Question 1

Aditya and David are the first-year data science students with Monash University. They are discussing how parallel and distributed processing can help data scientists perform the computation faster. They would like your help to understand and get answers to the following questions:

1. Using the current processing resources, we can finish processing 1TB (one terabyte) of data in 1 hour. Recently the volume of data has increased to 2TB and the management has decided to double up the processing resources. Using the new processing resources, we can finish processing the 2TB in 60 minutes. Aditya wants to know **(1 + 1 = 2 Marks)**

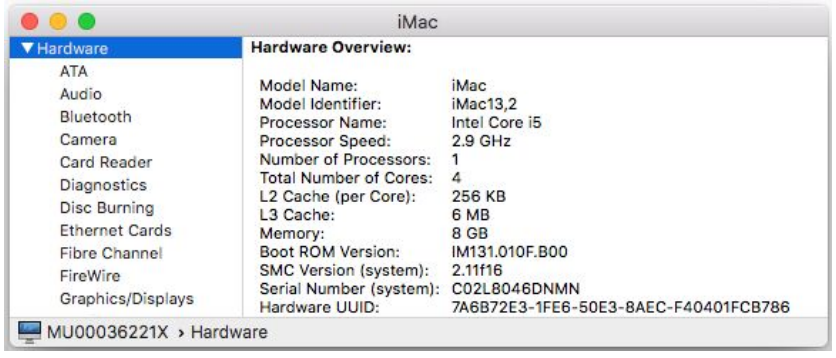
    a. Is this speed-up or scale-up? Please explain your answer.
    b. Also, please explain what type of speed-up or scale-up is it (**linear, superlinear or sub-linear**)?

    It is a scale-up. It is data scale up (not transaction scale-up). Scale-up is not about speeding up a job. Scale-up is about maintaining the same performance when the workload is increased by increasing the processing resources, proportionally.
    (1 mark for scale-up)

    It is a linear scale-up. Using x resources (current resources), 1TB queries = 60 minutes When the resources are doubled (e.g. x becomes 2x now), a linear scale up is being able to complete 2TB in 60 minutes.

    In the question, using 2x resources, it finishes 2TB in 60 minutes. Therefore, it is linear scale up.
    (1 mark, including the reason)

2. David is using his iMac desktop to do parallel query processing. The iMac has the following specifications:



    He wants to know what type of parallel database architecture is he using to do the parallel query processing. Please explain the reason for your answer. **(2 Marks)**

    It is a Shared-Memory Architecture. (1 mark)

The memory is shared among all cores within a single computer (4 cores per iMac). There is no interconnection network involved. (1 mark)

3. David read in the textbook that “Random unequal partitioning is sometimes inevitable in parallel search.” However, he could not understand why? Please give two reasons why random unequal partitioning is sometimes inevitable in parallel search. **(1 + 1 = 2 Marks)**

- If the initial data placement is based on a particular attribute (say attribute x), whereas the parallel search is based on a different attribute (say attribute y), then all processors must be used, and the data is random unequal in all processors.

- Even if the initial data placement is equal (random equal or round-robin), if the search is part of a large query which has some initial operations, such as join, then the parallel search which follows the previous operations, will have the data distributed unequally to all processors. Hence, it is random and unequal.

4. Aditya now understands that skewness is the unevenness of workload and skewed workload distribution is generally undesirable. He found the figure below in the textbook that shows the skewed workload distribution. He wants to know **(1 + 1 = 2 Marks)**
    a. Is the figure below **processing skew** or **data skew**? Please explain with reason.
    b. Is it possible to have an equal distribution of data? Please explain how.

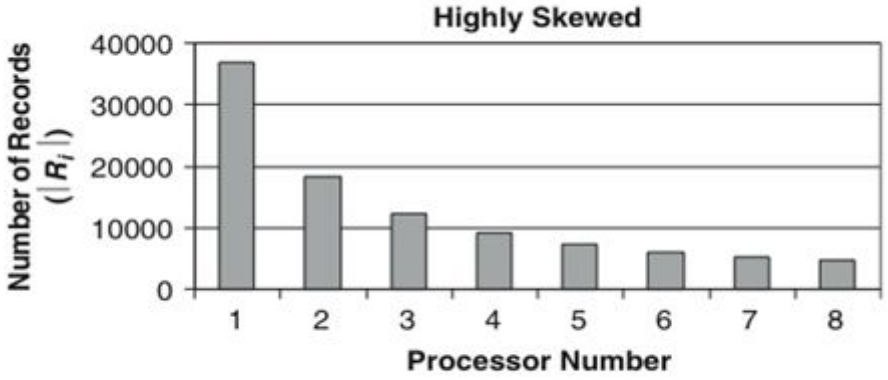


Figure 2.2 Highly skewed distribution

It is a data skew. Data skew is uneven distribution of data in terms of size or number of records.
(1 Mark for the correct choice and explanation; 0 Mark for incorrect answer.)

Yes, it is possible to have equal distribution of data using random-equal data partitioning method. (1 Mark)

5. David was given a task to perform log analysis in the lab. The input data consisted of log messages of varying degrees of severity, along with some blank lines. He has to compute

how many log messages appear at each level of severity. The contents of the “input.txt” file are shown below.

```
INFO This is a message with content
INFO This is some other content
(empty line)
INFO Here are more messages
WARN This is a warning
(empty line)
ERROR Something bad happened
WARN More details on the bad thing
INFO back to normal messages
```

The expected output of the operations is as below.

```
[(‘INFO’, 4), (‘WARN’, 2), (‘ERROR’, 1)]
```

However, he is not sure how to begin. Please explain to him assuming ‘sc’ as a SparkContext object. **(1 + 1 = 2 Marks)**

a. What is an RDD?
b. How can it be created in this case to perform a log analysis of “input.txt” file?

Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. (1 Mark)

It can be created using sc.textFile() method e.g. sc.textFile("input.txt")(1 Mark)

Question 2

Petadata is an enterprise software company that develops and sells database analytics software subscriptions. The company provides three main services: business analytics, cloud products, and consulting. It operates in North and Latin America, Europe, and Australia.

Petadata is headquartered in Melbourne, Victoria, and has additional major Australian locations in Sydney and Adelaide, where its data center research and development is housed. Peter Liu has served as the company's president and chief executive officer since 2014. The company reported $2.8 billion in revenue, with a net income of $112 million, and 15,026 employees globally, as of March 15, 2020.

Chin is a recent graduate from Monash University and preparing for the job interview in Petadata. He needs your help to understand aspects of parallel processing especially parallel joins in shared-nothing architecture.

1. Using a more general notation, table R has |R| number of records, and table S has |S| number of records. The first step of ROJA is to redistribute the records from both tables according to hash/range partitioning. What is the **cost model** of the **Redistribution Step** of **ROJA**? **(4 marks)**

Symbol	**Description**
Data Parameters	
R	Size of table in bytes
R_i	Size of table fragment in bytes on processor i
|R|	Number of records in table R
$\|R_i\|$	Number of records in table R on processor i
Systems Parameters	
N	Number of processors
P	Page size
Time Unit Cost	
IO	Effective time to read a page from disk
t_r	Time to read a record in the main memory
t_w	Time to write a record to the main memory
t_d	Time to compute destination
Communication Cost	
m_p	Message protocol cost per page
m_l	Message latency for one page

Scan cost for loading tables R and S from local disk in each processor is: ((Ri / P) + (Si / P)) IO

Select cost for getting the record out of data page is: (|Ri| + |Si|) (tr + tw)

Finding destination cost is: (|Ri| + |Si|) (td)

Data transfer cost is: ((Ri / P) + (Si / P)) (mp + ml)

Receiving records cost is: ((Ri / P) + (Si / P)) (mp)

Both data transfer and receiving costs look similar, as also mentioned previously in the divide and broadcast cost. However, for disjoint partitioning, the size of Ri and Si in the data transfer cost is likely to be different from that of the receiving cost. The reason is as follows. Ri and Si in the data transfer cost are the size of each fragment of both tables in each processor. Again, assuming that the initial data placement is done using a round-robin or any other equal partitioning, each fragment size will be equal. Therefore, Ri and Si in the data transfer cost are simply dividing the total table size by the available number of processors.

However, Ri and Si in the receiving cost are most likely skewed. Consequently, the values of Ri and Si in the receiving cost are different from those of the data transfer cost.

Disk cost for storing the result of data distribution is:
((Ri / P) + (Si / P)) IO

2. Chin found the code below in stackunderflow.com that counts the errors and warnings in the text file using Apache Spark.

```
from pyspark import SparkContext

sc = SparkContext(master="local[2]", appName="Errors and warnings Count")
twitter_rdd = sc.textFile('twitter.txt', 3)
blank_lines = 0 # global variable

def extract_blank_lines(line):
    if line == "":
        blank_lines += 1
    return line.split(" ")

word_rdds = twitter_rdd.flatMap(extract_blank_lines)
word_rdds.collect()

print("Blank lines: %d" %blank_lines)
```

However, the code produces the error shown below.

```
Caused by: org.apache.spark.api.python.PythonException: Traceback (most
recent call last):
  File
"/home/.../.local/lib/python3.8/site-packages/pyspark/python/lib/pyspark.zip
/pyspark/util.py", line 107, in wrapper
    return f(*args, **kwargs)
  File "<ipython-input-29-b8182ac1646a>", line 6, in extract_blank_lines
UnboundLocalError: local variable 'blank_lines' referenced before assignment
        at
        org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handleP
        ythonException(PythonRunner.scala:503)
        at
        org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner
        .scala:638)
        at
        org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner
        .scala:621)
        at
        org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(
        PythonRunner.scala:456)
        ... 1 more
```

a. Why does the code above generate errors? **(1 Marks)**
b. How can you fix the error? Please explain and write the fixed code below. **(3 Marks)**

blank_lines variable is a global variable to driver but is not accessible by the executors, because of which the flatMap operation in each executor produces the error *UnboundLocalError: local variable 'blank_lines' referenced before assignment*

This error can be fixed using accumulators. Accumulators provides a simple syntax for aggregating values from worker nodes back to the driver program. They are only "added" to through an associative and commutative operation and can therefore be efficiently supported in parallel.

```
twitter_rdd = sc.textFile('twitter.txt', 3)
blank_lines = sc.accumulator(0) # Create Accumulator[int] intitialized to 0

def extract_blank_lines(line):
    global blank_lines # make the global variable accessible
    if line == "":
        print(type(line))
        blank_lines += 1
    return line.split(" ")

word_rdds = twitter_rdd.flatMap(extract_blank_lines)
word_rdds.collect()

print("Blank lines: %d" %blank_lines.value)
```

3. Finally, Chin wants to know, if we have two tables (let's say Table R and Table S) and we want to perform an Outer Join query, if we use OJSO algorithm to process this outer join query, OJSO algorithm will be the same as ROJA algorithm. Is this statement correct? Please explain why. **(2 Marks)**

   OJSO is a load balancing algorithm for outer join. Load imbalance addressed by OJSO is applicable if there are more than 2 tables to be join (e.g. R join S join T), because after joining the first two tables, there will be load imbalance problem, and OJSO will solve this load imbalance problem with joining with the third table.

   In this question, there are only 2 tables (R and S), hence OJSO is the same as ROJA, because OJSO is based on ROJA.

Question 3

Tooba is a sessional lecturer and data scientist in Monash University and loves to bake cookies with M&Ms in them. She rewards her students in the university where she frequently teaches machine learning and data science courses with batches of those cookies. But she's data-driven and wants to ensure that she gets the right colours of M&Ms in the cookies for students from different states in Australia.

1. She has a computer with four processors. But she is planning to use only three processors to avoid resource contention. Given a data set D = {55; 30; 68; 39; 1; 4; 49; 90; 34; 76; 82; 56; 31; 25; 78; 56; 38; 32; 88; 9; 44; 98; 11; 70; 66; 89; 99; 22; 23; 26} and three processors, show step-by-step how the Parallel Redistribution Merge-All Sort works. **(3 Marks)**

   Assume random equal partitioning has been applied, where each processor has 10 records. The first processor will get the first 10 records, etc.

   Processor 1 = {55; 30; 68; 39; 1; 4; 49; 90; 34; 76}

   Processor 2= {82; 56; 31; 25; 78; 56; 38; 32; 88; 9}

   Processor 3 = {44; 98; 11; 70; 66; 89; 99; 22; 23; 26}

   Parallel Redistribution Merge-All Sort

   Step 1: Local Sort

   Processor 1 = {1; 4; 30; 34; 39; 49; 55; 68; 76; 90}

   Processor 2= {9; 25; 31; 32; 38; 56; 56; 78; 82; 88}

   Processor 3 = {11; 22; 23; 26; 44; 66; 70; 89; 98; 99}

   Step 2: Redistribution

   Assume Processor 1=1-33, Processor 2=34-66; Processor 3=67-99

   Processor 1 =

   {1; 4; 30}

   {9; 25; 31; 32}

   {11; 22; 23; 26}

   Results = {1; 4; 9; 11; 22; 23; 25; 26; 30; 31; 32}

   Processor 2=

   {34; 39; 49; 55}

   {38; 56; 56}

{44; 66}

Results = {34; 38; 39; 44; 49; 55; 56; 56; 66}

Processor 3 =

{68; 76; 90}

{78; 82; 88}

{70; 89; 98; 99}

Results = {68; 70; 76; 78; 82; 88; 89; 90; 98; 99}

2. She was thinking of using internal sorting to perform the sort. However, she read on the internet that "**External Sorting** is different from **Internal Sorting**. Therefore, external sorting cannot use any of the Internal sorting methods". Is this statement True or False? Explain the reason as well. **(2 Marks)**

   The statement is False.

   External sorting method is used when the entire dataset to be sorted cannot fit into the main memory. Internal sorting method is a sorting method when the entire dataset to be sorted can fit into main memory.

   The way external sorting works is by dividing the dataset into a smaller dataset so that each smaller dataset can fit into the main memory and these smaller datasets are sorted using an Internal Sorting method.

   So, External sorting method uses Internal sorting.

3. Upon further reading, Tooba found that there are two types of skewness: data skew and processing skew, that can hinder the efficient performance of parallel sorting.
   a. Explain what is **data skew** and **processing skew**. **(2 Marks)**
   b. Considering data skew and processing skew, when should we use Parallel Redistribution Merge-All Sort, and when should we use Parallel Partitioned Sort? Also, explain why. **(3 Marks)**

   Data skew is caused by the unevenness of data placement in a disk in each local processor, or by the previous operator. Unevenness of data placement is caused by the fact that data value distribution, which is used in the data partitioning function, may well be non-uniform due to the nature of data value distribution. If initial data placement is based on a round-robin data partitioning function, data skew will not occur. However, it is common for database processing to not involve a single operation only. It sometimes involves many operations, such as selection first, projection second, join third, and sort last. In this case, although initial data placement is even, other operators may have rearranged the data – some data are eliminated, or joined, and consequently, data skew may occur when the sorting is about to start.

Processing skew is caused by the processing itself, and may be propagated by the data skew initially. For example, a parallel external sorting processing consists of several stages. Somewhere along the process, the workload of each processing element may not be balanced, and this is called processing skew. Notice that even when data skew may not exist at the start of the processing, skew may exist at a later stage of processing. If data skew exists in the first place, it is very likely that processing skew will also occur.

If the processing skew degree is high, then use Parallel Redistribution Merge-All Sort. If both data skew and processing skew degrees are high OR no skew, then use Parallel Partitioned Sort.

When there is a high processing skew degree, parallel partitioned sort performs poorly. Why? One reason is that the skew occurs in the second phase of processing, that is in the sorting phase of parallel partitioned sort, and in the final merging of parallel redistribution merge-all sort. The second phase of parallel partitioned sort is similar to the first phase of parallel redistribution merge-all sort. With processing skew exists, the second phase of parallel partitioned sort now becomes so expensive, whereas the first phase of parallel redistribution merge-all sort remains the same, since no processing skew is involved in the first phase of processing. This results in an extreme overhead to parallel partitioned sort, and this is why the performance of parallel partitioned sort is degraded.

When both data and processing skews exist, the performance of parallel partitioned sort is now slightly better than parallel redistribution merge-all sort. The main reason is that data skew now affects the first phase of parallel redistribution merge-all sort (i.e. local sorting phase) very badly. On the other hand, data skew effect in the first phase of parallel partitioned sort (i.e. scanning phase) is not as bad, since the scanning phase, only a few operations are involved, particularly disk loading, reading, and partitioning. The first phase of parallel redistribution merge-all sort involves many more operations and they are all affected by data skew.

Parallel partitioned sort outperforms parallel redistribution merge-all sort when no skew is involved. However, it is unreasonable to assume that in the parallel partitioned sort, where the first phase is a redistribution phase does not involve any data skew.

Conclusion: parallel partitioned sort is suitable for only when no skew or both skew are involved. When processing skew exists without data skew, parallel partitioned sort does not perform as well as parallel redistribution merge-all sort.

Question 4

2020 has been the year of Big Data – the year when big data and analytics made tremendous progress through innovative technologies, data-driven decision making and outcome-centric analytics. You are applying for the job as a Data Scientist. Mohammad is a senior lecturer and data scientist at Monash University, and a good friend of yours. He has prepared a list of questions regarding Apache Spark and Machine Learning to help you prepare for the job interview. Please answer the following questions.

1. In Apache Spark, machine learning pipelines provide a uniform set of high-level APIs built on top of DataFrames. It makes it easier to combine multiple algorithms into a single

pipeline, or workflow. The key concepts introduced by the Pipelines API are DataFrame, Transformer, Estimator, Pipeline, and Parameter.

a. What is Machine Learning and why should you use machine learning with Spark? **(2 Marks)**
b. What is a Transformer and an Estimator? **(2 Marks)**

Write your answer below

Machine learning algorithms attempt to make predictions or decisions based on training data, often maximizing a mathematical objective about how the algorithm should behave. In machine learning, computers are taught to spot patterns in data. They adapt their behaviour based on automated modelling and analysis of the task they are trying to perform.

MLlib is designed to run in parallel on clusters. MLlib contains a variety of learning algorithms and is accessible from all of Spark's programming languages such as scala, java or python.

A Transformer is an abstraction that includes feature transformers and learned models. Technically, a Transformer implements a method transform(), which converts one DataFrame into another, generally by appending one or more columns.

An Estimator is an algorithm which can be fit on a DataFrame to produce a Transformer. Technically, an Estimator implements a method fit(), which accepts a DataFrame and produces a Model, which is a Transformer.

2. According to McKinsey study, 35% of what consumers purchase on Amazon and 75% of what they watch on Netflix is driven by machine learning–based product recommendations.
   a. Mohammad wants to know if you have understood how these recommendation systems work. So, please use the dataset below to recommend Top-2 movies to Mohammad. Please show all the calculations. **(3 Marks)**

Name	StarTrek	StarWars	Superman	Batman	Hulk
Mohammad	4	2	?	5	4
Paras	5	3	4	?	3
Huashun	3	**?**	4	4	3

b. You are given a dataset "ratings" which contains movie ratings consisting of user, movie, rating and timestamp columns. The column names are *userId, movieId, rating and ts* respectively. Write a basic Machine Learning Program in PySpark to build and evaluate the recommendation model. Write the missing code snippets in the program given below. **(3 Marks)**

Write your answer below

Sim(Mohammad, Paras) = (4x5)+(2x3)+(4x3)/(Sqrt(42+22+42) x Sqrt(52+32+32))=0.97

Sim(Mohammad,Huashun) = (4x3)+(5x4)+(4x3)/(Sqrt(42+52+42) x Sqrt(32+42+32))=1.00

Calculate k as a normalising factor

k = 1/((0.97+1)) = 0.51

R(Mohammad, Superman) = k x
((*sim*(Mohammad,Paras)∗*R*(Paras,*Superman*))+(*sim*(Mohammad,
Huashun)∗*R*(Huashun,*Superman*)))
= 0.51((0.97∗4)+(1 ∗4))
= 4.02
Top-2(Harry, movies)= Batman, Superman

```
from pyspark.ml.recommendation import _____
from pyspark.ml.evaluation import_____

Task #1: # split the dataset into training and test data (70% training and 30% test)
(trainingData, testData) = _____

Task #2: Build the recommendation model using ALS on the training data
# Use maxIter = 5, regParam = 0.01, coldStartStrategy = "drop",
# implicitPrefs = False

# make predictions
predictions = model.transform(testData)

Task #3: Find and print the accuracy of the model
# Write code below
```

```
from pyspark.ml.evaluation import RegressionEvaluator

from pyspark.ml.recommendation import ALS

# split the dataset into training and test data (70% training and 30% test)

(trainingData, testData) = ratings.randomSplit([0.7, 0.3])

 # build the recommendation model using ALS on the training data

# Use maxIter = 5, regParam = 0.01, coldStartStrategy = “drop”,

# implicitPrefs = False

als = ALS(maxIter=5, regParam=0.01, userCol="userId", itemCol="movieId",
ratingCol="rating", coldStartStrategy="drop")

model = als.fit(trainingData)

# make predictions

predictions = model.transform(testData)

# find and print the accuracy of the model

evaluator = RegressionEvaluator(metricName="rmse", labelCol="rating",

                                predictionCol="prediction")

rmse = evaluator.evaluate(predictions)

print("Accuracy is = " + str(1-rmse))
```

Question 5

StopHacking is a start-up incubated in Monash University to develop cloud service to detect and stop computer hackers. Although they have some rule-based service to identify certain hacks, they would like to add machine learning models which can integrate with their Spark cluster to process large amounts of data and detect any potential hacks. The dataset contains an “attack” column representing whether the request was an attack or not.

They hired you as the Lead *Data Scientist* and Peter (your intern) to investigate the open data from the Cyber Range Labs of UNSW Canberra and build a model based on the data to identify abnormal system behaviour.

Before proceeding with the development of ML models, Peter has some questions in mind that he would like your input on.

1. Peter is not sure whether this is a classification or a regression problem. Is this a classification or a regression problem? Briefly discuss when do we use classification and regression with examples. **(2 Marks)**
   This is a classification problem. Here we are required to decide whether the row is an attack or not, i.e. predicting which class the target variable belongs to which qualifies it as a regression problem. On the other hand, regression is used when the output variable is real or continuous. For example, if we want to predict the real-estate prices based on historical data, it is a regression problem.

2. Upon investigation of the data, Peter has found that the data is imbalanced. Please suggest ways to handle an imbalanced dataset. **(1 Marks)**
   One way to handle imbalanced data is resampling using stratified sampling to represent the data from both classes in a balanced ratio.
   Other approaches include using different performance metrics like Precision/Recall/F1-Score instead of accuracy to gauge the model performance.

3. You have prepared an estimator for the Decision Tree Model. Executing a Decision tree algorithm is a simple task. But, Peter still has some doubts. **(2 + 3 = 5 Marks)**
   a. How does a tree splitting take place? Explain in the context of the ID3 algorithm.
   b. The models perform great on the training data but generalize poorly to new instances. Peter is not sure what is happening. Can you explain what is happening and suggest two possible solutions.
   a. The variable with highest information gain is selected as the splitting node and this process is repeated.
   b. Generalizing poorly with new instances means the model is overfitted where it performs very well with training dataset but poorly with test dataset. We can handle overfitting using techniques like Cross validation and regularization.

4. What are False Positive(FP) and False Negative(FN) in a confusion matrix? Which value should we try to reduce in this scenario, discuss briefly? **(2 Marks)**

False Positive means when the predicted class is positive but the label is negative whereas False Negative means the predicted class is negative but the actual label is positive.

In our use case, False Negative would mean the algorithm predicts an event as “non-attack” even when the event is an “attack”. This could mean, the attack goes undetected which could compromise the security resulting in serious consequences. On the contrary, False positive would mean the event is detected as an “attack” despite being a “non-attack”, which wouldn’t be so critical.

Question 6

Spectroscopy products developed at Divergent Technologies generate a lot of performance and diagnostic data. The data is typically stored locally on the controlling PC’s hard disk drive and only analysed for the purpose of reviewing function and performance as a part of short term test requirements. Further analysis (such as trend analysis, predictive analytics, comparative studies, regression / correlation, etc.) is currently very challenging and is done manually on an as-needs basis.

You and Neha have been hired as summer interns to implement machine learning algorithms with the data generated by the spectroscopy products. These spectroscopy products have sensor arrays installed and it is anticipated that using ML techniques could prove extremely valuable that enable timely preventative maintenance of the sensors and / or responsive lower cost repairs. Ultimately, it may lead to the development of a sale-able product in this area, with potential use across the broader Divergent instrument portfolio. You are working on streaming data from the sensors and Neha has some questions for you before she can develop the machine learning models.

1. The spectroscopy product has multiple sensors attached to it that measures different things for example light, gas and heat emission. Can you please explain two different methods that can be used to lower the granularity of the sensor arrays? **(3 Marks)**

   There are two methods to lower the granularity of sensor arrays that measure the different thing.
   (a) Method 1: Reduce, Normalise, and then Merge

   The first step is to reduce the granularity level of each sensor’s raw Data. Each sensor measures different things, so we need to normalise the raw data of each sensor, by categorising each raw data into several categories. The final step is to Merge this normalised data. The merging process can use a mean function which calculates the average of the categories. The mean could be a floating point number which indicates the average.

   (b) Method 2: Normalise, Merge and then Reduce

   The first step of the second method is to normalise each stream according to the categories. This normalisation is a Value Normalisation, which is basically reducing the granularity of the raw data into categories.The second step is to merge the normalised data streams. The merging is basically a 1-1 join operation based on the Timestamp, and then the matched records are aggregated based on the

normalised sensor values. The third step is to reduce the granularity of the merged results.

2. There are three main sensors in the Spectroscopy products. So, Neha is planning to send the data using three Kafka producers using the same topic "spectroscopy_streams". The sensors are producing data as key value pairs in the format below and sent as bytes.

"gas": 125
"light": 3298
"heat": 78

In the Apache Spark Streaming, the received data looks like below.

```
+----------------+-------------+--------------------+---------+------+--------------------+-------------+
|             key|        value|               topic|partition|offset|           timestamp|timestampType|
+----------------+-------------+--------------------+---------+------+--------------------+-------------+
|      [67 61 73]|   [31 34 30]|spectroscopy_streams|        0|   261|2020-10-18 00:06:...|            0|
|[6C 69 67 68 74]|[33 31 31 31]|spectroscopy_streams|        0|   262|2020-10-18 00:06:...|            0|
|   [68 65 61 74]|   [31 32 33]|spectroscopy_streams|        0|   263|2020-10-18 00:06:...|            0|
+----------------+-------------+--------------------+---------+------+--------------------+-------------+
```

Please complete the code below for Apache Spark Streaming to find the average for each sensor every 10 seconds.

```
from pyspark.sql import SparkSession

spark = SparkSession. ...

Task #1: # Subscribe to the topic "spectroscopy_streams". The server is running on 192.168.0.10, port 9092.

Task #2:  Find the average for each sensor.

Task #3: # Start running the query that prints the running counts to the console every 10 seconds.


query.awaitTermination()
```

The output will be as shown in the example below.

```
+-----+------------------+
|  key|        avg(value)|
+-----+------------------+
| heat| 90.22222222222223|
|  gas|126.88888888888889|
|light|3348.3333333333335|
+-----+------------------+
```

(3 Marks)

```
from pyspark.sql import SparkSession

spark = SparkSession. ...
```

```
Task #1: # Subscribe to the topic “spectroscopy_streams”. The server is running on 192.168.0.10, port 9092.

df = spark \
  .readStream \
  .format("kafka") \
  .option("kafka.bootstrap.servers", "192.168.0.10:9092") \
  .option("subscribe", "spectroscopy_streams") \
  .load()

Task #2:  Find the average for each sensor.
average_df = df.withColumn("key", decode(col("key"),'utf-8'))\
.withColumn("value",decode(col("value"),'utf-8'))\
.groupBy("key")\
.agg({'value':'mean'})

Task #3: # Start running the query that prints the running counts to the console every 10 seconds.

query = average_df \
  .writeStream \
  .outputMode("Complete")\
  .format("console") \
  .trigger(processingTime='10 seconds') \
  .start()

query.awaitTermination()
```

3. Is the windowing method mentioned in the question time based window or tuple based window? Please explain. How can you enable time based overlapping sliding windows in Apache Spark Structured Streaming? **(4 Marks)**

   It is a time based window. We have a window size of 10 seconds and we process the records that appear within the window. There is no overlapping between the windows in this case.

   We can enable time based overlapping sliding in Apache Spark Structured streaming using window-based aggregations.

   In window-based aggregations, the aggregate values are maintained for each window the event-time of a row falls into. In code, we can use window() operations to express windowed aggregations.

   words = ... # streaming DataFrame of schema { timestamp: Timestamp, word: String }

   # Group the data by window and word and compute the count of each group

   windowedCounts = words.groupBy(

```
        window(words.timestamp, "10 minutes", "5 minutes"),

        words.word

    ).count()
```

https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html#window-operations-on-event-time