

# ABC CONSUMER ELECTRONICS CASE STUDY

## ABSTRACT

This project helps to develop an inventory management data warehouse that will help the business analyst to make data-driven decisions. There was a provision for the dimension model, identifying the granularity, dimensions, attributes, and fact tables. There was consideration for the business requirements of ABC Consumer Electronics Outlet Ltd and the commencement of the in-depth day-to-day business processes analysis. The graphical representation of the various tables implementation to determine the relationships between the dimensions and the fact tables. There was a discussion of the strengths, weaknesses, and methods. After everything, there were various recommendations to increase the data warehouse's usefulness for the organization.

## INTRODUCTION

In the present age, there cannot be an overemphasis on the importance of Data Warehousing. It has become an essential tool for data analysis, science, and engineering. It is no secret that data warehousing can be applied to businesses to make better and good data-driven decisions. It is no surprise that there is a claim that data is the new oil. Organizations need to have reliable reporting and analysis of large sets of data. Businesses need the data generated or derived from their daily activities to be consolidated and integrated for different stages of clustering, from customer service to top-level executive business decisions [1]. These decisions drive organizations, as failure to meet the needs of their customers may result in poor customer service. As a result, ABC Consumer Electronics Outlet Ltd felt there was a need for the services of a business analyst to build a data warehouse for its inventory management system. Hiring the services of a business analyst was needed so that the organization could make better decisions and use data to make more business-driven changes.

Because of stock-level management, many businesses understand the importance of good warehouse management in an e-commerce company, especially the supply chain operations. There will be in-depth, step-by-step instructions on creating a data warehouse design that is worthy of approval by the top echelon of an organization.

## BUSINESS REQUIREMENT SPECIFICATIONS

The requirements of ABC Consumer Electronics Outlet Limited were due to the issues of understocking and overstocking, which has led to the mismanagement of orders and supplies, thereby impacting customer-entrepreneur trust and then negatively impacting the decisions made by the top brass of the organization.

There was a request by the organization that the data be easily accessible to foster better customer service and better data-driven decisions. Sample Inventory Data was provided from the Vend database to populate the Data warehouse. Listed below are the data files in txt and CSV file formats.

- Received Purchase Orders
- Sent Purchase Orders
- Suppliers
- Daily Product (3 days sample)

## DATA FILES

There was due consideration for all the files given as test data. There will be a brief explanation of them.

1. Received Purchase Orders  
This file contains sample stock purchase orders from suppliers on the 30th of Oct 2015. The attributes include
  - PurchaseOrderCode,
  - ProductSKU,
  - SupplierName,
  - DestinationOutletID,
  - SentDate,
  - ReceivedDate,
  - ReceivedQty,
  - OrderedQty.
2. Sent Purchase Orders

This file contains purchase orders to suppliers from the 31st of Oct 2016 to the 7th of Nov 2016. The attributes include

- PurchaseOrderCode,
- ProductSKU,
- SupplierName,
- DestinationOutlet,
- SentDate,
- OrderedQty.

3. Suppliers

This file contains information about the suppliers of the ABC Consumer Electronics company. The attributes include

- SupplierName,
- Description,
- Phone
- Email
- Fax
- FirstLineAddress
- Postcode
- City
- State
- CountryID

Only the supplierName field has data, and the other columns contain null values. This condition means only the Supplier Name field is compulsory on the Vend application. While the information on the other attributes may or may not be necessary. We will capture these values as "Unknown" during the design of our dimension table.

4. Products

This file has a snapshot of the product database daily. It keeps records of the different attributes of the products and a status column to show if it is active or not. The attributes include

- SKU
- ProductName
- Description
- Condition
- ProductType
- Brand
- SupplierName
- Tags
- CostPrice
- RetailPrice
- CurrentStockLevel
- DateCreatedAt
- DateDiscontinuedAt
- IsActive

## **DIMENSIONAL DATA MODELING**

The purpose of dimensional modeling is to optimize the database for faster retrieval of data [2]. Data Warehousing systems help to acquire and integrate information from different systems and to process that data efficiently.

Ralph Kimball developed the concept of Dimensional Modeling. It includes a set of methods and techniques for data warehouse design. It focuses on identifying and implementing the essential business process before moving on to the other additional processes [3].

Ross and Kimball laid out some objectives of dimensional data modeling, and they are:

- Protect information assets

- Consistent information presentation
- Easily accessible information
- Adaptable and receptive to change
- Timely presentation of information

Dimensional data modeling has two main requirements;

- Faster retrieval of data
- Present data that is straightforward to the business user

Dimensional models have a central fact table that links to various dimension tables in the shape of a star or snowflakes.

### GRAIN

Grain is the least on the hierarchy of detail on a table. It is the lowest level of detail on a table. The business process identified is inventory management. There are three known grain levels for ABC Consumer Electronics Outlet Ltd.

Business Activity	Grain
Stock Level	Daily stock levels per product
Sent Purchase Orders	Sent purchase orders per product per day
Received Purchase Orders	Received purchase orders per product per day

**Table 1.** The Grain levels for ABC Consumer Electronics Outlet Ltd

### DIMENSIONS

The dimensional model represents descriptive data about the numerical values in the fact table. A dimension table contains attributes that describe the objects in a fact table, and It has a primary key column that uniquely identifies a row. There are four main dimensions to be used for this project.

#### Date Dimension

This table contains all the date attributes that will enable the organization to generate reports and analyze them over a period for easy aggregation. The script for the table generation is in Appendix A.

Column Name	Data Type	Description	Nullable	Constraint	Extra
Date Key	Int	Full date in yyyymmdd	Not Null	Primary Key	Surrogate Key
FullDate	Date	Full date in mm/dd/yyyy format	Not Null		
DaysOfWeek	Int	Day number of the week	Not Null		
DaysName	Varchar(10)	Name of a day in the week	Not Null		
DaysOfYear	Int	Day number of the year	Not Null		
WeekNumberYYYY	Int	Week number of the year	Not Null		
MonthsName	Char(15)	Name of the month	Not Null		
MonthsNumber	Int	Number of the month	Not Null		
Quarters	Int	A three-month period in the year	Not Null		
Years	int	Calendar year	Not Null		

**Table 2.** The fields in the Date Dimension table

### Product Dimension

This table contains all the products in ABC Consumer Electronics Outlet Ltd. The ProductKey is the surrogate key that links to the fact tables. In this table, the SKU is unique. This ProductKey will help to create reports or analyses by products and their attributes. The data in this table was obtained from the Sample CSV data Product sheet provided in the requirement specification.

Column Name	Data Type	Description	Nullable	Constraint	Extra
ProductKey	Int	Auto Incremental Key	Not Null	Primary Key	Auto Increment
SKU	varchar(10)	Product code	Not Null	Unique	
ProductName	varchar(60)	Name of product	Not Null		
Descriptions	Varchar(10)	Product description	Not Null		
Condition	varchar(10)	State of the product	Not Null		
ProductType	varchar(10)	Type of product	Not Null		
Brand	varchar(10)	Product brand	Not Null		
Tags	varchar(30)	Product Classification	Not Null		
CostPrice	float	The cost price of the product	Not Null		
RetailPrice	float	Retail price of the product	Not Null		
DateCreatedAt	date	Date Created (mm/dd/yyyy)	Not Null		
DateDiscontinuedAt	date	Date Discontinued (mm/dd/yyyy)	Not Null		
IsActive	date	Product Status Flag	Not Null		

**Table 3.** The fields in the Product Dimension table

### Supplier Dimension

This table contains all the information on the suppliers of ABC Consumer Electronics Outlet Ltd. The surrogate key, SupplierKey, joins the table to the fact tables. This SupplierKey will help to create reports or analyses using products and their attributes. The sample data was obtained from the Sample CSV Supplier sheet provided for the project. Only the SupplierName field was populated in the CSV file; the other values were "NULL," but there will be a replacement with "Unknown" as it is best practice not to have Null values in a dimension table.

Column Name	Data Type	Description	Nullable	Constraints	Extra
SupplierKey	Int	Auto Incremental Key	Not Null	Primary Key	Auto Increment
SupplierName	Varchar(60)	Trading name of the supplier	Not Null		
Phone	Varchar(50)	Phone number of supplier	Not Null		
Email	Varchar(50)	Email address of the supplier	Not Null		
Fax	Varchar(50)	Fax number of supplier	Not Null		
FirstLineAddress	Varchar(60)	Address of supplier	Not Null		
PostCode	Varchar(10)	Postcode of supplier	Not Null		

City	Varchar(50)	City location in which the supplier is	Not Null		
State	Varchar(50)	State location in which the supplier is	Not Null		
CountryID	Varchar(50)	Country location in which the supplier is	Not Null		

**Table 4.** The fields in the Supplier Dimension table

#### Location Dimension

This table contains the location of the company's warehouses of ABC Consumer Electronics Outlet Ltd. The sample data's derivation from the Sample Received Purchase Order CSV provided for the project. The Location Dimension table is a degenerate dimension. Only the DestinationOutlet was populated, and the other fields are Null, but there will be a replacement with "Unknown" as it is best practice not to have Null values in a dimension table.

Column Name	Data Type	Description	Nullable	Constraint	Extra
LocationKey	Int	Auto Incremental Key	Not Null	Primary Key	Auto Increment
DestinationOutlet	Varchar(30)	Location of Company's Warehouse	Not Null		
Region	Varchar(30)	The region in which the company is	Not Null		
City	Varchar(30)	City location in which the company is	Not Null		
PostalCode	Varchar(10)	PostalCode of Company	Not Null		
CountryID	Varchar(50)	CountryID of Company	Not Null		

**Table 5.** The fields in the Location Dimension table

#### FACT TABLES

A fact table is an entity in a star or snowflakes schema that stores measures that measure the business, such as sales, cost of goods, or profits. Fact tables deal with the numeric data of a business [4]. It contains foreign keys from the date, product, and supplier dimensions to create reports and analyses by the attributes of these dimensions. Depending on the data's size, an aggregate table should be created for future reports to reduce the script.

#### Sent Purchase Order Fact

This fact table contains all received order stocks received from suppliers. It only contains information on sent purchase orders. It contains the foreign keys from the date, product, location, and supplier dimensions to create reports and analyses by the attributes of these dimensions.

Column Name	Data Type	Description	Nullable	Constraint	Extra
DateKey	Int	Date ID	Not Null	Foreign Key	Surrogate Key
ProductKey	Int	Product ID	Not Null	Foreign Key	Surrogate Key
SupplierKey	Int	Supplier ID	Not Null	Foreign Key	Surrogate Key
LocationKey	Int	Location ID	Not Null	Foreign Key	Surrogate Key
QuantityOrdered	bigint	Units purchased	Not Null		Measure

**Table 6.** The fields in the sent purchase order fact table.

### Received Purchase Order Fact

This fact table contains all purchase order stocks received from suppliers. Even though It is similar to the Sent Purchase Order Fact table as it contains the number of quantities purchased, it only contains information on fulfilled purchase orders. It contains the foreign keys from the date, product, location, and supplier dimensions to create reports and analyses by the attributes of these dimensions.

Column Name	Data Type	Description	Nullable	Constraint	Extra
DateKey	Int	Date ID	Not Null	Foreign Key	Surrogate Key
ProductKey	Int	Product ID	Not Null	Foreign Key	Surrogate Key
SupplierKey	Int	Supplier ID	Not Null	Foreign Key	Surrogate Key
LocationKey	Int	Location ID	Not Null	Foreign Key	Surrogate Key
QuantityOrdered	bigint	Units purchased	Not Null		Measure
QuantityReceived	bigint	Units received	Not Null		Measure

**Table 7.** The fields in the received purchase order fact table

### Fact Stock Level

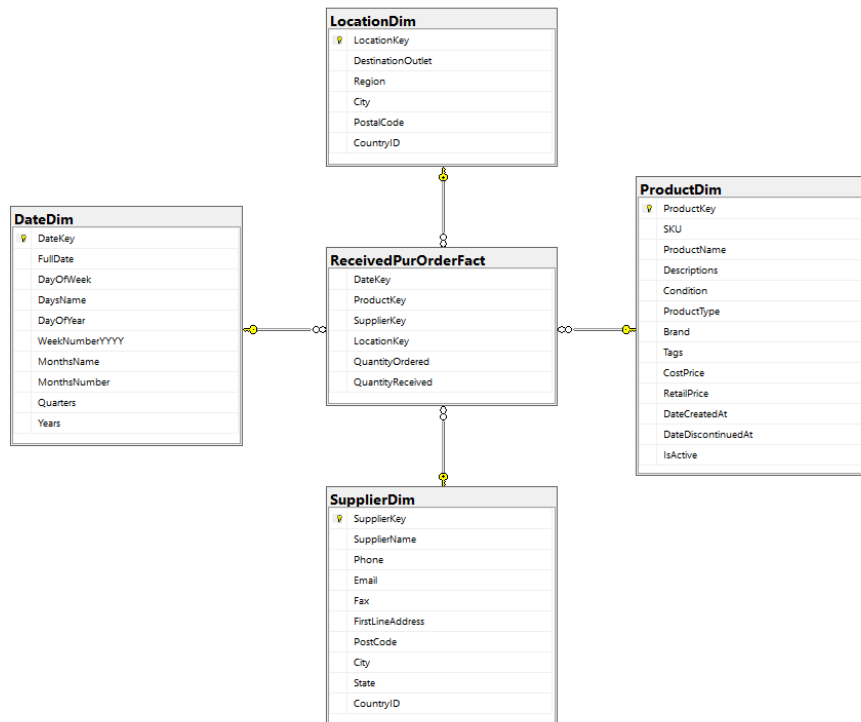
The stock level fact table shows the daily level of stocks in the warehouse. It contains the foreign keys from the date, product, and supplier dimension tables to create a detailed analysis of these dimension tables.

Column Name	Data Type	Description	Nullable	Constraint	Extra
DateKey	Int	Date ID	Not Null	Foreign Key	Surrogate Key
ProductKey	Int	Product ID	Not Null	Foreign Key	Surrogate Key
SupplierKey	Int	Supplier ID	Not Null	Foreign Key	Surrogate Key
CurrentStockLevel	bigint	The stock level at the current time	Not Null		Measure

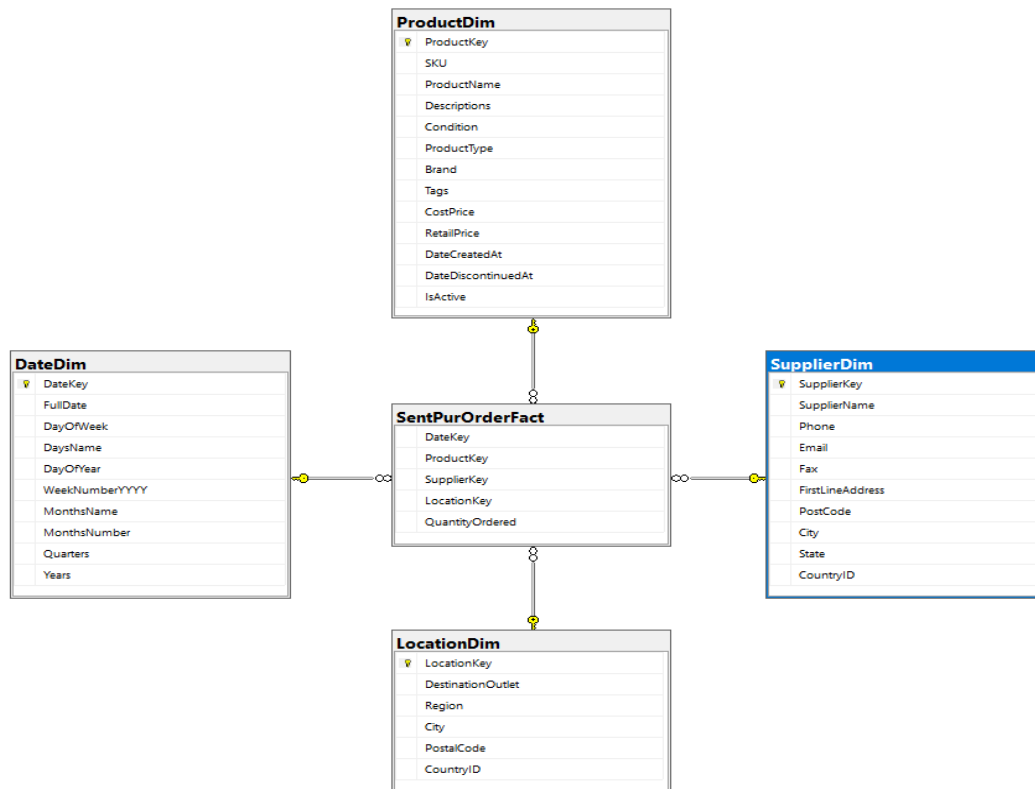
**Table 8.** The fields in the stock level fact table

### SCHEMA DIAGRAM

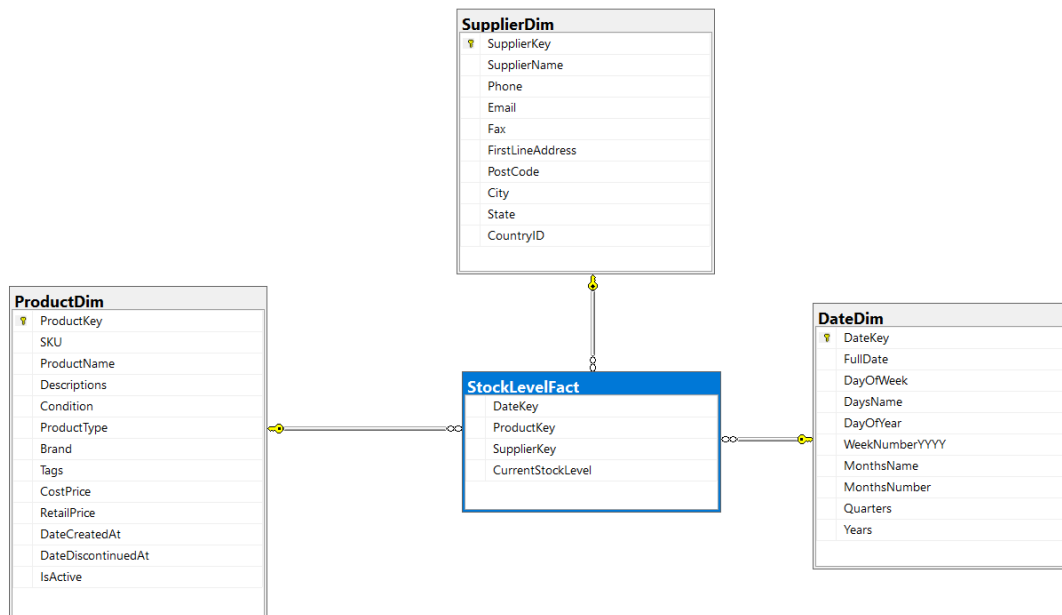
The schema diagrams use a simple star schema design. The Dimension tables have relationships with the fact tables joined by surrogate keys. Below are the schema diagrams



**Figure 1:** The received purchase order schema diagram



**Figure 2:** The sent purchase order schema diagram



**Figure 3:** The stock level schema diagram

### RECOMMENDATIONS

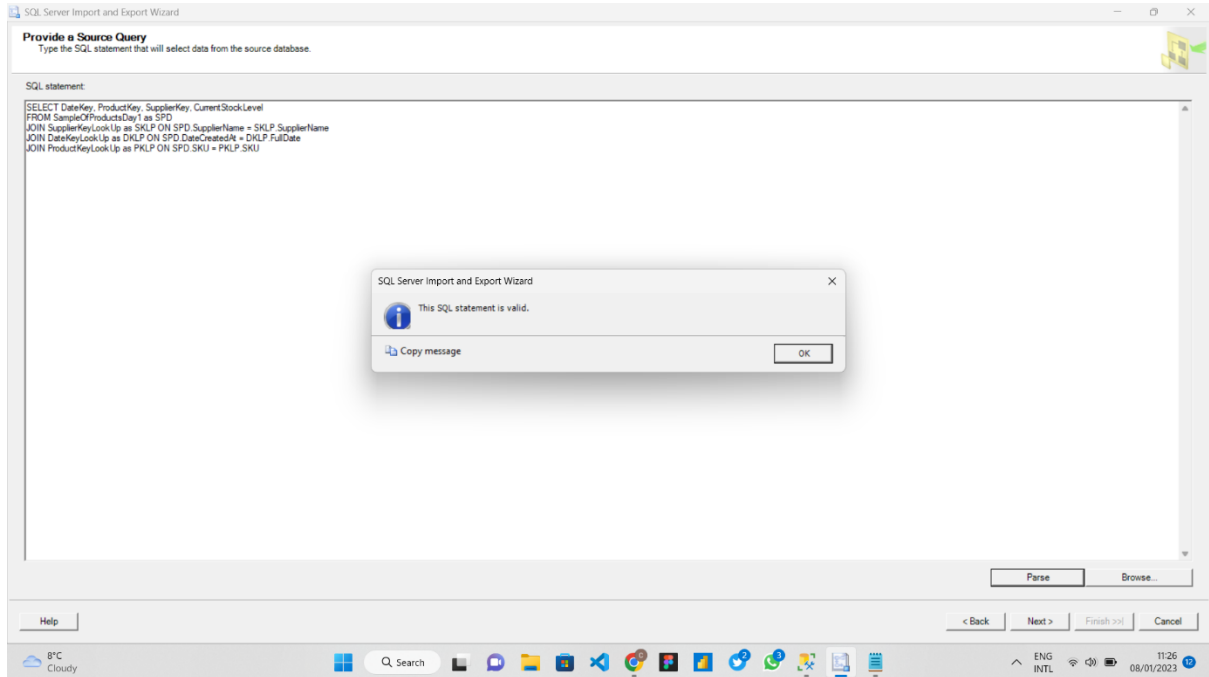
There can be an addition of another dimension to ascertain the condition of electrical items before and during delivery. This addition prevents delivering damaged items or getting claims for refunds for products in perfect condition during delivery. This dimension can store returned items that violate their warranty.

Further integration into other data sources will broaden the data warehouse and enable the business analyst to generate insights on sales, finances, revenues, delivery, and customer satisfaction measures and how all these are correlated and driven by the inventory.

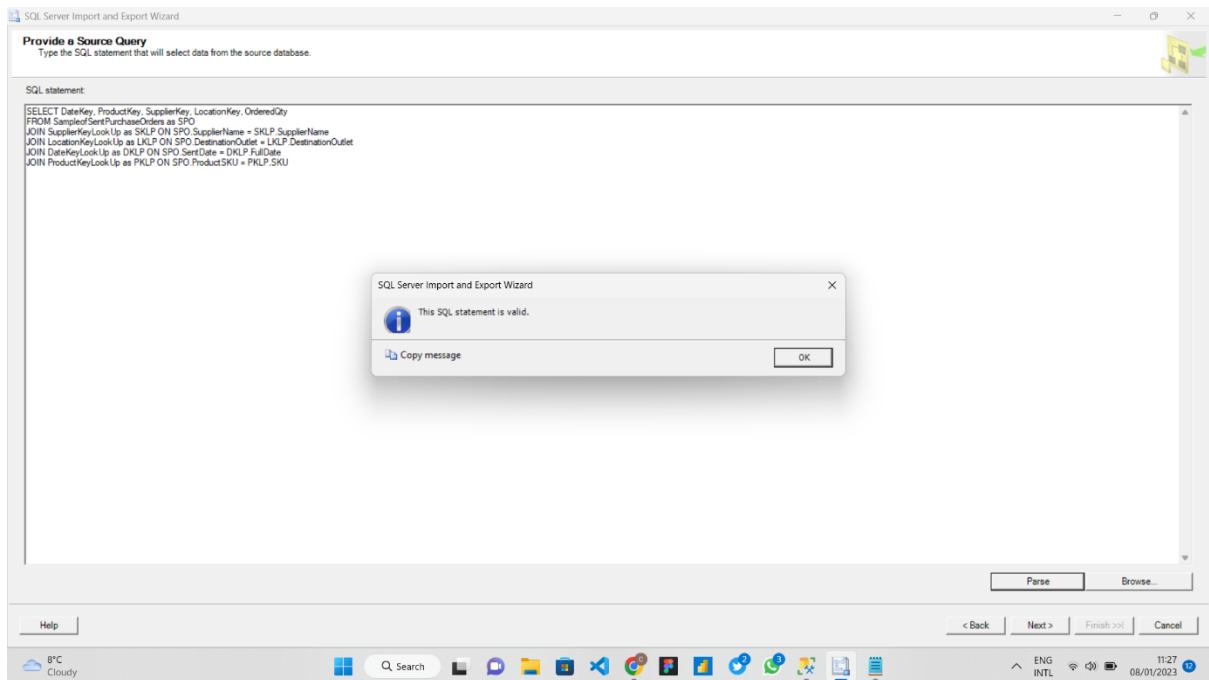


## IMPLEMENTATION AND TESTING OF THE DATA WAREHOUSE(SCREENSHOTS TO SHOW SQL COMMANDS AND THEIR RESULTS)

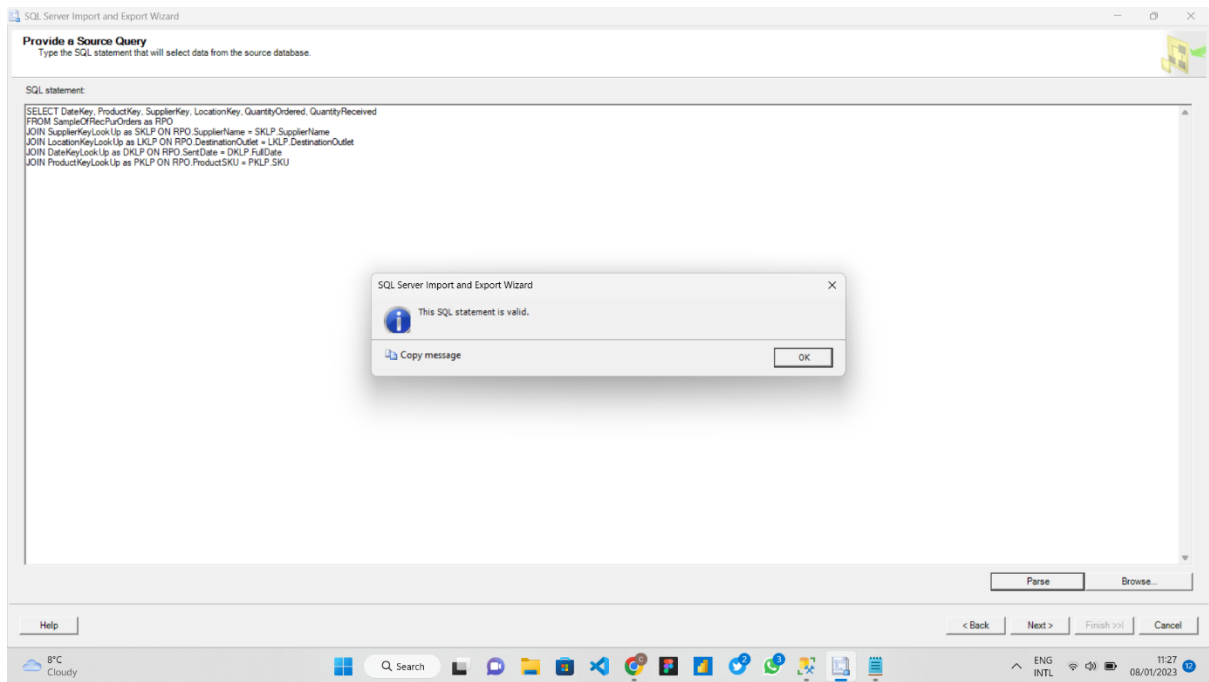
There were implementation and testing of the data warehouse on the Microsoft SQL Server Management Studio.



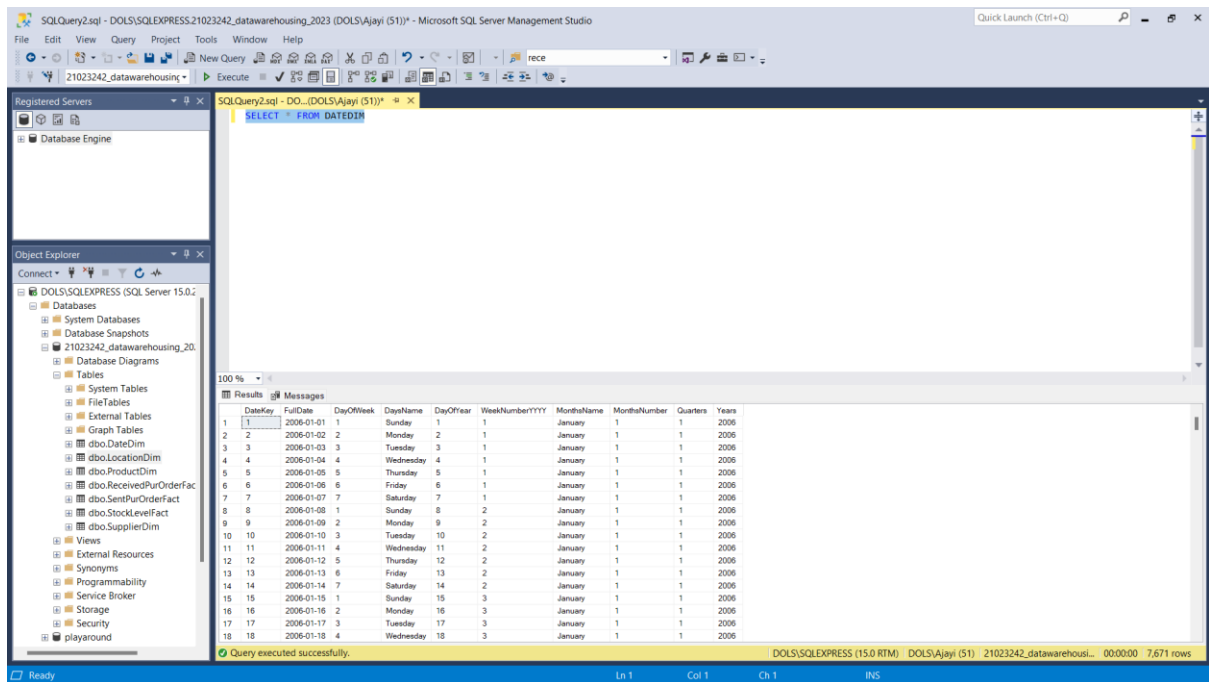
**Figure 4:** Query to import data into the Stock Level fact tables from the Staging Area



**Figure 5:** Query to import data into the Sent purchase order fact tables from the Staging Area



**Figure 6:** Query to import data into the Received purchase order fact tables from the Staging Area



**Figure 7:** Query to select all data in the Date Dimension table

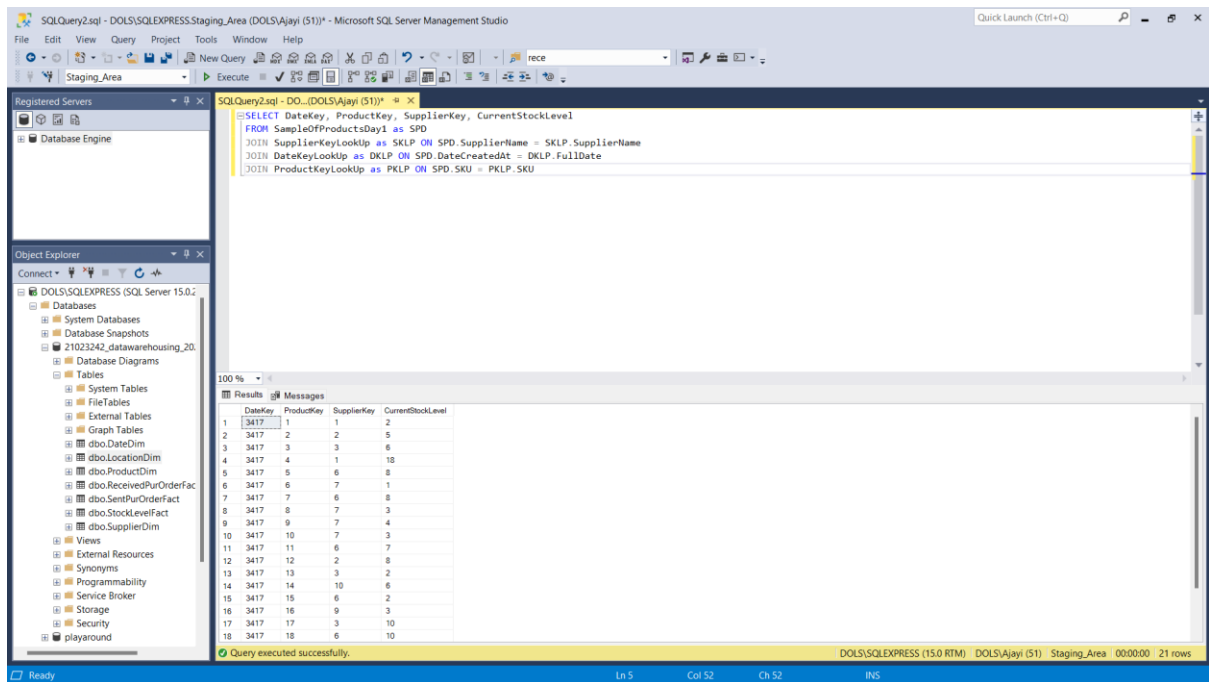


Figure 8: Query using JOIN and Aliases

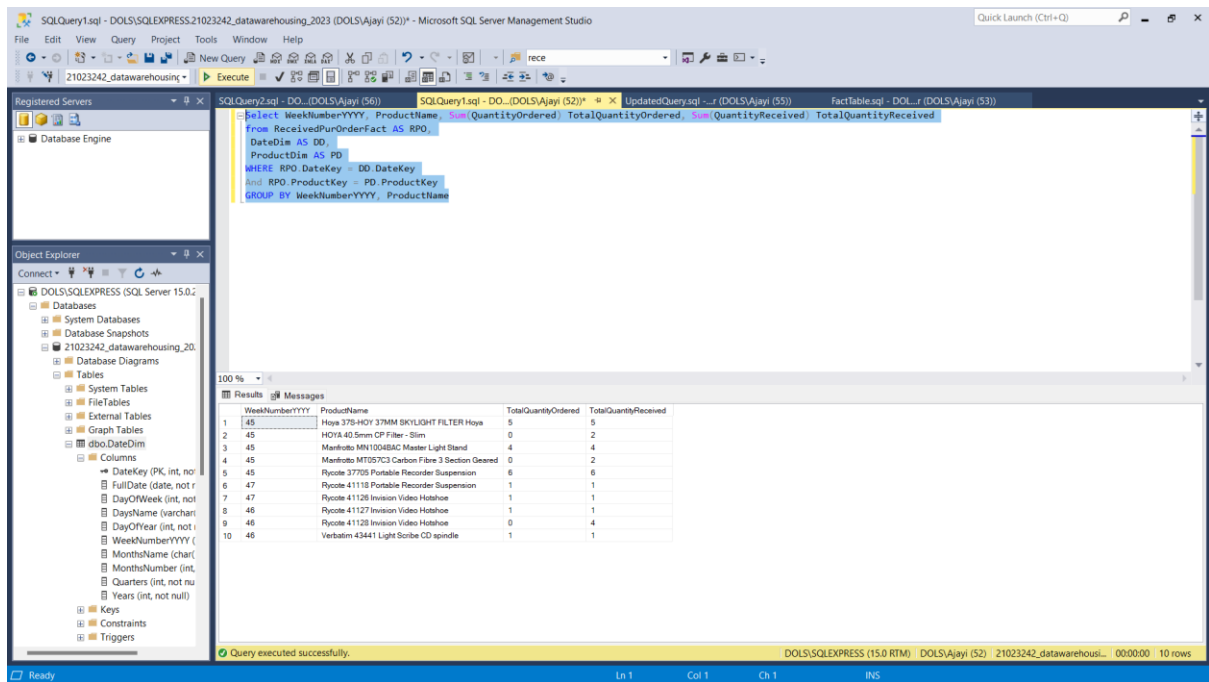
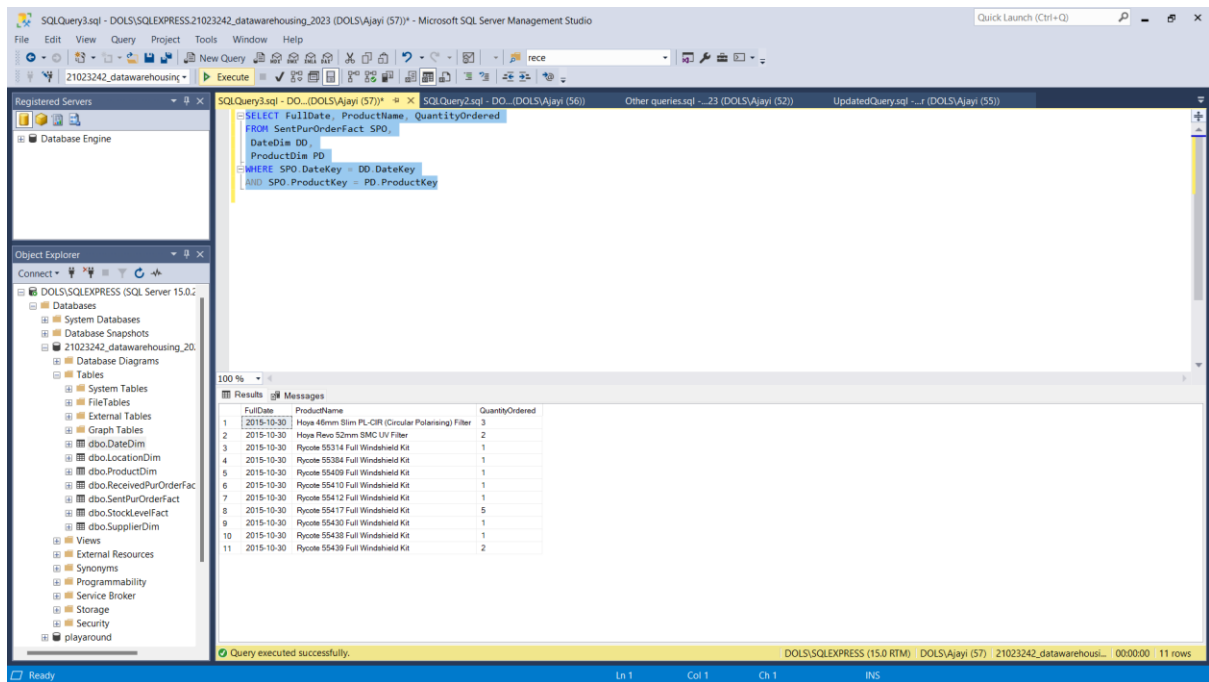
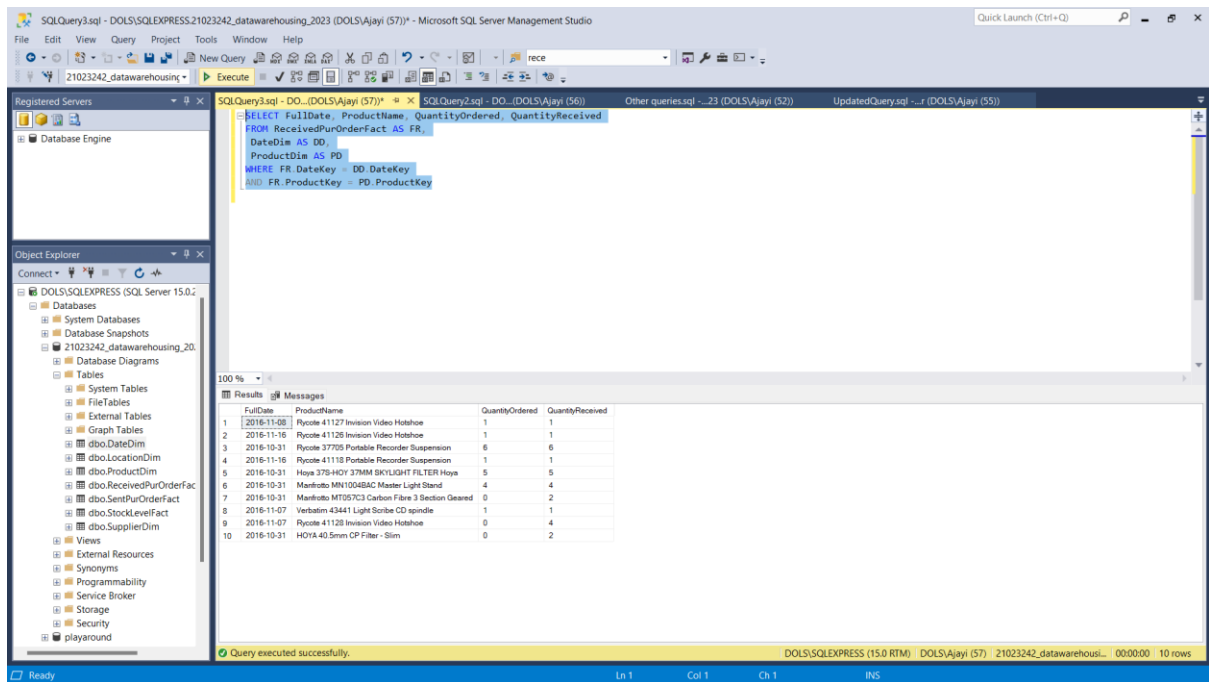


Figure 9: Stock Orders sent Weekly



**Figure 10:** Stock orders sent daily.



**Figure 11:** Stock orders received daily.

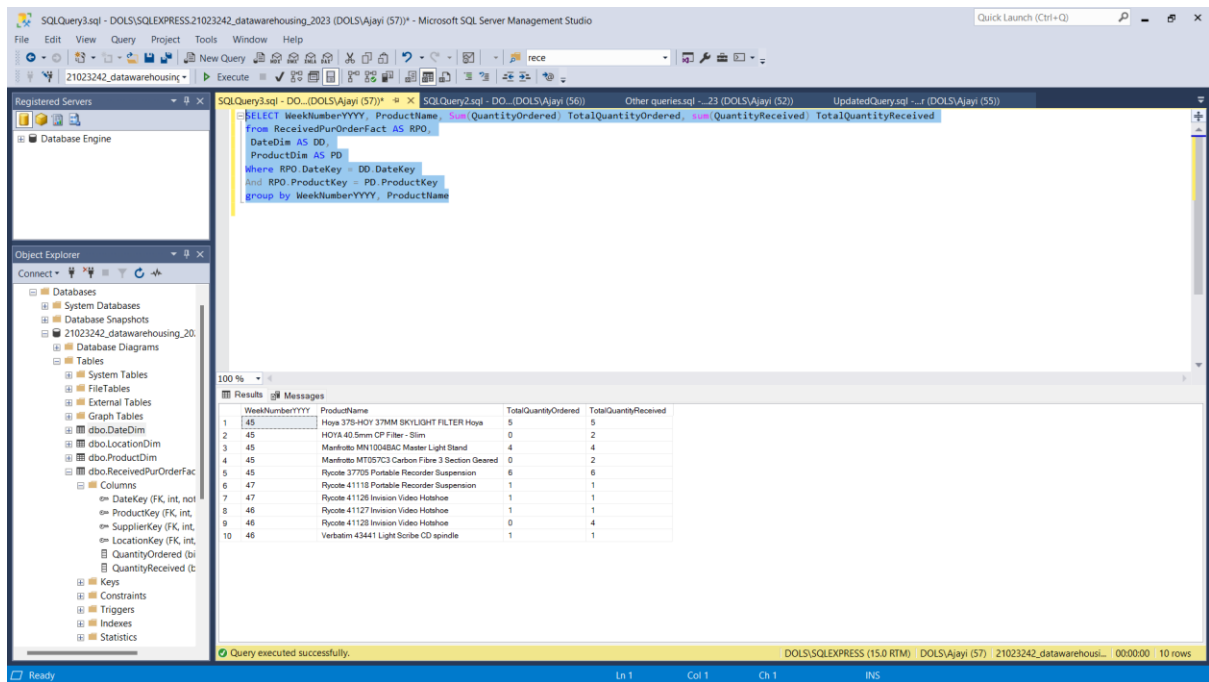


Figure 12: Received stock orders by the week

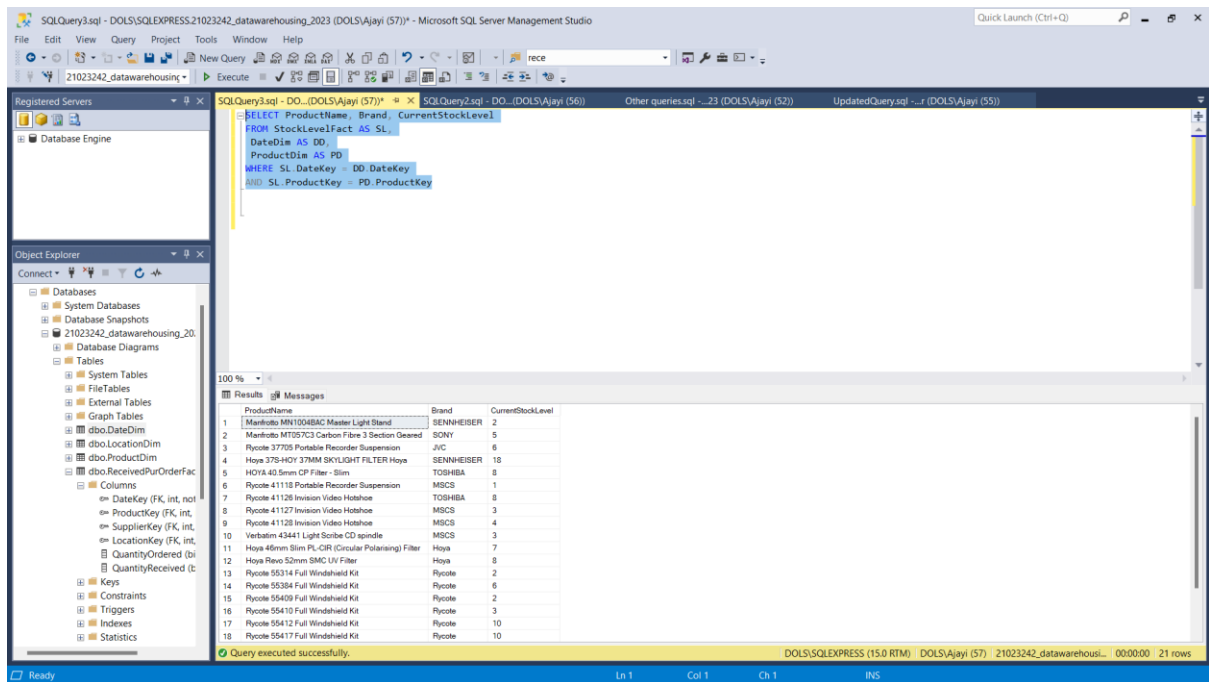
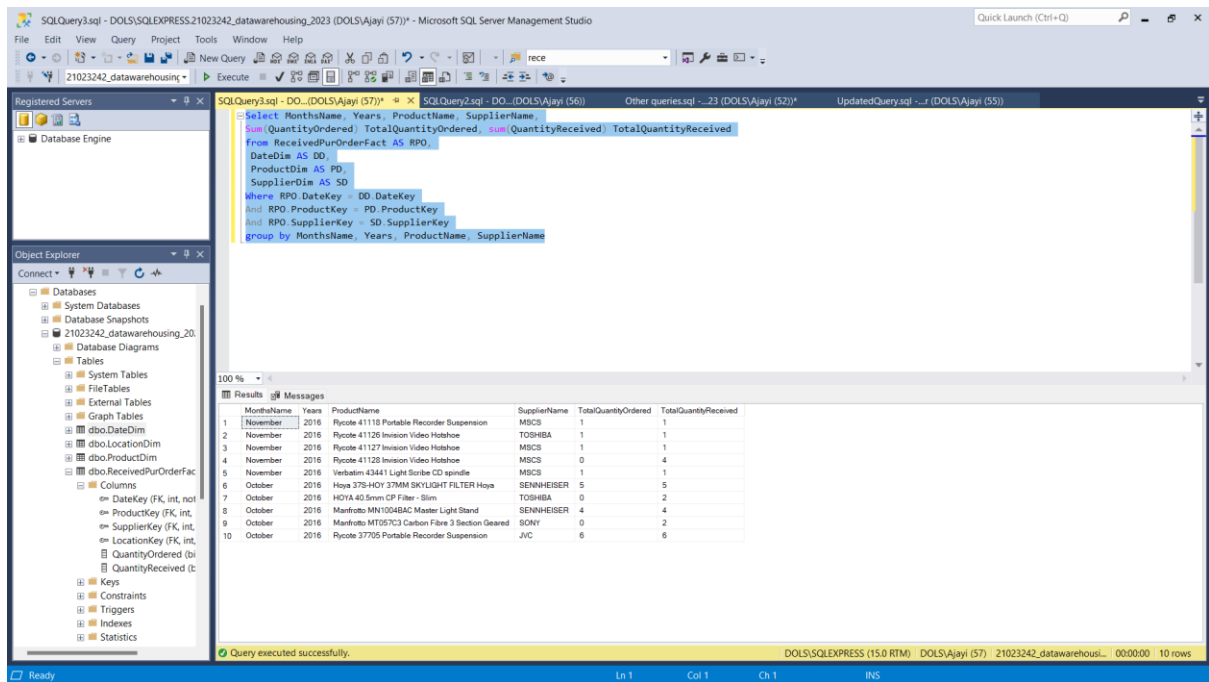
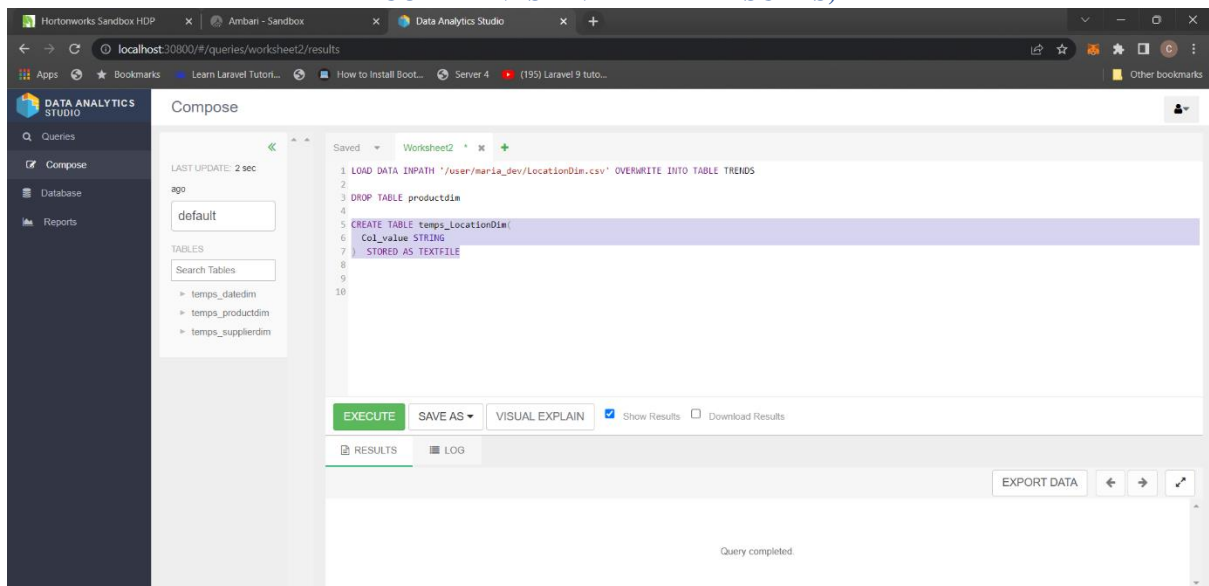


Figure 13: Stock level analyzed by brand



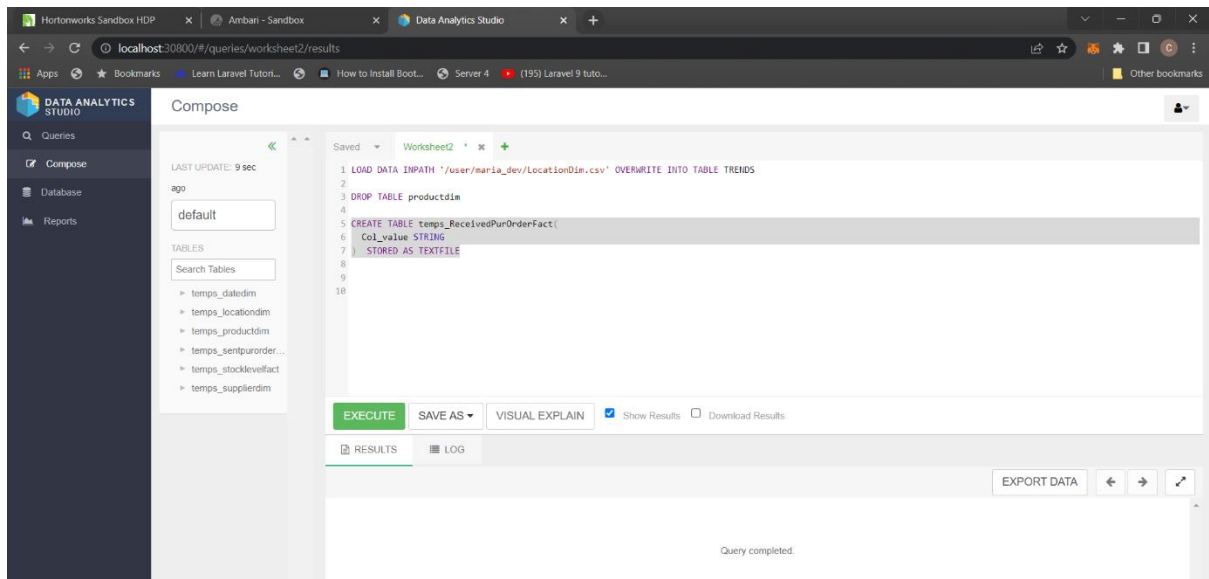
**Figure 14:** Received stock orders analyzed by supplier and month

## IMPLEMENTATION AND TESTING OF THE BIGDATA STORAGE ON HDFS(INCLUDING PIG COMMANDS AND THEIR RESULTS)

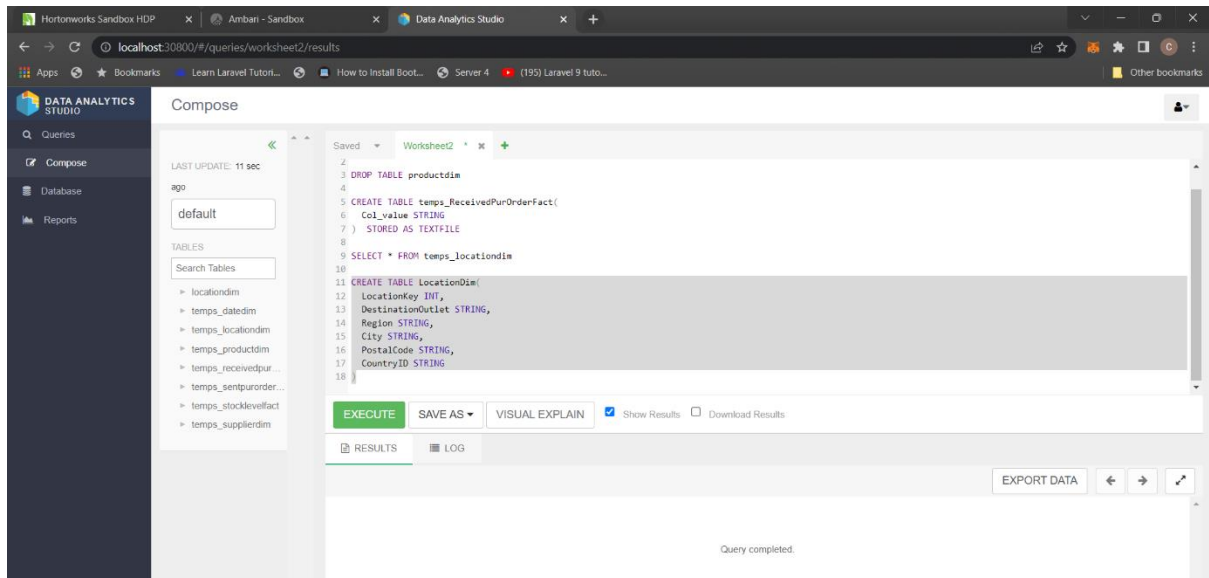


**Figure 15:** Hive Query showing how to create a temps\_LocationDim Table in Hadoop.

The temps\_LocationDim table temporarily holds the data from the CSV file imported into Hadoop. It stores the data in a single column in the temps' table before loading it into the main tables. Data cannot be loaded directly into the main tables.



**Figure 16:** Hive Query showing how to create a temps\_ReceivedPurOrderFact Table in Hadoop.



**Figure 17:** Hive Query showing how to create a LocationDim Table in Hadoop

The data's loading from the temps' tables directly into the main tables.

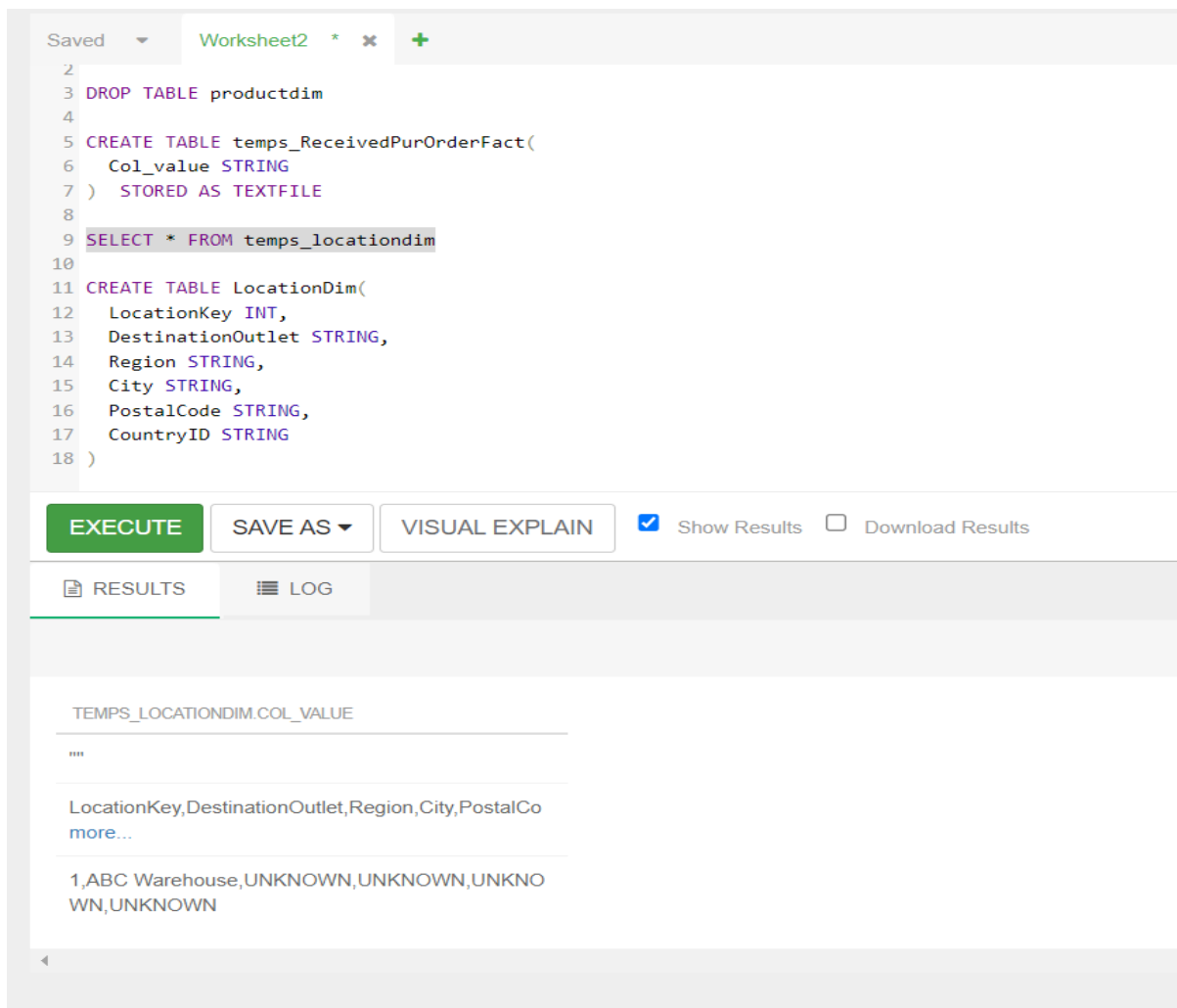


Figure 18: Query to show the data in the temps\_locationdim table.

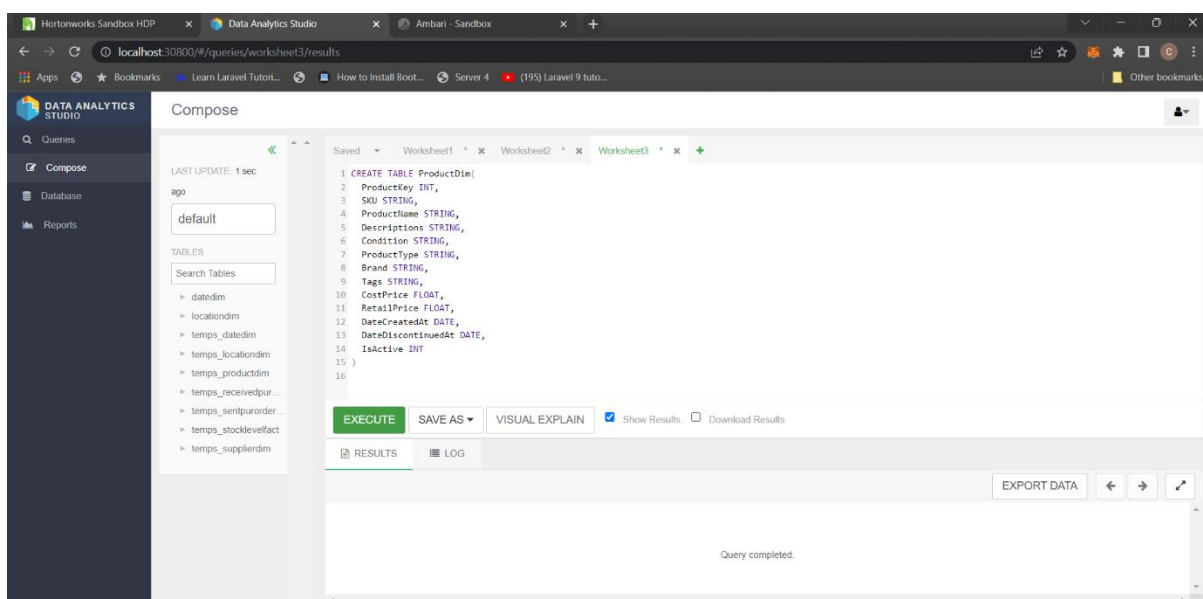
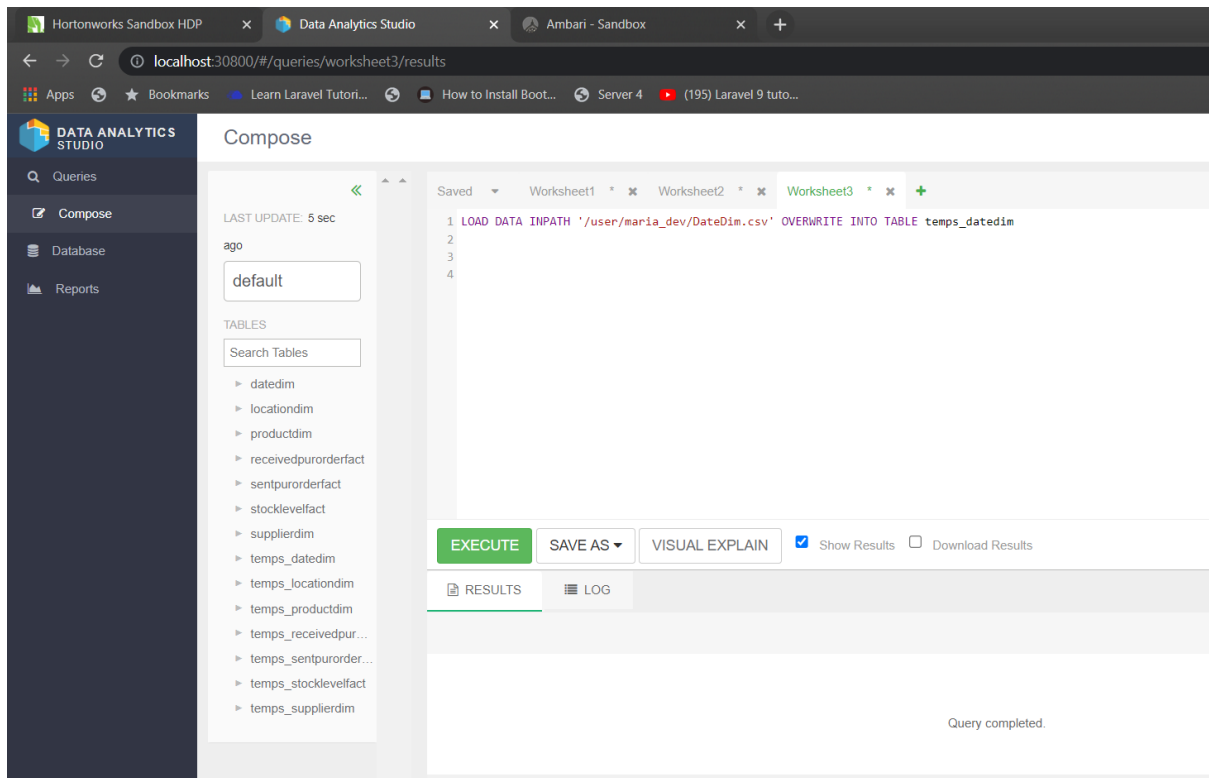


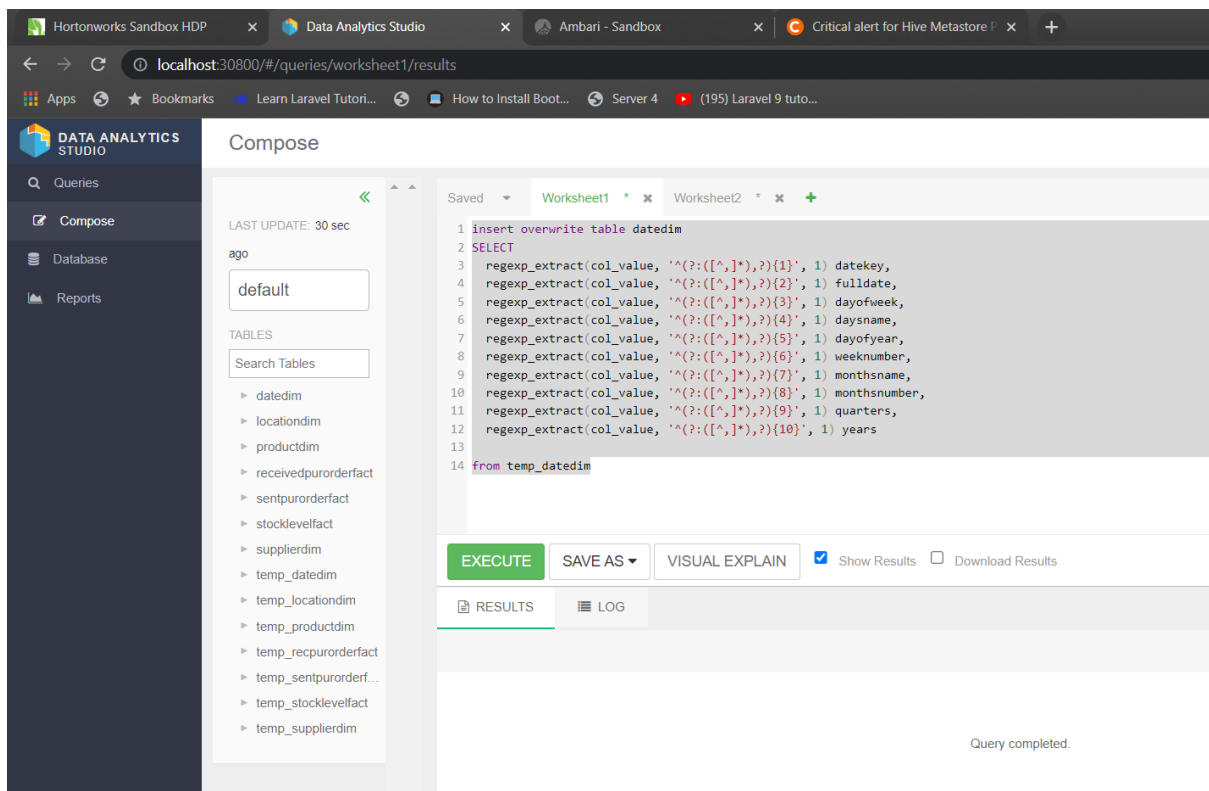
Figure 19: Hive Query Showing how to create a table.

In the figure above, there is the creation of the ProductDim table.





**Figure 20:** Hive Query showing how to load data into the temps\_datedim table from the CSV file uploaded in the marie\_dev folder in the file view.



**Figure 21:** Hive Query exporting the data from the temps' table into the main table.

As earlier speculated, the data exporting must first be into the temps' table and, in turn, exported into the main table, as shown above. The code above overwrites the main table and uses regular expressions to write the data into the main table.

DATEDIM DATEKEY	DATEDIM FULLDATE	DATEDIM DAYOFWEEK	DATEDIM DAYSNAME	DATEDIM DAYOFYEAR	DATEDIM WEEKNUMBER	DATEDIM MONTHSNAME	DATEDIM MONTHNUMBER	DATEDIM C
null	null	null	''	null	null	''	null	null
null	null	null	DaysName	null	null	MonthsName	null	null
1	2006-01-01	1	Sunday	1	1	January	1	1
2	2006-01-02	2	Monday	2	1	January	1	1
3	2006-01-03	3	Tuesday	3	1	January	1	1
4	2006-01-04	4	Wednesday	4	1	January	1	1

**Figure 22:** Hive Query showing the data in the datedim table.

The above query shows all the data inserted into the datedim table after importing from the temps\_datedim table.

```

date_dim = LOAD '/user/maria_dev/DateDim.csv' USING PigStorage(',')
AS (DateKey:int, FullDate:chararray, DayOfWeek:int,
DaysName:chararray, DayOfYear:int, WeekNumber:int,
MonthsName:chararray, MonthsNumber:int, Quarters:int,
Years:int);
DESCRIBE date_dim;
~
~
~
~
~
~
~
~
~
~

```

**Figure 23:** Pig Query showing how to load data with the schema into the pig file

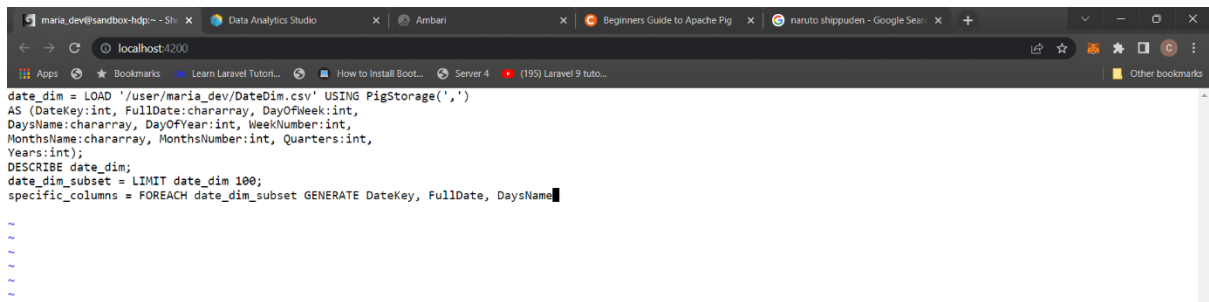
```

[maria_dev@sandbox-hdp ~]$ pig -f datedim
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
23/01/11 16:38:44 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
23/01/11 16:38:44 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
23/01/11 16:38:44 INFO pig.ExecTypeProvider: Trying ExecType : TEZ_LOCAL
23/01/11 16:38:44 INFO pig.ExecTypeProvider: Trying ExecType : TEZ
23/01/11 16:38:44 INFO pig.ExecTypeProvider: Picked TEZ as the ExecType
2023-01-11 16:38:44,404 [main] INFO org.apache.pig.Main - Apache Pig version 0.16.0.3.0.1.0-187 (rUnversioned directory) compiled Sep 19 2018, 10:13:33
2023-01-11 16:38:44,405 [main] INFO org.apache.pig.Main - Logging error messages to: /home/maria_dev/pig_1673455124397.log
2023-01-11 16:38:47,365 [main] INFO org.apache.pig.impl.util.Utils - Default bootstrap file /home/maria_dev/.pigbootstrap not found
2023-01-11 16:38:47,735 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: hdfs://sandbox-hdp.hortonwork
s.com:8020
2023-01-11 16:38:50,291 [main] INFO org.apache.pig.PigServer - Pig Script ID for the session: PIG-datedim-0e13724c-9447-4860-8539-a659f0333e55
2023-01-11 16:38:50,293 [main] WARN org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set to false
date_dim: {DateKey: int,FullDate: chararray,DayOfWeek: int,DaysName: chararray,DayOfYear: int,WeekNumber: int,MonthsName: chararray,MonthsNumber: int,Quarters: int,Year
s: int}
2023-01-11 16:38:54,235 [main] INFO org.apache.pig.Main - Pig script completed in 10 seconds and 502 milliseconds (10502 ms)
[maria_dev@sandbox-hdp ~]$

```

**Figure 24:** Results of the Pig Query executed above.

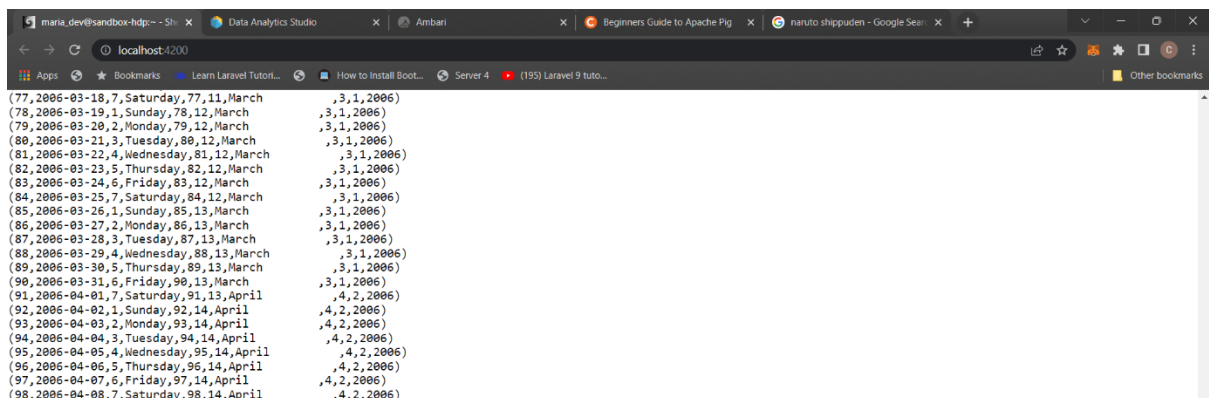
Figure 24 shows the results of the query run in figure 23.



```
date_dim = LOAD '/user/maria_dev/DateDim.csv' USING PigStorage(',')
AS (DateKey:int, FullDate:chararray, DayOfWeek:int,
DaysName:chararray, DayOfYear:int, WeekNumber:int,
MonthsName:chararray, MonthsNumber:int, Quarters:int,
Years:int);
DESCRIBE date_dim;
date_dim_subset = LIMIT date_dim 100;
specific_columns = FOREACH date_dim_subset GENERATE DateKey, FullDate, DaysName;
```

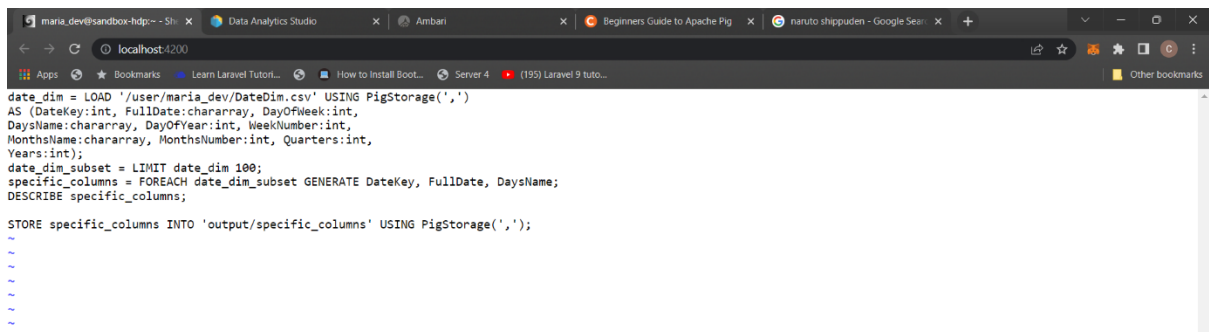
**Figure 25:** Query showing data from the DateDim table.

The query above has a limit of 100 outputs using the LIMIT function. The results also show the specific columns using the FOREACH function. It shows the first 100 entries of the DateKey, FullDate, and DaysName of the DateDim table.



```
(77, 2006-03-18, 7, Saturday, 77, 11, March, 3, 1, 2006)
(78, 2006-03-19, 1, Sunday, 78, 12, March, 3, 1, 2006)
(79, 2006-03-20, 2, Monday, 79, 12, March, 3, 1, 2006)
(80, 2006-03-21, 3, Tuesday, 80, 12, March, 3, 1, 2006)
(81, 2006-03-22, 4, Wednesday, 81, 12, March, 3, 1, 2006)
(82, 2006-03-23, 5, Thursday, 82, 12, March, 3, 1, 2006)
(83, 2006-03-24, 6, Friday, 83, 12, March, 3, 1, 2006)
(84, 2006-03-25, 7, Saturday, 84, 12, March, 3, 1, 2006)
(85, 2006-03-26, 1, Sunday, 85, 13, March, 3, 1, 2006)
(86, 2006-03-27, 2, Monday, 86, 13, March, 3, 1, 2006)
(87, 2006-03-28, 3, Tuesday, 87, 13, March, 3, 1, 2006)
(88, 2006-03-29, 4, Wednesday, 88, 13, March, 3, 1, 2006)
(89, 2006-03-30, 5, Thursday, 89, 13, March, 3, 1, 2006)
(90, 2006-03-31, 6, Friday, 90, 13, March, 3, 1, 2006)
(91, 2006-04-01, 7, Saturday, 91, 13, April, 4, 2, 2006)
(92, 2006-04-02, 1, Sunday, 92, 14, April, 4, 2, 2006)
(93, 2006-04-03, 2, Monday, 93, 14, April, 4, 2, 2006)
(94, 2006-04-04, 3, Tuesday, 94, 14, April, 4, 2, 2006)
(95, 2006-04-05, 4, Wednesday, 95, 14, April, 4, 2, 2006)
(96, 2006-04-06, 5, Thursday, 96, 14, April, 4, 2, 2006)
(97, 2006-04-07, 6, Friday, 97, 14, April, 4, 2, 2006)
(98, 2006-04-08, 7, Saturday, 98, 14, April, 4, 2, 2006)
```

**Figure 26.** The results of the query run in Figure 19.



```
date_dim = LOAD '/user/maria_dev/DateDim.csv' USING PigStorage(',')
AS (DateKey:int, FullDate:chararray, DayOfWeek:int,
DaysName:chararray, DayOfYear:int, WeekNumber:int,
MonthsName:chararray, MonthsNumber:int, Quarters:int,
Years:int);
date_dim_subset = LIMIT date_dim 100;
specific_columns = FOREACH date_dim_subset GENERATE DateKey, FullDate, DaysName;
DESCRIBE specific_columns;
STORE specific_columns INTO 'output/specific_columns' USING PigStorage(',');
```

**Figure 27:** Query shows the storing of specific columns in the PigStorage file folders.

```
Success!

DAG 0:
      Name: PigLatin:datedim-0_scope-0
      ApplicationId: job_1673441887774_0010
      TotalLaunchedTasks: 2
      FileBytesRead: 1081
      FileBytesWritten: 1081
      HdfsBytesRead: 131072
      HdfsBytesWritten: 2183
      SpillableMemoryManager spill count: 0
      Bags proactively spilled: 0
      Records proactively spilled: 0

DAG Plan:
Tez vertex scope-20 -> Tez vertex scope-22,
Tez vertex scope-22

Vertex Stats:
VertexId Parallelism TotalTasks InputRecords ReduceInputRecords OutputRecords FileBytesRead FileBytesWritten HdfsBytesRead HdfsBytesWritten Alias Feature0
scope-20 1 1 100 0 100 0 1081 131072 0 date_dim,date_dim_sub
set
scope-22 1 1 100 0 100 1081 0 0 2183 date_dim,date_dim_sub
set,specific_columns LIMIT hdfs://sandbox-hdp.hortonworks.com:8020/user/maria_dev/output/specific_columns,

Input(s):
Successfully read 100 records (131072 bytes) from: "/user/maria_dev/DateDim.csv"

Output(s):
Successfully stored 100 records (2183 bytes) in: "hdfs://sandbox-hdp.hortonworks.com:8020/user/maria_dev/output/specific_columns"

2023-01-11 18:19:14,283 [main] INFO org.apache.pig.Main - Pig script completed in 29 minutes, 41 seconds and 735 milliseconds (1781735 ms)
2023-01-11 18:19:14,283 [main] INFO org.apache.pig.backend.hadoop.executionengine.tez.TezLauncher - Shutting down thread pool
2023-01-11 18:19:14,532 [shutdown-hook-0] INFO org.apache.pig.backend.hadoop.executionengine.tez.TezSessionManager - Shutting down Tez session org.apache.tez.client.TezClient@6744c61d
2023-01-11 18:19:14,536 [shutdown-hook-0] INFO org.apache.tez.client.TezClient - Shutting down Tez Session, sessionName=PigLatin:datedim, applicationId=application_1673441887774_0010
[maria_dev@sandbox-hdp ~]$
```

Figure 28: The result of the Pig query ran in Figure 21.

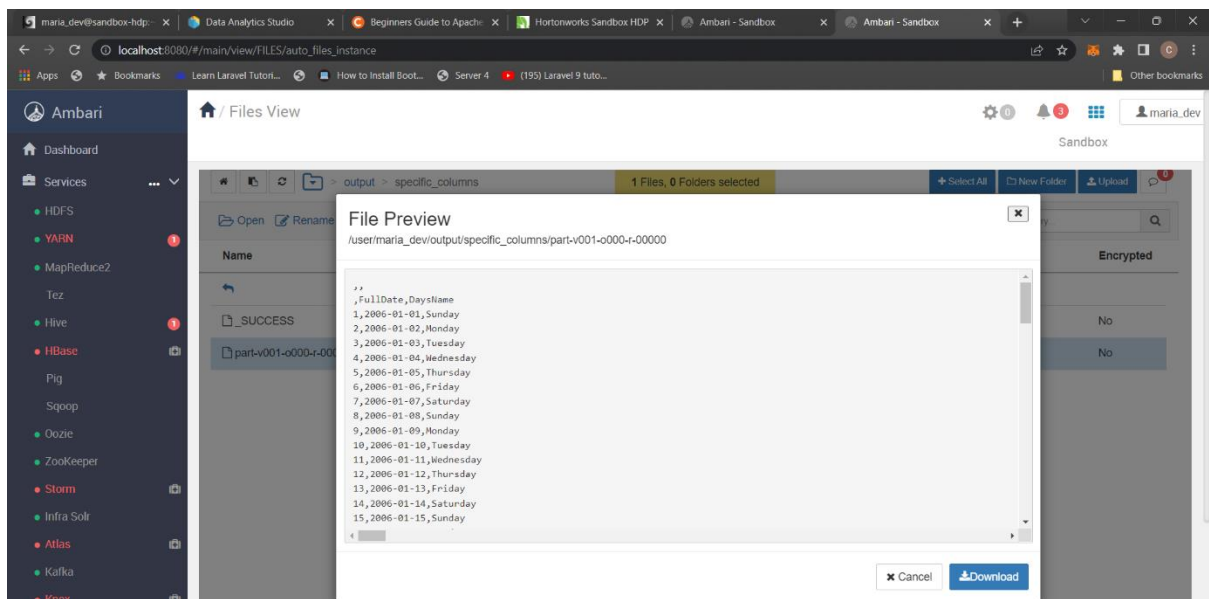
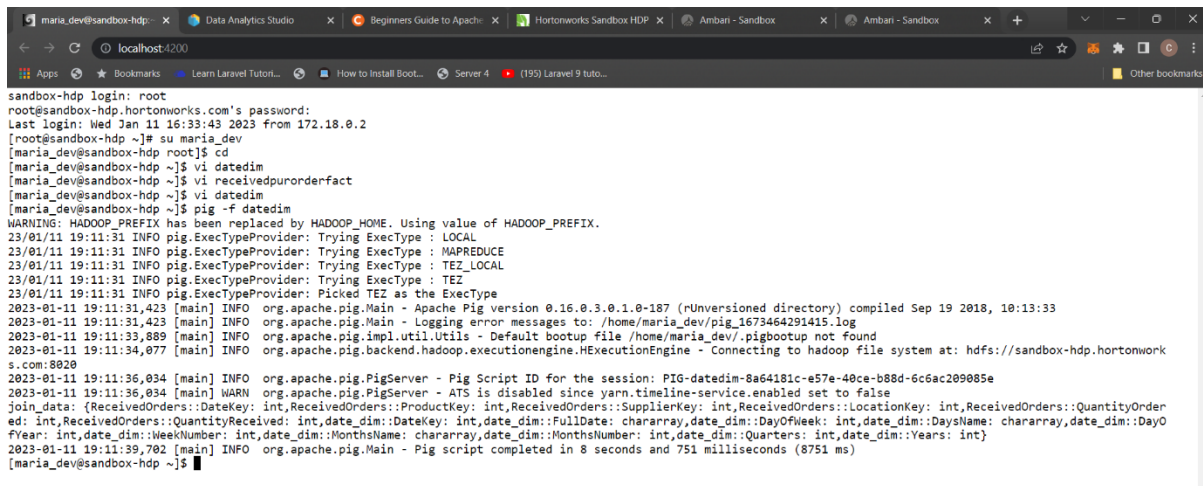


Figure 29: The file view of the query ran in Figure 21.

```
date_dim = LOAD '/user/maria_dev/DateDim.csv' USING PigStorage(',')
AS (DateKey:int, FullDate:chararray, DayOfWeek:int,
DayName:chararray, DayOfYear:int, WeekNumber:int,
MonthName:chararray, MonthsNumber:int, Quarters:int,
Years:int);
ReceivedOrders = LOAD '/user/maria_dev/ReceivedPurOrderFact.csv' USING PigStorage(',')
AS (DateKey:int, ProductKey:int, SupplierKey:int,
LocationKey:int, QuantityOrdered:int, QuantityReceived:int);
join_data = JOIN ReceivedOrders BY (DateKey), date_dim BY (DateKey);
DESCRIBE join_data;
~
~
```

Figure 30: Pig Query using the JOIN keyword to join the data in two tables into one. In this case, the DateDim and ReceivedPurchaseOrderFact tables.



```
sandbox-hdp login: root
root@sandbox-hdp.hortonworks.com's password:
Last login: Wed Jan 11 16:33:43 2023 from 172.18.0.2
[root@sandbox-hdp ~]# su maria_dev
[maria_dev@sandbox-hdp root]$ cd
[maria_dev@sandbox-hdp ~]$ vi datedim
[maria_dev@sandbox-hdp ~]$ vi receivedpurorderfact
[maria_dev@sandbox-hdp ~]$ vi datedim
[maria_dev@sandbox-hdp ~]$ pig -f datedim
[maria_dev@sandbox-hdp ~]$

WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
23/01/11 19:11:31 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
23/01/11 19:11:31 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
23/01/11 19:11:31 INFO pig.ExecTypeProvider: Trying ExecType : TEZ_LOCAL
23/01/11 19:11:31 INFO pig.ExecTypeProvider: Trying ExecType : TEZ
23/01/11 19:11:31 INFO pig.ExecTypeProvider: Picked TEZ as the ExecType
2023-01-11 19:11:31,423 [main] INFO org.apache.pig.Main - Apache Pig version 0.16.0.3.0.1.0-187 (rUnversioned directory) compiled Sep 19 2018, 10:13:33
2023-01-11 19:11:31,423 [main] INFO org.apache.pig.Main - Logging error messages to: /home/maria_dev/pig_1673464291415.log
2023-01-11 19:11:33,889 [main] INFO org.apache.pig.impl.util.Utils - Default bootstrap file /home/maria_dev/.pigbootstrap not found
2023-01-11 19:11:34,077 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: hdfs://sandbox-hdp.hortonwork
s.com:8020
2023-01-11 19:11:36,934 [main] INFO org.apache.pig.PigServer - Pig Script ID for the session: PIG-datedim-8a64181c-e57e-40ce-b88d-6c6ac209085e
2023-01-11 19:11:36,934 [main] WARN org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set to false
join_data: (ReceivedOrders::DateKey: int,ReceivedOrders::ProductKey: int,ReceivedOrders::SupplierKey: int,ReceivedOrders::LocationKey: int,ReceivedOrders::QuantityOrder
ed: int,ReceivedOrders::QuantityReceived: int,date_dim::DateKey: int,date_dim::FullDate: chararray,date_dim::DayOfWeek: int,date_dim::DaysName: chararray,date_dim::DayO
fYear: int,date_dim::WeekNumber: int,date_dim::MonthsName: chararray,date_dim::MonthsNumber: int,date_dim::Quarters: int,date_dim::Years: int)
2023-01-11 19:11:39,702 [main] INFO org.apache.pig.Main - Pig script completed in 8 seconds and 751 milliseconds (8751 ms)
[maria_dev@sandbox-hdp ~]$
```

**Figure 31:** The result of the Pig Query from Figure 24.

## CONCLUSION

The coursework has been an excellent learning experience. One has learned the terminologies and technical know-how of big data and data warehousing. The knowledge of data modeling has also been beneficial, including dimension, fact tables, and granularity. MSSQL has also played a big part in the learning process. One cannot overstate the importance of SQL in the learning process. It laid the building block for the other technologies used and learned in this coursework. Schema diagrams, tables, queries, and the import and export tools have all been beneficial, and one cannot wait to use them in the real world. As someone with previous knowledge of SQL, all the above listed have been very beneficial.

Hadoop proved difficult because of the issues and errors encountered when powering on the VM machine and the Hadoop platform. These issues made Hadoop run unsmooth. Even with the issues, one could quickly grasp Hadoop Hive and Pig knowledge and technical know-how. Learning how to create tables in Hive and create a temps table to hold the data before exporting it into the main table was possible. The know-how involved in what happens when loading data into the temps' table was also possible with the coursework.

Apache Pig ran smoothly, and one could quickly grasp the knowledge of the technology involved. From creating the pig table using the "vi" command and loading the CSV files into the Pig storage. Writing Pig scripts was also straightforward with the previous knowledge of the teachings in class and a small quantity of research. Performing queries like join, order by, creating a schema, loading data, and selecting specific columns were also easy.

The combination of all the technologies used in this project made knowing what Data Warehousing and Big Data are all about.

## REFERENCES

- [1] I. Ahmed, "What is Data Warehousing? Concepts, Features, and Examples," Astera Software, 6 November 2020. [Online]. Available: <https://www.astera.com/type/blog/what-is-data-warehousing/>. [Accessed 10 January 2022].
- [2] D. Taylor, "What is Dimensional Modeling in Data Warehouse? Learn Types," GURU99, 5 January 2023. [Online]. Available: <https://www.guru99.com/dimensional-model-data-warehouse.html>. [Accessed 10 January 2023].
- [3] A. Srivastava, "Dimensional Data Modeling," Medium, 11 December 2019. [Online]. Available: <https://towardsdatascience.com/dimensional-data-modeling-49038b96d95a>. [Accessed 10 January 2023].
- [4] IBM, "Fact tables and entities," IBM, 3 September 2021. [Online]. Available: <https://www.ibm.com/docs/it/ida/9.1.2?topic=models-fact-tables-entities>. [Accessed 10 January 2023].

## APPENDICES

### APPENDIX A

--Query to create Sent Purchase order table

```
CREATE TABLE dbo.SentPurOrderFact(  
    DateKey int NOT NULL,  
    ProductKey int NOT NULL,  
    SupplierKey int NOT NULL,  
    LocationKey int NOT NULL,  
    QuantityOrdered bigint NOT NULL  
);
```

--Query to create the Foreign Keys

```
ALTER TABLE dbo.SentPurOrderFact ADD  
CONSTRAINT FK_DateKey FOREIGN KEY (DateKey) REFERENCES  
dbo.DateDim (DateKey),  
CONSTRAINT FK_ProductKey FOREIGN KEY (ProductKey) REFERENCES  
dbo.ProductDim (ProductKey),  
CONSTRAINT FK_SupplierKey FOREIGN KEY (SupplierKey) REFERENCES  
dbo.SupplierDim (SupplierKey),  
CONSTRAINT FK_LocationKey FOREIGN KEY (LocationKey) REFERENCES  
dbo.LocationDim (LocationKey);  
GO
```

--Query to create Date Dimension

```
CREATE TABLE dbo.DateDim(  
    DateKey int Not NULL IDENTITY(1,1),  
    FullDate date NOT NULL,  
    DayOfWeek int Not NULL,  
    DaysName varchar(10) NOT NULL,  
    DayOfYear int NOT NULL,  
    WeekNumberYYYY int Not NULL,  
    MonthsName Char(3) Not NULL,  
    MonthsNumber int Not NULL,  
    Quarters int Not NULL,  
    Years int Not NULL  
CONSTRAINT PK_DateDim PRIMARY KEY (DateKey)  
) ON [PRIMARY];
```

--Query to alter the Product Dimension table

```
ALTER TABLE ProductDim  
ALTER COLUMN Tags varchar(40) Not NULL  
GO
```

--Query to create the LocationKey Lookup table

```
CREATE TABLE dbo.LocationKeyLookUp(  
    FirstLineAddress varchar(60) NOT NULL PRIMARY KEY,  
    LocationKey int NOT NULL  
)
```

--Query to create the ProductKey Lookup table

```
CREATE TABLE dbo.ProductKeyLookUp(  
    SKU varchar(10) NOT NULL PRIMARY KEY,  
    ProductName varchar(60) NOT NULL,  
    ProductKey int NOT NULL  
)
```



## APPENDIX B

--Query to get stock orders sent Weekly

```
Select WeekNumberYYYY, ProductName, Sum(QuantityOrdered) TotalQuantityOrdered,
Sum(QuantityReceived) TotalQuantityReceived
from ReceivedPurOrderFact AS RPO,
    DateDim AS DD,
    ProductDim AS PD
WHERE RPO.DateKey = DD.DateKey
And RPO.ProductKey = PD.ProductKey
GROUP BY WeekNumberYYYY, ProductName
```

--Query to get stock orders sent daily

```
SELECT FullDate, ProductName, QuantityOrdered
FROM SentPurOrderFact SPO,
    DateDim DD,
    ProductDim PD
WHERE SPO.DateKey = DD.DateKey
AND SPO.ProductKey = PD.ProductKey
```

--Query to get stock orders received daily.

```
SELECT FullDate, ProductName, QuantityOrdered, QuantityReceived
FROM ReceivedPurOrderFact AS FR,
    DateDim AS DD,
    ProductDim AS PD
WHERE FR.DateKey = DD.DateKey
AND FR.ProductKey = PD.ProductKey
```

--Query to get received stock orders by the week

```
SELECT WeekNumberYYYY, ProductName, Sum(QuantityOrdered) TotalQuantityOrdered,
sum(QuantityReceived) TotalQuantityReceived
from ReceivedPurOrderFact AS RPO,
    DateDim AS DD,
    ProductDim AS PD
Where RPO.DateKey = DD.DateKey
And RPO.ProductKey = PD.ProductKey
group by WeekNumberYYYY, ProductName
```

--Query to get stock level analyzed by brand

```
SELECT ProductName, Brand, CurrentStockLevel
FROM StockLevelFact AS SL,
    DateDim AS DD,
    ProductDim AS PD
WHERE SL.DateKey = DD.DateKey
AND SL.ProductKey = PD.ProductKey
```

--Query to get received stock orders analyzed by supplier and month

```
Select MonthsName, Years, ProductName, SupplierName,
Sum(QuantityOrdered) TotalQuantityOrdered, sum(QuantityReceived) TotalQuantityReceived
from ReceivedPurOrderFact AS RPO,
    DateDim AS DD,
    ProductDim AS PD,
    SupplierDim AS SD
Where RPO.DateKey = DD.DateKey
And RPO.ProductKey = PD.ProductKey
And RPO.SupplierKey = SD.SupplierKey
group by MonthsName, Years, ProductName, SupplierName
```



## APPENDIX C

--Hive query to create a Date Dim table

```
CREATE TABLE DateDim(  
    DateKey INT,  
    FullDate DATE,  
    DayOfWeek INT,  
    DaysName STRING,  
    DayOfYear INT,  
    WeekNumber INT,  
    MonthsName STRING,  
    MonthsNumber INT,  
    Quarters INT,  
    Years INT  
)
```

--Hive query to create a Received Purchase Order Fact table

```
CREATE TABLE ReceivedPurOrderFact(  
    DateKey INT,  
    ProductKey INT,  
    SupplierKey INT,  
    LocationKey INT,  
    QuantityOrdered INT,  
    QuantityReceived INT  
)
```

--Hive query to create a temps\_datedim table

```
CREATE TABLE temps_datedim(  
    Col_value STRING  
) STORED AS TEXTFILE
```

--Hive query to load data into a temps table

```
LOAD DATA INPATH '/user/maria_dev/SupplierDim.csv' OVERWRITE INTO TABLE temps_supplierdim
```

## APPENDIX D

--Hive query to copy the data in a temps table into the main table using regular expressions

insert overwrite table datedim

SELECT

```
regexp_extract(col_value, '^(:([^\,]*)?)\{1\}', 1) datekey,  
regexp_extract(col_value, '^(:([^\,]*)?)\{2\}', 1) fulldate,  
regexp_extract(col_value, '^(:([^\,]*)?)\{3\}', 1) dayofweek,  
regexp_extract(col_value, '^(:([^\,]*)?)\{4\}', 1) daysname,  
regexp_extract(col_value, '^(:([^\,]*)?)\{5\}', 1) dayofyear,  
regexp_extract(col_value, '^(:([^\,]*)?)\{6\}', 1) weeknumber,  
regexp_extract(col_value, '^(:([^\,]*)?)\{7\}', 1) monthsname,  
regexp_extract(col_value, '^(:([^\,]*)?)\{8\}', 1) monthsnumber,  
regexp_extract(col_value, '^(:([^\,]*)?)\{9\}', 1) quarters,  
regexp_extract(col_value, '^(:([^\,]*)?)\{10\}', 1) years
```

from temps\_datedim

## APPENDIX E

--Query to create a new Pig file

vi DateDim

--Query to load the Date Dimension into the specified pig file

```
date_dim = LOAD '/user/maria_dev/DateDim.csv' USING PigStorage(',');
```

```
DESCRIBE date_dim;
```

--Query to run the query written in the pig file

```
pig -f DateDim
```

--Query to load the Received Purchase Order Fact table into Pig

```
ReceivedOrders = LOAD '/user/maria_dev/ReceivedPurOrderFact.csv' USING PigStorage(',')
```

```
AS (DateKey:int, ProductKey:int, SupplierKey:int,
```

```
LocationKey:int, QuantityOrdered:int, QuantityReceived:int);
```

--Query to create a schema for the table created in Pig

```
date_dim = LOAD '/user/maria_dev/DateDim.csv' USING PigStorage(',')
```

```
AS (DateKey:int, FullDate:chararray, DayOfWeek:int,
```

```
DaysName:chararray, DayOfYear:int, WeekNumber:int,
```

```
MonthsName:chararray, MonthsNumber:int, Quarters:int,
```

```
Years:int);
```