple attributes.
- ( ) There may be multiple relationships between two entity sets.

Question 19 — 2 / 2 points

Consider a table defined as follows:

Create table R (a int primary key, b int unique, c int);

Which of the following statements can NOT be successfully executed?

- ( ) Create table S (a int, foreign key (a) references R(a));
- ( ) Create table S (a int, foreign key (a) references R(b));
- (●) Create table S (c int, foreign key (a) references R(c));
- ( ) Create table S (a int primary key, foreign key (a) references R(a));

Question 20 — 2 / 2 points

Which of the following MongoDB statements can correctly find the number of products in each category? For example, 3 cell phones, 2 laptops, etc.

- ( ) db.product.aggregate({$group: {_id: "category", cnt: {$sum: 1}}})
- ( ) db.product.aggregate({$group: {_id: "$category"}, cnt: {$sum: 1}})
- ( ) db.product.aggregate({$group: {_id: "category", cnt: {$count: 1}}})
- (●) db.product.aggregate({$group: {_id: "$category", cnt: {$sum: 1}}})

Question 21 — 2 / 2 points

Which of the following Spark RDD scripts correctly implements data.count()?

- ○ data.reduce(lambda U, x: U + 1)
- ○ data.aggregate(0, lambda U, x: U + x, lamba U, x: U + 1)
- ◉ data.aggregate(0, lambda U, x: U + 1, lamba U, x: U + x)
- ○ data.reduce(0, lambda U, x: U + 1, lamba U, x: U + x)

Question 22 — 2 / 2 points

Which of the following is an advantage of materialized view over virtual view?

- ○ Save storage space
- ○ Always return fresh data
- ◉ Often run faster
- ○ Supported by MySQL

Question 23 — 2 / 2 points

Consider two tables product(id, price) and sales(pid, date). Note that id is primary key of product and pid is primary key of sales. Also pid is a foreign key referring to id of product.

Suppose that product has 100 rows and sales has 1000 rows. How many rows will the natural join of these two tables produce?

- ○ 100
- ◉ 1000
- ○ 1100
- ○ 900

Question 24 — 2 / 2 points

Which attribute in the following table is NOT fixed length?

Create table person(ssn int, age tinyint, name varchar(20), program char(20));

- ○ ssn
- ○ age
- ◉ name
- ○ program

Question 25 2 / 2 points

Is it possible that an insertion into a B+-tree may increase the number of levels of the tree?

- ◉ Yes
- ○ No

Question 26 2 / 2 points

Which of the following RPC calls is NOT used in the process of writing a file in HDFS?

- ◉ getBlockLocations
- ○ create
- ○ append
- ○ addBlock

Done