



**UNIVERSITY  
OF MALAYA**

**FACULTY OF COMPUTER SCIENCE AND  
INFORMATION TECHNOLOGY  
MASTER IN DATA SCIENCE**

**WQD7009 Big Data Applications and  
Analytics**

**GROUP PROJECT**

**Smart Supply Chain Big Data Application  
with Google Cloud Platform**

# Contents

<b>1. Introduction</b>	<b>3</b>
1.1. Overview	3
1.2. Current Challenges	3
1.3. Objectives	4
1.4. Project Scope and Data Understanding	4
<b>2. Five (5) layers data lifecycle</b>	<b>5</b>
2.1. Overview of Google Cloud Platform (GCP)	5
2.2. Framework lifecycle	5
2.3. Data Ingestion	6
2.4. Data Storage	6
2.5. Data Processing	7
2.6. Data Analytics	7
2.7. Data Visualisation	7
<b>3. Cloud Based Tools Selection and Justification</b>	<b>8</b>
<b>4. Implementation</b>	<b>10</b>
<b>5. Evaluation Metric</b>	<b>15</b>
<b>6. Conclusion</b>	<b>20</b>
<b>References</b>	<b>21</b>

# 1. Introduction

## 1.1. Overview

Supply Chain is the network of all the individuals, organisations, resources, activities and technology involved in the creation and sale of a product. A supply chain encompasses everything, from the delivery of source materials from the supplier to the manufacturer through to its eventual delivery to the end user (Lutkevich, 2021). The fundamental steps of supply chain comprises of six (6) basic steps as follows;

- Sourcing of raw materials
- Refining maternal into basic material
- Order fulfilment/sales
- Product delivery
- Customer support and return services

Smart Supply Chain Management is an innovative approach in managing supply chains and it has become increasingly popular in the current complex business climate. Companies are leveraging the power of advanced technologies, such as Artificial Intelligence (AI), machine learning, and the Internet of Things, to optimise their supply chains in order to remain competitive

As the scope of Supply Chain is currently worldwide, the volume of data collected from its numerous processes and the velocity at which it is being generated can be qualified as Big Data. In addition to this, entities such as marketing and sales are now relying on analysis of the unstructured data along with the structured data to gain better insights on customer needs and improve upon the cost aspects of Supply Chain processes (Awwad, et al., 2018).

## 1.2. Current Challenges

There are several challenges that are currently experienced by the supply chain sector as below but not limited to:

- **Complexity:** Supply chains can be complex, involving many different stakeholders and processes. This can make it difficult to manage effectively and efficiently.
- **Visibility:** It can be challenging to get a complete and accurate view of what is happening across the entire supply chain, particularly if there are multiple layers or if there is a lack of transparency
- **Cost management:** Supply chains can be expensive to operate, with costs associated with transportation, warehousing, and other logistics. Finding ways

to reduce these costs while maintaining the quality of the supply chain is a key challenge.

- **Changing customer demands:** Customers' needs and preferences are constantly evolving, and supply chains need to be able to adapt to these changes in order to remain competitive. This can be challenging, particularly if the supply chain is not flexible or responsive.
- **Sustainability:** Ensuring that the supply chain is sustainable, both environmentally and socially, is becoming increasingly important. This can involve finding ways to reduce waste, minimise the carbon footprint, and ensure that suppliers and partners are operating ethically.

By tapping into data and analytics, companies can track and monitor key metrics like inventory levels, transportation routes, and supplier performance in real-time. This data can be used to optimise logistics, improve inventory management, enhance supplier collaboration, and increase visibility and transparency. Additionally, smart supply chains can help reduce costs and improve efficiency throughout the entire supply chain.

### 1.3. Objectives

The objectives of this project are:

- To develop Smart Supply Chain data framework lifecycle using Google Cloud Platform
- To implement the proposed framework into the selected supply chain dataset
- To evaluate and assess suitability of the framework based on the identified performance metrics

### 1.4. Project Scope and Data Understanding

In this study, an open source dataset from Kaggle website (*DataCo SMART SUPPLY CHAIN FOR BIG DATA ANALYSIS*, n.d.) is being used to further develop a Smart Supply Chain framework lifecycle using Google Cloud Platform. This dataset, which was sourced from DataCo Global, consists of structured data about products related to clothing, sports, and electronics supplies as described below:

Data type	Dataset name	Dataset format	Size	#of records
Structured	DataCoSupplyChainDataset	csv	93,553 KB	180,519

**Table 1.4.1 : Dataset Description**

## 2. Five (5) layers data lifecycle

### 2.1. Overview of Google Cloud Platform (GCP)

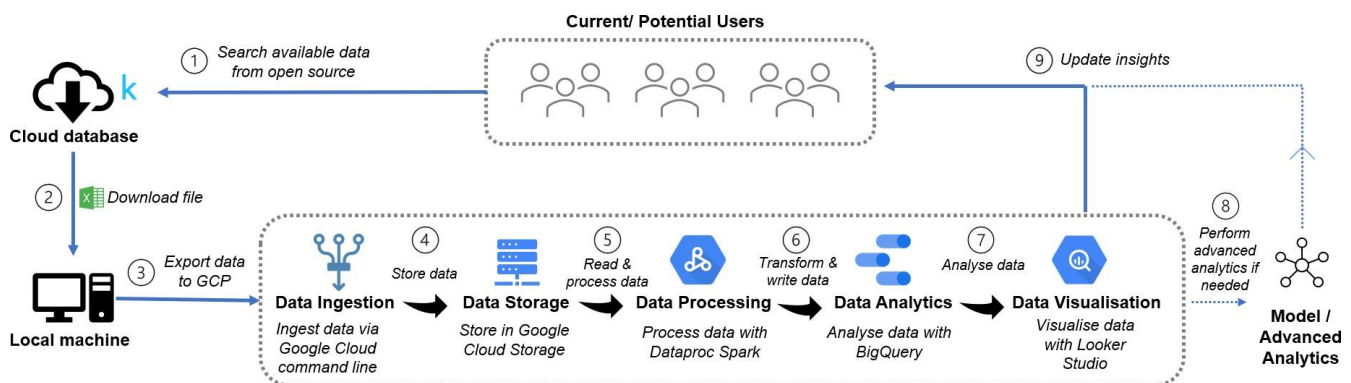
Google Cloud Platform (GCP) is a set of cloud computing services that utilises the same infrastructure as Google's internal products. In addition, it offers a variety of modular cloud services such as computing, data storage, data analytics, and machine learning. GCP offers infrastructure as a service, platform as a service, and serverless computing environments (*Google Cloud Platform*, n.d.).

### 2.2. Framework lifecycle

This project has proposed five (5) layers of data lifecycle to evaluate the smart supply chain management based on DataCo Global data. The goal of this framework is to understand and appraise the ability of GCP to conduct comprehensive data analytics on customer purchase and sales trends, as well as evaluate the logistics process based on product delivery status.

The aim of this framework is also to assess the execution time and determine memory consumption used to complete the overall process. In addition, this project will also share the user experience feedback based on UI/UX while conducting the project.

The 5 data lifecycle used and its selected tools along with version number are shown and described in Figure 2.2.1 and Table 2.2.1 below:



**Figure 2.2.1 : Proposed framework for Smart Supply Chain Management strategy using Google Cloud Platform**

Phase	Tool	Version	Description
Data Ingestion	GSutil	5.17	Load data from local machine and export to GCP Cloud Storage
Data Storage	Google Cloud Storage	2.7.0	Store raw dataset in the cloud storage
Data Processing	Dataproc Spark	2.0.x	Read and process the data from Bigtable
Data Analytics	BigQuery	1.40.0	Execute various command line to analyse the data
Data Visualisation	Looker Studio	22.20	Visualise the processed and analysed data with appropriate graph

**Table 2.2.1 Framework Lifecycle, Tools and Description**

### **2.3. Data Ingestion**

In this framework, the first step is to search appropriate dataset from any open source available that will be used for this study. The chosen dataset is then being downloaded from the website (kaggle) to our local machine and next being ingested into the Google Cloud Platform by using GSutil.

GSutil is a command-line tool to manage objects in Google Cloud Storage. It allows us to perform tasks such as uploading and downloading files, deleting and copying objects, and managing access controls.

### **2.4. Data Storage**

The data is stored in Google Cloud Storage as the main data storage with a dedicated bucket established that is named as 'data-co-supply-chain. Google Cloud Storage can store large amounts of data and make it available for big data analytics. As per described in Table 2.2.1, there are a total of 180k rows of data that need to be processed and analysed.

### **2.5. Data Processing**

Next, data processing is executed via Dataproc Spark to perform data cleaning, data scrubbing and data transformation before further data analysis with BigQuery. Dataproc Spark is a powerful and flexible tool for data processing and analytics and can help us to quickly analyse and understand the data.

## **2.6. Data Analytics**

Subsequently, data analytics is performed using BigQuery with a series of SQL queries. In the context of supply chain, BigQuery enables us to analyse data on orders, deliveries, inventory levels, and supplier performance, as well to identify trends and patterns that could help the business to optimise the supply chain.

## **2.7. Data Visualisation**

Finally, all the processed and analysed data is visualised using Looker Studio. Looker is a data analytics and visualisation platform that is available on Google Cloud. It allows us to explore, analyse, and visualise the data using a variety of tools and techniques such as by creating charts, graphs, and other visualisations as well identify trends and patterns in the supply chain data that could help business to optimise the operations.

Lastly, the proposed framework also includes Modeling/Advanced Analytics as part of the potential phase in the framework lifecycle that can be applied in the smart supply chain. However, this phase will not be further elaborate as this project identified Modelling/Advanced Analytics as an optional scope and best practice.

### 3. Cloud Based Tools Selection and Justification

#### 3.1. Data Ingestion Tool Justification

Google Cloud Storage (GCS) provides a command-line tool called GSutil, which serves as a powerful tool for data ingestion. Some of the reasons why GSutil is so effective for data ingestion includes:

- **High-speed data transfer:** GSutil allows for high-speed data transfer to and from GCS, making it possible to quickly ingest large amounts of data.
- **Parallel processing:** GSutil can perform multiple operations in parallel, which can greatly increase the speed of data ingestion.
- **Resumable uploads:** GSutil supports resumable uploads, which means that if an upload is interrupted, it can be resumed from where it left off, which is particularly useful when uploading large files.
- **Support for different data sources:** GSutil can upload data from a variety of sources, such as local files, other cloud storage services, and even directly from websites.
- **Support for multiple file formats:** GSutil can handle a wide variety of file formats, including CSV, JSON, Avro, and Parquet, making it suitable for different data types.
- **Scriptable:** GSutil can be used in scripts and automation, which makes it easy to schedule or automate data ingestion tasks.

#### 3.2. Data Storage Tool Justification

Google Cloud Storage (GCS) is a scalable and highly available object storage service that can be used to store and retrieve large amounts of data. It can be considered as a better option than Cloud SQL and Bigtable for certain use cases because of its scalability, durability, and accessibility features.

GCS is designed to handle extremely large amounts of unstructured data, and it can automatically scale to store petabytes of data. It also provides durability and high availability by replicating data across multiple geographic locations, making it a good choice for storing data that needs to be accessible from all sites. Although our supply chain dataset is structured data, GCS is still preferred due to its built-in integration with other GCP services such as Cloud Dataflow and Dataproc, which facilitates easy data processing and analysis.



### 3.3. Data Processing Tool Justification

Google Cloud Platform's (GCP) Dataproc Spark is an efficient tool for processing large quantities of data in the cloud. It is effective due to its scalability, which enables the formation of clusters of any size and automatically adjusts the number of worker nodes as necessary. This feature leads to cost-effective data processing as it only charges for the resources used. Additionally, Dataproc Spark integrates seamlessly with other GCP services such as Bigtable, BigQuery, and Cloud Storage, making it easy to process and analyse data stored in these services. The tool is user-friendly with its web-based interface and command-line tools which simplify the process of creating and managing Spark clusters and submitting jobs. Moreover, Dataproc Spark is designed for high performance big data processing and allows for utilising the latest performance advancements in Spark.

### 3.4. Data Analytics Tool Justification

Google BigQuery is a data analytics tool in Google Cloud, and it has several advantages over other analytics tools that make it a better choice for certain use cases. Some of the reasons why BigQuery is better than other analytics tools in Google Cloud include:

- **Scalability:** BigQuery is designed to handle extremely large amounts of data, and it can automatically scale to store and analyse petabytes of data. This makes it well-suited for use cases that require the analysis of big data.
- **Performance:** BigQuery is built on top of a columnar storage and a parallel query processing engine, which allows it to execute complex queries and aggregations on large datasets at lightning-fast speeds.
- **Ease of use:** BigQuery provides a web-based user interface and a SQL-like query language that makes it easy to analyse data, even for users without a lot of technical expertise.
- **Cost-effective:** BigQuery offers a pay-as-you-go model, and it provides an option to store data in a columnar format which makes it cost-effective for storing and querying large datasets.

### 3.5. Data Visualisation Tool Justification

Looker Studio is a free data visualisation tool that enables the creation of interactive dashboards with customised reporting. The tool is user-friendly and facilitates easy report sharing and scheduling. In this project, we utilise Looker Studio to visualise sales across multiple dimensions, including product sales trending, customer purchase patterns, and more. The tool's ability to generate real-time, dynamic, and interactive dashboards, in addition to its capability to collect real-time data from sources such as YouTube, Google Ads, and Google Analytics, makes it an ideal

choice for our project. Furthermore, Looker Studio can assist in integrating data from a warehouse in BigQuery and other sources, enabling the visualisation of results from disparate sources

## 4. Implementation

### 4.1. Data Ingestion

GSutil is a Python-based application that allows access to Google Cloud Storage from the command line interface. Those who are comfortable working around CLI commands due to its high speed command execution capabilities, can use Gsutil to perform wide variety of bucket and object management operations such as:

- Bucket creation and deletion
- Object uploading, downloading and deletion
- Buckets and objects listing
- Objects moving, copying and renaming
- Editing of object and bucket ACLs (Access Control List)

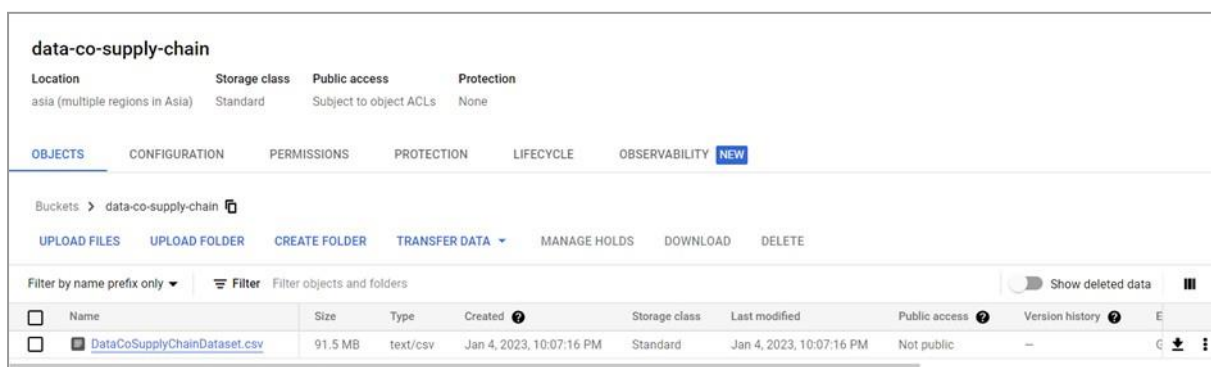
Firstly, a new bucket is created in cloud storage to store the raw dataset.

```
s2151665@cloudshell:~$ gsutil mb -b off -c STANDARD -l ASIA -p luminous-bond-371916 gs://data-co-supply-chain/
```

Figure 4.1.1 : Bucket creation under project

The raw dataset is then copied from the local machine into cloud storage. The project name is then renamed according to the assignment name.

```
s2151665@cloudshell:~ (luminous-bond-371916) $ gsutil cp Desktop/DataCoSupplyChainDataset.csv gs://data-co-supply-chain/
```



data-co-supply-chain									
Location	Storage class	Public access	Protection						
asia (multiple regions in Asia)	Standard	Subject to object ACLs	None						
<b>OBJECTS</b> CONFIGURATION PERMISSIONS PROTECTION LIFECYCLE OBSERVABILITY <b>NEW</b>									
Buckets > data-co-supply-chain									
UPLOAD FILES UPLOAD FOLDER CREATE FOLDER TRANSFER DATA MANAGE HOLDS DOWNLOAD DELETE									
Filter by name prefix only Filter objects and folders Show deleted data									
<input type="checkbox"/>	Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	E
<input type="checkbox"/>	DataCoSupplyChainDataset.csv	91.5 MB	text/csv	Jan 4, 2023, 10:07:16 PM	Standard	Jan 4, 2023, 10:07:16 PM	Not public	—	

Figure 4.1.2 : Copying of dataset from Local Machine

## 4.2. Data Storage

As per justification described in Section 3.2, GSC is used for data storage due to its scalability, durability, and accessibility features. On the other hand, Cloud SQL and Bigtable are both relational databases, which are designed to store structured data, and they are best suited for use cases that require complex, multi-row transactions and advanced querying capabilities. While they are highly performant, they may not be able to handle the scale and accessibility requirements for supply chain data as well as GCS.

Once the raw data is stored into the bucket, the project name is renamed and the storage is made available for all members of the team by making everyone owner of the project and bucket. This allows permissions for the members to perform operations on the raw dataset.

```
s2151665@cloudshell:~ (luminous-bond-371916)$ gcloud alpha projects update luminous-bond-371916 \
> --name="Data Co Supply Chain"
Updated [https://cloudresourcemanager.googleapis.com/v1/projects/luminous-bond-371916].
PROJECT_ID: luminous-bond-371916
NAME: Data Co Supply Chain
PROJECT_NUMBER: 454939068479

s2151665@cloudshell:~ (luminous-bond-371916)$ gsutil acl ch -u s2141806@siswa.um.edu.my:O gs://data-co-supply-chain
Updated ACL on gs://data-co-supply-chain/
s2151665@cloudshell:~ (luminous-bond-371916)$ gsutil acl ch -u s2110194@siswa.um.edu.my:O gs://data-co-supply-chain
Updated ACL on gs://data-co-supply-chain/
s2151665@cloudshell:~ (luminous-bond-371916)$ gsutil acl ch -u s2176584@siswa.um.edu.my:O gs://data-co-supply-chain
Updated ACL on gs://data-co-supply-chain/
```

VIEW BY PRINCIPALS

VIEW BY ROLES

Filter

Enter property name or value

?

≡

<input type="checkbox"/> Type	Principal <span>↑</span>	Name	Role	Security insights <span>?</span>	Inheritance	
<input type="checkbox"/>	s2110194@siswa.um.edu.my		Owner			
<input type="checkbox"/>	s2141806@siswa.um.edu.my		Owner			
<input type="checkbox"/>	s2151665@siswa.um.edu.my	KELVIN RAJJ	Owner			
<input type="checkbox"/>	s2176584@siswa.um.edu.my		Owner			

**Figure 4.2.1 : Allowing permissions to members to perform operations on the dataset**

## 4.3. Data Processing

Cloud Dataproc, Cloud Dataflow and Cloud Composer are all services offered by Google Cloud for data processing tasks. Each of these services has its own advantages and disadvantages, and the choice of which one to use will depend on the specific requirements of our use case and the trade-offs we are willing to make between ease of use, cost, and performance. Cloud Dataproc is a fully managed service that allows us to run Spark and Hadoop workloads. It provides us with more flexibility in terms of the types of data processing tasks we can perform and is generally less expensive than Dataflow and Composer, especially for simple jobs such as the processing made on the data in this project. Additionally, Dataproc gives us more fine-grained control over our resources allocated to your Spark and Hadoop clusters, allowing us to optimise performance and cost and It also can be easily

integrated with other Google Cloud services such as Bigtable, Cloud Storage, and Cloud SQL, which can simplify data pipeline management.

For this project, Dataproc spark cluster was used to process the supply chain data, the main processing logic is to ingest the raw dataset from Google Cloud Storage and rename the columns to make it suitable to be written to BigQuery, finally to write the formatted dataset to a BigQuery table as shown in code displayed in the Figure below.

In this project, we deployed a Dataproc Spark cluster (1 master node and 2 worker nodes 8 GB ram each) to process the supply chain data. The primary objective of the processing was to gather the raw dataset from Google Cloud Storage, reformat the column names to make it compatible for writing to BigQuery, and ultimately, write the formatted dataset to a BigQuery table as demonstrated in the code depicted in figure below:

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as F

spark = SparkSession.builder.appName('processing').getOrCreate()

SOURCE = 'gs://data-co-supply-chain/DataCoSupplyChainDataset.csv'

def main():
    df = spark.read.csv(SOURCE, header=True)

    df = df\
        .select(*[F.col(col).alias(col.replace(' ', '_').replace('(','_').replace(')','_')) for col in df.columns])

    (df
     .write
     .mode('overwrite')
     .option("temporaryGcsBucket", "temp-bucket-10")
     .format('bigquery')
     .option('table', 'luminous-bond-371916.supply_chain_data.cleaned_supply_chain_data')
     .save()
    )

if __name__ == "__main__":
    main()
```

**Figure 4.3.1 : Writing of Spark Job to be executed**

The code for this spark job was developed and executed on a Dataproc Spark virtual machine instance. Access to the master node was obtained through SSH, and the Spark job was written using the vim text editor in a file named "spark\_job.py". The job was then submitted using the Spark-submit CLI command, as demonstrated in the Figures below

Name	spark-cluster
Cluster UUID	4c9405a1-7bc2-4f5b-a3ab-6e1cd327490e
Type	Dataproc Cluster
Status	Running

MONITORING	JOB	VM INSTANCES	CONFIGURATION	WEB INTERFACES
Filter Filter instances				
	Name	Role		
	spark-cluster-m	Master		SSH

**Figure 4.3.2 : Spark job creation**

```
s2141806@spark-cluster-m:~$ ls
spark_job.py
s2141806@spark-cluster-m:~$ cat spark_job.py
from pyspark.sql import SparkSession
from pyspark.sql import functions as F

spark = SparkSession.builder.appName('processing').getOrCreate()

SOURCE = 'gs://data-co-supply-chain/DataCoSupplyChainDataset.csv'

def main():
    df = spark.read.csv(SOURCE, header=True)

    df = df\
        .select(*[F.col(col).alias(col.replace(' ', '_').replace('(','_').replace(')','_')) for col in df.columns])

    (df
     .write
     .mode('overwrite')
     .option("temporaryGcsBucket", "temp-bucket-10")
     .format('bigquery')
     .option('table', 'luminous-bond-371916.supply_chain_data.cleaned_supply_chain_data')
     .save()
    )

if __name__ == "__main__":
    main()

s2141806@spark-cluster-m:~$ spark-submit spark_job.py
```

**Figure 4.3.3 : Processing of dataset using Dataproc Spark (multi-node)**

## 4.4. Data Analytics

In Google Cloud, there are several tools that can be used for data analytics. Some of these tools include:

- **Cloud SQL:** This is a fully-managed relational database service that can be used to store and query structured data.
- **Cloud Spanner:** This is a fully-managed, globally distributed relational database service that can be used to store and query structured data.
- **Cloud Firestore:** This is a fully-managed NoSQL document database service that can be used to store and query semi-structured data.

Although there are other options available, BigQuery is still deemed the optimal choice for data analytics for our projects, as discussed in Section 3.4. The processed data that was obtained after executing the Spark job, was loaded into a BigQuery table for deeper analysis. In this project, BigQuery was mainly used to analyse sales in three (3) dimensions: sales per customer, sales per product, and sales per country.

### a) Query 1: Sales per customer

The following query analyses the sales per customer in a descending order and saves the result in the **best\_customers** table.

```
CREATE OR REPLACE TABLE `luminous-bond-371916.supply_chain_data.best_customers` AS
SELECT Customer_Id, Customer_Fname, Customer_Lname, SUM(CAST(Sales AS FLOAT64)) total_sales
FROM `luminous-bond-371916.supply_chain_data.cleaned_supply_chain_date`
GROUP BY Customer_Id, Customer_Fname, Customer_Lname
ORDER BY total_sales DESC
LIMIT 100;
```

**Figure 4.4.1 : Running queries to obtain data on sales per customer**

#### **b) Query 2: Sales per product**

The following query analyses the sales per product in a descending order and saves the result in the **best\_selling\_products** table.

```
CREATE OR REPLACE TABLE `luminous-bond-371916.supply_chain_data.best_selling_products` AS
SELECT Product_Name, SUM(CAST(Sales AS FLOAT64)) total_sales
FROM `luminous-bond-371916.supply_chain_data.cleaned_supply_chain_date`
GROUP BY Product_Name
ORDER BY total_sales DESC
LIMIT 100;
```

**Figure 4.4.2 : Running queries to obtain data on sales per product**

#### **c) Query 3: Sales per country**

The following query analyses the sales per country in a descending order and saves the result in the **sales\_per\_country** table.

```
CREATE OR REPLACE TABLE `luminous-bond-371916.supply_chain_data.sales_per_country` AS
SELECT Order_Country AS country, SUM(CAST(Sales AS FLOAT64)) total_sales
FROM `luminous-bond-371916.supply_chain_data.cleaned_supply_chain_date`
GROUP BY Order_Country
ORDER BY total_sales DESC
```

**Figure 4.4.3 : Running queries to obtain data on sales per country**

### **4.5. Data Visualisation**

This project utilised Looker for data visualisation. It is noted that there are also several alternative tools that can provide similar functions, such as Tableau and PowerBI. Despite the presence of these alternatives, Looker is still considered a viable option for this projects based on below explanation:

- **Integration:** Looker can be easily integrated with other Google Cloud services such as BigQuery, which can simplify data pipeline management and data analysis.
- **Flexibility:** Looker provides a flexible data modelling layer that allows us to create a semantic layer on top of your data, making it easier to understand and analyse your data.

- **Ease of use:** Looker provides an intuitive, easy-to-use interface that allows users to create and share interactive dashboards and reports without the need for extensive training.

The Figure below shows data visualisations on looker studio that are based on the 3 big query tables – **best\_customers**, **best\_selling\_products** & **sales\_per\_country**

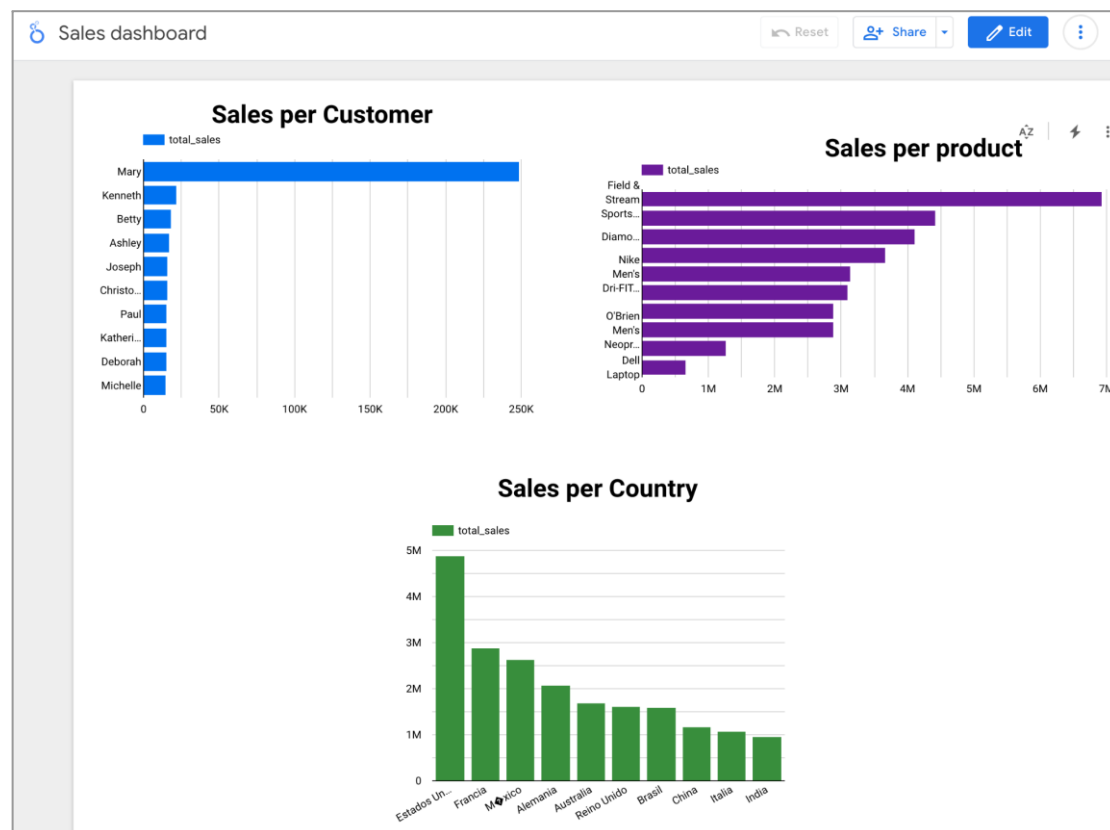


Figure 4.5.1 : Visualisation of queries obtained from BigQuery

## 5. Evaluation Metric

The following evaluation metrics are selected to analyse the performance of GCP with regards to its memory consumption, execution time, CPU utilisation and UI/UX in completing the data pipeline process:

- **Memory consumption** - analysing memory consumption in Spark is important because it can help to identify and prevent performance issues that can occur as a result of insufficient or poorly allocated memory. Memory is a critical resource for Spark, as it is used to store data structures, metadata, and intermediate results during the execution of a job.
- **Execution time** - specifies the time taken in seconds to complete the execution. For our case, the execution time is influenced by the size of the data.
- **CPU utilisation** - amount of CPU resources that are being used by the virtual machines (VMs) and other resources in GCP environment
- **UI/UX** - evaluation will be measured qualitatively based on the user experience of our team member while executing the task using the platform for instance the looks and feel of the UI, user friendliness of the features/tools, codeless/simplified codes available for implementation, etc.

### 5.1. Memory consumption

In Spark, the JVM memory is divided into two regions: on-heap and off-heap.

On-heap memory is the memory that is allocated on the Java heap, which is the memory space that is managed by the Java Virtual Machine (JVM). This memory is used to store objects that are created by the JVM, such as Spark data structures and metadata.

Off-heap memory is the memory that is allocated outside of the Java heap. This memory is used to store data that is not managed by the JVM, such as memory-mapped files and off-heap storage for RDDs.

Peak JVM Memory OnHeap / OffHeap refers to the maximum amount of on-heap and off-heap memory that Spark is using at a given time during the execution of a job.

As shown in the figure below, only the Peak JVM memory OnHeap (217.7 MiB, 157.2 MiB) and OffHeap (86.5 MiB, 88.6 MiB) for the spark cluster executor nodes were within the limit of the allocated storage memory (1.3 GiB) thus the cluster is safe that the peak JVM memory usage will never exceeds the amount of memory that is allocated to the JVM as this can cause performance issues such as garbage collection pauses and even lead to out-of-memory errors.



## Executors

Show 20 entries

Executor ID	Address	Status	RDD Blocks	Storage Memory	Peak JVM Memory OnHeap / OffHeap
driver	spark-multi-node-m.c.luminous-bond-371916.internal:37837	Active	0	0.0 B / 912.3 MiB	0.0 B / 0.0 B
1	spark-multi-node-w-1.c.luminous-bond-371916.internal:42351	Active	0	0.0 B / 1.3 GiB	217.7 MiB / 86.5 MiB
2	spark-multi-node-w-0.c.luminous-bond-371916.internal:38303	Active	0	0.0 B / 1.3 GiB	157.2 MiB / 88.6 MiB

Figure 5.1.1 : Memory consumption for Spark job

## 5.2. Execution Time

The tasks revolving around data ingestion and data storage (bucket creation and project permissions configurations) were effortless and executed within seconds.

```
> gsutil cp DataCoSupplyChainDataset.csv gs://data-co-supply-chain/DataCoSupplyChainDataset.csv
Copying file://DataCoSupplyChainDataset.csv [Content-Type=text/csv]...
- [1 files][ 91.5 MiB/ 91.5 MiB] 2.4 MiB/s
Operation completed over 1 objects/91.5 MiB.
```

Figure 5.2.1 : Execution time for dataset ingestion into storage

While the time taken to run the data processing spark job on the multi node spark cluster was 1 minute and 9.6 seconds as shown in the Figure 5.2.1 below

```
23/01/14 11:11:34 INFO com.google.cloud.spark.bigquery.BigQueryWriteHelper: Done loading to luminous-bond-371916.supply_chain_data.cleaned_supply_chain_date. jobId: JobId{project=luminous-bond-371916, job=69f3091d-6e7a-449d-839b-2c38a99f8e49, location=US}
23/01/14 11:11:35 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark@269deee7{HTTP/1.1, (http/1.1)}{0.0.0.0:0}

real    1m9.653s
user    0m32.791s
sys     0m2.139s
```

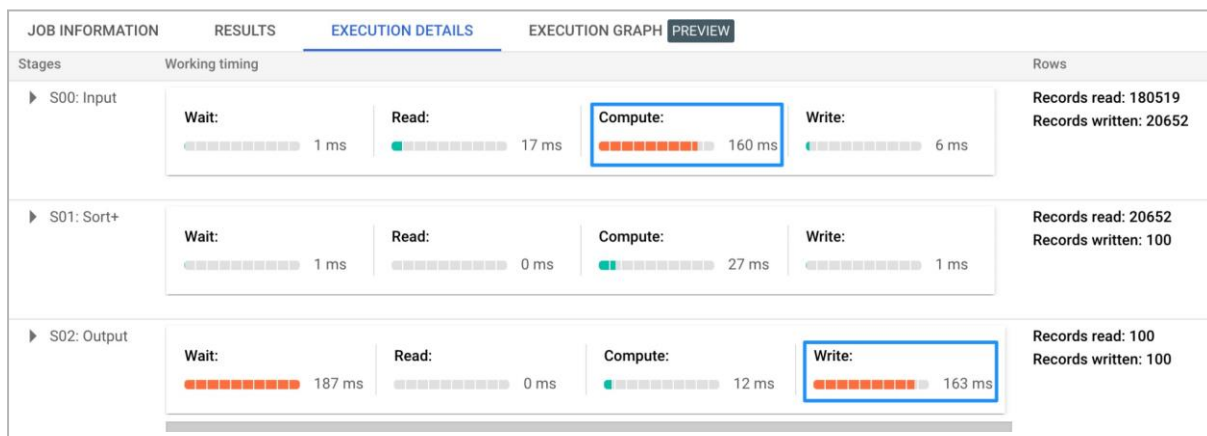
Figure 5.2.2 : Execution time for spark job on multi node cluster

Data analytics was done on BigQuery using 3 main queries as shown in section 4.1, for the first query shown in the Figure below

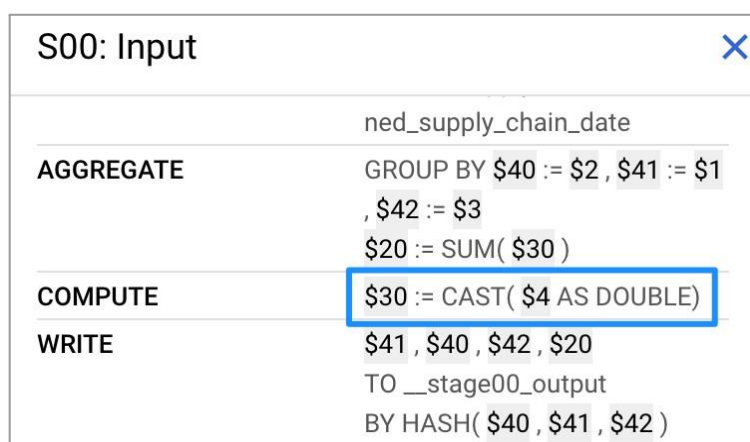
```
CREATE OR REPLACE TABLE `luminous-bond-371916.supply_chain_data.best_customers` AS
SELECT Customer_Id, Customer_Fname, Customer_Lname, SUM(CAST(Sales AS FLOAT64)) total_sales
FROM `luminous-bond-371916.supply_chain_data.cleaned_supply_chain_date`
GROUP BY Customer_Id, Customer_Fname, Customer_Lname
ORDER BY total_sales DESC
LIMIT 100;
```

Figure 5.2.3 : Sales per customer query

It took 160 ms in the input phase of the query, this was mainly due to casting the Sales column as a Float as shown in the figures below



**Figure 5.2.4 : Sales per customer query execution details**



**Figure 5.2.5 : Sales per customer query compute details**

While writing the data to the new table took 163 ms which is not that significant due to the small size of the project's dataset.

In the second and third query, the execution time was similar where the execution consisted of 3 phases, input, sort & output where the compute section in the input phase was 64 & 53 ms respectively due to the casting of the Sales column to Float as well.

```

CREATE OR REPLACE TABLE `luminous-bond-371916.supply_chain_data.best_selling_products` AS
SELECT Product_Name, SUM(CAST(Sales AS FLOAT64)) total_sales
FROM `luminous-bond-371916.supply_chain_data.cleaned_supply_chain_date`
GROUP BY Product_Name
ORDER BY total_sales DESC
LIMIT 100;
  
```

**Figure 5.2.6 : Sales per product query**

JOB INFORMATION	RESULTS	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Stages	Working timing	Rows		
► S00: Input	Wait: 1 ms Read: 7 ms Compute: 64 ms Write: 6 ms	Records read: 180519 Records written: 164		
► S01: Aggregate	Wait: 1 ms Read: 0 ms Compute: 7 ms Write: 5 ms	Records read: 164 Records written: 164		
► S02: Output	Wait: 182 ms Read: 0 ms Compute: 13 ms Write: 161 ms	Records read: 164 Records written: 164		

**Figure 5.2.7 : Sales per product query execution details**

```
CREATE OR REPLACE TABLE `luminous-bond-371916.supply_chain_data.sales_per_country` AS
SELECT Order_Country AS country, SUM(CAST(Sales AS FLOAT64)) total_sales
FROM `luminous-bond-371916.supply_chain_data.cleaned_supply_chain_date`
GROUP BY Order_Country
ORDER BY total_sales DESC
```

**Figure 5.2.8 : Sales per country query**

JOB INFORMATION	RESULTS	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Stages	Working timing	Rows		
► S00: Input	Wait: 1 ms Read: 4 ms Compute: 53 ms Write: 4 ms	Records read: 180519 Records written: 118		
► S01: Sort+	Wait: 1 ms Read: 0 ms Compute: 6 ms Write: 1 ms	Records read: 118 Records written: 100		
► S02: Output	Wait: 226 ms Read: 0 ms Compute: 18 ms Write: 151 ms	Records read: 100 Records written: 100		

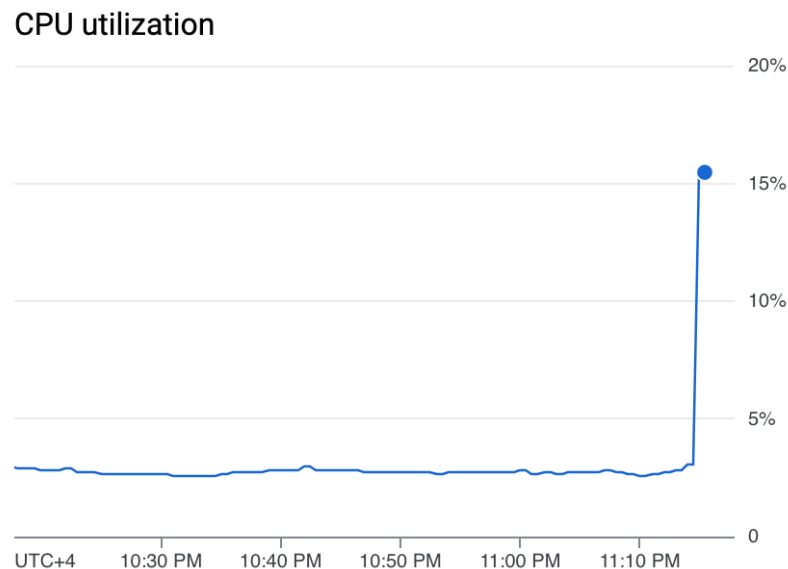
**Figure 5.2.9 : Sales per country query execution details**

The query execution time is not significantly high for all our queries which shows the great performance of BigQuery.

### 5.3. CPU Utilisation

The CPU utilisation figure shown below shows that 15.4% of the CPU resources were used in the master node to run the job. CPU utilisation on the master node of a Spark cluster can increase when running a job because the master node is responsible for coordinating the distribution of tasks to the worker nodes and

managing their execution. This coordination can require significant CPU resources. To mitigate this, we can increase the number of worker nodes in the cluster. However, 15% is not that significantly high thus it can be concluded that the spark resources are sufficient for the data processing job



**Figure 5.3.1 : CPU Utilisation graph**

#### **5.4. User Interface and User Experience (UI/UX)**

UI/UX (user interface/user experience) is important in data processing and analytics tools like Dataproc, BigQuery, and Looker because it can greatly impact the ease and efficiency of working with the data.

- Basic CLI commands were sufficient to create the storage bucket, import the raw dataset and allow permissions to specific users for operations to be performed on the dataset.
- Dataproc provided a web-based user interface that allowed us to easily create and manage clusters, as well as view job status and logs. It also allowed us to easily integrate with other GCP services such as Bigtable, BigQuery, and Cloud Storage, which made data processing more efficient.
- BigQuery provides a web-based user interface that allows us to easily run SQL queries and view query results, as well as view and manage datasets and tables. The UI is intuitive, user-friendly and allows us to share the results among each other.
- Looker provided an intuitive and user-friendly web-based interface that allowed us to easily create and share data visualisations and dashboards. It also allowed for the creation of custom data models that can be used to join and transform data from multiple sources, making it easy for users to explore and analyse data.

## 6. Conclusion

Google Cloud Platform (GCP) is a suitable choice for ingesting, storing, processing, analysing, and visualising supply chain data as it offers a wide array of services and tools that can be utilised to support each phase of data management. In this project, GSUtils command line was used to ingest the raw dataset and store it in Google Cloud Storage, the Dataproc Spark cluster was used to process the raw dataset and load it into a BigQuery table, and Looker Studio was employed to visualise the data in BigQuery. Performance metrics such as execution time, memory allocation, CPU utilisation, and UI/UX were evaluated. The results showed that the execution time of the Spark job in the data processing phase and analysis in the data analytics phase were feasible, memory allocation was within the Spark cluster range, CPU utilisation peaked at only 15.45% in the Spark cluster, and the UI/UX of all the tools were user-friendly and integrative.

Based on the execution of the framework and performance results, it can be concluded that this framework can be used to manage the various stages of a supply chain, from sourcing raw materials to delivering products to the customer, and to aid in decision-making to improve and strategize the supply chain system

## References

- Awwad,, M., Kulkarni, P., Bapna, R., & Marathe, A. (2018). Big Data Analytics in Supply Chain: A Literature Review. *IEOM Society International*.  
[https://www.researchgate.net/publication/327979282\\_Big\\_Data\\_Analytics\\_in\\_Supply\\_Chain\\_A\\_Literature\\_Review](https://www.researchgate.net/publication/327979282_Big_Data_Analytics_in_Supply_Chain_A_Literature_Review)
- Best Uses for Bigtable - Introduction to Google Cloud Bigtable Course*. (n.d.). Cloud Academy. Retrieved January 10, 2023, from  
<https://cloudacademy.com/course/introduction-to-google-cloud-bigtable/best-uses-for-bigtable-1/>
- DataCo SMART SUPPLY CHAIN FOR BIG DATA ANALYSIS*. (n.d.). Kaggle. Retrieved January 10, 2023, from  
[https://www.kaggle.com/datasets/shashwatwork/dataco-smart-supply-chain-for-big-d  
ata-analysis](https://www.kaggle.com/datasets/shashwatwork/dataco-smart-supply-chain-for-big-data-analysis)
- Google Cloud Platform*. (n.d.). Wikipedia. Retrieved January 10, 2023, from  
[https://en.wikipedia.org/wiki/Google\\_Cloud\\_Platform](https://en.wikipedia.org/wiki/Google_Cloud_Platform)
- Lutkevich, B. (2021, June 2). *supply chain*. WhatIs.com.  
<https://www.techtarget.com/whatis/definition/supply-chain>