WQD7007 BIG DATA MANAGEMENT
2/2022/2023

BIG DATA TECHNOLOGY IMPLEMENTATION IN SUPPLY CHAIN DATA

INDIVIDUAL PROJECT

Matric Number	S2151665
Name	Kelviin Rajj A/L Karupaya

TABLE OF CONTENTS

QUESTION 1

According to Biswas and Sen (2016), supply chain data is considered as big data due to its volume, velocity, variety, value and requires advanced analytic approaches to extract meaningful insights from the data.

a. Supply chain data in relation to 7Vs of Big Data

VOLUME: Supply chain **data can exist in large volumes** as a result of numerous transactions, processes, and entities taking place. Information such inventory status, sales, procurement, logistics, and much more can be found in supply chain data.

VELOCITY: In most cases, supply chain **data is being generated and updated in real-time.** This includes streamed data from sources such as sensors, RFID tags, point-of-sale systems, and more. The velocity feature is crucial to be monitored to optimize supply chain operations as it allows quick decision-making to respond to changing demands or challenges.

VARIETY: Supply chain data come in **multiple types and formats**. Structured data, such as sales records and inventory statuses, as well as unstructured data, like emails, customer feedback, social media posts are among the examples. Combining and analysing these variety of data types can provide fruitful insights for enhancing supply chain operations.

VERACITY: There are many cases where supply chain data may suffer **data quality issues, such as inaccuracies, inconsistencies, and missing values**. This could be due to manual data entry errors, outdated or incomplete data sources, or data integration issues. Ensuring data veracity is crucial to make informed reliable decisions and avoid incorrect information transfer throughout the supply chain.

VARIABILITY: Supply chain data can display variability due to **periodic variations, demand spikes, or unexpected disruptions** such as catastrophe or supplier issues. Analysis and management of supply chain data helps organizations expect and respond to changes effectively.

VISUALIZATION: Visualization is substantial in understanding supply chain data. Supply chain professionals often use visualizations like **dashboards, charts, and graphs** to observe data patterns and trends, and communicate insights.

VALUE: The key objective in supply chain data is extracting value. Organizations can **gain insights** into demand patterns, optimize inventory management, identify bottlenecks and enhance overall supply chain performance by analysing the data.

b. Supply chain data can be useful to manufacturing sector as below:

Forecast demands: Analysis of historical sales data, market trends, and customer behaviour from supply chain data can allow manufacturers precisely forecast demand, enabling them to optimize production schedules and minimize inventory costs, stockouts or overstocks.

Inventory management: Supply chain data can give real-time view into inventory status. As a result, manufacturers can optimize stock levels, reduce excess inventory and ensure availability of raw materials and components.

Cost optimization: Supply chain data analysis can assist manufacturers identify cost drivers, optimize transportation routes, streamline logistics processes, negotiate better deals with providers and create opportunities for cost savings.

c. Organizations that can benefit greatly from supply chain data are global retail corporations with complex network of suppliers, distribution centres, and stores such as **Walmart**. Analysis of supply chain data such as sales, inventory and logistics information can make room for optimization of inventory levels and logistics operations and improve demand forecasting accuracy and minimize costs which results in overall efficiency in the supply chain system.

QUESTION 2

In the field of supply chain management, data is generated mostly in structured/semi-structured format to alleviate analytical processes. Alam et al. (2023) presented that big data technologies are implemented to analyse large datasets and allow insights to be gained seamlessly to allow data-driven decision making. Below is the comparison of big data tools commonly used for data access stage:

Tools	Relational Database (MYSQL)	HBase	MongoDB	Hive
Advantages	- Well-defined structure, enforcing predefined schema - Efficient in performing complex queries that involved multiple tables and relationships	- Built for scalability, making it appropriate for large datasets - Allows dynamic schema design - Optimized for high speed read-writes, suitable for real time data ingestion	- Flexible and dynamic schema design - Highly scalable - Efficient querying capabilities	- Designed to process and analyse large datasets stored in Hadoop - Allows ad-hoc querying, data exploration and analysis - Can integrate with other Hadoop tools and framework
Disadvantages	- Limited scalability, can be challenging for large datasets - In certain high-concurrency situations, relational databases may experience limitations compared to NoSQL databases	- Limited querying abilities (supports key based lookups and scans), but unable to perform complex ad-hoc queries - Steep learning curve to use the tool efficiently	- Increased management complexity due to its flexible schema design - Increased storage overhead compared to relational database due to its flexible schema	- High latency due to scalability, resulting in longer query execution times

Based on the comparison of the above tools, **Hive** is the most efficient among all tools, especially in the use case of supply chain related data analysis. In Hive, OLAP is primarily used which is useful to gain insights from the data being analysed. It has the capabilities to **perform ad-hoc querying and data exploration tasks upon large datasets without increased complexity and overhead** such as in MongoDB. Hive also uses HiveQL to **perform SQL-like queries** (easier language) to analyse data compared to HBase that has limited querying functionalities and requires high learning curve to perform analysis (using dedicated Java API that require coding or 3rd party tools). Hive **stores data in a distributed storage system such as Hadoop (HDFS) that is well-suited to handle big data workloads**, such as in supply chain data where large number of datasets are being generated on a regular basis. MySQL stores data in a traditional relational database that is more suitable for small-medium sized datasets.

QUESTION 3

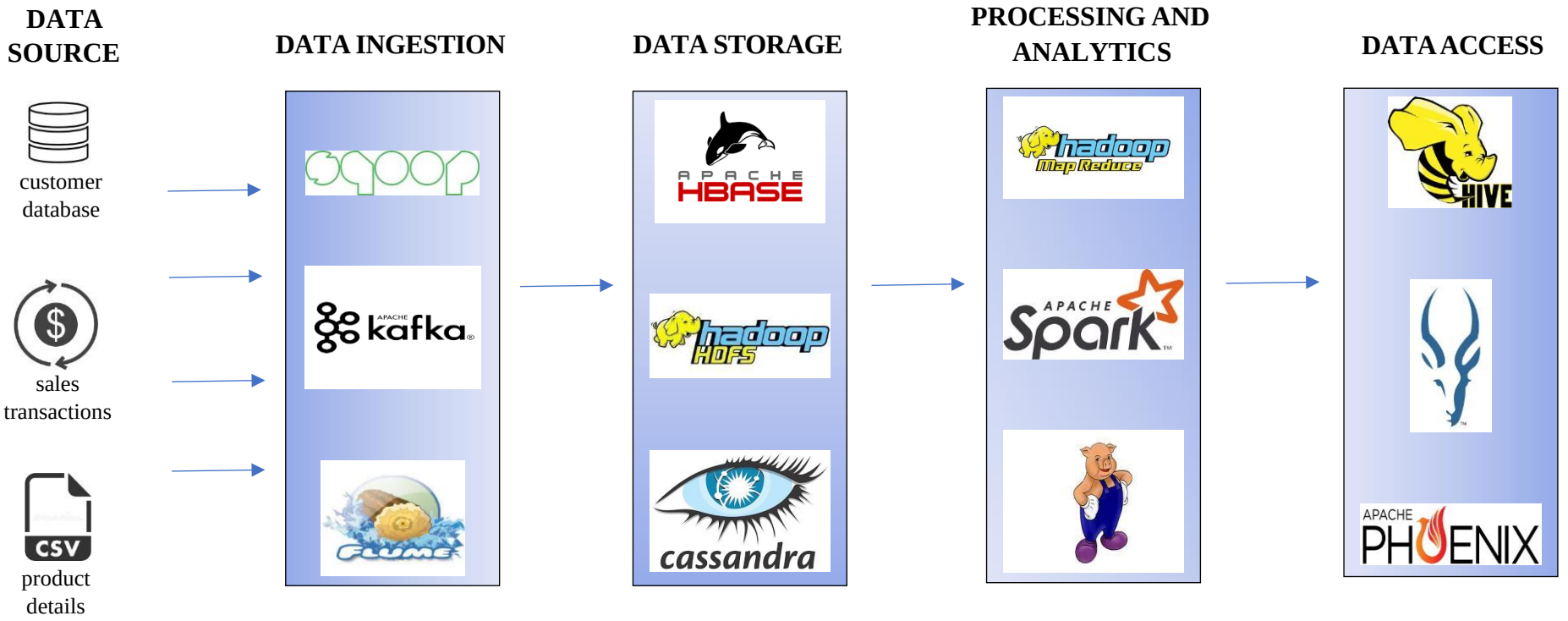
Demonstration of supply chain data storage and access using HDFS and Hive.

Data source: https://www.kaggle.com/datasets/harshsingh2209/supply-chain-analysis.

Figures (appendix)	Discussion and insights
Figure 1	Data is imported into HDFS which is capable in handling big data storage workloads
Figure 2	A new table 'scm' is created in Hive and linked to the dataset stored in Hadoop to be imported. The table can now be analysed by performing SQL-like queries.
Figure 3	The type of product is counted to analyse occurrences in the dataset. **Skincare has more occurrences in the dataset** followed by haircare and cosmetics.
Figure 4	The **highest average selling price of products are cosmetics** followed by skincare and haircare.
Figure 5	**Skincare and haircare products had higher manufacturing costs** compared to cosmetic products.
Figure 6	**As expected, cosmetic products generated the highest average revenue due to its high selling price and low manufacturing costs.**
Figure 7	The most preferred choice for shipment was **Carrier B as it has performed more shipments** compared to other 2 carriers in delivering the products.
Figure 8	When analysed shipment costs, Carrier B had the lowest average shipment fees compared to others. **This explains why Carrier B was the preferred logistics provider.**
Figure 9	In terms of gender, **highest count fall under 'Unknown'** which implies there are **inconsistencies** present in this dataset and further transformation tasks can be implemented to counter these data errors.
Figure 10	**Shipment by road was the most preferred** transportations mode for deliveries compared to rail, air and sea.
Figure 11	When compared among suppliers, **Supplier 1 was the preferred manufacturer to provide more goods** compared to the other suppliers.
Figure 12	It can be assumed that **Supplier 1 is the most preferred manufacturer** because **despite having the low production volumes compared to other suppliers, it had the fastest manufacturing lead time**.

QUESTION 4

As per the architecture proposed by Biswas and Sen (2016) for supply chain management system, the below big data pipeline can be implemented. Big data pipeline for supply chain data typically consists of several stages and components that help ingest, store, process and access to analyse and gain insights from the data.



QUESTION 5

Glavic (2021) presented importance of data provenance in handling data inaccuracy as below:

ERROR DETECTION AND ISOLATION

In a big data pipeline, data provenance aids in the **detection and isolation of unexpected data errors**. It is simpler to pinpoint the location and time of the error by capturing the lineage of the data, which includes its source, adjustments, and intermediary steps. This information **accelerates the mistake identification and isolation** process by enabling data experts to concentrate on examining specific elements or processes that caused the incorrect data.

DATA VALIDATION AND QUALITY ASSURANCE

The **veracity and quality of data** may be verified using data provenance at several points throughout the pipeline. **Discrepancies or unanticipated inaccuracies can be found** by evaluating the actual data recorded in the provenance with the expected data attributes or metrics. Data provenance allows for the implementation of data quality checks and validations, allowing for automatic or human inspections to make sure the data fulfils the required standards.

PERFORMANCE OPTIMIZATION AND TUNING

Data provenance offers insights into the data pipeline's performance features. Data engineers may **discover possible bottlenecks, ineffective procedures, or data quality problems** that may be a factor in data errors by evaluating the lineage. Based in the gathered information, performance optimisation and tuning can be done to increase the pipeline's effectiveness and dependability while lowering the likelihood of unforeseen inaccuracies.

CONTINUOUS MONITORING AND ALERTING

Data integrity and accuracy in the big data pipeline may be continuously monitored through data provenance. Organisations can **build up monitoring methods to identify unexpected data errors** in real-time by regularly gathering and analysing the lineage and metadata. **Alerts or notifications** can be sent out in response to allow quick analysis and remedial action.

AUDITING AND COMPLIANCE

For auditing and compliance purposes, data provenance is a useful asset. In order to prove the quality and compliance of data workflows, it gives organisations a full record of the **data history, transformations, and processes**. When unexpected mistakes occur, the provenance data may be utilised to **carry out audits and investigations** to find the root causes and implement necessary corrective measures.

REFERENCES.

Biswas, S. & Sen, J. (2016). A Proposed Architecture for Big Data Driven Supply Chain Analytics, pp 1-24.

Alam, M. N., Singh, V., Kaur, R. & Kabir, S. (2023). Big Data: An Overview with Legal Aspects and Future Prospects. *Journal of Emerging Technologies and Innovative Research (JETIR), Vol. 10: No. 5,* pp 476-485.

Glavic, B. (2021), "Data Provenance", *Foundations and Trends in Databases: Vol. 9: No. 3-4,* pp 209-441.

APPENDIX.

```
cloudera@Ubuntu:~/Desktop/hadoop/sbin$ hdfs dfs -put /home/cloudera/Desktop/sc_
data/supply_chain_data.csv /scm
cloudera@Ubuntu:~/Desktop/hadoop/sbin$ hdfs dfs -cat /scm/supply_chain_data.csv
 | head
type,sku,price,availability,num_of_prod_sold,revenue,customer_type,stock_level,
lead_time,order_quant,ship_time,ship_carrier,ship_costs,supplier_name,location,
lead _time,prod_volumes,manufact_lead_time,manufact_costs,inspect_result,defect
 _rate,transport_mode,routes,costs
haircare,SKU0,69.80800554211577,55,802,8661.996792392383,Non-binary,58,7,96,4,C
arrier B,2.956572139430807,Supplier 3,Mumbai,29,215,29,46.27987924050832,Pendin
g,0.22641036084992516,Road,Route B,187.75207545920392
skincare,SKU1,14.843523275084339,95,736,7460.900065445849,Female,53,30,37,2,Car
rier A,9.71657477143131,Supplier 3,Mumbai,23,517,30,33.61676895373,Pending,4.85
4068026388706,Road,Route B,503.0655791496692
haircare,SKU2,11.319683293090566,34,8,9577.74962586873,Unknown,1,10,88,2,Carrie
r B,8.054479261732155,Supplier 1,Mumbai,12,971,27,30.688019348284204,Pending,4.
580592619199229,Air,Route C,141.92028177151906
skincare,SKU3,61.163343016437736,68,83,7766.836425685233,Non-binary,23,13,59,6,
Carrier C,1.7295685635434288,Supplier 5,Kolkata,24,937,18,35.62474139712503,Fai
l,4.74664862064775,Rail,Route A,254.77615921928663
skincare,SKU4,4.805496036345893,26,871,2686.505151567447,Non-binary,5,3,56,8,Ca
rrier A,3.8905479158706715,Supplier 1,Delhi,5,414,3.92,06516059871285,Fail,3.14
```

Figure 1: Import of dataset from local machine into HDFS

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS scm(type string,sku string,price floa
t,availability int,num_of_prod_sold int,revenue float,customer_type string,stoc
k_level int,stock_lead_time int,order_quant int,ship_time int,ship_carrier stri
ng,ship_costs float,supplier_name string,location string,lead_time int,prod_vol
umes int,manufact_lead_time int,manufact_costs float,inspect_result string,defe
ct_rate float,transport_mode string,routes string,costs float)
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE
    > location '/scm';
OK
Time taken: 0.058 seconds
hive> select * from scm limit 1;
OK
haircare        SKU0    69.80801        55      802     8661.997        Non-bin
ary     58      7       96      4       Carrier B       2.956572        Supplie
r 3     Mumbai  29      215     29      46.27988        Pending 0.22641036    R
oad     Route B 187.75208
Time taken: 0.093 seconds, Fetched: 1 row(s)
```

Figure 2: New table creation in Hive

```
hive> select type, count(*) as instance_count from scm group by type;
Query ID = cloudera_20230621002936_72fcd499-0bfc-4c55-a448-e271961d8d1a
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-06-21 00:29:38,135 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local307271392_0005
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 207020 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
cosmetics       26
haircare        34
skincare        40
```

Figure 3: Count distinct type variables

```
hive> select type,avg(price) from scm group by type;
Query ID = cloudera_20230622204610_e78b12f1-538e-4e27-ad13-ae47247fe3e1
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-06-22 20:46:11,523 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1266613539_0003
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 124212 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
cosmetics       57.36105780418102
haircare        46.014279148157904
skincare        47.25932885408402
```

Figure 4: Calculate average price of product in respect to product types

```
hive> select type,sum(manufact_costs) from scm group by type;
Query ID = cloudera_20230622203932_edce9ebc-3956-4c1f-9e79-9279c6c797a6
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-06-22 20:39:35,335 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1737123633_0001
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 41404 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
cosmetics       1119.3712593317032
haircare        1647.5717697143555
skincare        1959.726304769516
```

Figure 5: Calculate sum of manufacturing costs in respect to product types

```
hive> select type,avg(revenue) from scm group by type;
Query ID = cloudera_20230621003924_6976a012-ccef-4d3c-9f26-4cbbaf0eb60c
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-06-21 00:39:26,216 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local2002978705_0011
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 455444 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
cosmetics       6212.356384277344
haircare        5131.040947409238
skincare        6040.704055786133
```

Figure 6: Calculate average total revenue in respect to product types

```
hive> select ship_carrier, count(*) as instance_count from scm group by ship_ca
rrier;
Query ID = cloudera_20230621003049_57a0106f-a88b-4675-bf00-64ffb1129d5b
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-06-21 00:30:51,000 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local622666752_0006
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 248424 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Carrier A       28
Carrier B       43
Carrier C       29
```

Figure 7: Calculate the frequency of specific shipping carrier in dataset

```
hive> select ship_carrier,avg(ship_costs) from scm group by ship_carrier;
Query ID = cloudera_20230621003408_b61c954f-89fa-4ec2-bc1c-925c2d31437e
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-06-21 00:34:09,946 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1081789876_0009
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 372636 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Carrier A       5.554922504084451
Carrier B       5.5092470008273455
Carrier C       5.5992916288047
```

Figure 8: Calculate total average shipping costs of shipping carriers

```
hive> select customer_type, count(*) as instance_count from scm group by custom
er_type;
Query ID = cloudera_20230621003141_c7c89ae3-4d11-468c-a7f5-6503b6725351
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-06-21 00:31:42,566 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local373961705_0007
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 289828 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Female  25
Male    21
Non-binary      23
Unknown 31
```

Figure 9: Determine frequency of different genders in the dataset

```
hive> select transport_mode, count(*) as instance_count from scm group by trans
port_mode;
Query ID = cloudera_20230621210211_fafc9c98-fc1f-4e05-82ec-79520655fddf
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-06-21 21:02:12,640 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local106704731_0015
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 621060 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Air     26
Rail    28
Road    29
Sea     17
```

Figure 10: Determine frequency of different modes of transports

```
hive> select supplier_name, count(*) as instance_count from scm group by suppli
er_name;
Query ID = cloudera_20230621003758_7f7feffa-9829-48d5-a27e-7cea40942d66
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-06-21 00:37:59,524 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local251449225_0010
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 414040 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Supplier 1      27
Supplier 2      22
Supplier 3      15
Supplier 4      18
Supplier 5      18
```

Figure 11: Determine frequency of different suppliers in the dataset

```
hive> select supplier_name,avg(prod_volumes),avg(manufact_lead_time) from scm g
roup by supplier_name;
Query ID = cloudera_20230621210415_47bf5979-b2be-40a4-b046-c3d3645c2cf7
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-06-21 21:04:17,435 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1522700954_0016
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 662464 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Supplier 1      501.6666666666667       12.592592592592593
Supplier 2      641.1363636363636       15.590909090909092
Supplier 3      533.1333333333333       14.933333333333334
Supplier 4      653.1111111111111       15.333333333333334
Supplier 5      521.1666666666666       16.333333333333332
```

Figure 12: Calculate total average production volumes and manufacturing lead time for suppliers