Santa Clara University
School of Engineering

Honor Code of the School of Engineering

"All students taking courses in the School of Engineering agree, individually and collectively, that they will not give or receive unpermitted aid in examinations or other course works that is to be used by the instructor as the basis of grading."

-From the Graduate/Undergraduate Bulletin

I have read, understood, and agree to abide by the Honor Code of the School of Engineering.

Name: _____

ID: _____

Signature: _____

Date: _____

1. [40 points]
2. [30 points]
3. [30 points]
4. [30 points]
5. [30 points]
6. [30 points]
7. [30 points]
8. [60 points]
9. [50 points]
10. [50 points]

Total Score:

Midterm Examination

COEN 242 Introduction to Big Data
Department of Computer Engineer
Santa Clara University

Dr. Ming-Hwa Wang
Phone: (408) 805-4175
Course website:
Office Hours:

Spring Quarter 2023

Email address: mwang2@cse.scu.edu

http://www.cse.scu.edu/~m1wang/bigdata/
Wednesday 9:00pm-9:30pm & Saturday 10:00am-11:00am

1. [40 points] True or false problems with wrong-answer penalties:
   a) A Spark task starts from reading input data from disk until ends by saving the result to disk if requested by the user. During this process, intermediate data always and can only stores in memory.
   b) Each Spark task contains one driver process and a set of executor processes. Each executor process has it own local set of variables which the executor can modify without affect other executors.
   c) The join transformation in Spark can be either narrow dependency if input is co-partitioned or wide dependency if input is not co-partitioned.
   d) Spark uses replications (like any other distributed system) other than checkpoints for recovery from loss of data.
   e) SparkSessions can only run on the master node instead of any of worker/executor nodes.
   f) Spark transformations are always lazy evaluations but eventually well be evaluated.
   g) All results showed in Spark console are generated by spark actions.
   h) All reducers can be combinable reducers.

2. [30 points] What is the output of the following pyspark code:

```
from functools import reduce
x = ['Python', 'programming', 'is', 'awesome!']
print(reduce(lambda val1, val2: val1 + val2, x))
```

3. [30 points] What is the output of the following code:

```
import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
simpleData = [("James","Sales","NY",90000,34,10000), \
    ("Michael","Sales","NY",86000,56,20000), \
    ("Robert","Sales","CA",81000,30,23000), \
    ("Maria","Finance","CA",90000,24,23000) \
  ]
columns= ["employee_name","department","state",
```

```
    "salary","age","bonus"]
df = spark.createDataFrame(data = simpleData,
    schema = columns)
simpleData2 = [("James","Sales","NY",90000,34,10000), \
     ("Maria","Finance","CA",90000,24,23000), \
     ("Jen","Finance","NY",79000,53,15000), \
     ("Jeff","Marketing","CA",80000,25,18000), \
     ("Kumar","Marketing","NY",91000,50,21000) \
  ]
columns2= ["employee_name","department","state",
    "salary","age","bonus"]
df2 = spark.createDataFrame(data = simpleData2,
    schema = columns2)
unionDF = df.union(df2)
unionDF.show(truncate=False)
disDF = df.union(df2).distinct()
disDF.show(truncate=False)
unionAllDF = df.unionAll(df2)
unionAllDF.show(truncate=False)
```

4. [30 points] Given a test file "input.dat" with empty lines, please write a PySpark code to output number of empty lines in the file.

5. [30 points] Please write a MapReduce pseudocode to solve the grouping and aggregation problem. Let $R(A, B, C)$ be a relation to which we apply the operator $\gamma_{A,\theta(B)}(R)$.

6. [30 points] Suppose there is a repository of 2^{23} documents, and word *w* appears in 512 of them. In a particular document *d*, the maximum number of occurrences of a word is 70. Approximately what is the TF.IDF score for *w* if that word appears (a) once (b) 5 times?

7. [30 points] Using the matrix-vector multiplication, applied to the matrix and vector:

```
 1  2  3  4       1
 5  6  7  8       2
 9 10 11 12       3
13 14 15 16       4
```

apply the Map function to this matrix and vector (assume the vector can load into memory fully). Then, identify in the list below, one of the key-value pairs that are output of Map.
a) (3, 45) b) (1, 13) c) (2, 70) d) (4, 28)

8. [60 points] Which of the following will cause core dump? What is the output if it can execute without core dump?

```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
```

a)

```
data = sc.parallelize([[1, 2], [3, 4], [5, 6])
r = data.flatMap(lambda x: x)
print(r.collect())
```

b)

```
data = sc.parallelize([{'a':1, 'b':2}, {'c':3, 'd':4}])
r = data.flatMap(lambda x: x)
print(r.collect())
```

c)

```
data = sc.parallelize([1.0, 2.0, 3.0, 4.0])
r = data.flatMap(lambda x: x)
print(r.collect())
```

9. [50 points] The follow program changes CourseName to upper case, reduces price by 1000 for each course, and then apply different discount to reach course. Please replace pass with correct code to make it working. [Hint: use df.withColumn()].

```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
            .appName('SparkByExamples.com') \
            .getOrCreate()
simpleData = (("Java",4000,5), \
    ("Python", 4600,10),  \
    ("Scala", 4100,15),   \
    ("PHP", 3000,20),  \
  )
columns= ["CourseName", "fee", "discount"]
df = spark.createDataFrame(data = simpleData, schema
= columns)
from pyspark.sql.functions import upper
def to_upper_str_columns(df):
    pass
def reduce_price(df,reduceBy):
    pass
def apply_discount(df):
    pass
df2 = df.transform(to_upper_str_columns) \
        .transform(reduce_price,1000) \
        .transform(apply_discount)
df2.show()
```

The output is:

```
+----------+----+--------+-------+--------------+
|CourseName| fee|discount|new_fee|discounted_fee|
+----------+----+--------+-------+--------------+
|      JAVA|4000|       5|   3000|        2850.0|
```

```
|    PYTHON|4600|      10|   3600|        3240.0|
|     SCALA|4100|      15|   3100|        2635.0|
|       PHP|3000|      20|   2000|        1600.0|
+----------+----+--------+-------+--------------+
```

10. [50 points] Please replace pass with correct code to make it working.

```
from pyspark import SparkConf, SparkContext
from pyspark.sql import SQLContext
sc = SparkContext()
sqlContext = SQLContext(sc)
company_df =
sqlContext.read.format('com.databricks.spark.csv') \
               .options(header='true',inferschema='true') \
               .load('Fortune5002017.csv')
company_df.printSchema()
    DataFrame[Rank: int, Title: string, Website: string,
    Employees: Int, Sector: string]
    root
    |-- Rank:  integer (nullable = true)
    |-- Title:  string (nullable = true)
    |-- Website:  string (nullable = true)
    |-- Employees:  integer (nullable = true)
    |-- Sector:  string (nullable = true)
from pyspark.ml.feature import VectorAssembler
vectorAssembler = VectorAssembler(pass)
tcompany_df = vectorAssembler.transform(company_df)
tcompany_df= tcompany_df.select(pass)
tcompany_df.show(3)
```

Features	Employees
[1.0,2300000.0]	2300000
[2.0,367700.0]	367700

```
splits = tcompany_df.randomSplit([0.7, 0.3])
train_df = splits[0]
test_df = splits[1]
from pyspark.ml.regression import LinearRegression
lr = LinearRegression(featuresCol = pass, labelCol= pass,
       maxIter=10, regParam=0.3, elasticNetParam=0.8)
lr_model = pass
print("Coefficients: " + str(lr_model.coefficients))
    Coefficients: [-32251.88812374517, 0.9255193858709874]
print("Intercept: " + str(lr_model.intercept))
    Intercept: 140317.88600801243
predictions = pass
```