**Optimizing Sales and Inventory Management for a Beverage Distributor: Insights and Efficiency Through BigQuery, Hive, and Spark**

**ADTA 5240**

**Harvesting, Storing, and Retrieving Data**

**Presented by**

Ashok Kumar Mettu, 11640819

Shanmukha Sai Ram Tummuri, 11703315

Kushal Kumar Reddy Kolli, 11639360

Laxmi Narsimha Reddy Racha, 11695049

Harshith Mamidishetty, 11634943

**Introduction:**

This documents provides step-by-step process and technical plow for Sales and inventory Management projects. This includes the screenshots and explanation for each of the step. Steps start from uploading and storing the dataset to visualization to ensure reproducibility and clarity.

The main aim of this project is to improve the decision making in inventory allocation, sale forecasting, and supplier management.

**Data Lifecycle:**

1. **Data Collection:** This dataset consists of historical sales data from a beverage distributor downloaded from data.gov website.
2. **Data Storage**: To store this dataset we used a cloud data warehouse called BigQuery provided by Google Cloud Platform.

GCP Setup and upload:

1. Login GCP
2. Create a storage bucket to store the dataset
3. Upload the dataset to that bucket.
4. Set up a BigQuery Table and upload the dataset.

A screenshot of a computer

Description automatically generated

The dataset is successfully uploaded to the GCP bucket. This is a centralized storage for subsequent processing. The screenshots shows out dataset “Updated\_Sales” uploaded successfully into the bucket with correct schema.

1. **Data Processing**: The data is processed by handling null values, renaming columns, and cleaning the data.

Now we will do some data exploration. To understand the structure of the dataset. Try to identify if any missing data or outliers in the dataset as this might affect the analysis.

Check if everything is imported correctly.

A screenshot of a computer

Description automatically generated

Looks like there are some null values present in particular problem which will be handled in later steps.

A screenshot of a computer

Description automatically generated

1. **Data Analysis**: performed analysis using SQL in BigQuery to generate total sales by supplier, retail sales trends, and sales ratios.

Lets first run some simple queries in BigQuery.

**Query 1.** Check table content

SELECT \* FROM `Sales\_Beverages.Updated\_Sales` LIMIT 10;

A screenshot of a computer

Description automatically generated

This query verified that the dataset was imported correctly into BigQuery.

**Query 2.** Print distinct SUPPLIER from the table

A screenshot of a computer

Description automatically generated

This query displays all the different suppliers who supply beverages to this company.

Now, Lets check for null values:

SELECT

  COUNTIF(YEAR IS NULL) AS null\_years,

  COUNTIF(MONTH IS NULL) AS null\_months,

  COUNTIF(SUPPLIER IS NULL) AS null\_suppliers,

  COUNTIF(ITEM\_CODE IS NULL) AS null\_item\_codes,

  COUNTIF(ITEM\_DESCRIPTION IS NULL) AS null\_item\_descriptions,

  COUNTIF(ITEM\_TYPE IS NULL) AS null\_item\_types,

  COUNTIF(RETAIL\_SALES IS NULL) AS null\_retail\_sales,

  COUNTIF(RETAIL\_TRANSFERS IS NULL) AS null\_retail\_transfers,

  COUNTIF(WAREHOUSE\_SALES IS NULL) AS null\_warehouse\_sales

FROM `Sales\_Beverages.Updated\_Sales`

A screenshot of a computer

Description automatically generated

This query identified that there are some columns with missing values which we will address in subsequent steps.

**Remove NULL values.**

To remove NULL values we substituted null values to the corresponding values as below

1. All NULL values in SUPPLIER columns are replaced by “Unknown”

Update SUPPLIER column where it's NULL

UPDATE `Sales\_Beverages.Updated\_Sales`

SET SUPPLIER = 'Unknown'

WHERE SUPPLIER IS NULL;

1. All NULL values in ITEM\_TYPE columns are replaced by “Unknown”

Update ITEM\_TYPE column where it's NULL

UPDATE `Sales\_Beverages.Updated\_Sales`

SET ITEM\_TYPE = 'Unknown'

WHERE ITEM\_TYPE IS NULL;

1. All NULL values in RETAIL\_SALES columns are replaced by “0”

Update RETAIL\_SALES column where it's NULL

UPDATE `Sales\_Beverages.Updated\_Sales`

SET RETAIL\_SALES = 0

WHERE RETAIL\_SALES IS NULL;

`

A screenshot of a computer

Description automatically generated

Verifying the above activity.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Successfully all null values are removed.

1. **EDA:**

Lets now understand the distribution and structure of data.

Find mean, median, standard deviation and count for numeric fields **RETAIL\_SALES**, **RETAIL\_TRANSFERS**, **WAREHOUSE\_**SALES to find the central tendency and variability of the data.

SELECT

  AVG(RETAIL\_SALES) AS avg\_sales,

  APPROX\_QUANTILES(RETAIL\_SALES, 2)[OFFSET(1)] AS median\_retail\_sales,  -- 50th percentile is the median

  STDDEV(RETAIL\_SALES) AS stddev\_sales,

  COUNT(\*) AS total\_records

FROM `Sales\_Beverages.Updated\_Sales`

A screenshot of a computer

Description automatically generated

SELECT

  AVG(RETAIL\_SALES) AS avg\_sales,

  STDDEV(RETAIL\_SALES) AS stddev\_sales,

  COUNT(\*) AS total\_records

FROM `Sales\_Beverages.Updated\_Sales`;

A screenshot of a computer

Description automatically generated

**Get total sales by grouping Supplier and Item Type:**

SELECT

  SUPPLIER,

  ITEM\_TYPE,

  SUM(RETAIL\_SALES) AS total\_retail\_sales,

  SUM(RETAIL\_TRANSFERS) AS total\_retail\_transfers,

  SUM(WAREHOUSE\_SALES) AS total\_warehouse\_sales

FROM `Sales\_Beverages.Updated\_Sales`

GROUP BY SUPPLIER, ITEM\_TYPE

ORDER BY total\_retail\_sales DESC

A screenshot of a computer

Description automatically generated

**Creating performance metrics:**

1. Sales Ratio:
2. Understand how the sales relate to what is in the warehouse, calculate ratio between RETAIL\_SALES and WAREHOUSE\_SALES
3. calculate the ratio of RETAIL SALES to RETAIL TRANSFERS to analyze if there's a high sales-to-transfer rate.

SELECT

SUPPLIER,

ITEM\_TYPE,

SUM(RETAIL\_SALES) / NULLIF(SUM(WAREHOUSE\_SALES), 0) AS sales\_to\_warehouse\_ratio,

SUM(RETAIL\_SALES) / NULLIF(SUM(RETAIL\_TRANSFERS), 0) AS sales\_to\_transfer\_ratio

FROM `Sales\_Beverages.Updated\_Sales`

GROUP BY SUPPLIER, ITEM\_TYPE

A screenshot of a computer

Description automatically generated

**Find Monthly trends**

Analyze how the sales, transfers, and warehouse sales change over time (by **MONTH** and **YEAR**).

SELECT

  YEAR,

  MONTH,

  SUM(RETAIL\_SALES) AS monthly\_retail\_sales,

  SUM(RETAIL\_TRANSFERS) AS monthly\_retail\_transfers,

  SUM(WAREHOUSE\_SALES) AS monthly\_warehouse\_sales

FROM `Sales\_Beverages.Updated\_Sales`

GROUP BY YEAR, MONTH

ORDER BY YEAR, MONTH

A screenshot of a computer

Description automatically generated

1. **Data Presentation**:

Created visualizations in Tableau, including total retail sales by supplier, sales trends over time, and sales ratios.

Now export the data to TABLEAU.

1. Visualize total retail sales by supplier.

A screenshot of a computer

Description automatically generated

We can see the supplier and total retail sales of each of them.

A screenshot of a graph

Description automatically generated

Extract data for Sales Trends Over Time

SELECT

  YEAR,

  MONTH,

  SUM(RETAIL\_SALES) AS total\_retail\_sales

FROM

  `Sales\_Beverages.Updated\_Sales`

GROUP BY

  YEAR, MONTH

ORDER BY

  YEAR ASC, MONTH ASC;

A screenshot of a computer

Description automatically generated

A screenshot of a graph

Description automatically generated

Similarly for sales ratio

SELECT

  ITEM\_TYPE,

  RETAIL\_SALES,

  WAREHOUSE\_SALES,

  IF(WAREHOUSE\_SALES = 0, NULL, RETAIL\_SALES / WAREHOUSE\_SALES) AS sales\_ratio

FROM

   `Sales\_Beverages.Updated\_Sales`

WHERE

  WAREHOUSE\_SALES IS NOT NULL AND RETAIL\_SALES IS NOT NULL

ORDER BY

  ITEM\_TYPE;

A screenshot of a computer

Description automatically generated

A screenshot of a graph

Description automatically generated

STEP 2: Hadoop Clusters

Hive and Spark

Step 1 create tables in Hive and Spark simultaneously.

1. Creating table in Hive

Step 1: Search for clusters in GCP and start the cluster.A screenshot of a computer

Description automatically generated  
  
Now navigate to compute engine>>VM Instances

A screenshot of a computer

Description automatically generated

Open SSH of the manager node.

Step 1 will be Moving data to HDFS: to do it we do below steps.

A screenshot of a computer

Description automatically generated

Check existing directories available.

hdfs dfs -ls /

hdfs dfs -ls /user

A screenshot of a computer

Description automatically generated

Now we will create a personal directory.

hdfs dfs -mkdir /user/lnarasimha

hdfs dfs -mkdir /user/lnarasimha/data

hdfs dfs -mkdir /user/lnarasimha/data/sales

A screenshot of a computer

Description automatically generated

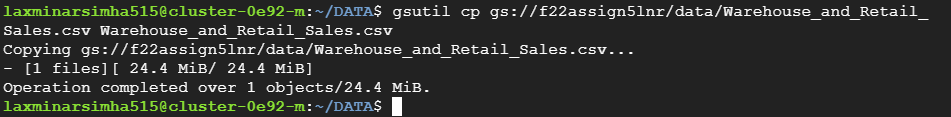
First move the files from GCP cloud storage to manager node.

mkdir DATA

cd DATA

Copy the dataset from the GCP bucket to the manager node.

gsutil cp gs://**f22assign5lnr**/data/ [Warehouse\_and\_Retail\_Sales.csv](https://console.cloud.google.com/storage/browser/_details/f22assign5lnr/data/Warehouse_and_Retail_Sales.csv?pageState=(%22StorageObjectListTable%22:(%22f%22:%22%255B%255D%22))&inv=1&invt=AbiGKA&project=adta5240a5lnr) [Warehouse\_and\_Retail\_Sales.csv](https://console.cloud.google.com/storage/browser/_details/f22assign5lnr/data/Warehouse_and_Retail_Sales.csv?pageState=(%22StorageObjectListTable%22:(%22f%22:%22%255B%255D%22))&inv=1&invt=AbiGKA&project=adta5240a5lnr)



Load Data into HDFS:

hdfs dfs -put [Warehouse\_and\_Retail\_Sales.csv](https://console.cloud.google.com/storage/browser/_details/f22assign5lnr/data/Warehouse_and_Retail_Sales.csv?pageState=(%22StorageObjectListTable%22:(%22f%22:%22%255B%255D%22))&inv=1&invt=AbiGKA&project=adta5240a5lnr) /user/lnarasimha/data/sales

hdfs dfs -ls /user/lnarasimha/data/sales

A black screen with white text

Description automatically generated

SETUP:HIVE

Access hive using beeline -u jdbc:hive2://localhost:10000

A screen shot of a computer

Description automatically generated

CREATE EXTERNAL TABLE IF NOT EXISTS sales\_data (

year INT,

month INT,

supplier STRING,

item\_code INT,

item\_desc STRING,

item\_type STRING,

retail\_sales FLOAT,

retail\_transfers FLOAT,

warehouse\_sales FLOAT

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE

LOCATION '/user/lnarasimha/data/sales/';

A screenshot of a computer

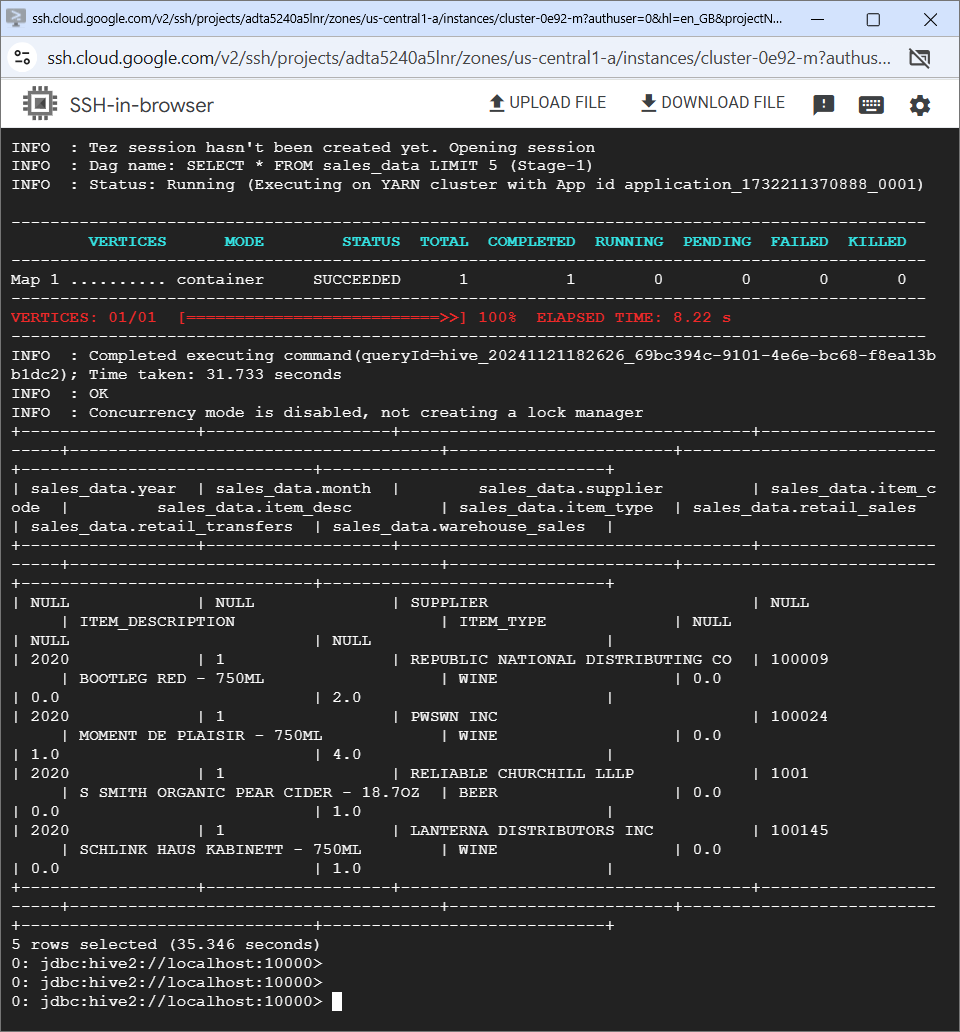
Description automatically generated A screenshot of a computer

Description automatically generated

Run a sample query.

SELECT \* FROM sales\_data LIMIT 5;

A screenshot of a computer program

Description automatically generated 

Now Open New SSH from manager node.

Open spark-sql

>>spark-sql

A screenshot of a computer program

Description automatically generated

Create table in spark.

CREATE TABLE spark\_sales\_data(

year INT,

month INT,

supplier STRING,

item\_code INT,

item\_desc STRING,

item\_type STRING,

retail\_sales FLOAT,

retail\_transfers FLOAT,

warehouse\_sales FLOAT

)

USING CSV

OPTIONS (

path '/user/lnarasimha/data/sales/Warehouse\_and\_Retail\_Sales.csv',

header 'true',

inferSchema 'true'

);

A screenshot of a computer

Description automatically generated

Now Lets run Queries to compare the response times in hive and spark.

**Query 1: Total Warehouse Sales by Supplier**

**Hive:**

SELECT supplier, SUM(warehouse\_sales) AS total\_warehouse\_sales

FROM sales\_data

GROUP BY supplier

ORDER BY total\_warehouse\_sales DESC

LIMIT 5;

A screenshot of a computer program

Description automatically generated A black screen with white text

Description automatically generated

**Spark:**

SELECT supplier, SUM(warehouse\_sales) AS total\_warehouse\_sales

FROM spark\_sales\_data

GROUP BY supplier

ORDER BY total\_warehouse\_sales DESC;

Output:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Query | Hive Execution Time | Spark Execution Time | Performance Improvement |
| Query 1 | 15.57 (S) | 11.561(S) | 25.75% |

**Query 2: Total Sales Metrics by Year**

**Hive:**

SELECT year,

SUM(retail\_sales) AS total\_retail\_sales,

SUM(retail\_transfers) AS total\_retail\_transfers,

SUM(warehouse\_sales) AS total\_warehouse\_sales

FROM sales\_data

GROUP BY year;

A screenshot of a computer

Description automatically generated A screenshot of a computer program

Description automatically generated

**Spark:**

SELECT year,

SUM(retail\_sales) AS total\_retail\_sales,

SUM(retail\_transfers) AS total\_retail\_transfers,

SUM(warehouse\_sales) AS total\_warehouse\_sales

FROM spark\_sales\_data

GROUP BY year;

A black screen with white text

Description automatically generated

The table below compares Hive and Spark SQL execution times, showing Spark’s superior performance.

|  |  |  |  |
| --- | --- | --- | --- |
| Query | Hive Execution Time | Spark Execution Time | Performance Improvement |
| Query 2 | 9.117 (S) | 2.567(S) | 71.84% |

**Query 3: Top 5 Items with Highest Retail Sales**

**Hive:**

SELECT item\_desc,

SUM(retail\_sales) AS total\_sales

FROM sales\_data

GROUP BY item\_desc

ORDER BY total\_sales DESC

LIMIT 5;

A screenshot of a computer program

Description automatically generated

**A screenshot of a computer

Description automatically generated**

**SPARK**

SELECT item\_desc,

SUM(retail\_sales) AS total\_sales

FROM spark\_sales\_data

GROUP BY item\_desc

ORDER BY total\_sales DESC

LIMIT 5;

A screenshot of a computer

Description automatically generated

The table below compares Hive and Spark SQL execution times, showing Spark’s superior performance.

|  |  |  |  |
| --- | --- | --- | --- |
| Query | Hive Execution Time | Spark Execution Time | Performance Improvement |
| Query 3 | 12.934 (S) | 2.507(S) | 80.62% |

**Query 4: Top 5 Suppliers by Warehouse Sales**

**Hive:**

SELECT supplier,

SUM(warehouse\_sales) AS total\_warehouse\_sales

FROM sales\_data

GROUP BY supplier

ORDER BY total\_warehouse\_sales DESC LIMIT 5;

A screenshot of a computer program

Description automatically generated A screenshot of a computer

Description automatically generated

**Spark:**

SELECT supplier,

SUM(warehouse\_sales) AS total\_warehouse\_sales

FROM spark\_sales\_data

GROUP BY supplier

ORDER BY total\_warehouse\_sales DESC

LIMIT 5;

A screen shot of a computer

Description automatically generated

The table below compares Hive and Spark SQL execution times, showing Spark’s superior performance.

|  |  |  |  |
| --- | --- | --- | --- |
| Query | Hive Execution Time | Spark Execution Time | Performance Improvement |
| Query 4 | 10.38 (S) | 1.323 (S) | 87.25% |

**Sample queries for analysis:**

**Query 5: Comparison of Liquor, Beer, and Wine Sales:**

**Hive:**

SELECT item\_type,

SUM(retail\_sales) AS total\_retail\_sales,

SUM(warehouse\_sales) AS total\_warehouse\_sales

FROM sales\_data

WHERE item\_type IN ('LIQUOR', 'BEER', 'WINE')

GROUP BY item\_type;

A screenshot of a computer program

Description automatically generated A screen shot of a computer

Description automatically generated

**Query 5: Retail and Warehouse Sales by Item Type**

**Spark:**

SELECT item\_type,

SUM(retail\_sales) AS total\_retail\_sales,

SUM(warehouse\_sales) AS total\_warehouse\_sales

FROM spark\_sales\_data

WHERE item\_type IN ('LIQUOR', 'BEER', 'WINE')

GROUP BY item\_type;

A screenshot of a computer program

Description automatically generated