



## ELG 5166 – Cloud Analytics Exam

### Personal Ethics & Academic Integrity Statement

Student name: **Hosam Mahmoud Ibrahim Mahmoud**

Student ID: **300327269**

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 my work in its entirety.

I attest to the fact that this submission adheres to the fraud policies as outlined in the Academic Regulations in the University's Undergraduate Studies Calendar. I further attest that I have knowledge of and have respected the "Beware of Plagiarism" brochure found on the University of Ottawa website.

I, by typing in my name and student ID on this form and submitting it electronically,

- warrant that the work submitted herein is your work and not the work of others.
- acknowledge that I 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.

## Q1)

- A **service level agreement** typically describes the parties involved, the nature of the service rendered, quality or performance metrics for the services, how and when the contract may be terminated, and how to resolve service quality issues.
- In their SLAs, most cloud service providers, for example, frequently guarantee 99.99% (often referred to as four nines) or even 99.999% uptime for their services. And if they don't deliver, they'll give service credits (points that can be redeemed for cloud services).

Reference: [What is a Service Level Agreement \(SLA\) in IT Service Delivery \(invgate.com\)](https://www.invgate.com/what-is-a-service-level-agreement-sla-in-it-service-delivery/)

- **There will be 2 cases:**

1- The first case that the **entire cluster** will be down 24 days in the year.

- which mean that the **Downtime** will be  $\left(\frac{24}{365}\right) * 100 = 6.575\%$
- So, the **Uptime** will be  $(1 - \text{Downtime}) * 100 = 93.424657\%$
- Which mean that the **Availability** = 1 nine.

2- The second case that **each node in the cluster** is expected to have up to 24 days Downtime in the year.

- which mean that the **Downtime** will be  $\left(\frac{24}{365}\right) * \left(\frac{24}{365}\right) * \left(\frac{24}{365}\right) * \left(\frac{24}{365}\right) * \left(\frac{24}{365}\right) * 100 = 0.000122911\%$
- So, the **Uptime** will be  $(1 - \text{Downtime}) * 100 = 99.99987\%$
- Which mean that the **Availability** = 5 nines.

## Q2)

- Upload the csv files into my account:

**Create New Table**

Data source

**Upload File** S3 DBFS Other Data Sources

DBFS Target Directory

/FileStore/tables/ /retail-data/daily

Files uploaded to DBFS are accessible by everyone who has access to this workspace. [Learn more](#)

Files

data-2010-12-01.csv 0.3 MB <a href="#">Remove file</a>	data-2010-12-02.csv 0.2 MB <a href="#">Remove file</a>	data-2010-12-03.csv 0.2 MB <a href="#">Remove file</a>
data-2010-12-05.csv 0.2 MB <a href="#">Remove file</a>		

File uploaded to /FileStore/tables/retail-data/daily/data\_2010\_12\_02.csv  
 File uploaded to /FileStore/tables/retail-data/daily/data\_2010\_12\_01.csv  
 File uploaded to /FileStore/tables/retail-data/daily/data\_2010\_12\_03.csv  
 File uploaded to /FileStore/tables/retail-data/daily/data\_2010\_12\_05.csv

[Create Table with UI](#) [Create Table in Notebook](#)

- Creating Cluster.

The screenshot shows the Databricks Clusters page for a cluster named 'Retail data'. The configuration is as follows:

- Configuration** tab is selected.
- Databricks Runtime Version:** 9.1 LTS (includes Apache Spark 3.1.2, Scala 2.12)
- Driver type:** Community Optimized (15.3 GB Memory, 2 Cores, 1 DBU)
- Instance:** Free 15 GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours. For more configuration options, please upgrade your Databricks subscription.
- Instances** tab is selected.
- Availability zone:** us-west-2c

- Read the data:

The screenshot shows a Databricks Notebook with the following Scala code:

```
1 val path = "/FileStore/tables/retail-data/daily/*.csv"
2 val Retail_Data = spark.read.option("header", "true").csv(path)
3 display(Retail_Data)
```

The command was executed successfully, resulting in a table with 7 rows and 9 columns. The table is truncated, showing the first 1,000 rows. The runtime took 2.44 seconds.

	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

- Create the Month column to be able to aggregate

The screenshot shows a Databricks Notebook with the following Scala code:

```
1 import org.apache.spark.sql.functions._
2 import spark.sqlContext.implicits._
3 spark.conf.set("spark.sql.legacy.timeParserPolicy", "LEGACY")
4 val Retail_Data_new = Retail_Data.withColumn("Date", to_date(col("InvoiceDate"), "dd/mm/yyyy"))
5 val Retail_Data_new1 = Retail_Data_new.withColumn("Month", month(col("Date")))
6 display(Retail_Data_new1)
```

The command was executed successfully, resulting in a table with 7 rows and 10 columns. The table is truncated, showing the first 1,000 rows. The runtime took 2.65 seconds.

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Date	Month
1	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850	United Kingdom	2010-01-12	1
2	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850	United Kingdom	2010-01-12	1
3	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850	United Kingdom	2010-01-12	1
4	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850	United Kingdom	2010-01-12	1
5	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850	United Kingdom	2010-01-12	1
6	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010 8:26	7.65	17850	United Kingdom	2010-01-12	1
7	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12/1/2010 8:26	4.25	17850	United Kingdom	2010-01-12	1

- Aggregation:

**Retail data** Scala

File Edit View Run Help Last edit was 2 minutes ago Give feedback

```

1 val monthly_invoice_summary = Retail_Data_new1.groupBy("Month").agg(
2   sum("Quantity").as("total products sold"),
3   avg("UnitPrice").as("average price"),
4   countDistinct("CustomerID").as("total customers"),
5   sum((Retail_Data_new1("Quantity")*Retail_Data_new1("UnitPrice"))).as("sales value")
6 )
7 display(monthly_invoice_summary)

```

(3) Spark Jobs

monthly\_invoice\_summary: org.apache.spark.sql.DataFrame = [Month: integer, total products sold: double ... 3 more fields]

Table +

	Month	total products sold	average price	total customers	sales value
1	1	79062	3.8177434936908563	323	181847.24999999985

Showing 1 row. | 3.78 seconds runtime Refreshed now

Command took 3.78 seconds -- by hosbadawy1@hotmail.com at 12/18/2022, 4:19:44 PM on Retail data

Q3)

### 1- Calculate the number of nodes.

- The disk size = **512GB**, and this disk has a used storage which is **112GB**, so HDFS will take the remaining empty storage space which is 400GB.
- **HDFS = disk size – the used storage = 512GB - 112GB = 400GB**
- **number of nodes = ((the size of data needs to be procced (in giga byte)) \* replication factor) / available size of HDFS.**
- **number of nodes = ((300\*1024) \*3)/400 = 2304 node.**
- It's required to be **2304 node** within the Hadoop YARN cluster to store the entire 300 TB of data in HDFS.

### 2- Calculate the number partitions and nodes.

- Log files need to be procced = **640GB**
- memory allocation per node = **64GB memory - 14GB memory = 50GB**
- Num of nodes = **log files / memory allocation per node = 640GB / 50GB = 12.8 = 13 node**
- Number of partitions = **number of containers per node \* number of partitions on each container \* number of nodes**
- Number of partitions = **4 \* 1 (assume that there is 1 partition in each container) \* 13 = 52 partitions**

Q4)

### 1- Calculate the number of sensors

- The Number of sensors = **meters / 1.5 Km for each sensor**
- The Number of sensors = **6000/ 1.5 = 4000 sensors**

### 2- Calculate the number of events and partitions

- As it's mentioned in the question the sensor receives an event each 30 seconds, which mean that the sensor receives **2 events per minute.**
- **The Total number of events for all sensors per minute = 4000 \* 2 = 8000 events per minute**
- **Then Total number of partitions = 8000/500 = 16 partitions**

3- Calculate the number of events hubs and namespaces.

- I will assume that we will use basic tier according to Microsoft tiers which has 10 event hubs per namespace, and 32 partitions per event hub.
- Then Number of event hubs = **number of partitions / partitions per event hub = 16 / 32 = 0.5 = 1 event hub**
- Then Number of namespaces = **number of event hubs / event hubs per namespace = 1/10 = 0.1 = 1 namespace**

4- ASA Job queries:

- **ASA job query for below 800 psi (leak alert)**

```
1 %sql
2 with input_data as(
3     select *
4     from [event_hub]
5     TIMESTAMP by process_date
6 ),
7
8 PSI_800_data as(
9     select *
10    from [input_data]
11    where PressureReading < 800
12 )
13
14 select COUNT(*), SerialNumber
15 into
16 [output_data]
17 from
18 [PSI_800_data]
19 group by SerialNumber, SlidingWindow(minute,5)
20 having COUNT (*) >= 3
```

- **ASA job query for above 1200 psi (obstruction or blockage downstream)**

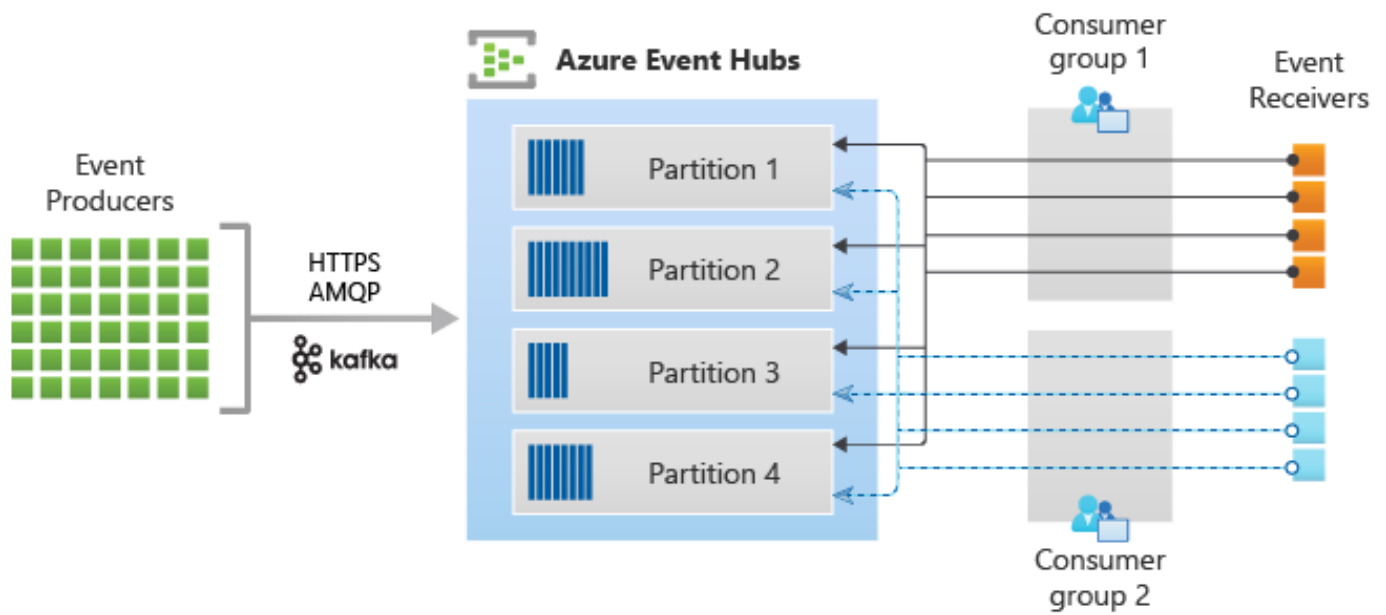
```
WITH input_data AS(
    SELECT *
    FROM [event_hub]
    TIMESTAMP BY process_date
),

PSI_1200_data AS(
    SELECT *
    FROM [input_data]
    WHERE PressureReading > 1200
)

SELECT COUNT(*), SerialNumber
INTO
[output_data]
FROM
[PSI_1200_data]
GROUP BY SerialNumber, SlidingWindow(minute,5)
HAVING COUNT (*) >= 3
```

5- Number of consumers groups, their roles, and what to do to support receivers:

- There must be **two consumers for each adverse event**, because there is 2 type of events, one for leakage and the other one for blockage.
- The consumer groups are responsible for allowing multiple consuming applications to have their own view of the event stream and read it independently at their own pace and offsets.
- Distribute the events across all partitions, and when publishing events on a regular basis, use the AMQP protocol whenever possible.



Preferred Event Hubs stream processing architecture.

Reference: [Overview of features - Azure Event Hubs - Azure Event Hubs | Microsoft Learn](#)