uOttawa

ELG5166 Cloud Analytics

Take Home Exam

Instructor: Dr. Benjamin Eze

Student Name: Ali El-Sherif

ID: 300327246

Table of Contents

Personal Ethics & Academic Integrity Statement

By typing in my name and student ID on this form and submitting it electronically, I am attesting to the fact that I have reviewed not only my work but the work of my team member, in its entirety.

I attest to the fact that my work in this project adheres to the fraud policies as outlined in the Academic Regulations in the University's Graduate Studies Calendar. I further attest that I have knowledge of and have respected the "Beware of Plagiarism" brochure for the university. To the best of my knowledge, I also believe that each of my group colleagues has also met the aforementioned requirements and regulations. I understand that if my group assignment is submitted without a completed copy of this Personal Work Statement from each group member, it will be interpreted by the school that the missing student(s) name is confirmation of non-participation of the aforementioned student(s) in the required work. We, by typing in our names and student IDs on this form and submitting it electronically,

- warrant that the work submitted herein is our own group members' work and not the work of others.

- acknowledge that we have read and understood the University Regulations on Academic Misconduct.

- acknowledge that it is a breach of University Regulations to give or receive unauthorized and/or unacknowledged assistance on a graded piece of work.

Question 1:

First assumption (All nodes will fail at the same time)

Probability = $(\frac{24}{365})^5 = 1.229 \times 10^{-6}$

Uptime = 1 – probability = 1 – 1.229×10^{-6} = 0.9999987

Uptime percentage = uptime × 100 = 99.99987%

Availability = 5 nines

Second assumption (All nodes failure calculated throughout the year)

Probability $= \left(\frac{24 \times 5}{365 \times 5}\right) = 0.065753$

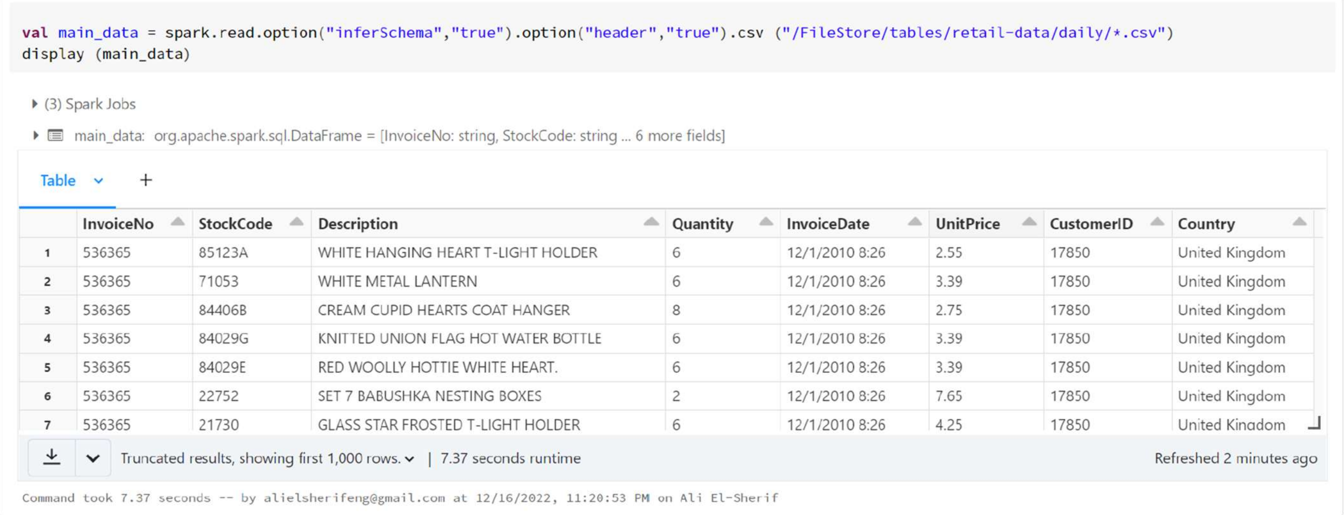
Uptime = 1 – probability = 1 – 0.065753 = 93.4247

Uptime percentage = uptime × 100 = 93.4247%

Availability = one nine

Question 2:

Reading the data as spark dataframe:



```
val main_data = spark.read.option("inferSchema","true").option("header","true").csv ("/FileStore/tables/retail-data/daily/*.csv")
display (main_data)
```

▸ (3) Spark Jobs

▸ main_data: org.apache.spark.sql.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]

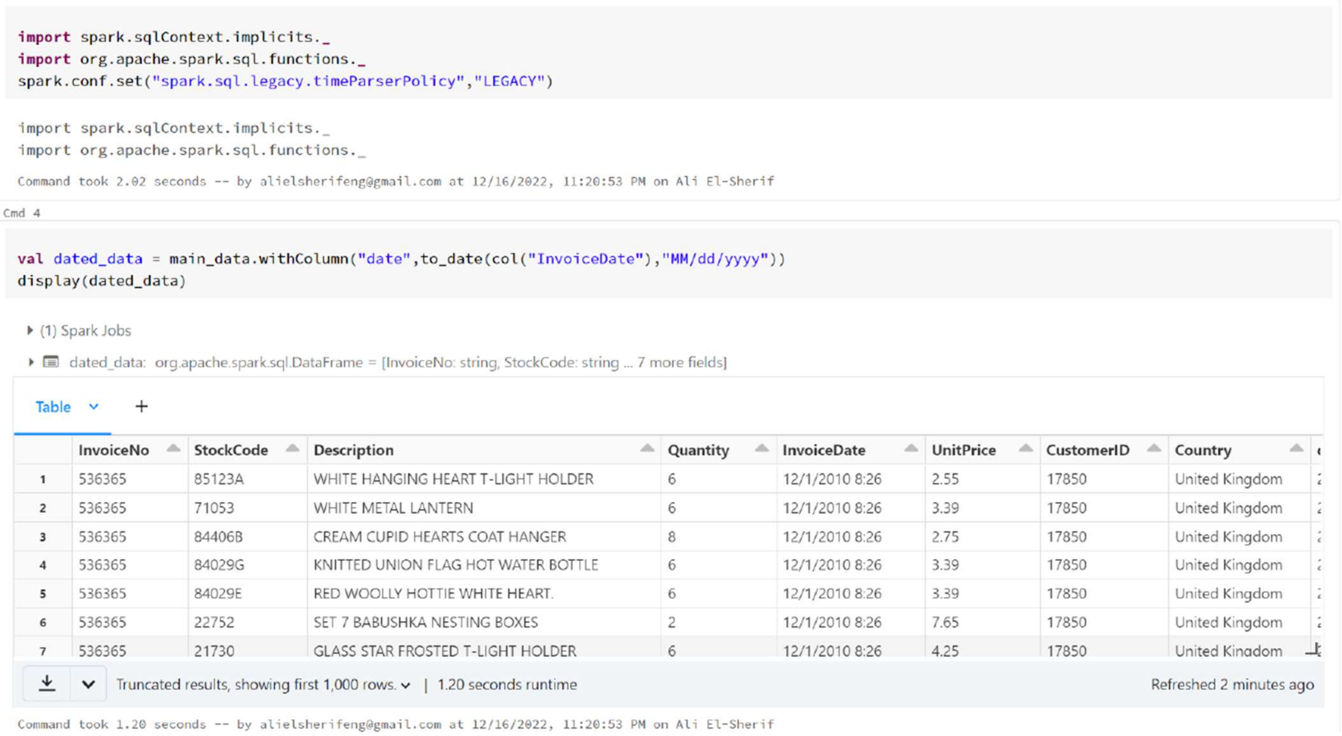
	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850	United Kingdom
2	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850	United Kingdom
3	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850	United Kingdom
4	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850	United Kingdom
5	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850	United Kingdom
6	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010 8:26	7.65	17850	United Kingdom
7	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12/1/2010 8:26	4.25	17850	United Kingdom

Truncated results, showing first 1,000 rows. | 7.37 seconds runtime

Refreshed 2 minutes ago

```
Command took 7.37 seconds -- by alielsherifeng@gmail.com at 12/16/2022, 11:20:53 PM on Ali El-Sherif
```

Change date column from string to date format:



```
import spark.sqlContext.implicits._
import org.apache.spark.sql.functions._
spark.conf.set("spark.sql.legacy.timeParserPolicy","LEGACY")
```

```
import spark.sqlContext.implicits._
import org.apache.spark.sql.functions._
```

```
Command took 2.02 seconds -- by alielsherifeng@gmail.com at 12/16/2022, 11:20:53 PM on Ali El-Sherif
```

Cmd 4

```
val dated_data = main_data.withColumn("date",to_date(col("InvoiceDate"),"MM/dd/yyyy"))
display(dated_data)
```

▸ (1) Spark Jobs

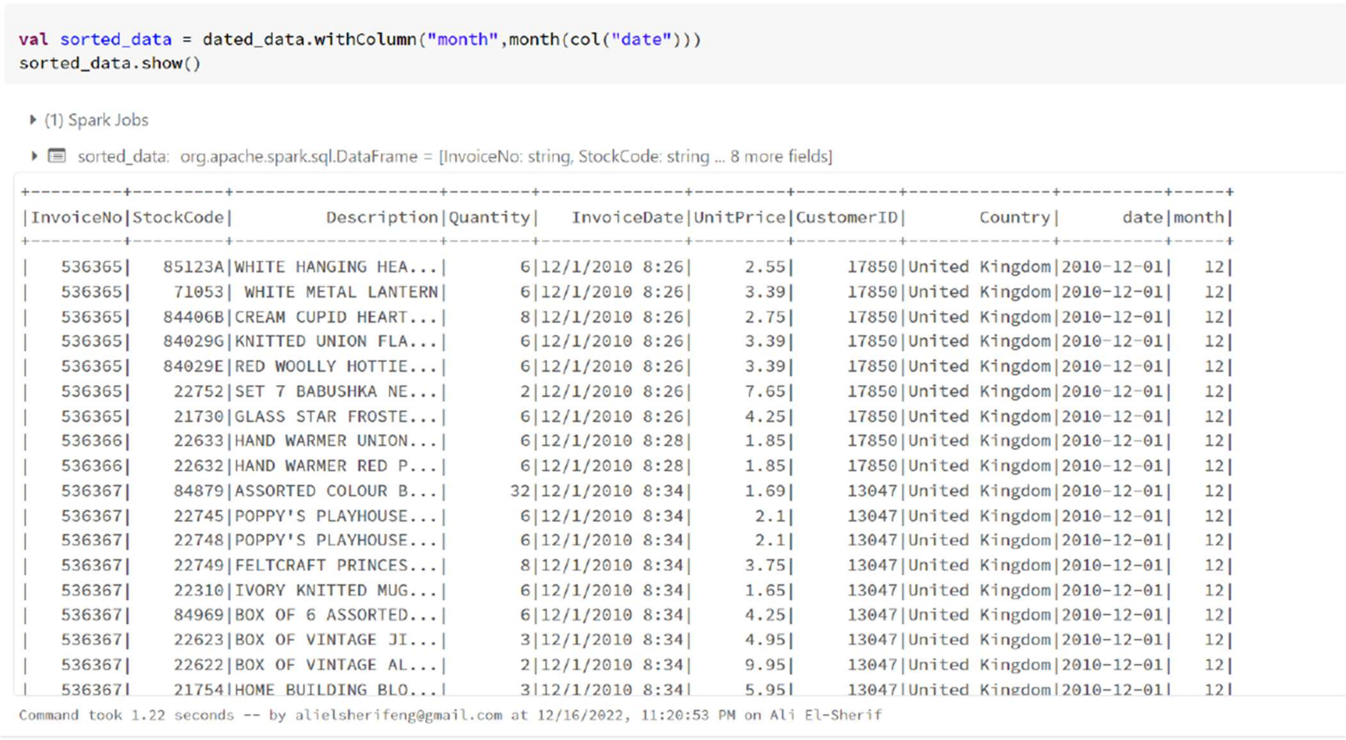
▸ dated_data: org.apache.spark.sql.DataFrame = [InvoiceNo: string, StockCode: string ... 7 more fields]

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850	United Kingdom
2	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850	United Kingdom
3	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850	United Kingdom
4	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850	United Kingdom
5	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850	United Kingdom
6	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010 8:26	7.65	17850	United Kingdom
7	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12/1/2010 8:26	4.25	17850	United Kingdom

Truncated results, showing first 1,000 rows. | 1.20 seconds runtime   Refreshed 2 minutes ago

```
Command took 1.20 seconds -- by alielsherifeng@gmail.com at 12/16/2022, 11:20:53 PM on Ali El-Sherif
```

Add a month column as follows:



```
val sorted_data = dated_data.withColumn("month",month(col("date")))
sorted_data.show()
```

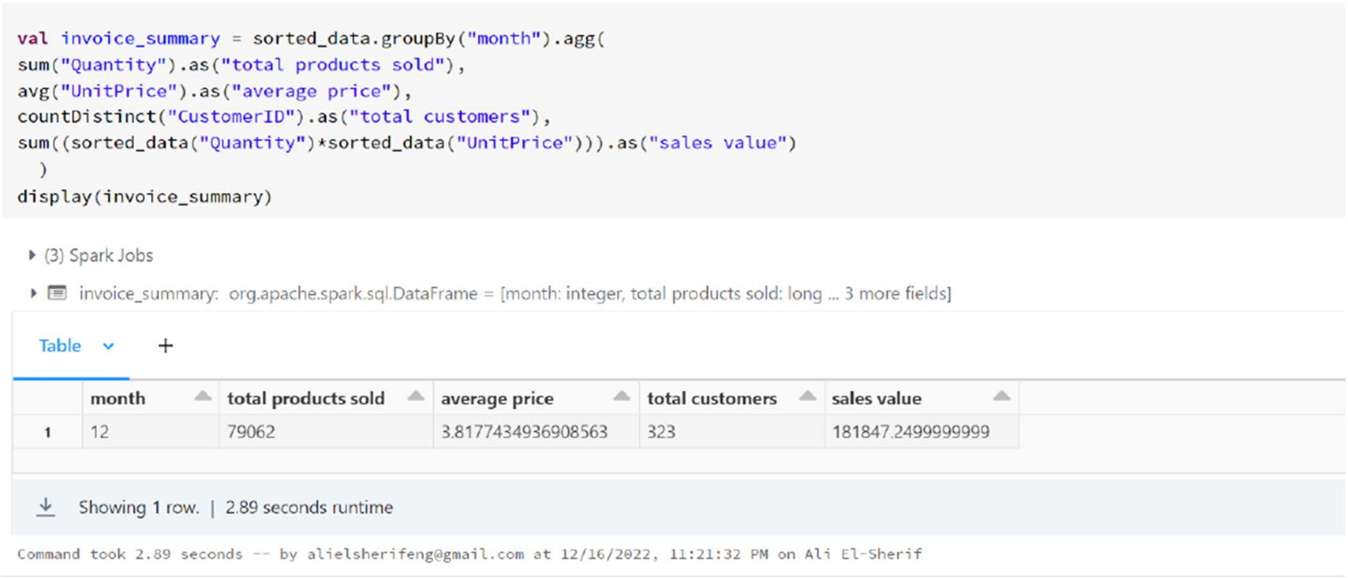
▸ (1) Spark Jobs

▸ sorted_data: org.apache.spark.sql.DataFrame = [InvoiceNo: string, StockCode: string ... 8 more fields]

```
+---------+---------+--------------------+--------+--------------+---------+----------+--------------+----------+-----+
|InvoiceNo|StockCode|         Description|Quantity|   InvoiceDate|UnitPrice|CustomerID|       Country|      date|month|
+---------+---------+--------------------+--------+--------------+---------+----------+--------------+----------+-----+
|   536365|   85123A|WHITE HANGING HEA...|       6|12/1/2010 8:26|     2.55|     17850|United Kingdom|2010-12-01|   12|
|   536365|    71053| WHITE METAL LANTERN|       6|12/1/2010 8:26|     3.39|     17850|United Kingdom|2010-12-01|   12|
|   536365|   84406B|CREAM CUPID HEART...|       8|12/1/2010 8:26|     2.75|     17850|United Kingdom|2010-12-01|   12|
|   536365|   84029G|KNITTED UNION FLA...|       6|12/1/2010 8:26|     3.39|     17850|United Kingdom|2010-12-01|   12|
|   536365|   84029E|RED WOOLLY HOTTIE...|       6|12/1/2010 8:26|     3.39|     17850|United Kingdom|2010-12-01|   12|
|   536365|    22752|SET 7 BABUSHKA NE...|       2|12/1/2010 8:26|     7.65|     17850|United Kingdom|2010-12-01|   12|
|   536365|    21730|GLASS STAR FROSTE...|       6|12/1/2010 8:26|     4.25|     17850|United Kingdom|2010-12-01|   12|
|   536366|    22633|HAND WARMER UNION...|       6|12/1/2010 8:28|     1.85|     17850|United Kingdom|2010-12-01|   12|
|   536366|    22632|HAND WARMER RED P...|       6|12/1/2010 8:28|     1.85|     17850|United Kingdom|2010-12-01|   12|
|   536367|    84879|ASSORTED COLOUR B...|      32|12/1/2010 8:34|     1.69|     13047|United Kingdom|2010-12-01|   12|
|   536367|    22745|POPPY'S PLAYHOUSE...|       6|12/1/2010 8:34|      2.1|     13047|United Kingdom|2010-12-01|   12|
|   536367|    22748|POPPY'S PLAYHOUSE...|       6|12/1/2010 8:34|      2.1|     13047|United Kingdom|2010-12-01|   12|
|   536367|    22749|FELTCRAFT PRINCES...|       8|12/1/2010 8:34|     3.75|     13047|United Kingdom|2010-12-01|   12|
|   536367|    22310|IVORY KNITTED MUG...|       6|12/1/2010 8:34|     1.65|     13047|United Kingdom|2010-12-01|   12|
|   536367|    84969|BOX OF 6 ASSORTED...|       6|12/1/2010 8:34|     4.25|     13047|United Kingdom|2010-12-01|   12|
|   536367|    22623|BOX OF VINTAGE JI...|       3|12/1/2010 8:34|     4.95|     13047|United Kingdom|2010-12-01|   12|
|   536367|    22622|BOX OF VINTAGE AL...|       2|12/1/2010 8:34|     9.95|     13047|United Kingdom|2010-12-01|   12|
|   536367|    21754|HOME BUILDING BLO...|       3|12/1/2010 8:34|     5.95|     13047|United Kingdom|2010-12-01|   12|
```

```
Command took 1.22 seconds -- by alielsherifeng@gmail.com at 12/16/2022, 11:20:53 PM on Ali El-Sherif
```

Showing the monthly invoice aggregate summary:



```
val invoice_summary = sorted_data.groupBy("month").agg(
sum("Quantity").as("total products sold"),
avg("UnitPrice").as("average price"),
countDistinct("CustomerID").as("total customers"),
sum((sorted_data("Quantity")*sorted_data("UnitPrice"))).as("sales value")
  )
display(invoice_summary)
```

```
▸ (3) Spark Jobs
▸ invoice_summary: org.apache.spark.sql.DataFrame = [month: integer, total products sold: long ... 3 more fields]
```

	month	total products sold	average price	total customers	sales value
1	12	79062	3.8177434936908563	323	181847.2499999999

```
Showing 1 row. | 2.89 seconds runtime
Command took 2.89 seconds -- by alielsherifeng@gmail.com at 12/16/2022, 11:21:32 PM on Ali El-Sherif
```

Question 3:

a)

Since disk size = 512, log size = 112, replication factor = 3

Available disk size per node = 512 – 112 = 400 GB/node

Number of data nodes = $\frac{(300 \times 1024)}{400} \times 3 = 2304$

Total number of nodes = 2304 + 768 = 3072

b)

Since log files = 640 GB, 4 containers per node

let the number of containers = number of partitions

so that available memory in HDFS per node = 64 – 14=50 GB/node

number of nodes = $\frac{640}{50} = 12.8 \ \cong 13\, nodes.$

number of partitions = $13 \times 4 = 52\; partitions$

Question 4:

a)

Number of sensors = $\frac{Pipeline\ length}{maintaenance\ point\ distance}$ = $\frac{6000}{1.5} = 4000\ sensors$

b)

Number of total events = $\frac{4000}{30}$ = 133.333 events per second

Handled by each partition = $\frac{1500}{60}$ = 25 event per second

Number of partitions = $\frac{\text{Number of total events}}{\text{Handled by each partition}} = 5.333\ \cong 6\ partitions$

c)

According to the documentation [1], in basic or standard tier:

we will need 1 event hub with number of partitions = 32 (6 partitions needed)

Number of event hubs per namespace = 10, so 1 namespace is enough in this case

d)

Job query for leakage (<800 PSI):

```
SELECT SerialNumber, Longitude, Latitude, ReadingTime,
PressureReading, Count(*) as CountofSensors
FROM StreamData
TIMESTAMP BY ReadingTime
WHERE PressureReading < 800
GROUP BY SerialNumber, SlidingWindow(Minute,5)
HAVING CountofSensors  >= 3
```

Job query for blockage (>1200 PSI):

```
SELECT SerialNumber, Longitude, Latitude, ReadingTime,
PressureReading, Count(*) as CountofSensors
FROM StreamData
TIMESTAMP BY ReadingTime
WHERE PressureReading >1200
GROUP BY SerialNumber, SlidingWindow(Minute,5)
HAVING CountofSensors  >= 3
```

e)

Since there are 2 actions that can be taken, 2 consumer groups are needed and can be divided as follows:

- In case of obstruction or blockage downstream (>$1200\ PSI$), the pump will shut off until pressure normalizes.
- In case of leakage ($< 800PSI$), an alert will be triggered for the maintenance point associated with the sensor.

References

[1] spelluru. (n.d.). *Quotas and limits - Azure Event Hubs - Azure Event Hubs*. Learn.microsoft.com. https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-quotas