ISIT312 Sample Exam Paper 2

Question 1

Assume that the input of the Map stage is as the following.

Table X:	Table Y:
A:1,B:a	A:1,C:b
A:1,B:b	A:2,C:d
A:4,B:c	A:4,C:f

In the Map stage, each row is mapped into a key-value pair in the Map stage, where the key is the value of A and the value is a concatenation of the column name and its value with colon (:) as a delimiter. The Java implementation of the map function is as follows.

```
public void map(Object key, Text value, Context context)
     throws IOException, InterruptedException {

     IntWritable key = new IntWritable(0);
     Text val = new Text();

     String[] input = value.toString().split(“,”);
     key.set(Integer.parseInt(input.split(“:”)[0]));
     val.set(input[1]);

     context.write(key, val);
}
```

The results of the map function for table X and table Y are the following.

Table X:	Table Y:
(1, B:a)	(1, C:b)
(1, B:b)	(2, C:d)
(4, B:c)	(4, C:f)

After the Map stage, the input received in the Reduce stage would be a key-value pair, where the key is the value of column A and the value is a list of Strings concatenated from the column name and its value separated by a colon (:). The input of the Reduce stage is the following.

```
(1, [B:a, B:b, C:b])
(2, [C:d])
(4, [B:c, C:f])
```

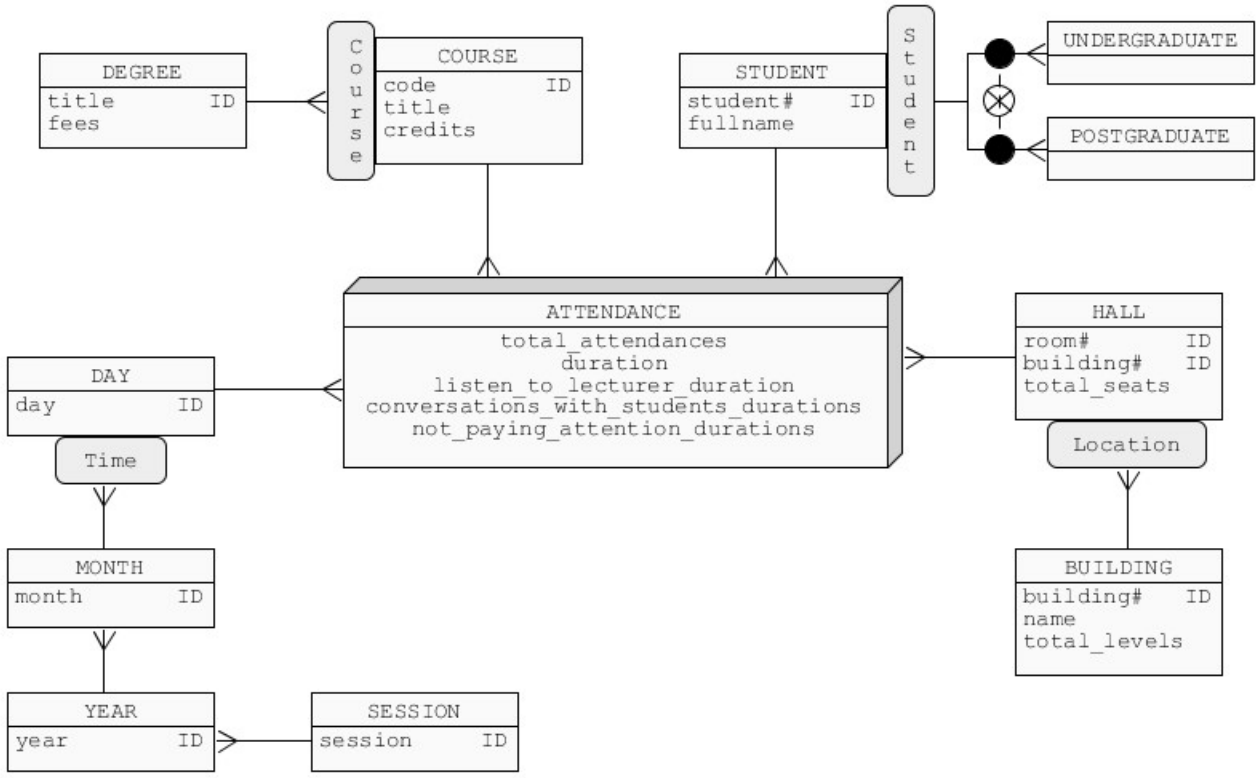
In the Reduce stage, each String in the list is grouped into two different lists based on the column name. After that, a cross join is done. The result of the cross join is a key-value pair, where the key is the value of column A and the value is a String concatenation of value in column B and C separated by a comma (,). The Java implementation of the reduce function is as follows.

```
public void reduce(IntWritable key, Iterable<Text> value, Context context)
     throws IOException, InterruptedException {

     Text val = new Text();

     ArrayList<String> columnB = new ArrayList<String>();
     ArrayList<String> columnC = new ArrayList<String>();

     for (Text t: value) {
          String[] str = t.toString().split(":");
          if (str[0].equals("B"))
               columnB.add(str[1]);
          else
               columnC.add(str[1]);
     }

     if (columnB.size() < 1) {
          for (String b: columnB) {
               val.set(b + ",");
               context.write(key, val);
          }
     }
     else if (columnC.size() < 1) {
          for (String c: columnC) {
               val.set("," + c);
               context.write(key, val);
          }
     }
     else {
          for (String b: columnB) {
               for (String c: columnC) {
                    val.set(b + "," + c);
                    context.write(key, val);
               }
          }
     }
}
```

The output of the reduce function is as follows.

```
(1, a,b)
(1, b,b)
(2, ,d)
(4, c,f)
```

Question 2



Question 3

Part (1)

```
$HADOOP_HOME/bin/hadoop fs -mkdir /user/hive/warehouse/ACCIDENT
$HADOOP_HOME/bin/hadoop fs -put /home/jrg/accidents/car.tbl
/user/hive/warehouse/ACCIDENT
$HADOOP_HOME/bin/hadoop fs -put /home/jrg/accidents/accident.tbl
/user/hive/warehouse/ACCIDENT
```

Part (2)

```
CREATE EXTERNAL TABLE CAR (
     registration   VARCHAR(50),
     make           VARCHAR(50),
     model          VARCHAR(50)
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’
STORED AS TEXTFILE LOCATION ‘/user/hive/warehouse/ACCIDENT’;

LOAD DATA LOCAL INPATH ‘car.tbl’ INTO TABLE CAR;
```

```
CREATE EXTERNAL TABLE ACCIDENT (
     day            NUMBER(2),
     month          CHAR(3),
     year           NUMBER(4),
     registration   VARCHAR(50),
     license        VARCHAR(15),
     street         VARCHAR(50),
     city           VARCHAR(20)
     severity       VARCHAR(15)
     damages        NUMBER(10,2)
)
ROW FORMAT DELIMITED FIELD TERMINATED BY ‘,’
STORED AS TEXTFILE LOCATION ‘/user/hive/warehouse/ACCIDENT’;

LOAD DATA LOCAL INPATH ‘accident.tbl’ INTO TABLE ACCIDENT;
```

Part (3)

(i)
```
SELECT model, year, COUNT(*)
FROM ACCIDENT JOIN CAR ON ACCIDENT.registration = CAR.registration
GROUP BY model, year WITH CUBE;
```

(ii)
```
SELECT year, month, AVG(damages)
FROM ACCIDENT JOIN CAR ON ACCIDENT.registration = CAR.registration
WHERE make = ‘Toyota’
GROUP BY year, month WITH ROLLUP;
```

(iii)
```
SELECT year, make, model, city, COUNT(*)
FROM ACCIDENT JOIN CAR ON ACCIDENT.registration = CAR.registration
WHERE severity = ‘serious’
GROUP BY year, make, model, city
GROUPING SETS((year), (make, model), (city));
```

Question 4

Part (1)

```
create 'question4', 'PERSON', 'CAR', 'LOCATION', 'DATE', 'ACCIDENT'

put 'question4', 'person|P001', 'PERSON:license', 'P001'
put 'question4', 'person|P001', 'PERSON:fname', 'Alex'
put 'question4', 'person|P001', 'PERSON:lname', 'Smith'

put 'question4', 'car|PKR856', 'CAR:registration', 'PKR856'
put 'question4', 'car|PKR856', 'CAR:make', 'Rolls Royce'
put 'question4', 'car|PKR856', 'CAR:model', 'Silver Shadow'

put 'question4', 'car|UUQ076', 'CAR:registration', 'UUQ076'
put 'question4', 'car|UUQ076', 'CAR:make', 'Toyota'
put 'question4', 'car|UUQ076', 'CAR:model', 'Corrola'

put 'question4', 'accident|A001', 'PERSON:license', 'P001'
put 'question4', 'accident|A001', 'CAR:registration', 'PKR856'
put 'question4', 'accident|A001', 'LOCATION:street', 'Victoria St.'
put 'question4', 'accident|A001', 'LOCATION:city', 'Sydney'
put 'question4', 'accident|A001', 'DATE:day', '25'
put 'question4', 'accident|A001', 'DATE:month', '9'
put 'question4', 'accident|A001', 'DATE:year', '2015'
put 'question4', 'accident|A001', 'ACCIDENT:severity', 'serious'
put 'question4', 'accident|A001', 'ACCIDENT:damages', '1234.56'

put 'question4', 'accident|A002', 'PERSON:license', 'P001'
put 'question4', 'accident|A002', 'CAR:registration', 'UUQ076'
put 'question4', 'accident|A002', 'LOCATION:street', 'Bong Bong St.'
put 'question4', 'accident|A002', 'LOCATION:city', 'Dapto'
put 'question4', 'accident|A002', 'DATE:day', '13'
put 'question4', 'accident|A002', 'DATE:month', '3'
put 'question4', 'accident|A002', 'DATE:year', '2019'
put 'question4', 'accident|A002', 'ACCIDENT:severity', 'minor'
put 'question4', 'accident|A002', 'ACCIDENT:damages', '100.00'
```

Part (2)

(i)
```
scan 'question4', {FILTER=>"QualifierFilter(=, 'binary:damages') AND
ValueFilter(>, 'binary:1000')"}
```

(ii)
```
scan 'question4', {COLUMNS=>['PERSON:fname', 'PERSON:lname'],
FILTER=>"SingleColumnValueFilter('LOCATION', 'city', =,
'binary:Sydney') AND SingleColumnValueFilter('DATE', 'year', =,
'binary:2019')"}
```

Question 5

Part (1)

```
students = load 'hdfs://localhost:8080/records/student.txt' using PigStorage(',') as (studentid:int, name:chararray, majorid:int);

majors = load 'hdfs://localhost:8080/records/majors.txt' using PigStorage(',') as (majorid:int, majorname:chararray);

gpas = load 'hdfs://localhost:8080/records/GPAs.txt' using PigStorage(',') as (studentid:int, majorid:int, gpa:float);
```

Part (2)

```
studentMajor = join students by majorid left outer, majors by majorid;
dump studentMajor;
```

Output:

```
(1, Henry, 100, 100, SoftwareEngineering)
(2, Karen, 100, 100, SoftwareEngineering)
(3, Paul, 101, 101, InformationManagement)
(4, Jimmy, 102, 102, BigData)
(5, Janice, 102, 102, BigData)
```

Part (3)

```
joinGM = join gpas by majorid, major by majorid;
GPAbyMajor = group joinGM by major::majorname;
GPAbyMajor = foreach GPAbyMajor generate group, AVG(joinGM.gpas::gpa);
dump GPAbyMajor;
```

Output:

```
(SoftwareEngineering, 3.5)
(InformationManagement, 2.5)
(BigData, 4.5)
```

Question 6

Part (a)

```
invoiceDF.count()
```

Part (b)

```
invoiceDF.select("StockCode").distinct().count()
```

Part (c)

```
invoiceDF.groupBy("StockCode").avg("UnitPrice").show()
```

Part (d)

```
invoiceDF.groupBy("CustomerID").count("InvoiceNo").show()
```