DSCI-551 Final Exam

Chaoyu Li, 95/100

QUESTION 1 [Query Execution]

Consider two tables:

Country(code, name, continent, GNP, capital) and **City(id, name, population)**, where code is primary key of Country and id is primary key of City table, capital in Country is a foreign key referring to id of City.

Consider joining the two tables on Country.capital = City.id.

Suppose Country data is stored in 200 blocks, city in 500 blocks, and there are 101 blocks of memory available for the join.

(1.a.) Design a partitioned-hash algorithm for the join. Be sure to specify the steps and details of the algorithm.

A: step1: Hash Country into 100 buckets using hash function h, and send them back to disk: Cost = 2 B(Coutry) = 400 (2 blocks/bucket)

step2: Hash city into 100 buckets using hash function h, and send them back to disk: Cost = 2 B(city) = 1,000 (5 blocks/bucket)

step3: Join corresponding buckets from Country and city: Cost: B(Country) + B(city) = 700.

(1.b.) What is I/O cost of the algorithm? (Ignore the cost of writing the output of the algorithm). Show the derivation.

A: Follow the steps in 1.1, the **total cost**: 3B(R) + 3B(S) = 2100 Block I/O's.

(1.c) Now suppose the city table has 50,000 blocks instead. Explain how you would modify the above algorithm to perform the join. What is the cost of the algorithm (with the new city table)? Show your derivation.

A: step1: Hash Country into 100 buckets using hash function h, and send them back to disk: Cost = 2 B(Coutry) = 400 (2 blocks/bucket)

step2: Hash city into 100 buckets using hash function h, and send them back to disk: Cost = 2 B(city) = 100,000 (500 blocks/bucket)

step3: Cause we can still store entire one bucket of Country data(2 blocks) at a time, so we can direct join corresponding buckets from Country and city: Cost = B(Country) + B(city) = 50,200

Total cost = 3B(Coutry) + 3B(city) = 150,600 Block I/O's.

QUESTION 2 [Hadoop MapReduce]

Now suppose the tables in Question 1 (shown below) are stored in CSV files: country.csv and city.csv

Country(code, name, continent, GNP, capital)

City(id, name, population).

For example, a row in county.csv may look like:

USA,United States,North America,123456.78,100

And a row in city.csv may look like:

100,Washington DC,700000

Suppose neither file has a header row.

Design a MapReduce program to answer the following SQL query on the data:

Select continent, avg(GNP)

From country

Where GNP >= 10000

Group by continent

Having count(*) >= 10;

(2.a.) Write a pseudocode for the map function of map task. Be sure to state input (what are the key and value), output, and steps of the function.

A: Input <key,value> = <0 "USA,United States,North America,123456.78,100">

Pseudocode:

```
    For each k,v that we get inside the map function:
        Split v based on comma and we would get array of tokens
        If float(token[3]) >= 10000:
            output(token[2], float(token[3]))
```

Output: [(North America, 123456.78)]

(2.b.) Write a pseudocode for the reduce function of reduce task. Be sure to state the input, output, and steps of the function.

A: #Suppose we have one more data row "USA,United States,North America,100000.01,100".

Input <key, value> = <North America, [123456.78,100000.01]>

Pseudocode:

```
    Count = 0 Sum = 0
    For each k, list of values we get inside the reduce function
    Loop over all the values:
        Sum += value
        Count += 1
    If Count >=10:
        Output(k, Sum/Count)
```

#the Avg of 123456.78 and 100000.01 = 111728.395.

Output [(North America, 111728.395]

QUESTION 3 [Spark DataFrame]

Consider the same two CSV files as in Question 2, but now assume that each file has a header row (i.e., the first row of the file lists the columns).

The two tables are repeated here for convenience:

Country(code, name, continent, GNP, capital)

City(id, name, population).

Suppose the following codes have been executed in a PySpark session

import pyspark.sql.functions as fc

country = spark.read.csv('country.csv', header=True)

city = spark.read.csv('city.csv', header=True)

Write a PySpark script using Spark DataFrame API for each of the following SQL queries:

(3.a.)

Select name, GNP

From country

Where GNP <= 10000 or GNP >= 20000;

A: country.filter((country['GNP'] <= '10000') | (country['GNP'] >= '20000')).select('name','GNP')

(3.b.)

Select continent, avg(GNP) as avg_gnp, count(*) as cnt
From country
Where GNP >= 10000
Group by continent
Having count(*) >= 10;

A: country.filter(country['gnp']>=10000).groupBy("continent").agg(fc.avg('gnp').alias('avg_gnp'), fc.count("*").alias('cnt')).where('cnt>=10')

(3.c.)

Select name, population
From city
Where population >= 10000
Order by population desc
Limit 10;

A: city.filter(city["population"]>=10000).select("name","population").orderBy("population", ascending=False)

QUESTION 4 [Spark RDD]

Suppose we covert the country dataframe in Question 3 into RDD:

country_rdd = country.rdd

The corresponding two tables are repeated here for convenience:

Country(code, name, continent, GNP, capital)

City(id, name, population)

(4.a.) Write a RDD script to implement the following SQL query:

Select continent, max(GNP)

From country

Where GNP > 10000

Group by continent;

A: country_rdd.filter(lambda x: x[3]>10000).map(lambda x: ((x[2]),x[3])).groupByKey().mapValues(max).collect()

(4.b.) Consider a text file hello.txt which contains lines of texts, e.g.,

hello world

hello this world

…

Write a Spark RDD script to find out how many lines contain the word "hello". Suppose your script starts with:

lines = sc.textFile('hello.txt')

A: lines1 = lines.filter(lambda x: "hello" in x).count()

QUESTION 5 [MongoDB]

Now suppose the table: Country(code, name, continent, GNP, capital) is stored in MongoDB collection country with country code as the primary key _id. Here is an example document in the collection:

{"_id": "USA", "name": "United States",

"continent":"North America",

"GNP":123456.78,capital:100}

For a MongoDB script using functions such as find, aggregate, insert, and update for each of the following questions:

(5.a.) Find names of countries in the North America continent. Return ONLY the names alphabetically (ascending order).

A: db.Country.find({"continent" : "North America"},{"name":1,_id:0}).sort({"name":1})

(5.b.) Find out how many countries in Asia with GNP between 10000 and 20000 (both are inclusive).

A: db.Country.find($and:[{GNP:{$gte: 10000, $lte: 20000}}, {continent: "Asia"}]).count()

(5.c.) Find out, for each continent with at least 10 countries, the number of countries and the maximum GNP of the countries in the continent. Order the results in the descending order of maximum GNP.

A: db.Country.aggregate([{$group:{_id: "$continent", count:{$sum: 1}, max_GNP:{$max:"$GNP"}}}, {$match: {count: {$gte: 10}}}, {$sort: {max_GNP: -1}}])

QUESTION 6 [Multiple choices]

The following questions are based on student answers in the midterms on the questions about Firebase commands, SQL queries, and XPath expressions. For each question, there are four statements about the student answer.

Note: **There may be multiple correct statements.** You need to select them all for the full credit. Note also that a half point will be deducted for each incorrect statement selected. Each question is worth 2 points.

Note also that a syntax error is an error that will lead to the rejection of commands, queries, or expressions by Firebase or MySQL or XPath evaluator. If a student answer does not have any syntax errors, but does not produce the expected results, we say that the answer has a semantic error.

QUESTION 6-1 [XPath]

Consider a person.xml file. Part of its content is shown below.

```
<persons>
  <person id=”123”>
    <name>John Smith</name>
    <age>25</age>
    <address><city>LA</city><state>CA</state></address>
  </person>
  <person id=”124”>
    <name>David Smith</name>
    <phone>312-123-4567</phone>
  </person>
  …
</persons>
```

(6.a.) Consider writing an XPath expression to find names of people who live in LA. Output actual names, not the elements. Which of the following expressions are NOT correct?

A. //person[//city="LA"]/name/text()
B. /persons/person/city="LA"
C. /persons/person/address[city = 'LA']../name/text()
D. /persons/person[address[city] = 'LA']/name/text()

A: BCD

ABCD

(6.b.) Consider writing an XPath expression to find the address element of person whose id is 123. Which of the following expressions are NOT correct?

A. /persons/person/@id='123'

B. /persons/person[@id = '123']/address/*

C. /persons/person[id = "123"]/address

A: AC
ABC

(6.c.) Consider writing an XPath expression to find names of people who are at least 25 years old and output names only. Which of the following expressions are NOT correct?

A. /persons/person/(age >= '25')/text()

B. //person[age>=25]/text()

C. //person[@age >= 25]/name/text()

A: AC
ABC

QUESTION 6-2 [SQL]

Consider the tables in a beers database as shown below. Note that it has one more table "Ales" than what you have seen in class.

Beers(name, manf)

Ales(name, color)

Drinkers(name, addr, phone)

Bars(name, addr)

Likes(drinker, beer)

Frequents(drinker, bar)

Sells(bar, beer, price)

(6.d.) Consider writing an SQL query to find out, for each manufacturer, how many times its beers are sold in bars with a price of at least 3 dollars. Which of the following queries is (are) NOT correct?

A. SELECT manf, COUNT(beer) as count

FROM Beers as b join Sells as s on b.name = s.beer

WHERE price>= 3;

B. SELECT COUNT(name), manf

FROM Beers

GROUP BY manf

HAVING name IN (SELECT beer FROM Sells WHERE price >= 3);

C. SELECT b.manf,COUNT(*)

FROM Beers AS b JOIN Sells AS s ON b.name=s.beer

GROUP BY b.manf

HAVING s.price>=3;

D. select count(*)

from Beers t1 join Sells t2 on t1.name = t2.beer

where t2.price >= 3 group by t1.manf;

A: BD
ABCD
(6.e.) Consider writing an SQL query to find manufacturers of beers that nobody likes. Which of the following queries is (are) NOT correct?

A. SELECT manf
FROM Beers
WHERE name NOT IN Likes;

B. SELECT Beers.manf
FROM Beers
WHERE Beers.name NOT IN (SELECT Likes.beer JOIN ON Beers.name = Likes.beer);

C. select name
from Beers
where name not in (select beer from Likes);

A: AC
ABC
(6.f.) Consider writing a query to find out drinkers who frequents at least two different bars. Which of the following queries is (are) NOT correct?

A. select f.drinker
from Frequents f
group by f.bar
having count(f.bar)>=2;

B. SELECT d.name
FROM Drinkers d join Frequents f on d.name = f.drinker
group by f.drinker
having count > 1;

A: AB
(6.g.) Consider writing a query to find colors of Ales beers NOT made by Heineken and output the same color once. Which of the following queries is (are) NOT correct?

A. SELECT distinct color
FROM Ales
WHERE name != 'Heineken';

B. Select distinct name
From Ales
where name in (select name from Beers where manf = 'Heineken');

C. select color
from Ales
where name not in
(select name from Beers where manf = "Heineken");

A: ABC

QUESTION 6-3 [Firebase]
Consider using Firebase to manage student data. Suppose all student data will be stored in ‘https://dsci551.firebaseio.com/students.json’, and are structured as follows:
{“students”:
{“200”: {“name”: “David”, “age”: 28}},

“300”: {“name”: “Bill”},
…
}
where 200 and 300 are student IDs.

(6.h.) Consider writing a curl command to add a new student with id = “100” and name = “john”. Which of the following commands is (are) NOT correct?

A. curl -X PUT ‘https://dsci551.firebaseio.com/students/student1.json' -d '{"id":"100","name";"john"}'
B. curl -X PUT ‘https://dsci551.firebaseio.com/students.json’ -d '{"id": "100", "name": "john"}'
C. curl -X PUT 'https://dsci551.firebaseio.com/students.json’ -d {100: name="john"}
D. curl -X POST "https://dsci551.firebaseio.com/students.json" -d {"100": {"name": "john"}}

A: AD

ABCD

(6.i.) Consider writing a curl command to add a new attribute age = 25 for the student 100 added by the command in the previous question. Which of the following commands is (are) NOT correct?

A. curl -X put -d {"age": 25} 'https://dsci551.firebaseio.com/students.json/id/100'
B. curl -X POST 'https://dsci551.firebaseio.com/students/100.json' -d '{"age":25}'
C. curl -X PATCH 'https://dsci551.firebaseio.com/students.json’ -d {100: age = 25}
D. curl -X PATCH -d '{"id":"100", "name":"john", "age":25}' 'https://dsci551.firbaseio.com/students.json'

A: AB

ABCD

(6.j.) Consider writing a curl command to find all students who are at least 25 years old. Which of the following commands is (are) NOT correct?

A. curl -X GET ‘https://dsci551.firebaseio.com/students.json?orderBy="age"&&startAt="25"’
B. curl 'https://dsci551.firebaseio.com/students.json?orderBy="$value"&startAt=25'
C. curl -X GET 'https://dsci551.firebaseio.com/students.json?orderBy="age"&startAt=26'

A: ABC