LONDON METROPOLITAN UNIVERSITY

Implementation of Data Warehouse for ABC Consumer Electronics Outlet ltd.

Name	**Tonny K Podiyan**
Student ID	**21052135**
Email ID	**Tok0165@my.londonmet.ac.uk**
Module Code	**CS7079**
Module Title	**Data Warehousing & Big Data**
Date	**12 January 2023**

Group - 12

(6 Musketeers)

Sr. No.	Name	Course Name	ID Number
1.	Tonny K Podiyan	MSc. Data Analytics	21052135
2.	Vishnu Venugopalan Nair	MSc. Data Analytics	21051665
3.	Srikanth Kanakambhatla	MSc. Data Analytics	22016858
4.	Dhruv Mevada	MSc. Data Analytics	22025213
5.	Nihir Chauhan	MSc. Data Analytics	22014400
6.	Tony Innocent	MSc. Data Analytics	22021154

TABLE OF CONTENTS

PART A - Group Work
Designing of Warehouse

1. Introduction

In the modern world, to stay in business many companies require market knowledge and analytics to grow their business. Here we have taken one company into consideration named ABC consumer electronics outlet ltd for our project. It is one of the electronic giants in London. ABC consumer electronic outlet ltd. has six stores in London, also it conducts online business in the UK as well as in Europe. The company currently has about 10,000 products available, arranged in over 200 brands and roughly ten categories. These numbers represent a single moment in time.

As new product lines are launched and outdated items are withdrawn, the total number of products fluctuates continuously. In order to support its business processes, ABC Consumer Electronics Outlet Ltd. has previously made investments in information and communication technologies.

Many of the company's retail operations are supported by cloud-based software called Vend. It uses Xerox to support its finance department and Linnworks to automate its online operations. Each day, these software programs produce a considerable amount of transactional data. Each software program stores and maintains its own copy of the created datasets.

2. Requirements for business

In this cutting-edge era, the most important requirement is inventory management. Basically, Its major objective is to create a strong purchase order plan to guarantee that product items are accessible when required. Now, the company is facing distinct types of problems like controlling over-stocks and under-stocks of product items. So, inventory management helps to reduce the over-stocks and under-stocks through which the company will gain revenue and customer satisfaction.

Inventory management is the process that is used by the business to identify the stock, order, receive, and sell. This process helps to watch for supply and demand trends so that it can guarantee the business always has necessary items in stock.

This data is in a different format and can be utilized in optimizing and improving the operations to increase the productivity and profits of the company. To achieve the objectives mentioned previously a Data Warehouse can be implemented. A Data Warehouse is a data management system that is developed to aid business intelligence (BI) activities such as analytics. All the relevant data from various sources in the business can be integrated into one central corporate data repository which is a data warehouse. Since all the data are stored in a centralized manner, they can be extracted according to the business requirement without any

time-consuming complex process. Data Warehousing has been proven to be the most popular and effective solution.

3. Understanding the Data

For efficient results in analysis and reporting we must understand data properly. For our coursework, we are designing a data warehouse for ABC Consumer Electronic Outlet. This business is operated by 6 stores located in and around London and by online means across the UK and Europe. At the movement, the company offers more than ten thousand products organized into ten categories and in more than 200 brands. ABC Consumer Electronics Outlet Ltd has already invested and gathered data from software like Vend, Linnworks, and Xerox. These software applications are generating huge transactional data every day. The provided data files are as follows:

1) Generated Date
2) Product details
3) Sent Purchase Order details
4) Received Purchase Order details
5) Supplier details

4. Data Files

The contents of all the files listed in point 3 are used to design the schema of the Data Warehouse. All these files are described below:

1. Generated Date:

The data in the generated data file is set to a date range from 03/01/2014 to 31/12/2023. The column attributes are as follows,

datekey - unique identification number of a given date
date - date in the format of yyyy/mm/dd
year - year in yyyy
month - month number in a particular year
monthname- name of the month
week- week of the year
day-day of the month
quarternumber- quarter of the year

2. Product details:

This dataset contains data on different products sold by the company. The column attributes are as follows:

sku - unique identification for each product (stock keeping unit)
productname - name of the product
description - description of the product
condition - condition of the product
producttype - type of the product
brand - manufacturer of the product
suppliername - supplier of the product
tags - additional information for the product
costprice - cost of the product
retail price - retail/selling price of the product
currentstocklevel - stock available of a product
datecreatedat - date at which the product was introduced into this business
datediscontinuedat - date at which the product was discontinued from this business
isactive - product active flag

3. Sent Purchase Order details:

This data set contains sent purchase order details. The column attributes are as follows:

purchaseordercode - unique purchase order code for every purchase request
productsku - unique identification for each product (stock keeping unit)
supplierid - Unique identifier of a supplier
sentorderid - Unique identification number of sent orders
sentdatekey - datekey when purchase order was requested
orderedqty - quantity of a product requested in purchase order

4. Received Purchase Order details:

This data set contains received purchase order details. The column attributes are as follows,

receivedorderid - Unique identification number of received orders
purchaseordercode - unique purchase order code for every purchase request
productsku - unique identification for each product (stock keeping unit)
supplierid- Unique identifier of a supplier
warehousekey - unique identification key of a warehouse
sentdate - date when purchase order was dispatched
receiveddate - date when purchase order was delivered
receivedqty - quantity of a product received in purchase order
orderedqty - quantity of a product requested in purchase order

5. Supplier details:

This data set contains Supplier details. The column attributes are as follows,

supplierid - Unique identifier of a supplier
suppliername - supplier name of the product
supplier description - additional information about supplier
phone - supplier's phone number
email - supplier's email id
fax - supplier's fax number
firstlineaddress - supplier's address
postcode - supplier's postcode
city - supplier's city
state - supplier's sate
countryid - supplier's country id

6. Warehouse Details:

This data set contains warehouse details. The column attributes are as follows,

wh outlet id- name of the warehouse where the item is stored once the supplier ships the
warehousekey - unique identification key of a warehouse
wh phone no.- phone number of the warehouse
wh postcode- postcode of the warehouse
wh city- city where the warehouse is located

Note:- All column attributes are not used for the Part A of the coursework , only those required to satisfy reporting requirements have been used.

5. Data Profiling

Data profiling is the process of analyzing data sets in order to better understand the information and to find errors and inconsistencies in the data sets.

We found the following details through data profiling:

1. In The DimProduct table, the DateCreatedAt attribute was found to be greater than DateDiscontinuedAt which cannot be logically correct.
2. Cost Price of multiple products has been captured as Zero.
3. There are 2 attributes with the same name 'Description', one in Product and the other in supplier.

4. Only the Supplier name was given while other details like phone number, email id, and fax all were null.
5. Destination Id was provided in varchar so we had to create a dimension as DimWarehouse where we have stored additional details like the Warehouse key as a primary key. Other details such as phone number, Postcode, and city name were also added but they are currently null values.
6. In the sample of Products data file we have added a Date column which is used to refer the stock level at that particular date.

6. Reporting Requirements

As per the BRD (Business requirement document) provided by the ABC Consumer Electronic Outlet, the following are the reporting requirements:

1. To identify daily stock levels of all products for the last month. We will meet this requirement by integrating SKU, Date, and current stock level from factstockcheck. Using the date key we can convert the data into a full date and month name.
2. To generate a weekly report of all products with minimum stock levels. This report can be created by using the SKU, date, and current stock level from factstockcheck. Using the date key we can convert the data into the week number.
3. To identify stock levels by brand or product type or supplier. We can write a query using join function on dimproduct , dimsupplier and factstockcheck table.
4. To create daily and weekly sent and received stock orders for the last four weeks.We can use join function and join factsentorders and factreceivedorders along with datekey.
5. To analyze received stock orders by the supplier and by the month. We use join function and use dimsupplier,dimdate and factreceivedorders.

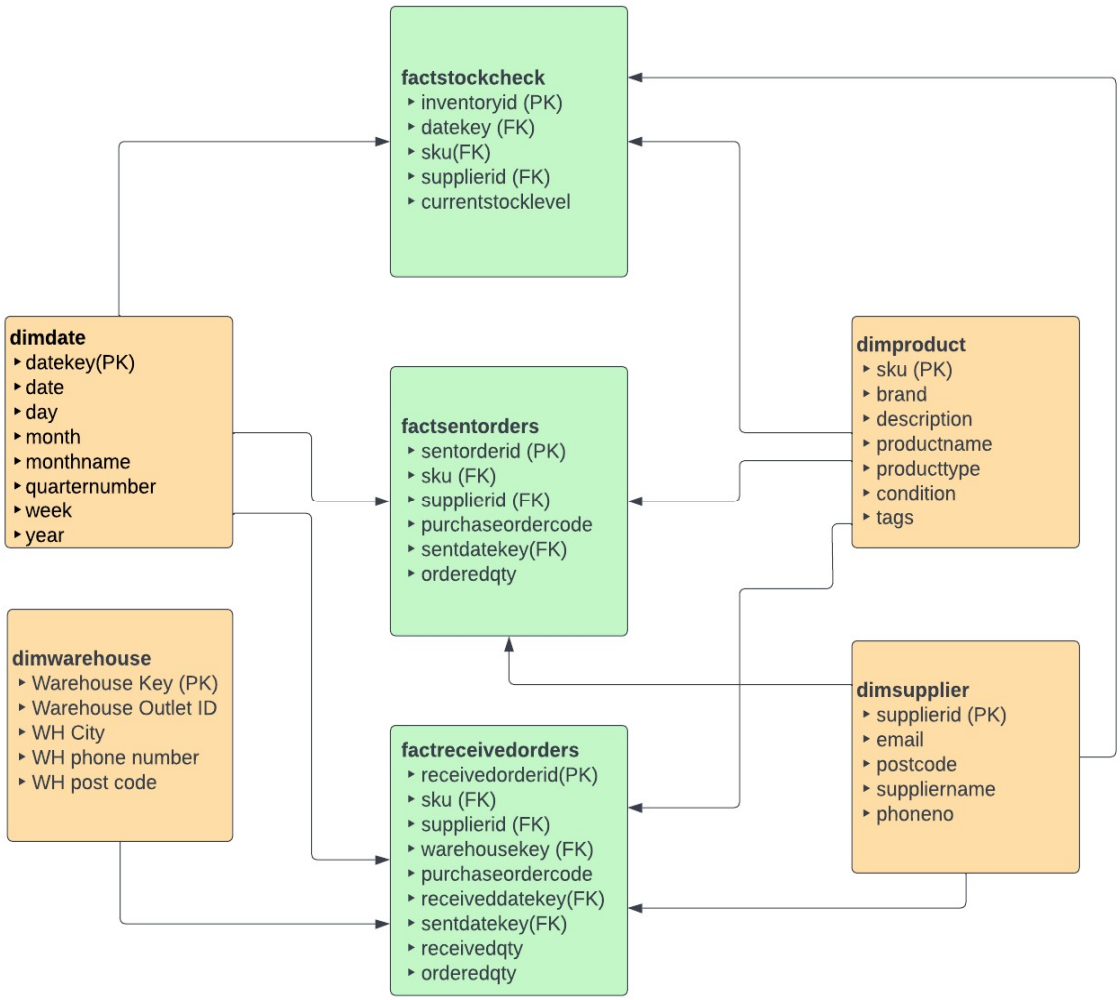
7. Data warehouse Schema Design

We have used a lucid chart to design the below schema design, there are 4 dimension tables representing the product, supplier, warehouse, and date. Based on the business requirement we have designed 3 fact tables:

- From factstockcheck we would be able to retrieve the current stock levels of different SKUs.

- factreceivedorders tells us what and when the items were received at the warehouse.

- factsentorders is where details regarding active items and discontinued items are stored.



8. Schema & Granularity

A schema is a logical description that describes the entire database. In this case, we are using a simple star schema, with 4 different dimension tables and 3 fact tables.

Note : For the sake of simplicity we have use varchar(100) for all text attributed columns , in real life length to be specified to increase query efficiency.

1.1. dimdate

The DimDate is the dimension table that is used to convert a particular date into different formats like the day of the month, day of the week, week number, Month Name, and Year. Date key is the unique identifier used here which can be later populated in the fact tables or lookup tables.

Column	***Type***	***Validation***	***Null***	***Note***
datekey	Integer	Identity (1,1)	N	PK
date	Date		N	
year	Integer		N	
month	Integer	>=1 and <=12	N	
monthname	Varchar	Varchar(10)	N	
week	Integer	>=1 and <=54	N	
day	Integer	>=1 and <=31	N	
quarternumber	Integer	0=> and <=4	N	

1.2. dimsupplier

dimsupplier is the dimension table where the data of the Supplier is stored. The supplier ID is the system-generated key whereas the supplier name, phone number, email, address are all the available data in the Purchase order. This dimension table is updated when a new supplier is added

Column	***Type***	***Validation***	***Null***	***Note***
Supplier ID	Integer	Identity (1,1)	N	PK
Supplier Name	Varchar	Varchar(100)	N	
Phone No	Integer	BigInt	Y	
Email	Varchar	%_@_%._%	Y	
PostCode	Varchar	Varchar(100)	Y	

For this coursework we have assumed values of supplier phone no., email and postcode

1.3. dimproduct

DimProduct is the dimension table where the Primary key is the SKU number and has stored all the detailed information of a product like brand name, description, and tags. This table gets updated when a new product is proposed and will be sent to the warehouse.

Column	***Type***	***Validation***	***Null***	***Note***
SKU	Varchar	Varchar(100)	N	PK
Product Name	Varchar	Varchar(100)	N	
Condition	Varchar	Varchar(100)	Y	
ProductType	Varchar	Varchar(100)	N	
Tags	Varchar	Varchar(100)	Y	
Description	Varchar	Varchar(100)	Y	
Brand	Varchar	Varchar(100)	N	

1.4. dimwarehouse

DimWarehouse is the dimension table where the Primary Key is the Warehouse Key which is a system-generated key and has stored all the detailed information of a warehouse-like its outlet Id, Phone number, Postcode, and City. This dimension table is updated when a new warehouse is added in the list

Column	***Type***	***Validation***	***Null***	***Note***
Warehouse Key	Integer	Identity (1,1)	N	PK
WH Outlet ID	Varchar	Varchar(100)	N	
WH Phone No.	Integer	Bigint	Y	
WH Postcode	Varchar	Varchar(100)	Y	
WH City	Varchar	Varchar(100)	Y	

2.1. factstockcheck

This is a Fact table that has the details of the Stock level on a particular date. This table gets updated daily bases while checking minimum stock and ordering the products. Here we mention the supplier details, SKU, Date key, available stock, Order Quantity, and sent date. We

have also created a system-generated key to identify the unique transactions happening which is known as Inventory ID.

Column	***Type***	***Validation***	***Null***	***Note***
Inventory ID	Integer	Identity (1,1)	N	PK
SKU	Varchar	Varchar(100)	N	FK
Supplier ID	Integer		N	FK
Date Key	Integer		N	FK
Current Stock Level	Integer		N	

2.2. factsentorders

This fact table is used to store purchase order details of sent orders, along with date key, sku , supplier id and ordered quantity. Since there was no unique key we have generated a primary key sentorder id.

Column	***Type***	***Validation***	***Null***	***Note***
sentorderid	Integer	Identity (1,1)	N	PK
purchaseordercode	Varchar		N	
SKU	Varchar	Varchar(100)	N	FK
Supplier ID	Integer		N	FK
sentdatekey	Integer		N	FK
orderqty	Integer		N	FK

2.3. factreceivedorder

This is a fact table where we store purchase order code details of received orders along with supplier,warehouse and product details.Also received and order quantity have been recorded in this fact table Since there was no unique key in this table we generated a primary key called receivedorderid.

Column	***Type***	***Validation***	***Null***	***Note***
receivedorderid	Integer	Identity (1,1)	N	PK
purchaseordercode	Varchar	Varchar(100)	N	
SKU	Varchar	Varchar(100)	N	FK
supplierid	Integer		N	FK
warehousekey	Integer		N	FK
sentdatekey	Integer		N	
receiveddatekey	Integer		N	
orderedqty	Integer		N	
receivedqty	Integer		N	

9. Recommendations for further development

In the above simple star schema only attributes required to run the below business activities have been used. Following are the mentioned business activities:

1. Send purchase orders to suppliers when there are product items with minimum levels of inventory.
2. Receiving purchase orders and storing stocks in appropriate locations.
3. Controlling and maintaining stocks which involves adding new products and adjusting stock levels of existing products

Further we have Cost Price and Retail Price which can be used to identify multiple functions of which some are listed below:

1. Sales
2. Profit
3. Annual Revenue
4. Top Performing Products
5. Least Performing Products
6. Supplier's with Best Price.
7. Supplier with max/min delivery times

PART B – Individual Work

10. Implementation Process in SSMS

Step 1: Create Database

In this step we are creating a database called ABC_Warehouse , further fact tables and dimension tables will then be created in this database.

```
CREATE DATABASE ABC_Warehouse
```

Step 2: Create Dimensional Tables.

a. Creating dimdate

For this we use the below query it creates a table called dimdate in ABC_Warehouse database and populates it with a record for each date within a specified range (from @StartDate to @EndDate).The dimdate table has several columns that store information about each date, such as the year, month, week, and day of the month.

```
-- declare variables to hold the start and end date
DECLARE @StartDate datetime
DECLARE @EndDate datetime

--- assign values to the start date and end date we
-- want our reports to cover (this should also take
-- into account any future reporting needs)
SET @StartDate = '2014-01-03'
SET @EndDate = '2018-12-31'

IF EXISTS (SELECT *
FROM sysobjects
WHERE type = 'U'
AND ID = OBJECT_ID('[dbo].[dimdate]') )
BEGIN
DROP TABLE [dbo].[dimdate]
PRINT 'Table dropped'
END
CREATE TABLE dbo.dimdate (
datekey int NOT NULL IDENTITY(1, 1),
[date] datetime NOT NULL,
[year] int NOT NULL,
[month] int NOT NULL,
```

```
[monthname] varchar(10) NOT NULL,
[week] int NOT NULL,
[day] int NOT NULL,
[quarternumber] int NOT NULL,
CONSTRAINT PK_dimDates PRIMARY KEY CLUSTERED (dateKey)
)


-- using a while loop increment from the start date
-- to the end date
DECLARE @LoopDate datetime
SET @LoopDate = @StartDate

WHILE @LoopDate <= @EndDate
BEGIN
-- add a record into the date dimension table for this date
INSERT INTO dimdate VALUES (
@LoopDate,
year(@LoopDate),
month(@LoopDate),
datename(MM,@LoopDate),
datepart(WK,@LoopDate),
day(@LoopDate),
CASE WHEN month(@LoopDate) IN (1, 2, 3) THEN 1
WHEN month(@LoopDate) IN (4, 5, 6) THEN 2
WHEN month(@LoopDate) IN (7, 8, 9) THEN 3
WHEN month(@LoopDate) IN (10, 11, 12) THEN 4
END

)

-- increment the LoopDate by 1 day before
-- we start the loop again
SET @LoopDate = dateadd(d, 1, @LoopDate)

END
```

b. Creating dimproduct

We create a table called dimproduct where in we can import data of different products and their SKU numbers and other information by using the sample of product data file which was provided.

```
CREATE TABLE dimproduct (
  sku varchar(100) PRIMARY KEY,
productname varchar(100),
condition  varchar(100) ,
producttype varchar(100) ,
tags varchar(100) ,
description varchar(100) ,
brand varchar(100) );
```

c. Creating dimsupplier

We create a table called dimsupplier where we store supplier details like supplier name, phone no., email and post code, we have assumed dummy data for supplier's phone no., email and post code for populating our dimension tables.

```
CREATE TABLE dimsupplier (
   supplierid int IDENTITY(1,1) PRIMARY KEY,
   suppliername varchar(100),
   phoneno bigint,
   email varchar(100),
   postcode varchar(100));
```

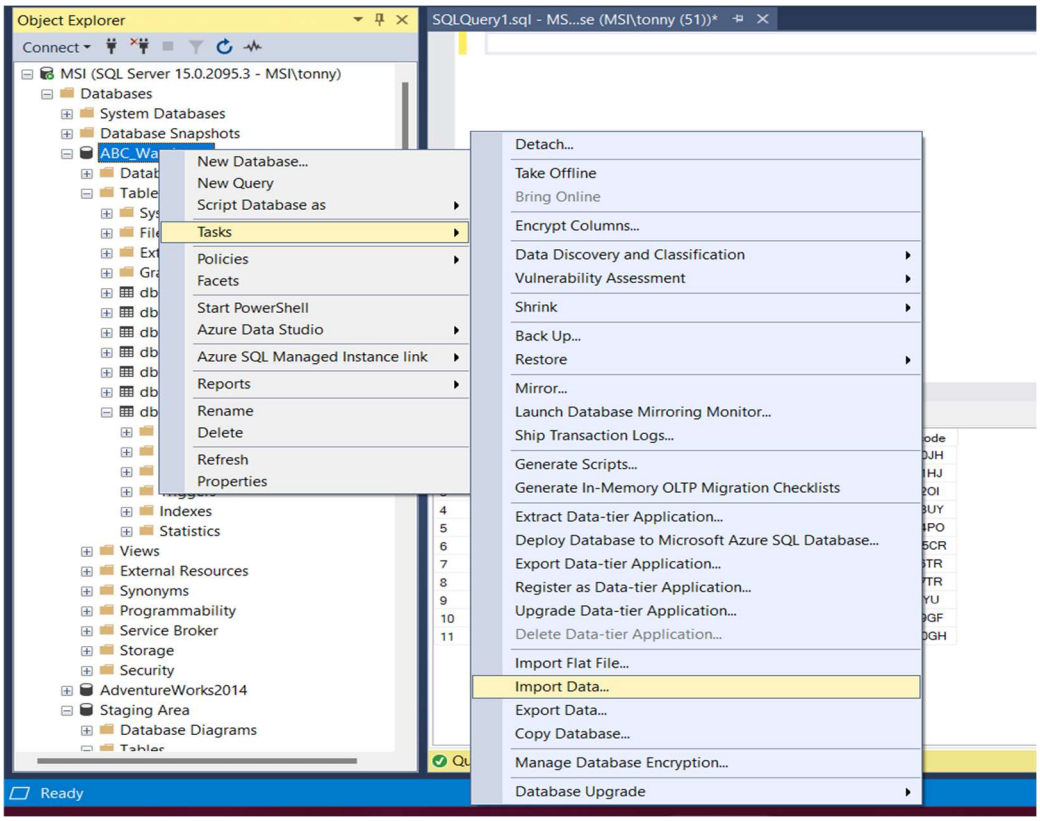
d. Creating dimwarehouse

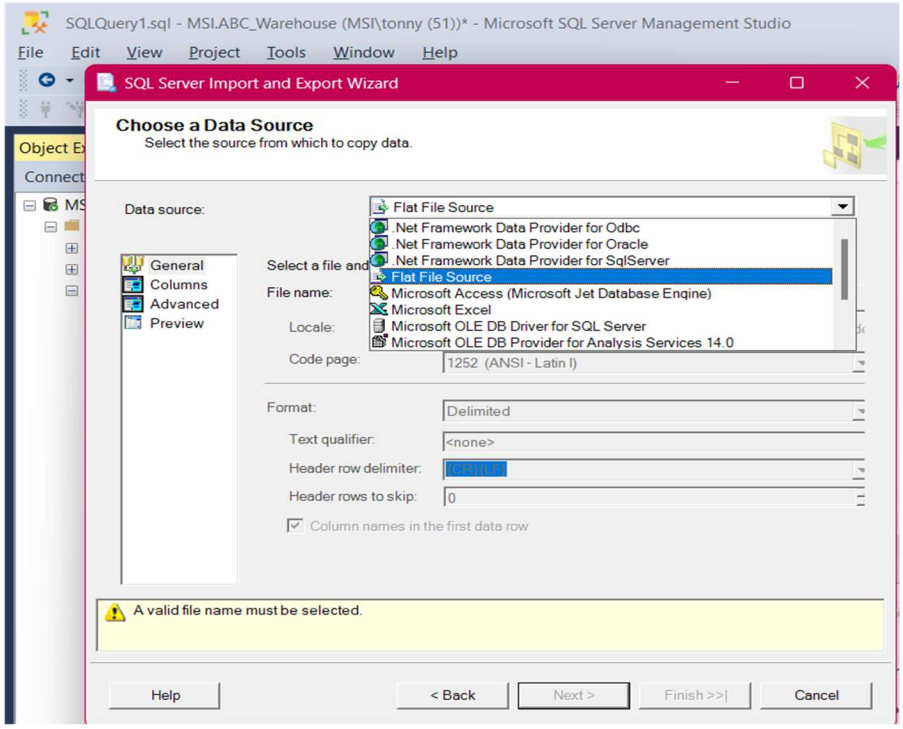
We create a table called dimwarehouse where we store warehouse details like warehouse outlet id, phone no, post code and city. We have assumed some dummy data for populating our dimension table with phone no, post code and city.

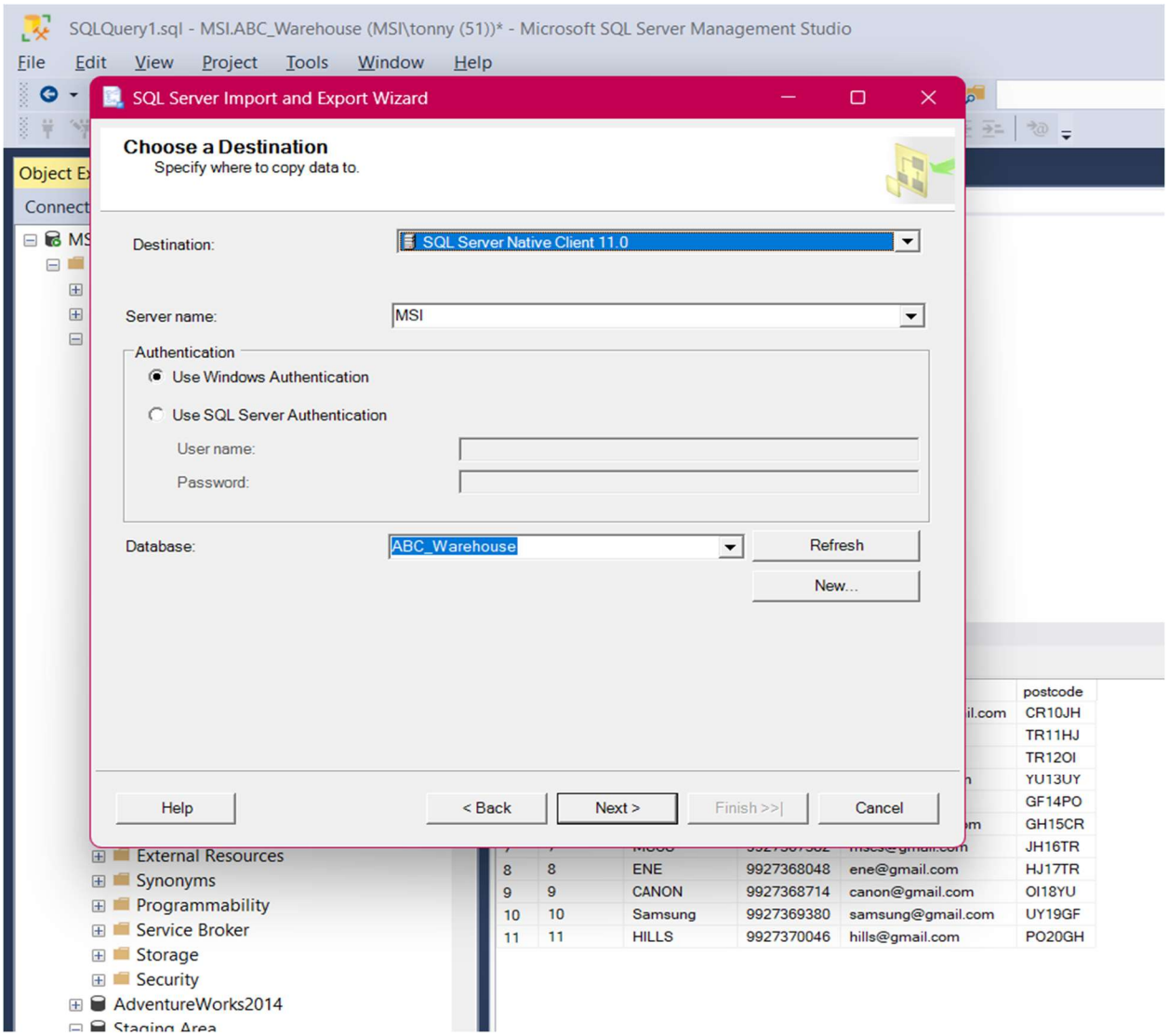
```
CREATE TABLE dimwarehouse (
   warehousekey int IDENTITY(1,1) PRIMARY KEY,
   warehouseoutletid varchar(100),
   whphoneno bigint,
   whpostcode varchar(100),
   whchity varchar(100) );
```

Step 3: Populate dimension tables with data.

Currently only the date dimension has been populated with data, in this stage we use the import export wizard to populate the dimension tables.







We use flat file as the source and SQL server native client as the destination and import product, supplier and warehouse details to their respective dimension tables which we created in step 2.

Below is dummy data used to populate dimsupplier and dimwarehouse

Dummy Supplier Data

supplierid	suppliername	phoneno	email	postcode
1	SENNHEISER	9927363386	sennheiser@gmail.com	CR10JH
2	SONY	9927364052	sony@gmail.com	TR11HJ
3	JVC	9927364718	jvc@gmail.com	TR12OI
4	HAMA	9927365384	hama@gmail.com	YU13UY
5	VIV	9927366050	viv@gmail.com	GF14PO
6	TOSHIBA	9927366716	toshiba@gmail.com	GH15CR
7	MSCS	9927367382	mscs@gmail.com	JH16TR
8	ENE	9927368048	ene@gmail.com	HJ17TR
9	CANON	9927368714	canon@gmail.com	OI18YU
10	Samsung	9927369380	samsung@gmail.com	UY19GF
11	HILLS	9927370046	hills@gmail.com	PO20GH

Dummy Warehouse Data

warehousekey	warehouseoutletid	whphonenumber	whpostcode	whcity
1	ABC Warehouse	9917363364	CRE0AG	Brighton
2	DEF Warehouse	7826227836	VG30GA	Cambridge
3	GHI Wareouse	8477362826	VD92OA	Windsor
4	JKL Warehouse	9173647782	BS50AT	Southwark

Step 4: Creating fact tables.

a. Creating factstockcheck

We create our first fact table factstockcheck which will be used to check current stock level of the product, after creating this fact table we populate it with data in the further steps. Primary key in this fact table is inventory id.

```
create table factstockcheck(
inventoryid int identity(1,1) primary key not null,
sku varchar(100) not null,
supplierid int not null,
datekey int not null,
currentstocklevel int
)
```

b. Creating factsentorders

Here we create our second fact table factsentorders we create a primary key sentorderid. This fact table is used to record purchase order codes of sent products along with dates and ordered quantities.

```
create table factsentorders (
sentorderid int identity (1.1) primary key not null
purchaseordercode varchar(100),
sku varchar(100) not null,
supplierid int not null,
sentdatekey int not null,
orderedqty int not null
)
```

c. Creating factreceivedorders

Here we create our final fact table which is used to store purchase order codes regarding orders along with sent and received dates and ordered quantities.

```
create table factreceivedorders(
receivedorderid int identity(1,1) primary key not null,
sku  varchar(100) not null,
purchaseordercode  varchar(100) not null,
supplierid int not null,
warehousekey int not null,
```

```
sentdatekey int not null,
receiveddatekey int not null,
orderedqty int not null,
receivedqty int not null
);
```

Step 5: Creating Staging Area and lookup tables.

In order to populate the fact tables we create a staging area

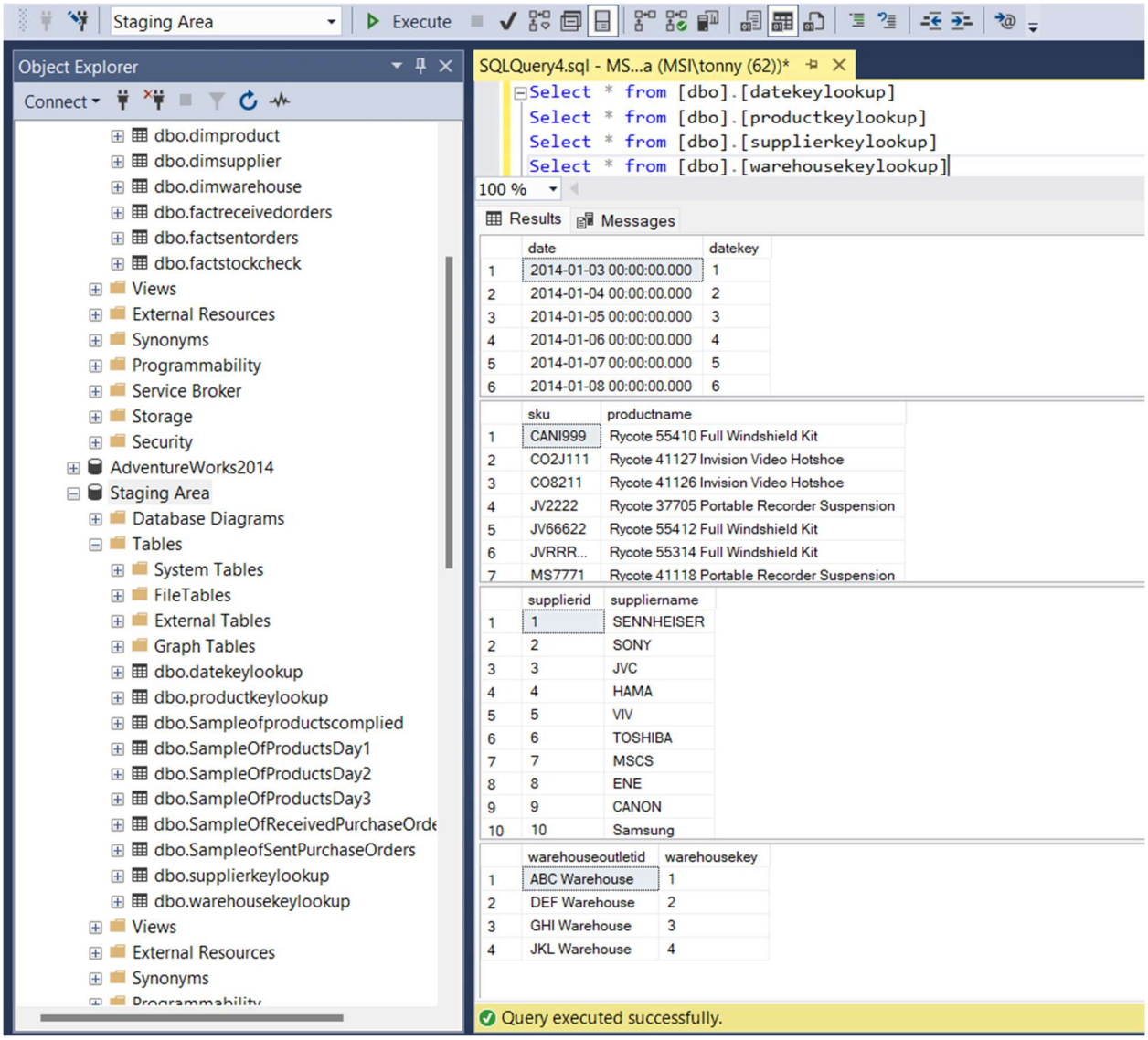
```
CREATE DATABASE Staging Area
```

Now we create lookup tables in the staging area.

```
use[Staging Area];

create table datekeylookup(
[date] date not null,
datekey int primary key not null);

create table productkeylookup(
sku varchar(100) primary key not null,
productname varchar(100) not null);

create table supplierkeylookup(
suppliername varchar(100) primary key not null,
supplierid int not null);

create table warehousekeylookup(
warehouseoutletid varchar(100) not null,
warehousekey int primary key not null ) ;
```

Step 6: Populating lookup tables using import export Wizard.

In this step we populate the lookup table with information from the respective dimension tables, we use sql server native client > ABC_Warehouse as the source and sql server native client > Staging Area as the destination.

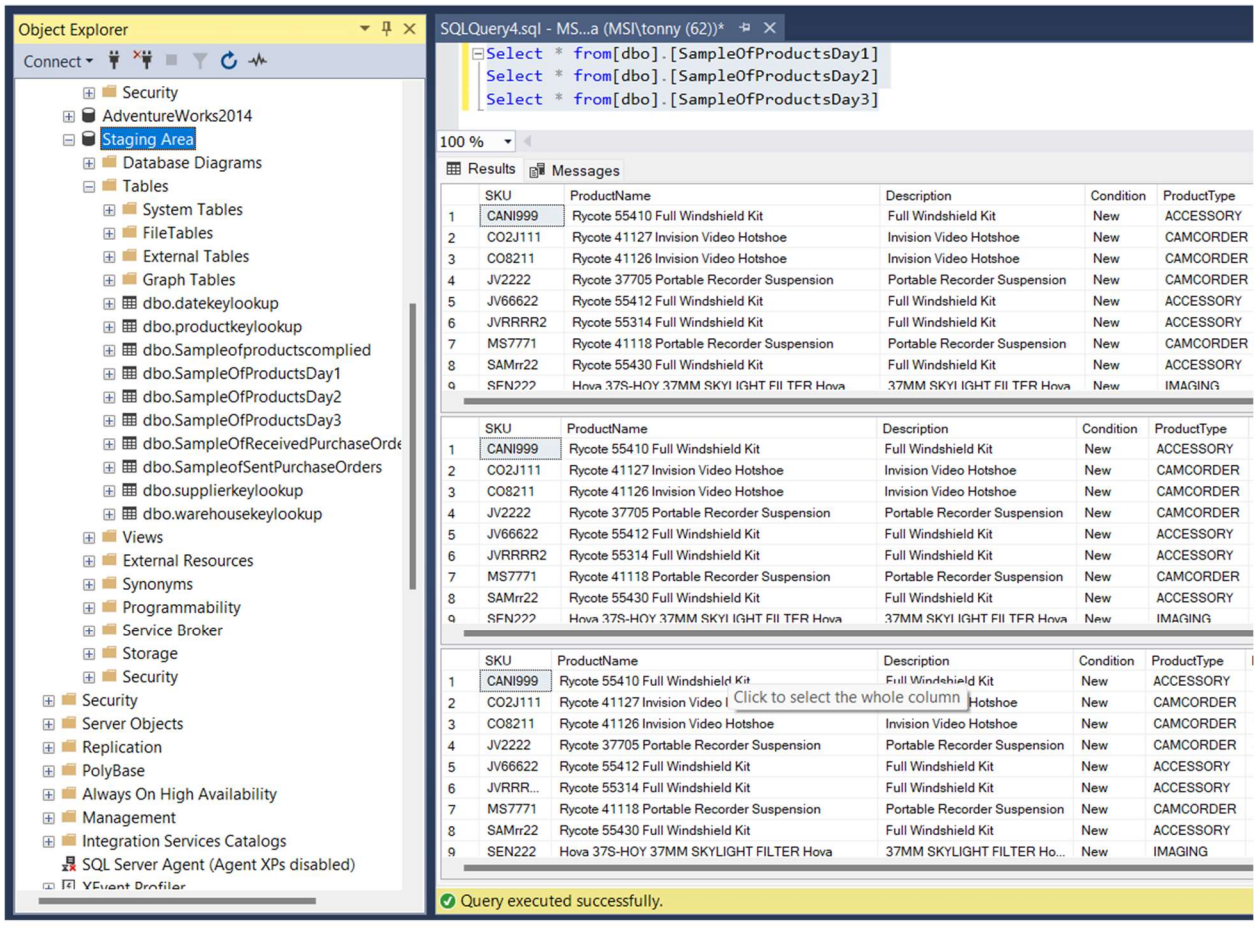


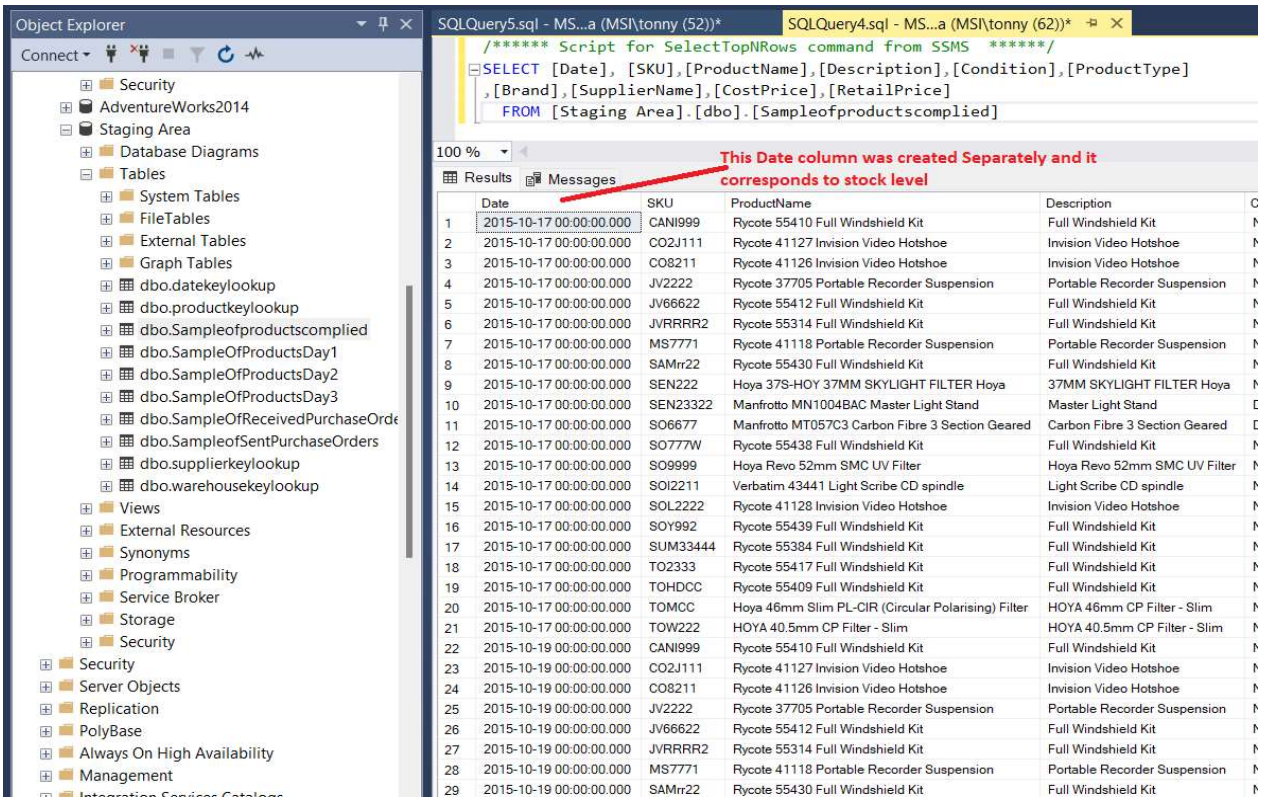
Step 7: Importing necessary flat files to staging area to import to fact table.

In this step we import the given data files ‘Sample of Products1’,‘Sample of Products2’,‘Sample of Products3’ ,’SampleofSentpurchaseorders’ and ‘SampleofReceivedpurchaseorders’.

Since sample of products does not have a date column corresponding to stock level, we create a Date table which corresponds to stock level of that particular date in a new table called sampleofproductscompiled and compile data from all 3 days into one table.

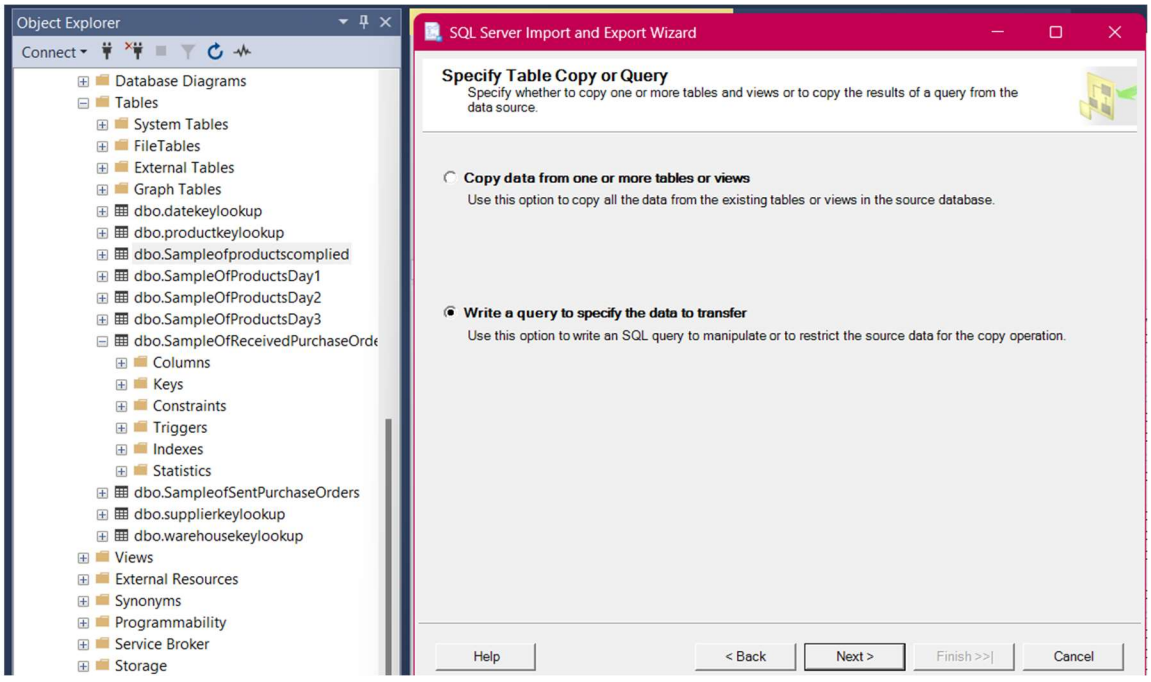
We create a table sample of products compiled table with date column and import Sample of Products 1 data into it, now we use the alter function to replace the date null values, now we repeat this process for Sample of Products 2 and 3.





Step 8: Populating fact table using query.

In this stage we use the import export wizard to populate the fact table, we use SQL server native client > Staging Area as the source and SQL server native client > ABC_Warehouse as the destination.



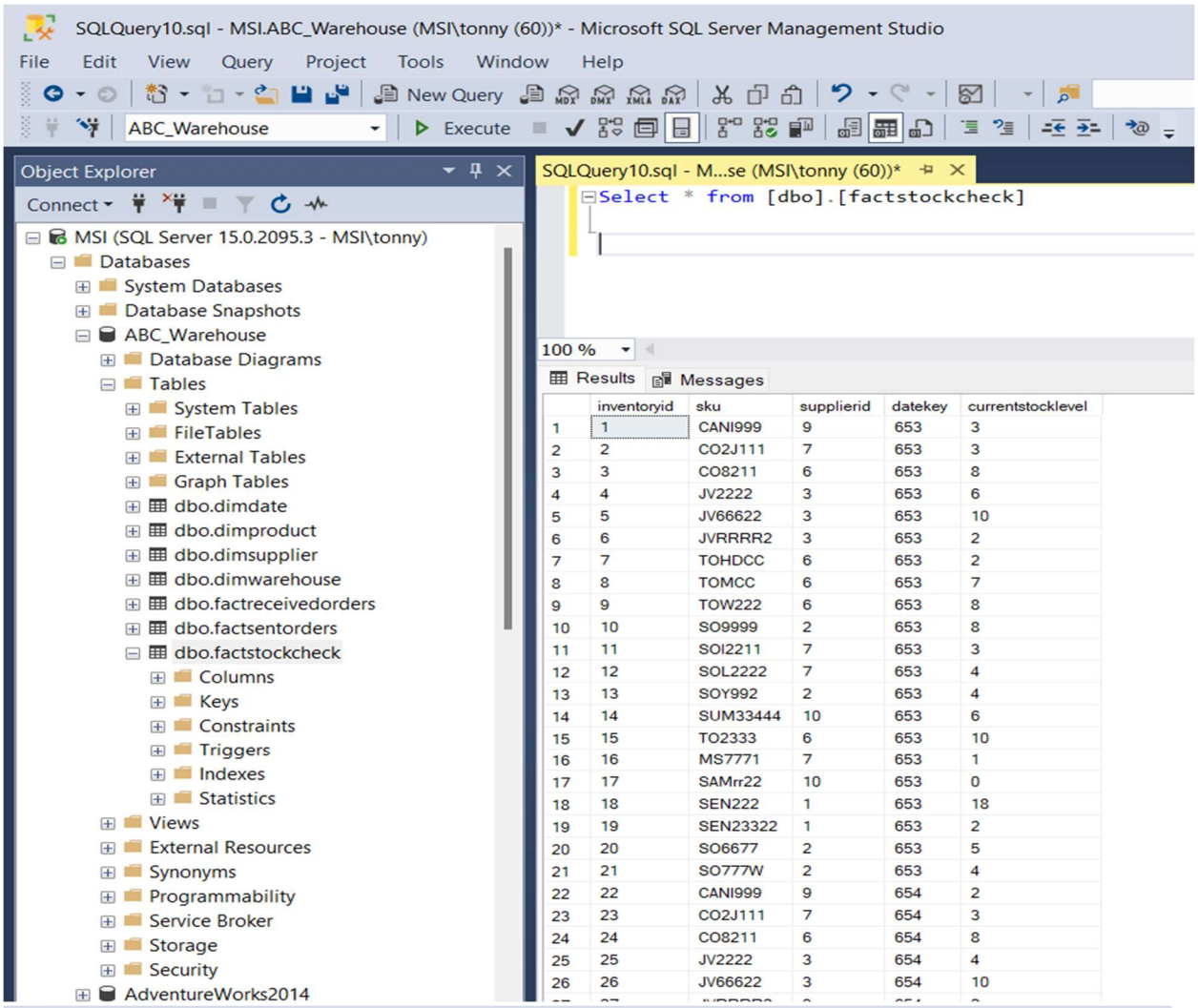
We use the below 3 queries to populate their respective fact tables.

```
SELECT  SSPO.productsku,
SUP.supplierid, DLT.datekey SSPO.currentstocklevel
FROM [dbo].[Sampleofproductscomplied] AS SSPO
INNER JOIN datekeylookup
 AS DLT ON DLT.date = SSPO.sentdate
INNER JOIN supplierkeylookup
 AS SUP ON SSPO.suppliername = SUP.suppliername;
```

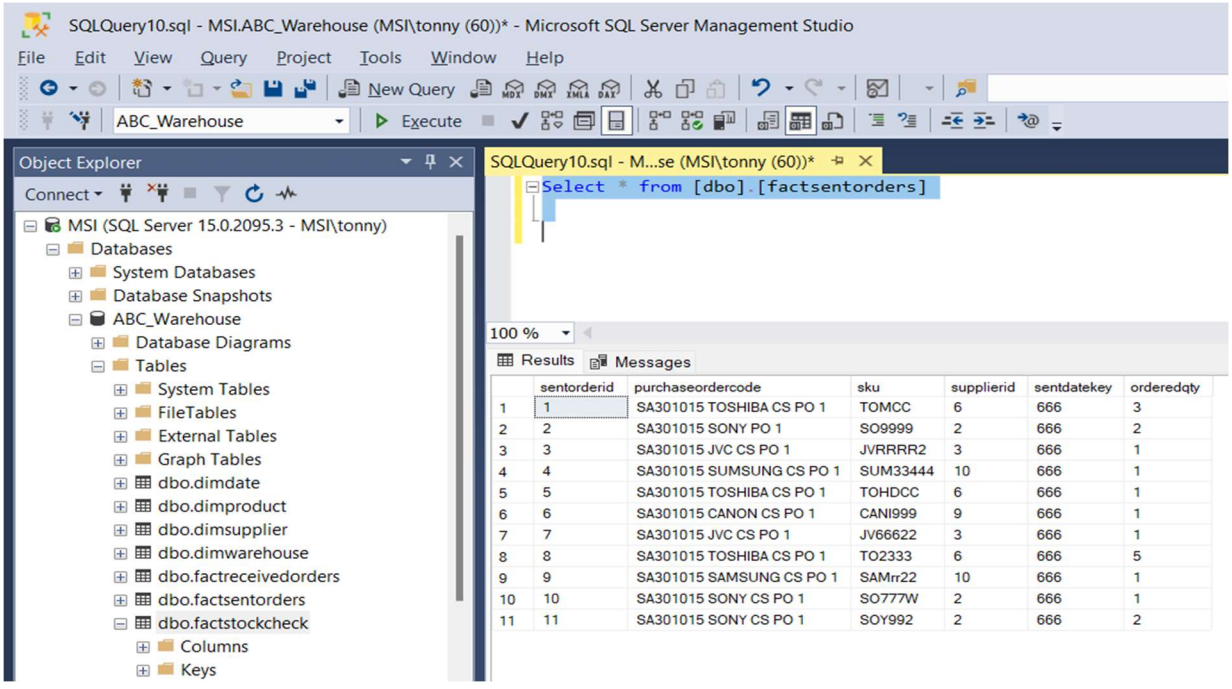
```
SELECT SSPO.purchaseordercode, SSPO.productsku,
SUP.supplierid, DLT.datekey AS sentdate, SSPO.orderedqty
FROM [dbo].[SampleofSentPurchaseOrders] AS SSPO
INNER JOIN datekeylookup
 AS DLT ON DLT.date = SSPO.sentdate
INNER JOIN supplierkeylookup
 AS SUP ON SSPO.suppliername = SUP.suppliername;
```

```
SELECT SRP.productsku as sku, SUP.supplierid, DLT1.datekey AS sentdate,
 DLT2.datekey AS receiveddate, SRP.orderedqty, SRP.receivedqty,
WH.warehousekey AS DestinationOutletID
FROM [dbo].[SampleOfReceivedPurchaseOrders] AS SRP
INNER JOIN supplierkeylookup AS SUP ON SUP.suppliername = SRP.suppliername
INNER JOIN datekeylookup AS DLT1 ON DLT1.date = SRP.sentdate
INNER JOIN datekeylookup AS DLT2 ON DLT2.date = SRP.receiveddate
INNER JOIN warehousekeylookup AS WH ON
WH.warehouseoutletid=SRP.destinationoutletid
```

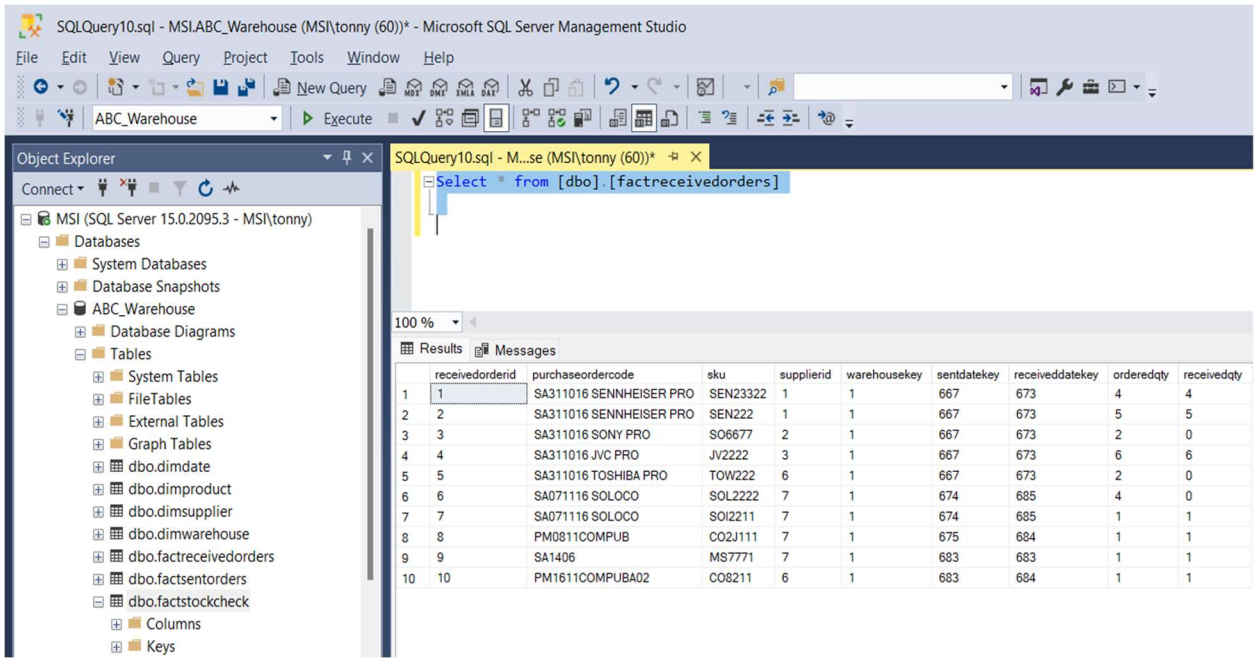
After successfully executing above queries we have successfully populated all 3 fact tables with data.



	inventoryid	sku	supplierid	datekey	currentstocklevel
1	1	CANI999	9	653	3
2	2	CO2J111	7	653	3
3	3	CO8211	6	653	8
4	4	JV2222	3	653	6
5	5	JV66622	3	653	10
6	6	JVRRRR2	3	653	2
7	7	TOHDCC	6	653	2
8	8	TOMCC	6	653	7
9	9	TOW222	6	653	8
10	10	SO9999	2	653	8
11	11	SOI2211	7	653	3
12	12	SOL2222	7	653	4
13	13	SOY992	2	653	4
14	14	SUM33444	10	653	6
15	15	TO2333	6	653	10
16	16	MS7771	7	653	1
17	17	SAMrr22	10	653	0
18	18	SEN222	1	653	18
19	19	SEN23322	1	653	2
20	20	SO6677	2	653	5
21	21	SO777W	2	653	4
22	22	CANI999	9	654	2
23	23	CO2J111	7	654	3
24	24	CO8211	6	654	8
25	25	JV2222	3	654	4
26	26	JV66622	3	654	10



	sentorderid	purchaseordercode	sku	supplierid	sentdatekey	orderedqty
1	1	SA301015 TOSHIBA CS PO 1	TOMCC	6	666	3
2	2	SA301015 SONY PO 1	SO9999	2	666	2
3	3	SA301015 JVC CS PO 1	JVRRRR2	3	666	1
4	4	SA301015 SUMSUNG CS PO 1	SUM33444	10	666	1
5	5	SA301015 TOSHIBA CS PO 1	TOHDCC	6	666	1
6	6	SA301015 CANON CS PO 1	CANI999	9	666	1
7	7	SA301015 JVC CS PO 1	JV66622	3	666	1
8	8	SA301015 TOSHIBA CS PO 1	TO2333	6	666	5
9	9	SA301015 SAMSUNG CS PO 1	SAMrr22	10	666	1
10	10	SA301015 SONY CS PO 1	SO777W	2	666	1
11	11	SA301015 SONY CS PO 1	SOY992	2	666	2



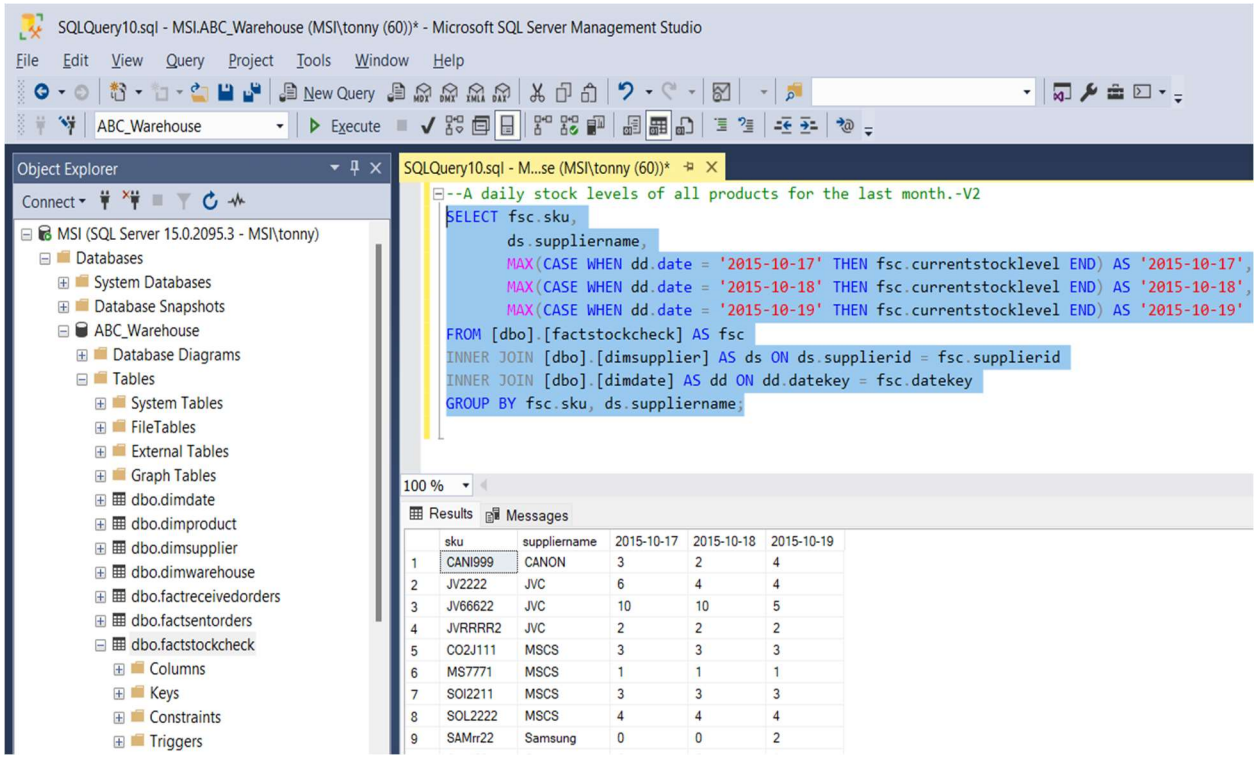
	receivedorderid	purchaseordercode	sku	supplierid	warehousekey	sentdatekey	receiveddatekey	orderedqty	receivedqty
1	1	SA311016 SENNHEISER PRO	SEN23322	1	1	667	673	4	4
2	2	SA311016 SENNHEISER PRO	SEN222	1	1	667	673	5	5
3	3	SA311016 SONY PRO	SO6677	2	1	667	673	2	0
4	4	SA311016 JVC PRO	JV2222	3	1	667	673	6	6
5	5	SA311016 TOSHIBA PRO	TOW222	6	1	667	673	2	0
6	6	SA071116 SOLOCO	SOL2222	7	1	674	685	4	0
7	7	SA071116 SOLOCO	SOI2211	7	1	674	685	1	1
8	8	PM0811COMPUB	CO2J111	7	1	675	684	1	1
9	9	SA1406	MS7771	7	1	683	683	1	1
10	10	PM1611COMPUBA02	CO8211	6	1	683	684	1	1

Output Queries to generate reports as per business requirement.

1.Daily stock levels of all products for the last month

The below query retrieves data from three tables: "factstockcheck", "dimsupplier", and "dimdate". The query uses an INNER JOIN to combine the tables on specific columns: "factstockcheck" and "dimsupplier" are joined on "supplierid", and "factstockcheck" and "dimdate" are joined on "datekey". The final result is a table showing for each sku, suppliername, the maximum current stock level on the dates '2015-10-17', '2015-10-18' and '2015-10-19' respectively.

```
SELECT fsc.sku,
    ds.suppliername,
    MAX(CASE WHEN dd.date = '2015-10-17' THEN fsc.currentstocklevel END) AS '2015-10-17',
    MAX(CASE WHEN dd.date = '2015-10-18' THEN fsc.currentstocklevel END) AS '2015-10-18',
    MAX(CASE WHEN dd.date = '2015-10-19' THEN fsc.currentstocklevel END) AS '2015-10-19'
FROM [dbo].[factstockcheck] AS fsc
INNER JOIN [dbo].[dimsupplier] AS ds ON ds.supplierid = fsc.supplierid
INNER JOIN [dbo].[dimdate] AS dd ON dd.datekey = fsc.datekey
GROUP BY fsc.sku, ds.suppliername;
```

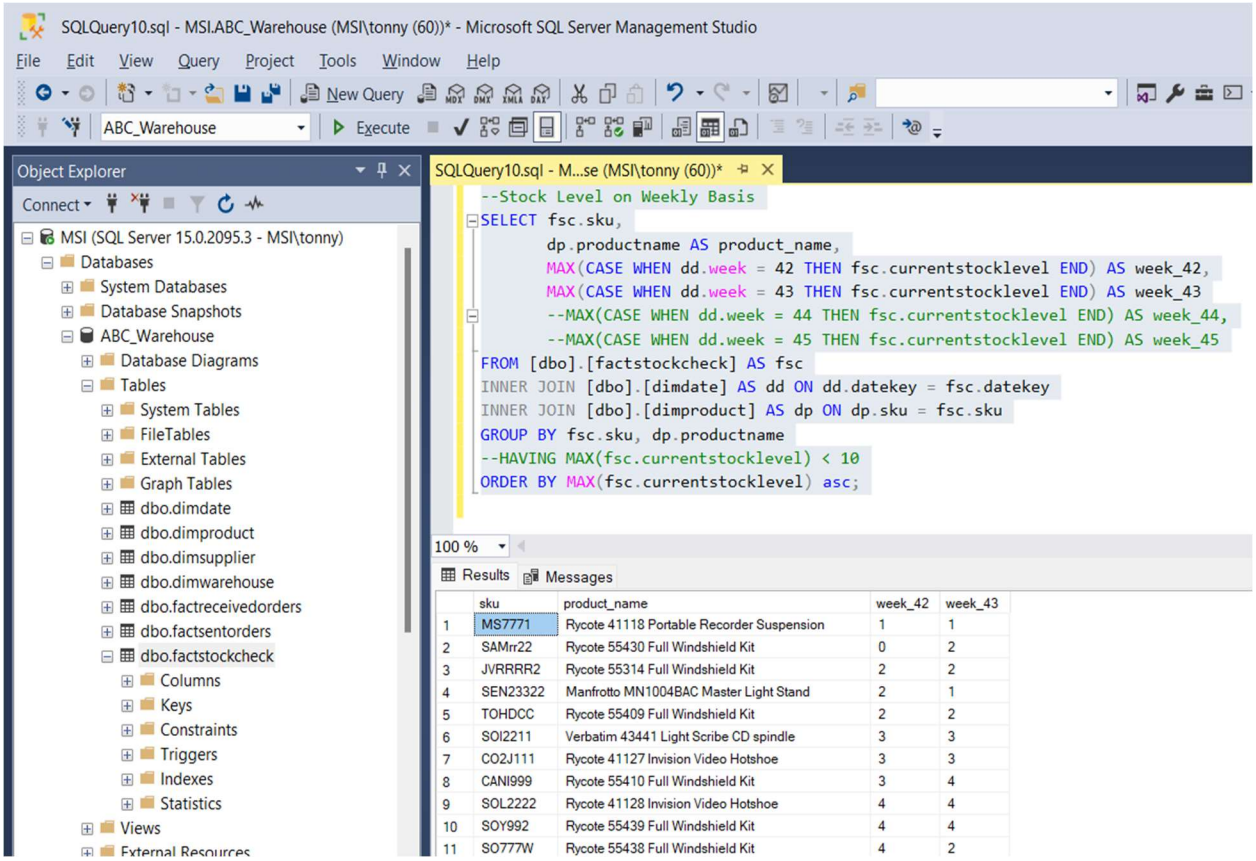


	sku	suppliername	2015-10-17	2015-10-18	2015-10-19
1	CANI999	CANON	3	2	4
2	JV2222	JVC	6	4	4
3	JV66622	JVC	10	10	5
4	JVRRRR2	JVC	2	2	2
5	CO2J111	MSCS	3	3	3
6	MS7771	MSCS	1	1	1
7	SOI2211	MSCS	3	3	3
8	SOL2222	MSCS	4	4	4
9	SAMrr22	Samsung	0	0	2

2.Stock Level on Weekly Basis

The below query joins "factstockcheck", "dimdate", and "dimproduct" tables using two INNER JOINs, selects specific columns, and uses MAX function with CASE statement to retrieve maximum currentstocklevel for specific weeks, grouped by "sku" and "productname" to show the maximum stock level for weeks 42 and 43

```
SELECT fsc.sku,
	dp.productname AS product_name,
	MAX(CASE WHEN dd.week = 42 THEN fsc.currentstocklevel END) AS week_42,
	MAX(CASE WHEN dd.week = 43 THEN fsc.currentstocklevel END) AS week_43
	--MAX(CASE WHEN dd.week = 44 THEN fsc.currentstocklevel END) AS week_44,
	--MAX(CASE WHEN dd.week = 45 THEN fsc.currentstocklevel END) AS week_45
FROM [dbo].[factstockcheck] AS fsc
INNER JOIN [dbo].[dimdate] AS dd ON dd.datekey = fsc.datekey
INNER JOIN [dbo].[dimproduct] AS dp ON dp.sku = fsc.sku
GROUP BY fsc.sku, dp.productname
--HAVING MAX(fsc.currentstocklevel) < 10
ORDER BY MAX(fsc.currentstocklevel) asc;
```



	sku	product_name	week_42	week_43
1	MS7771	Rycote 41118 Portable Recorder Suspension	1	1
2	SAMrr22	Rycote 55430 Full Windshield Kit	0	2
3	JVRRRR2	Rycote 55314 Full Windshield Kit	2	2
4	SEN23322	Manfrotto MN1004BAC Master Light Stand	2	1
5	TOHDCC	Rycote 55409 Full Windshield Kit	2	2
6	SOI2211	Verbatim 43441 Light Scribe CD spindle	3	3
7	CO2J111	Rycote 41127 Invision Video Hotshoe	3	3
8	CANI999	Rycote 55410 Full Windshield Kit	3	4
9	SOL2222	Rycote 41128 Invision Video Hotshoe	4	4
10	SOY992	Rycote 55439 Full Windshield Kit	4	4
11	SO777W	Rycote 55438 Full Windshield Kit	4	2

3.Analysing stock levels by brand or product type or supplier

Following SQL queries are used to retrieve data from the "factstockcheck" table and join it with "dimproduct" or "dimsupplier" table.

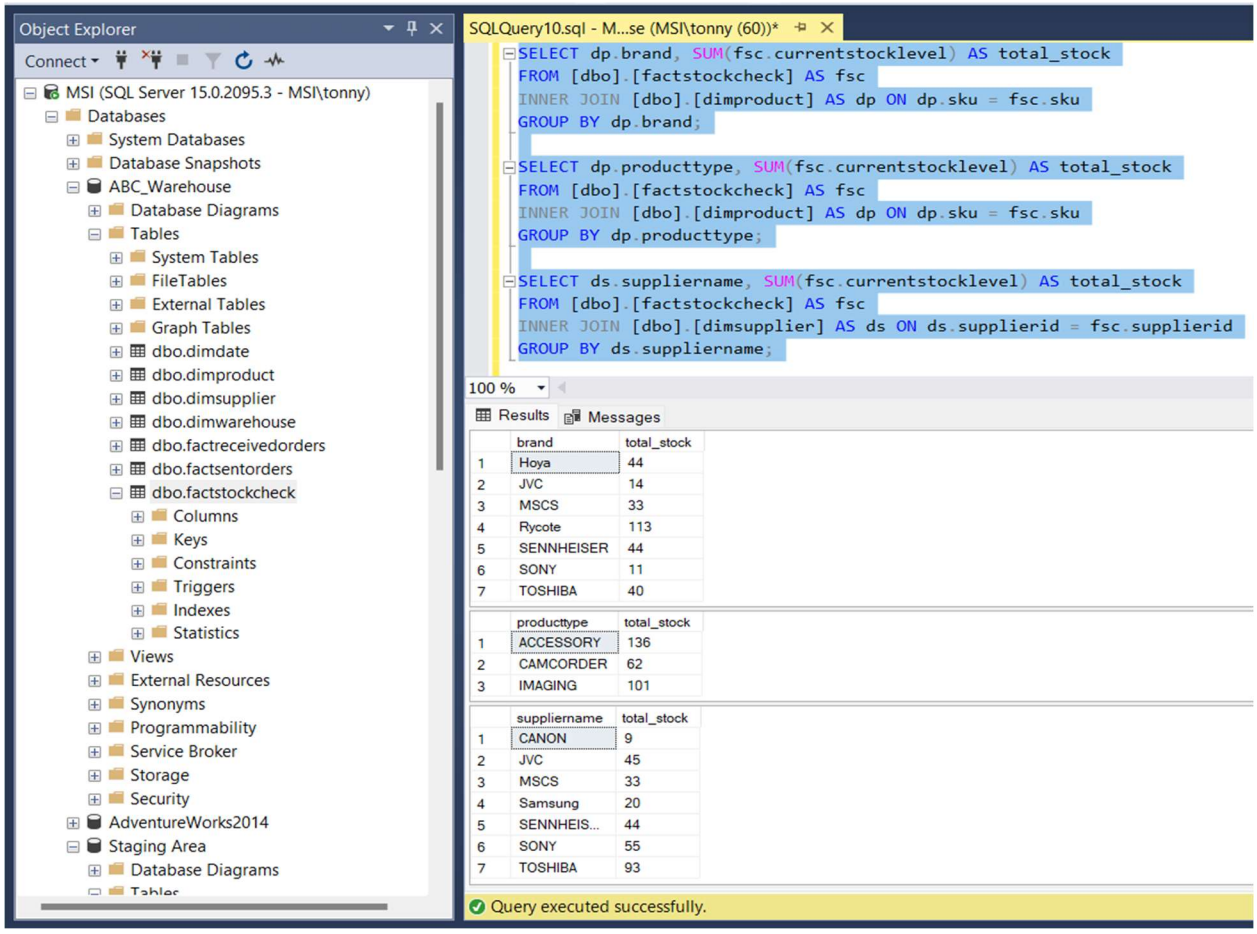
The first query uses an INNER JOIN to combine the "factstockcheck" and "dimproduct" tables on the "sku" column. The query then selects the "brand" and "currentstocklevel" columns from the "dimproduct" table and uses the SUM function to calculate the total stock level for each brand. The result is grouped by the "brand" column.

The second query is similar to the first query, but it uses the "producttype" column instead of the "brand" column to group the results, and it also calculates the total stock level for each producttype.

The third query uses an INNER JOIN to combine the "factstockcheck" and "dimsupplier" tables on the "supplierid" column. The query then selects the "suppliername" and "currentstocklevel" columns from the "dimsupplier" table, and uses the SUM function to calculate the total stock level for each suppliername. The result is grouped by the "suppliername" column.

All three queries return a table showing the total stock level grouped by brand, producttype, or suppliername respectively.

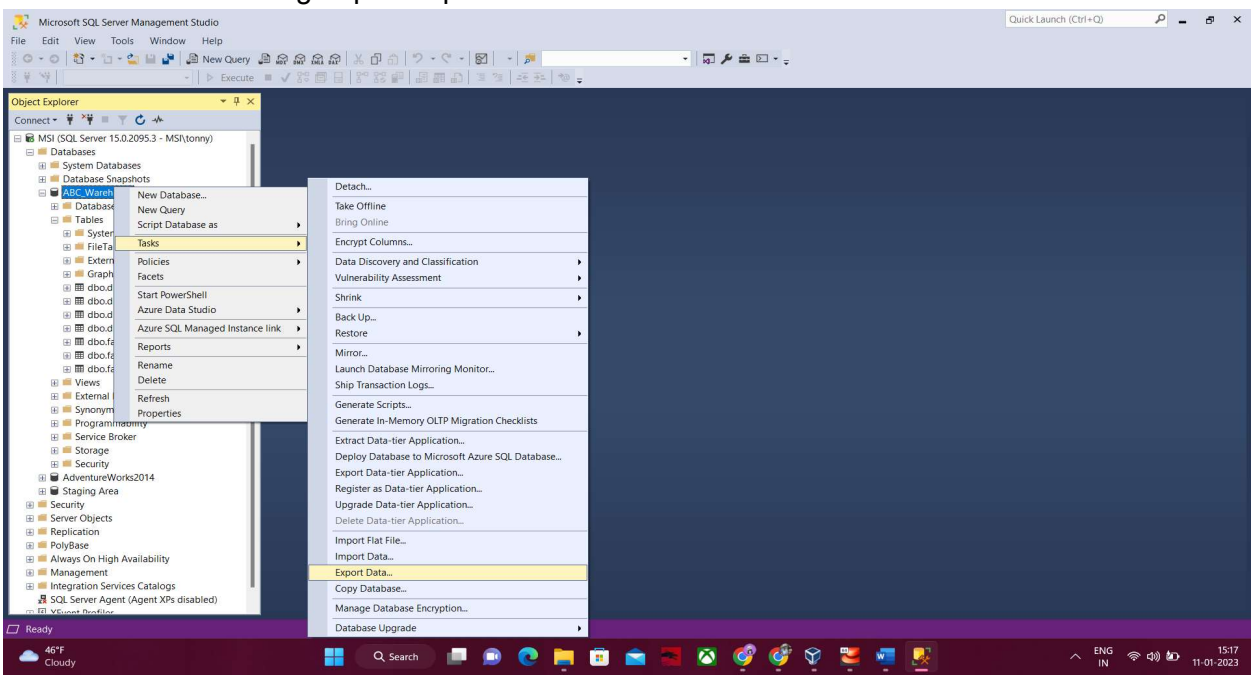
```
SELECT dp.brand, SUM(fsc.currentstocklevel) AS total_stock
FROM [dbo].[factstockcheck] AS fsc
INNER JOIN [dbo].[dimproduct] AS dp ON dp.sku = fsc.sku
GROUP BY dp.brand;

SELECT dp.producttype, SUM(fsc.currentstocklevel) AS total_stock
FROM [dbo].[factstockcheck] AS fsc
INNER JOIN [dbo].[dimproduct] AS dp ON dp.sku = fsc.sku
GROUP BY dp.producttype;

SELECT ds.suppliername, SUM(fsc.currentstocklevel) AS total_stock
FROM [dbo].[factstockcheck] AS fsc
INNER JOIN [dbo].[dimsupplier] AS ds ON ds.supplierid = fsc.supplierid
GROUP BY ds.suppliername;
```

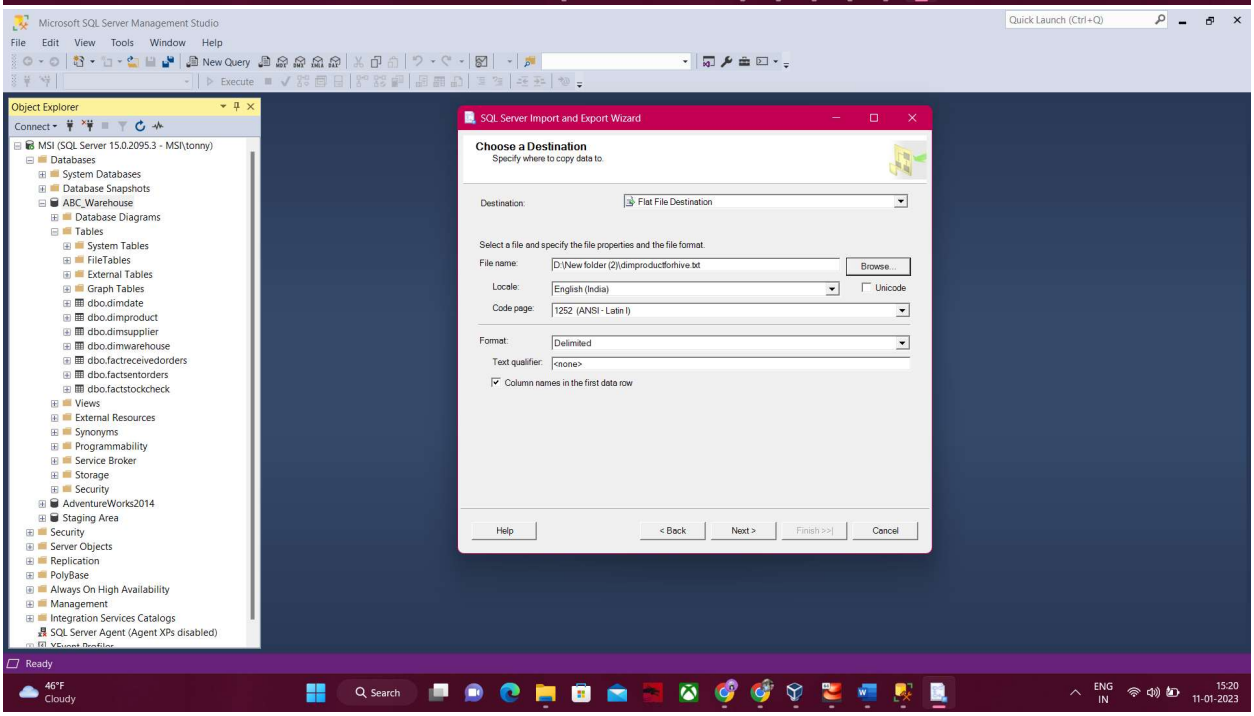

Object Explorer
Connect
MSI (SQL Server 15.0.2095.3 - MSI\tonny)
Databases
System Databases
Database Snapshots
ABC_Warehouse
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.dimdate
dbo.dimproduct
dbo.dimsupplier
dbo.dimwarehouse
dbo.factreceivedorders
dbo.factsentorders
dbo.factstockcheck
Columns
Keys
Constraints
Triggers
Indexes
Statistics
Views
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
AdventureWorks2014
Staging Area
Database Diagrams
SQLQuery10.sql - M...se (MSI\tonny (60))*
SELECT dp.brand, SUM(fsc.currentstocklevel) AS total_stock
FROM [dbo].[factstockcheck] AS fsc
INNER JOIN [dbo].[dimproduct] AS dp ON dp.sku = fsc.sku
GROUP BY dp.brand;
SELECT dp.producttype, SUM(fsc.currentstocklevel) AS total_stock
FROM [dbo].[factstockcheck] AS fsc
INNER JOIN [dbo].[dimproduct] AS dp ON dp.sku = fsc.sku
GROUP BY dp.producttype;
SELECT ds.suppliername, SUM(fsc.currentstocklevel) AS total_stock
FROM [dbo].[factstockcheck] AS fsc
INNER JOIN [dbo].[dimsupplier] AS ds ON ds.supplierid = fsc.supplierid
GROUP BY ds.suppliername;
100 %
Results
Messages
brand total_stock
1 Hoya 44
2 JVC 14
3 MSCS 33
4 Rycote 113
5 SENNHEISER 44
6 SONY 11
7 TOSHIBA 40
producttype total_stock
1 ACCESSORY 136
2 CAMCORDER 62
3 IMAGING 101
suppliername total_stock
1 CANON 9
2 JVC 45
3 MSCS 33
4 Samsung 20
5 SENNHEIS... 44
6 SONY 55
7 TOSHIBA 93
Query executed successfully.

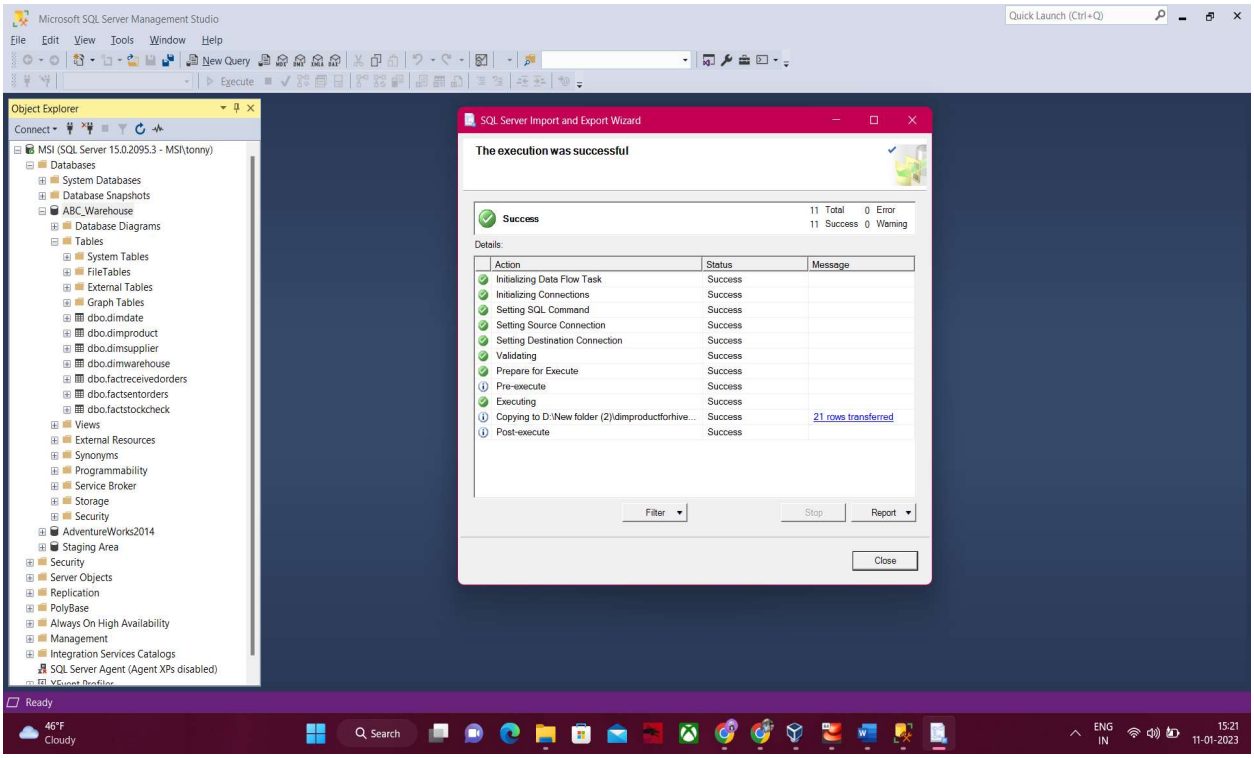
11.Implementation Process in Hive

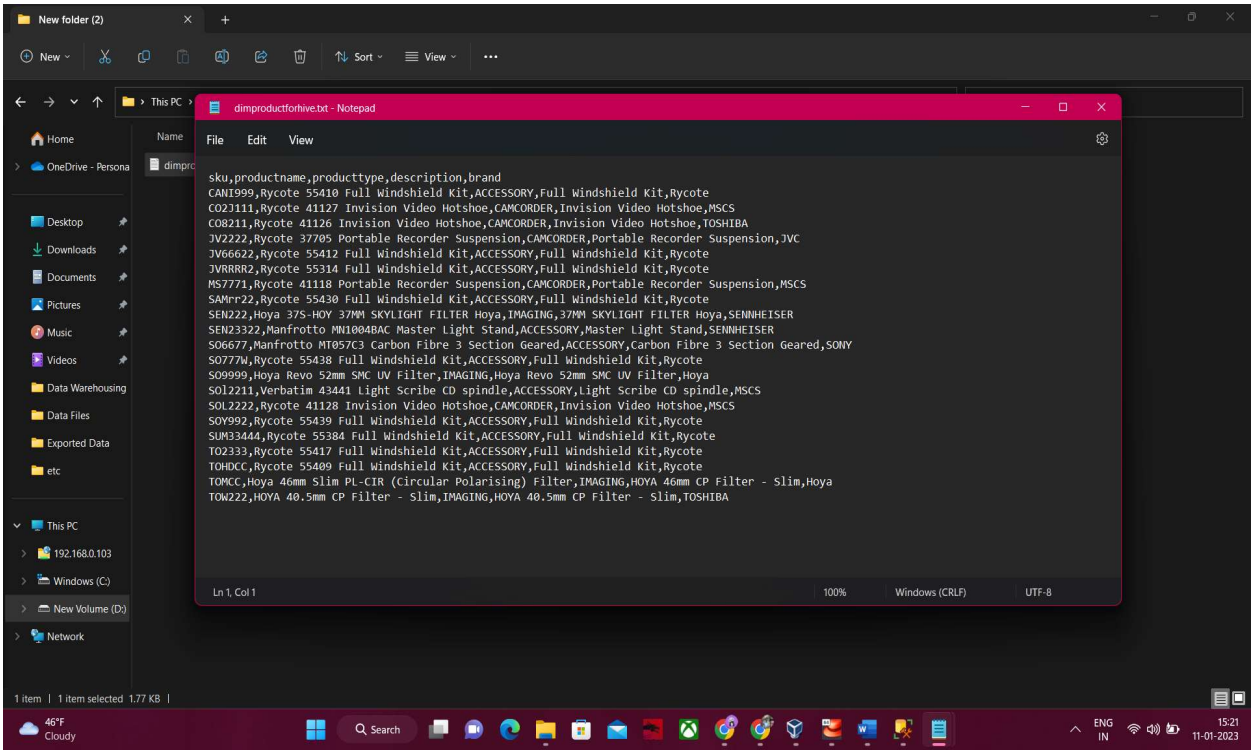
We transfer data from our SSMS database (**ABC_Warehouse**) to an external location by utilizing the import export wizard tool and saving the dimension or fact tables as a text file. An alternate method is to connect to the SQL server in Excel and use the "Get data" option to achieve the same result.

Method 1: Using Import export Wizard.

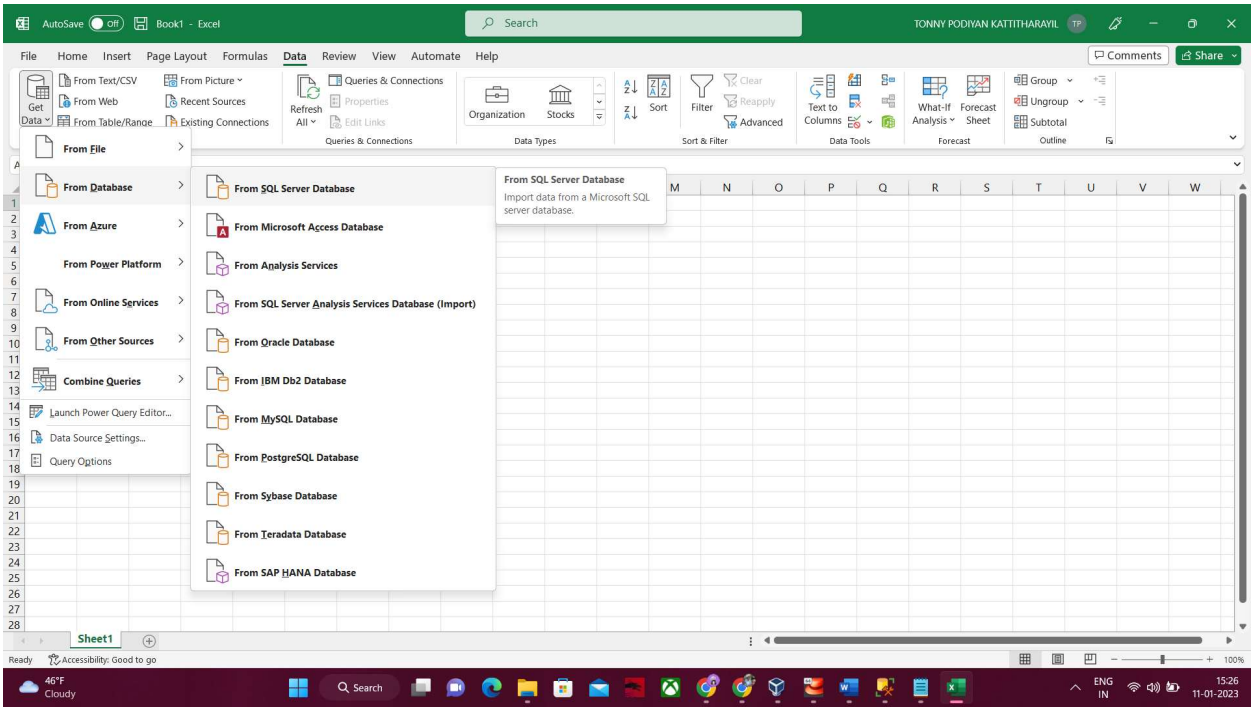


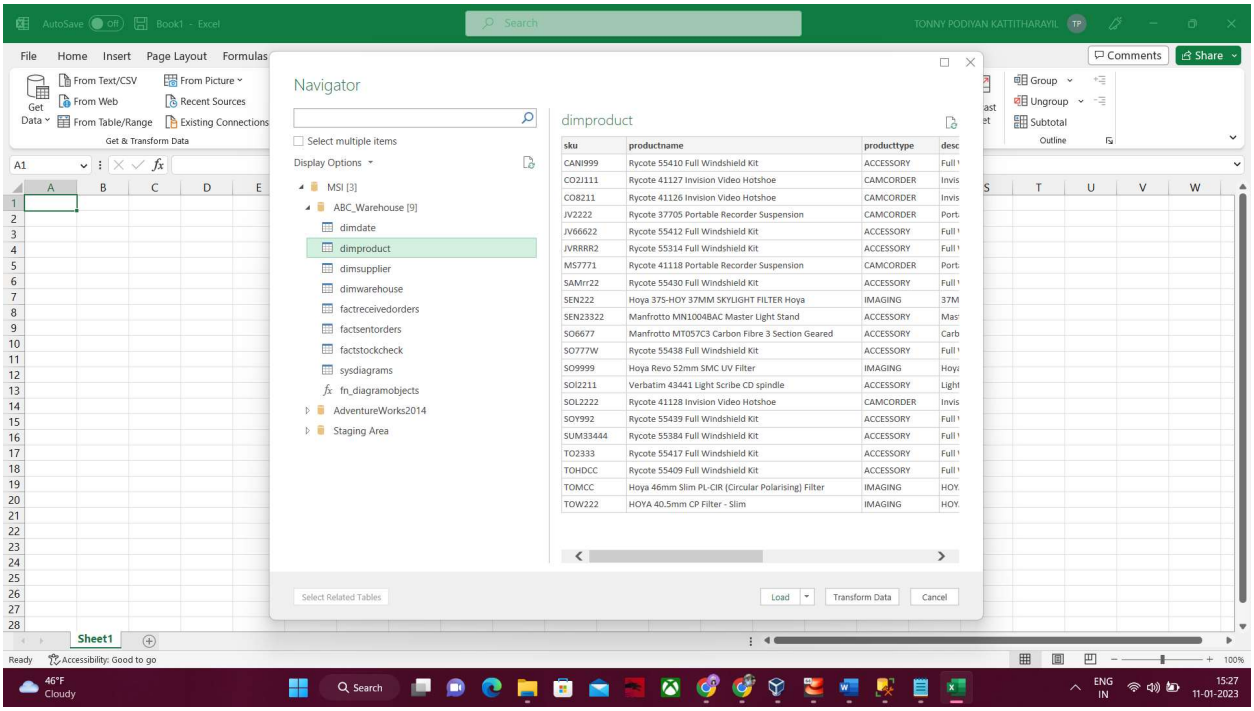


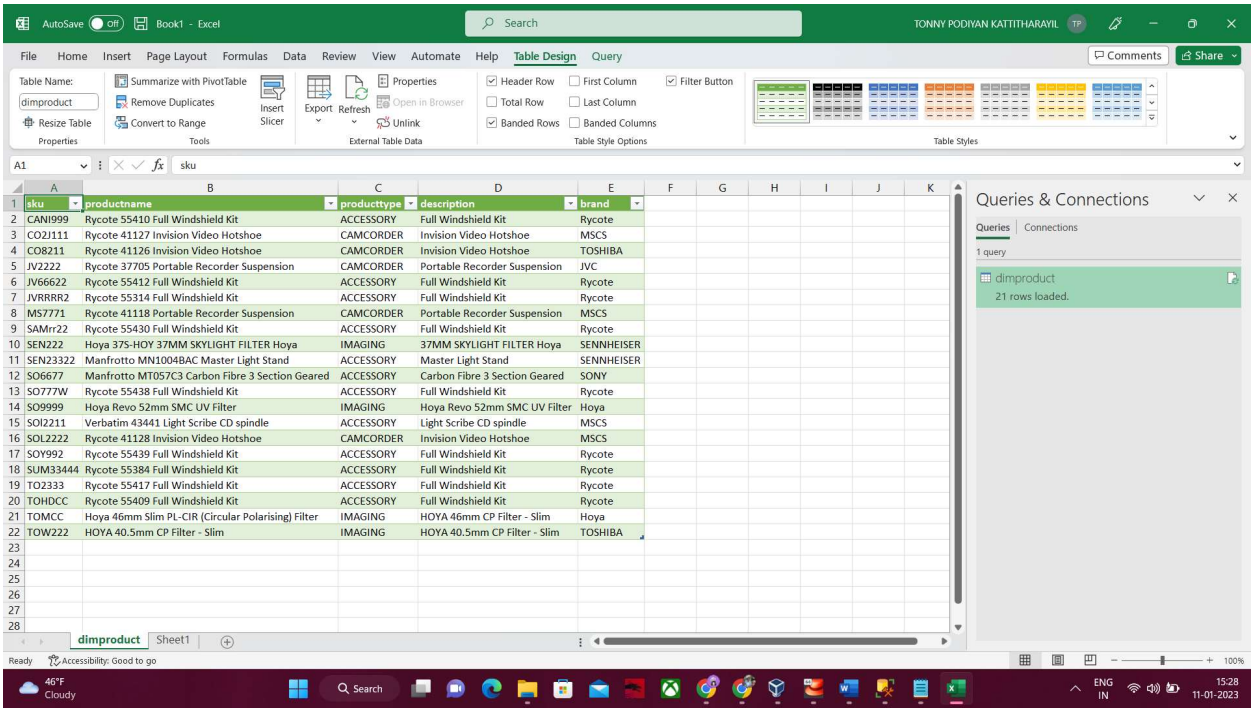

Microsoft SQL Server Management Studio
File Edit View Tools Window Help
New Query
Execute
Object Explorer
Connect
MSI (SQL Server 15.0.2095.3 - MSI\tonny)
Databases
System Databases
Database Snapshots
ABC_Warehouse
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.dimdate
dbo.dimproduct
dbo.dimsupplier
dbo.dimwarehouse
dbo.factreceivedorders
dbo.factsentorders
dbo.factstockcheck
Views
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
AdventureWorks2014
Staging Area
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabled)
SQL Server Import and Export Wizard
The execution was successful
Success
11 Total 0 Error
11 Success 0 Warning
Details:
Action
Status
Message
Initializing Data Flow Task
Initializing Connections
Setting SQL Command
Setting Source Connection
Setting Destination Connection
Validating
Prepare for Execute
Pre-execute
Executing
Copying to D:\New folder (2)\dimproductforhive...
21 rows transferred
Post-execute
Filter
Stop
Report
Close
Ready
46°F
Cloudy
Search
ENG
IN
15:21
11-01-2023


New folder (2)
New
Sort
View
This PC
dimproductforhive.txt - Notepad
File Edit View
sku,productname,producttype,description,brand
CANI999,Rycote 55410 Full Windshield Kit,ACCESSORY,Full Windshield Kit,Rycote
CO2J111,Rycote 41127 Invision Video Hotshoe,CAMCORDER,Invision Video Hotshoe,MSCS
CO8211,Rycote 41126 Invision Video Hotshoe,CAMCORDER,Invision Video Hotshoe,TOSHIBA
JV2222,Rycote 37705 Portable Recorder Suspension,CAMCORDER,Portable Recorder Suspension,JVC
JV66622,Rycote 55412 Full Windshield Kit,ACCESSORY,Full Windshield Kit,Rycote
JVRRRR2,Rycote 55314 Full Windshield Kit,ACCESSORY,Full Windshield Kit,Rycote
MS7771,Rycote 41118 Portable Recorder Suspension,CAMCORDER,Portable Recorder Suspension,MSCS
SAMrr22,Rycote 55430 Full Windshield Kit,ACCESSORY,Full Windshield Kit,Rycote
SEN222,Hoya 37S-HOY 37MM SKYLIGHT FILTER Hoya,IMAGING,37MM SKYLIGHT FILTER Hoya,SENNHEISER
SEN23322,Manfrotto MN1004BAC Master Light Stand,ACCESSORY,Master Light Stand,SENNHEISER
SO6677,Manfrotto MT057C3 Carbon Fibre 3 Section Geared,ACCESSORY,Carbon Fibre 3 Section Geared,SONY
SO777W,Rycote 55438 Full Windshield Kit,ACCESSORY,Full Windshield Kit,Rycote
SO9999,Hoya Revo 52mm SMC UV Filter,IMAGING,Hoya Revo 52mm SMC UV Filter,Hoya
SOl2211,Verbatim 43441 Light Scribe CD spindle,ACCESSORY,Light Scribe CD spindle,MSCS
SOL2222,Rycote 41128 Invision Video Hotshoe,CAMCORDER,Invision Video Hotshoe,MSCS
SOY992,Rycote 55439 Full Windshield Kit,ACCESSORY,Full Windshield Kit,Rycote
SUM33444,Rycote 55384 Full Windshield Kit,ACCESSORY,Full Windshield Kit,Rycote
TO2333,Rycote 55417 Full Windshield Kit,ACCESSORY,Full Windshield Kit,Rycote
TOHDCC,Rycote 55409 Full Windshield Kit,ACCESSORY,Full Windshield Kit,Rycote
TOMCC,Hoya 46mm Slim PL-CIR (Circular Polarising) Filter,IMAGING,HOYA 46mm CP Filter - Slim,Hoya
TOW222,HOYA 40.5mm CP Filter - Slim,IMAGING,HOYA 40.5mm CP Filter - Slim,TOSHIBA
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8
Home
OneDrive - Persona
Desktop
Downloads
Documents
Pictures
Music
Videos
Data Warehousing
Data Files
Exported Data
etc
This PC
192.168.0.103
Windows (C:)
New Volume (D:)
Network
1 item | 1 item selected 1.77 KB
46°F
Cloudy
Search
ENG
IN
15:21
11-01-2023

Method 2: Using Get Data option in Microsoft Excel

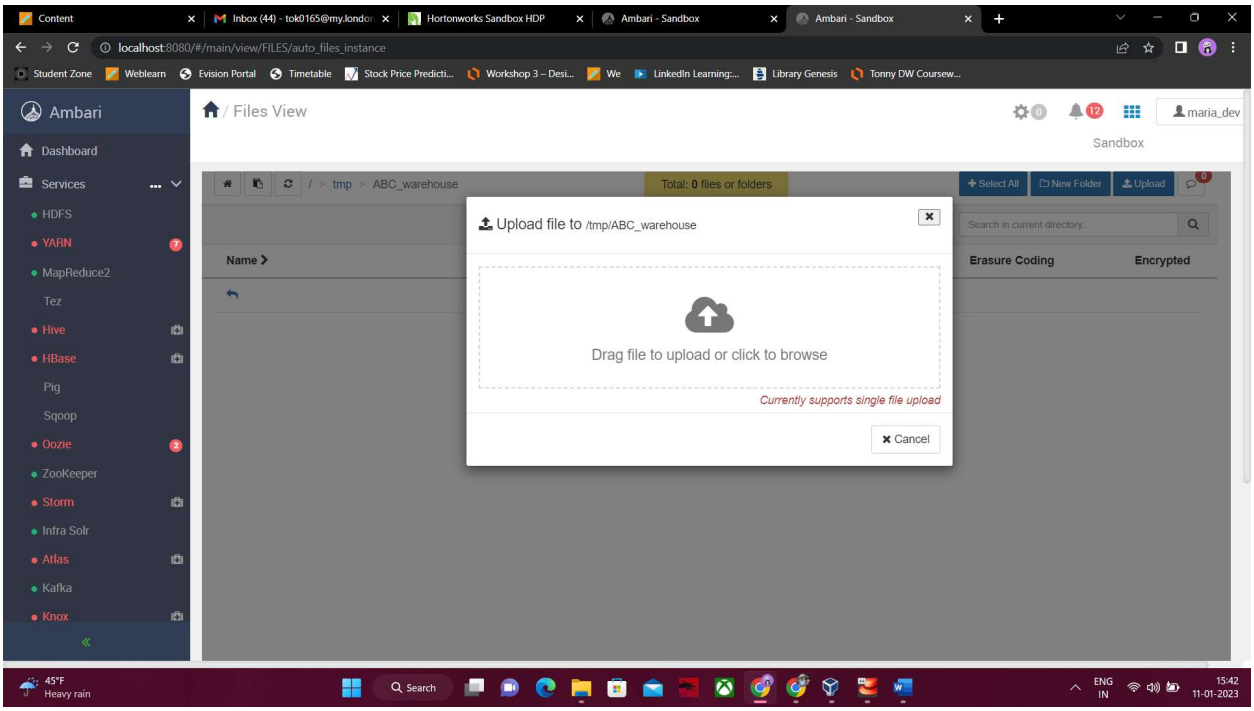


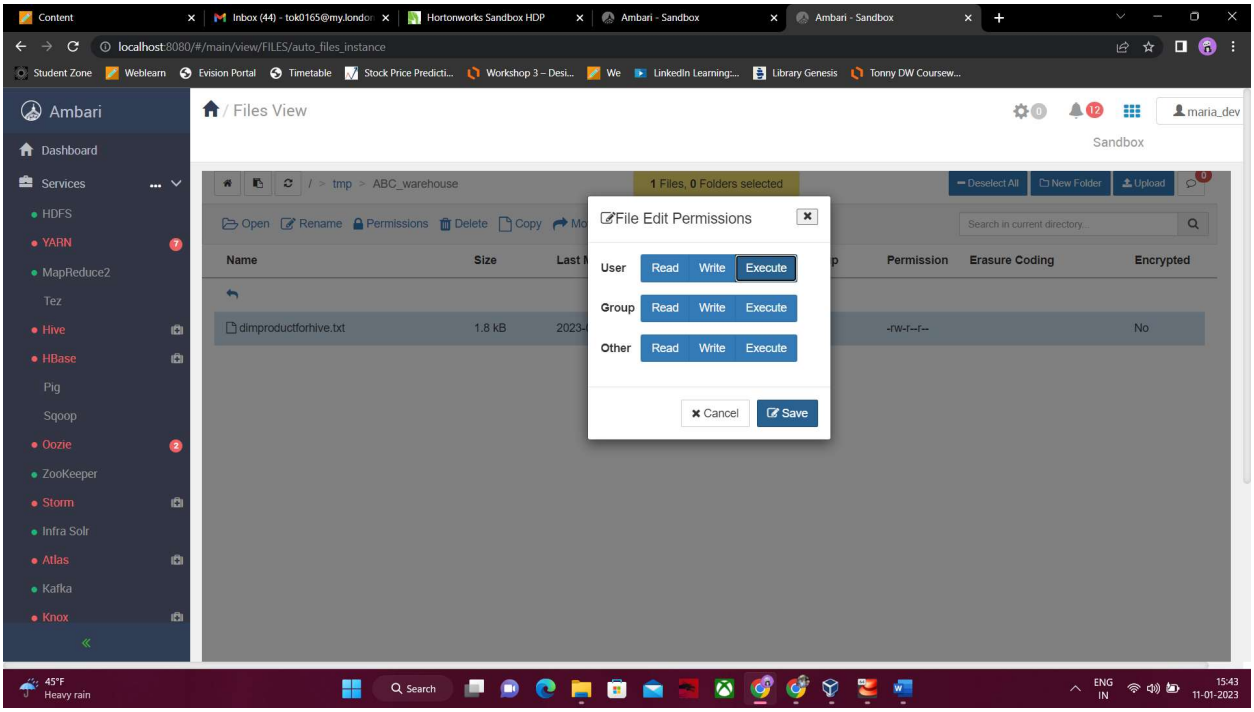




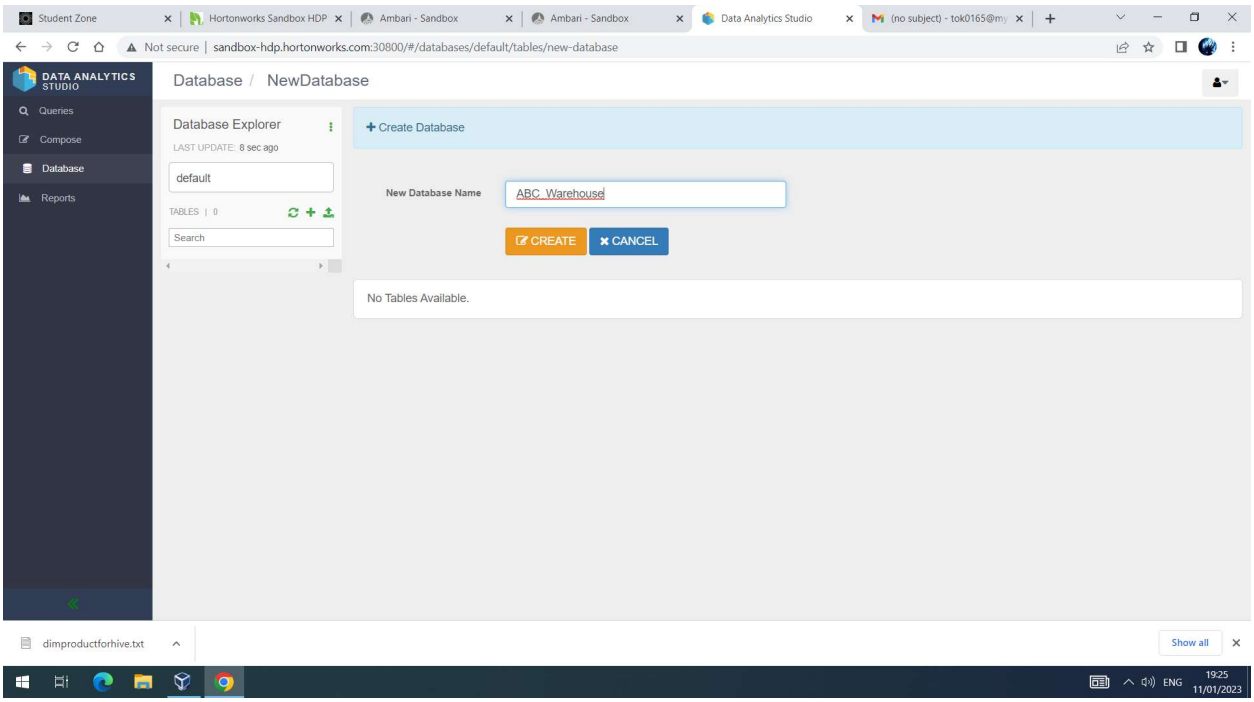
for our coursework we will use Method 1 and import our text file to Ambari by using the HDFS File Browser, for further Process.

We created a new folder name ABC_Warehouse using the files view option in Ambari and imported our text file and gave all relevant read, write, and execute permissions.





Now we create our database in Hive and call it ABC_Warehouse



We create a table called "dimproduct" with specific column names and data types, and then loads data into the table from a file located in the HDFS file system.

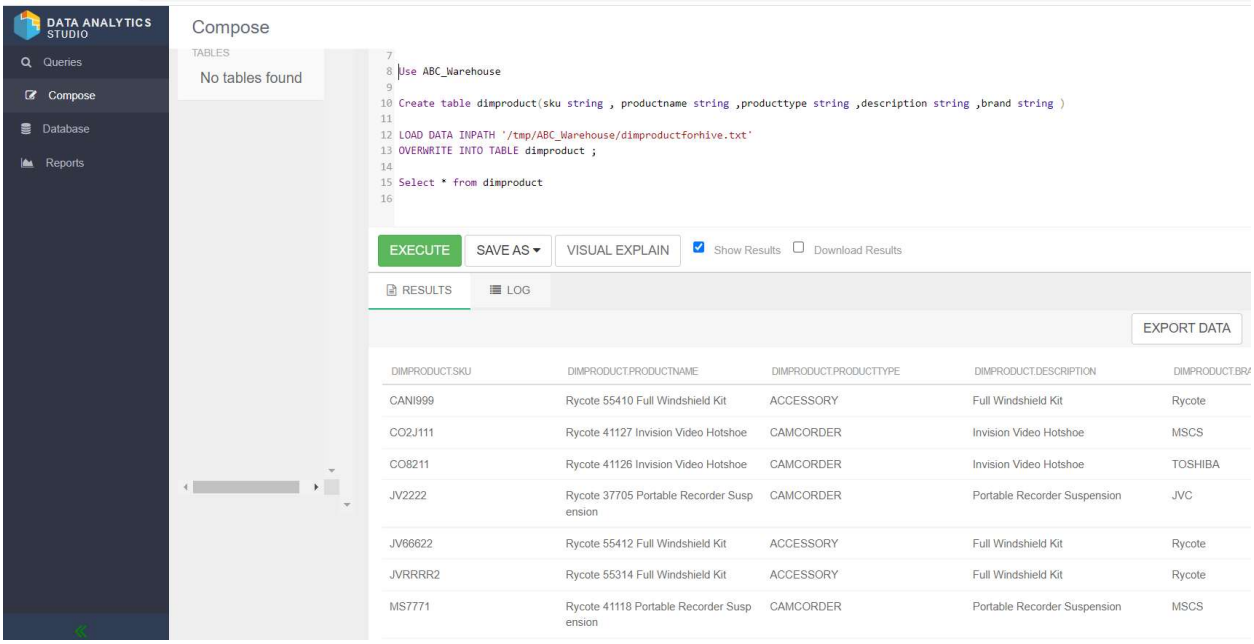
The first command uses the "USE" statement to switch to the "ABC_Warehouse" database.

The second command creates a new table called "dimproduct" with five columns: "sku", "productname", "producttype", "description", and "brand", with all columns are of type string.

The third command uses the "LOAD DATA" statement to load data from a file located at '/tmp/ABC_Warehouse/dimproductforhive.txt' and loads it into the "dimproduct" table. The "INPATH" clause specifies the location of the file on HDFS, and the "OVERWRITE INTO" clause overwrites any existing data in the table.

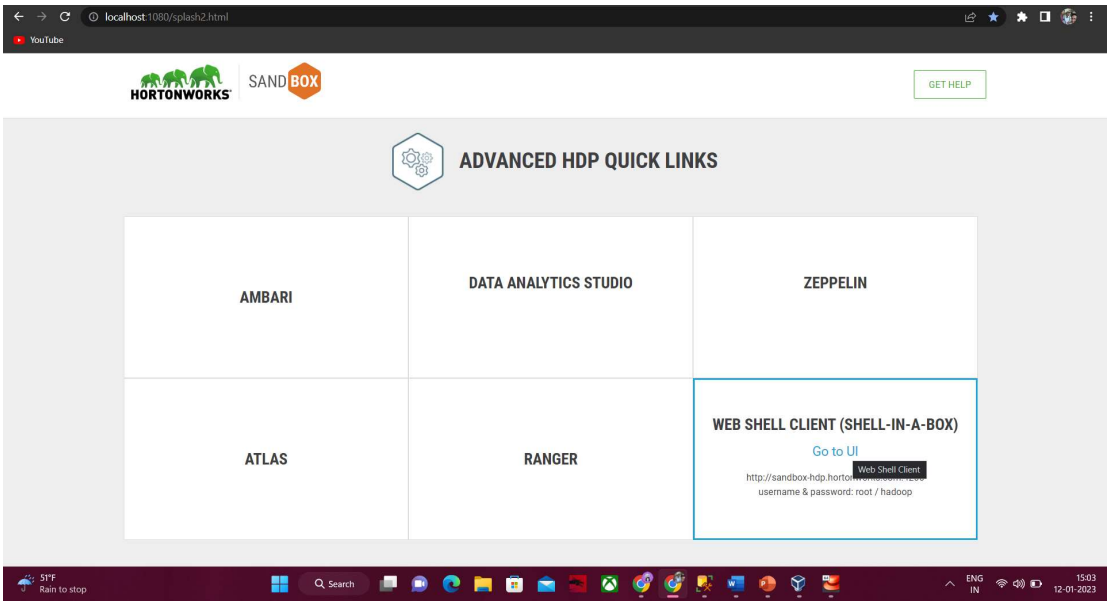
The last command selects all columns and rows from the "dimproduct" table, this will display all the rows and columns of the table on the screen.

```
Use ABC_Warehouse

Create table dimproduct(sku string , productname string ,producttype string ,description string ,brand string )

LOAD DATA INPATH '/tmp/ABC_Warehouse/dimproductforhive.txt'
OVERWRITE INTO TABLE dimproduct ;

Select * from dimproduct
```

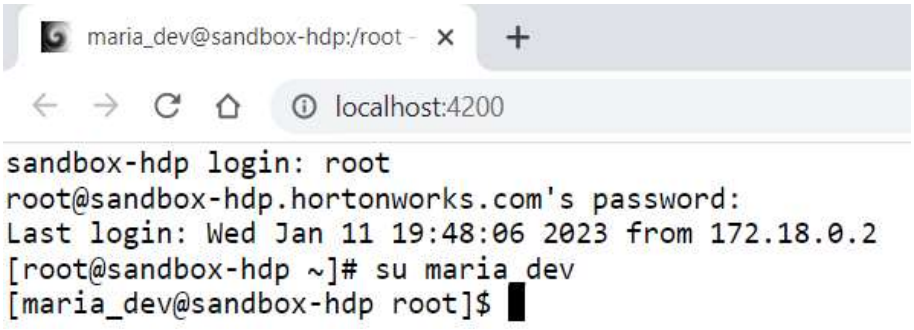


12.Accessing and Use of Pig

Apache Pig is an open-source tool that allows analyzing large data sets using a high-level programming language (Pig Latin) and a runtime environment on a distributed cluster. It is designed to work with data stored in Hadoop Distributed File System (HDFS) and other Hadoop-compatible data storage systems. It is commonly used in big data analytics and data warehousing projects and can be integrated with other big data tools like Apache Hive and Apache Spark.



We can access Pig by directly going to the link: http://sandbox-hdp.hortonworks.com:4200. This link takes us to the Apache Ambari Web UI, which is the web-based management console for the Hortonworks Data Platform (HDP) Sandbox. From there, we can access the Pig View, which allows us to interact with Pig through a web-based interface. We will be able to run Pig Latin scripts and commands, we will also be able to view the output of Pig commands and see the results of their queries in a tabular format. We login using the username as **root** and password as **Londonmet2000**, we then use the command **su** to switch user to Maria_Dev. We can use **cd** command to change our working directory. By using the **ls** command which stands for list files, we can see our previous worked files, we can use the **rm** command to remove/delete those listed files.

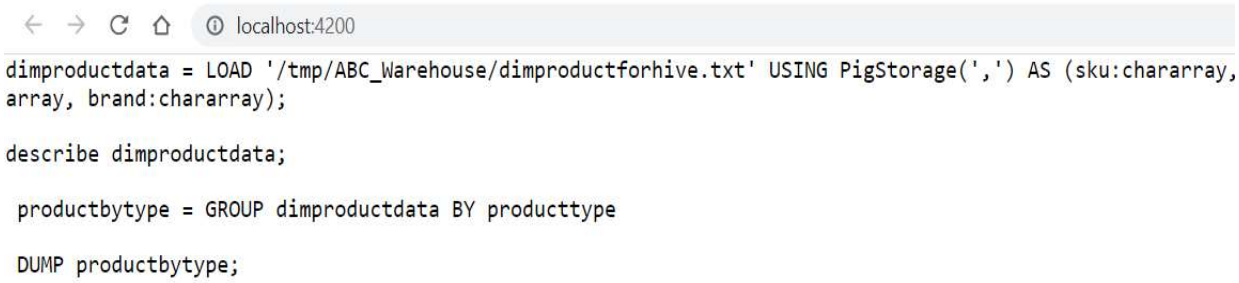


We use the command "**vi dimproductdata**" to open and edit a file named "dimproductdata" using the vi editor. Since the file does not exist, vi will create it. Once the file is open, we can edit the contents of the file and save our changes.

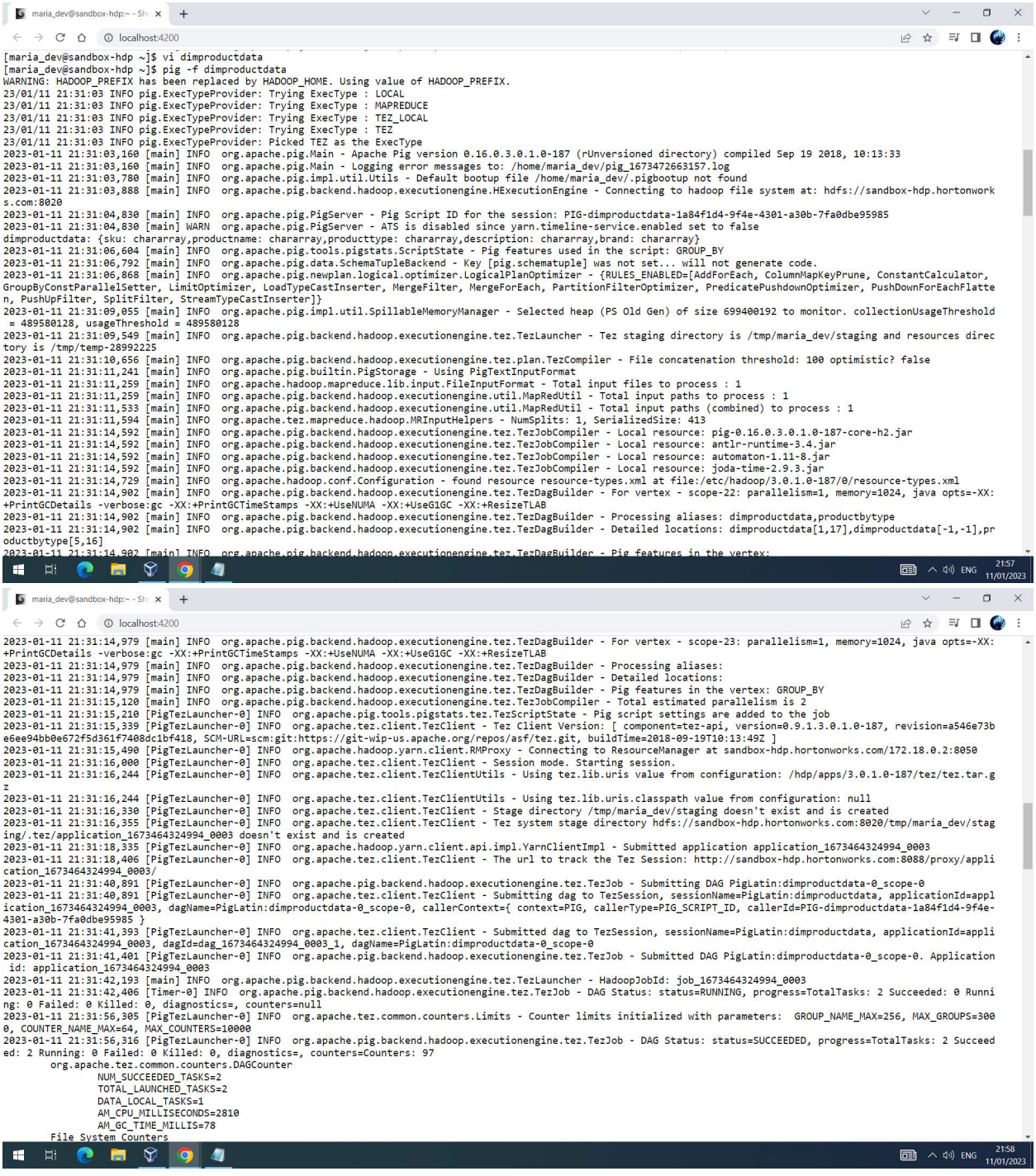
Code used in dimproductdata:

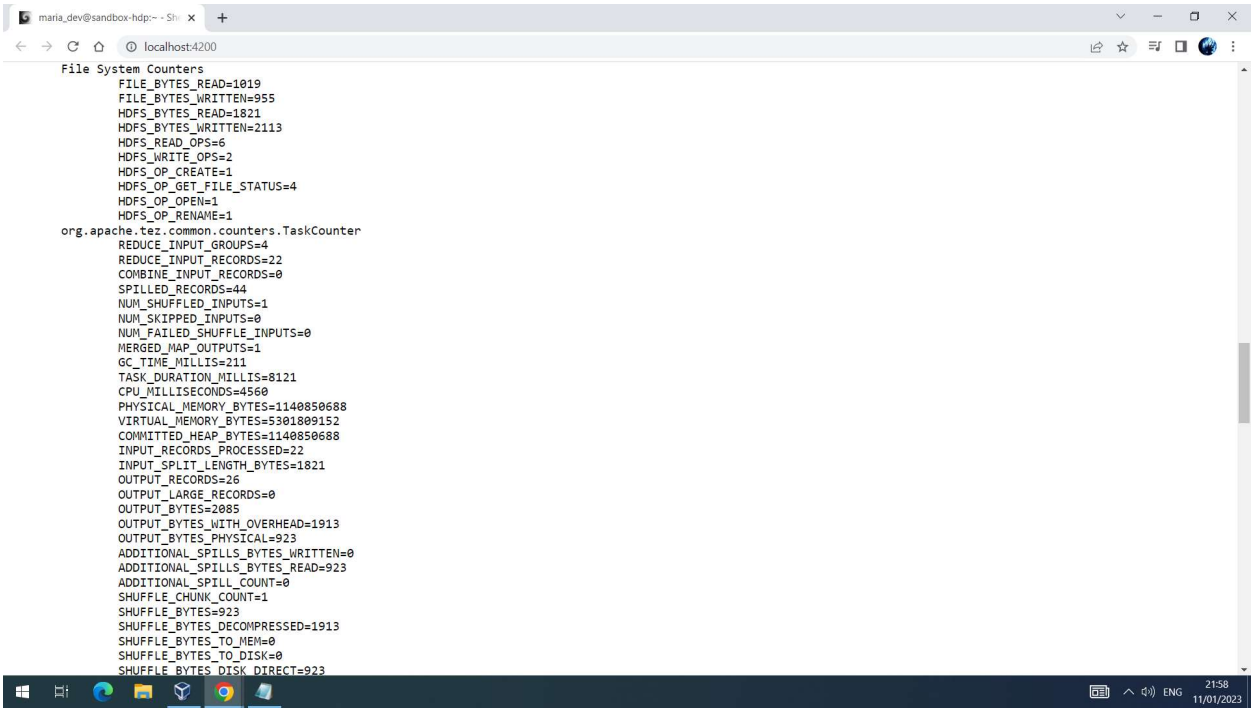
```
dimproductdata = LOAD '/tmp/ABC_Warehouse/dimproductforhive.txt' USING PigStorage(',') AS (sku:chararray, productname:chararray, producttype:chararray, description:chararray, brand:chararray);

describe dimproductdata

productbytype= GROUP dimproductdata  BY producttype

DUMP productbytype;
```

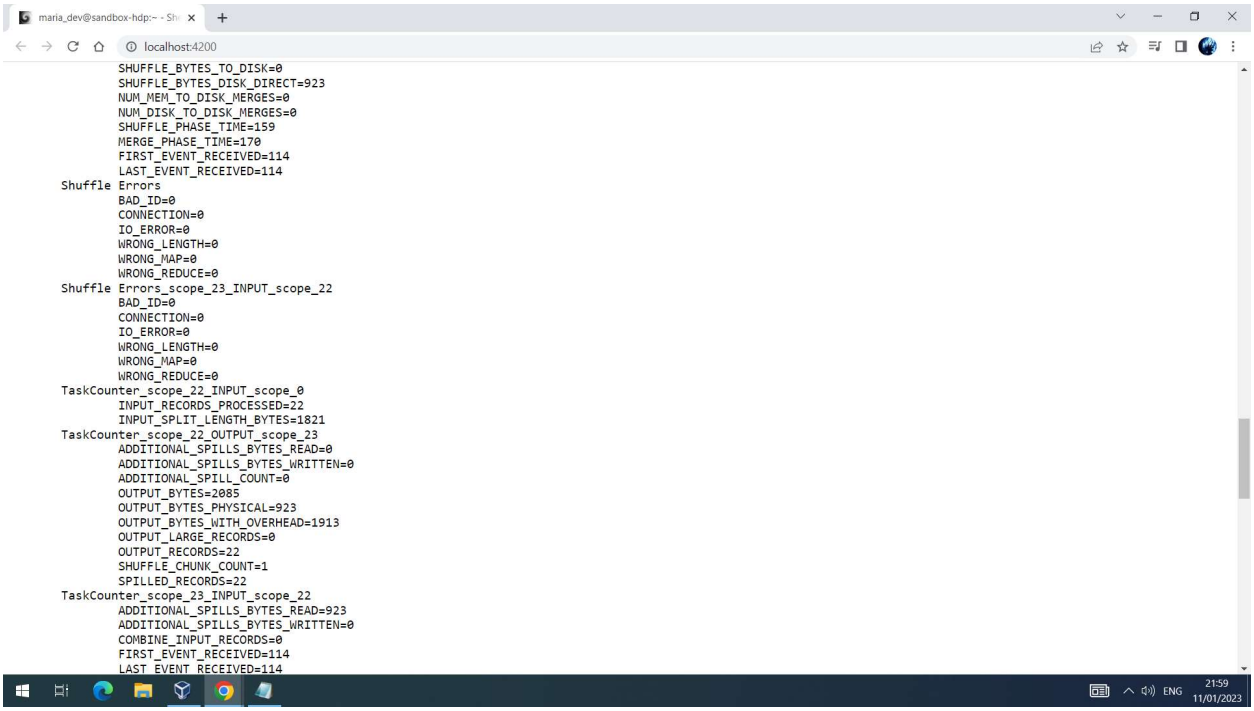
This is a Pig script that loads data from a file, groups it by a specific column, and shows the result of the grouping. Data is loaded from '/tmp/ABC_Warehouse/dimproductforhive.txt' into a relation "dimproductdata" and groups it by "producttype" column. The result is stored in the relation "productbytype" and is displayed on the screen.

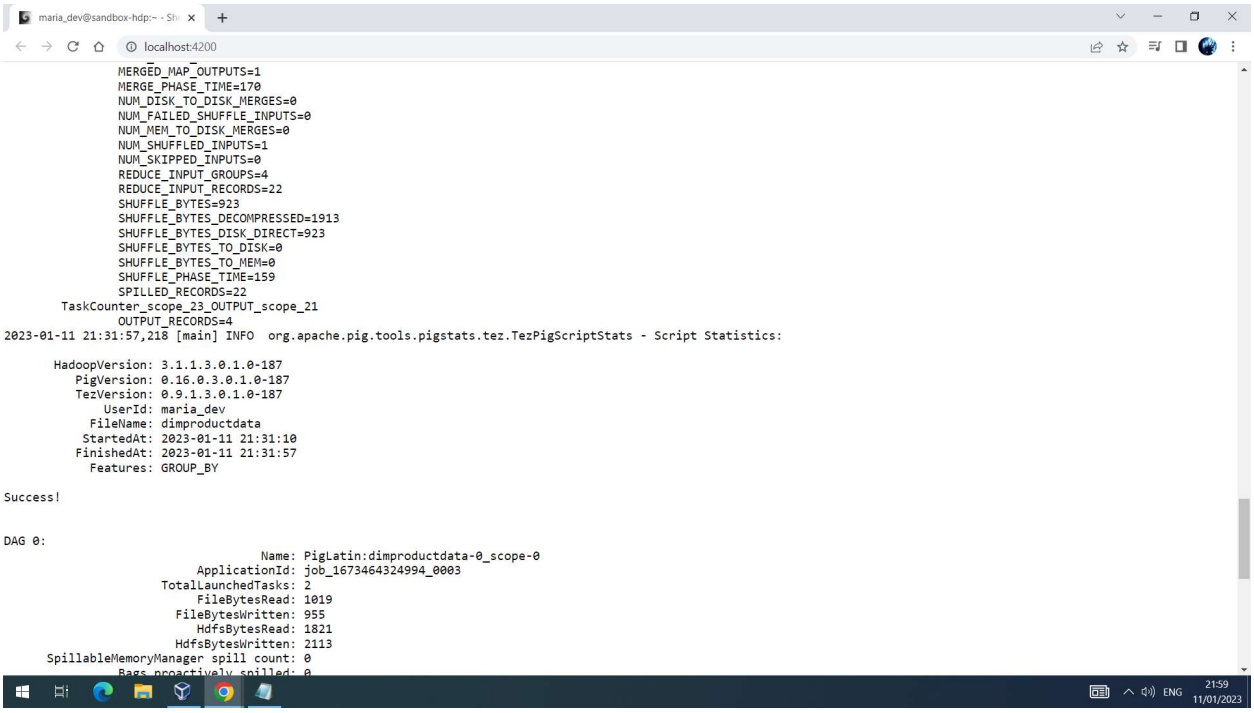


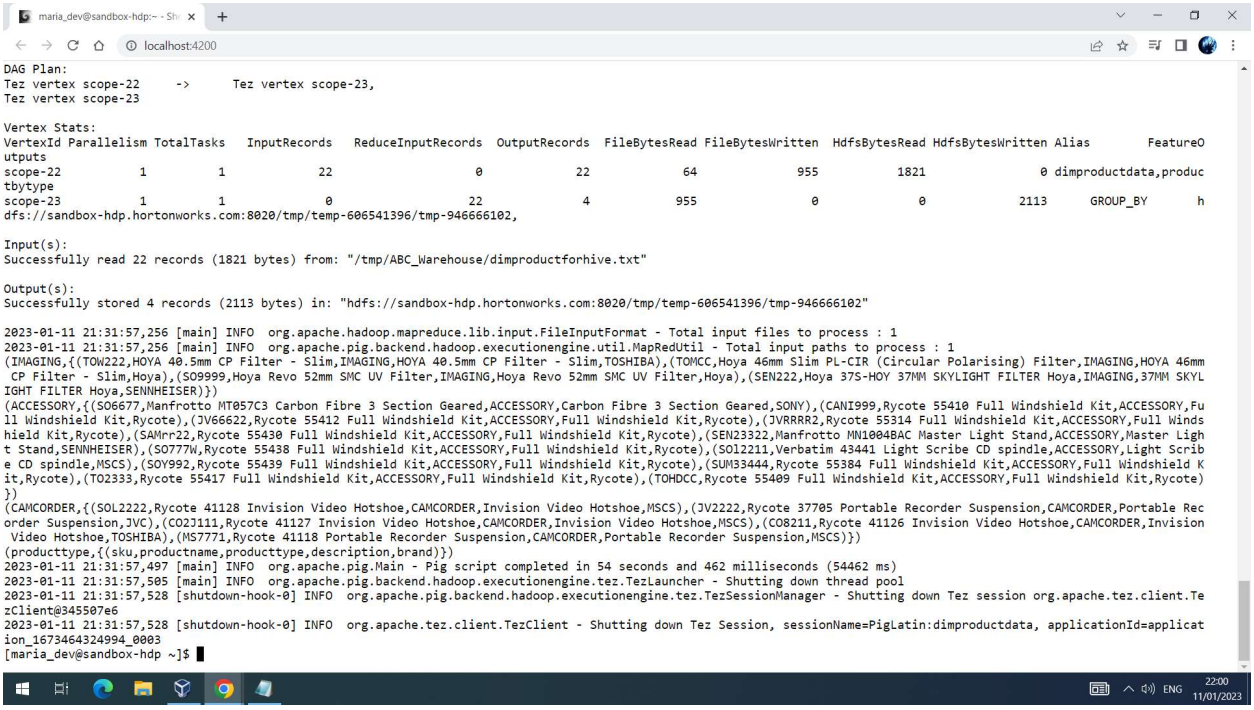
After executing the above script, we can see the below screenshot showing the output of the script file




maria_dev@sandbox-hdp:~ - Sh
localhost:4200
File System Counters
FILE_BYTES_READ=1019
FILE_BYTES_WRITTEN=955
HDFS_BYTES_READ=1821
HDFS_BYTES_WRITTEN=2113
HDFS_READ_OPS=6
HDFS_WRITE_OPS=2
HDFS_OP_CREATE=1
HDFS_OP_GET_FILE_STATUS=4
HDFS_OP_OPEN=1
HDFS_OP_RENAME=1
org.apache.tez.common.counters.TaskCounter
REDUCE_INPUT_GROUPS=4
REDUCE_INPUT_RECORDS=22
COMBINE_INPUT_RECORDS=0
SPILLED_RECORDS=44
NUM_SHUFFLED_INPUTS=1
NUM_SKIPPED_INPUTS=0
NUM_FAILED_SHUFFLE_INPUTS=0
MERGED_MAP_OUTPUTS=1
GC_TIME_MILLIS=211
TASK_DURATION_MILLIS=8121
CPU_MILLISECONDS=4560
PHYSICAL_MEMORY_BYTES=1140850688
VIRTUAL_MEMORY_BYTES=5301809152
COMMITTED_HEAP_BYTES=1140850688
INPUT_RECORDS_PROCESSED=22
INPUT_SPLIT_LENGTH_BYTES=1821
OUTPUT_RECORDS=26
OUTPUT_LARGE_RECORDS=0
OUTPUT_BYTES=2085
OUTPUT_BYTES_WITH_OVERHEAD=1913
OUTPUT_BYTES_PHYSICAL=923
ADDITIONAL_SPILLS_BYTES_WRITTEN=0
ADDITIONAL_SPILLS_BYTES_READ=923
ADDITIONAL_SPILL_COUNT=0
SHUFFLE_CHUNK_COUNT=1
SHUFFLE_BYTES=923
SHUFFLE_BYTES_DECOMPRESSED=1913
SHUFFLE_BYTES_TO_MEM=0
SHUFFLE_BYTES_TO_DISK=0
SHUFFLE_BYTES_DISK_DIRECT=923
ENG
21:58
11/01/2023


maria_dev@sandbox-hdp:~ - Sh
localhost:4200
SHUFFLE_BYTES_TO_DISK=0
SHUFFLE_BYTES_DISK_DIRECT=923
NUM_MEM_TO_DISK_MERGES=0
NUM_DISK_TO_DISK_MERGES=0
SHUFFLE_PHASE_TIME=159
MERGE_PHASE_TIME=170
FIRST_EVENT_RECEIVED=114
LAST_EVENT_RECEIVED=114
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
Shuffle Errors_scope_23_INPUT_scope_22
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
TaskCounter_scope_22_INPUT_scope_0
INPUT_RECORDS_PROCESSED=22
INPUT_SPLIT_LENGTH_BYTES=1821
TaskCounter_scope_22_OUTPUT_scope_23
ADDITIONAL_SPILLS_BYTES_READ=0
ADDITIONAL_SPILLS_BYTES_WRITTEN=0
ADDITIONAL_SPILL_COUNT=0
OUTPUT_BYTES=2085
OUTPUT_BYTES_PHYSICAL=923
OUTPUT_BYTES_WITH_OVERHEAD=1913
OUTPUT_LARGE_RECORDS=0
OUTPUT_RECORDS=22
SHUFFLE_CHUNK_COUNT=1
SPILLED_RECORDS=22
TaskCounter_scope_23_INPUT_scope_22
ADDITIONAL_SPILLS_BYTES_READ=923
ADDITIONAL_SPILLS_BYTES_WRITTEN=0
COMBINE_INPUT_RECORDS=0
FIRST_EVENT_RECEIVED=114
LAST_EVENT_RECEIVED=114
ENG
21:59
11/01/2023


maria_dev@sandbox-hdp:~ - Sh
localhost:4200
MERGED_MAP_OUTPUTS=1
MERGE_PHASE_TIME=170
NUM_DISK_TO_DISK_MERGES=0
NUM_FAILED_SHUFFLE_INPUTS=0
NUM_MEM_TO_DISK_MERGES=0
NUM_SHUFFLED_INPUTS=1
NUM_SKIPPED_INPUTS=0
REDUCE_INPUT_GROUPS=4
REDUCE_INPUT_RECORDS=22
SHUFFLE_BYTES=923
SHUFFLE_BYTES_DECOMPRESSED=1913
SHUFFLE_BYTES_DISK_DIRECT=923
SHUFFLE_BYTES_TO_DISK=0
SHUFFLE_BYTES_TO_MEM=0
SHUFFLE_PHASE_TIME=159
SPILLED_RECORDS=22
TaskCounter_scope_23_OUTPUT_scope_21
OUTPUT_RECORDS=4
2023-01-11 21:31:57,218 [main] INFO org.apache.pig.tools.pigstats.tez.TezPigScriptStats - Script Statistics:
HadoopVersion: 3.1.1.3.0.1.0-187
PigVersion: 0.16.0.3.0.1.0-187
TezVersion: 0.9.1.3.0.1.0-187
UserId: maria_dev
FileName: dimproductdata
StartedAt: 2023-01-11 21:31:10
FinishedAt: 2023-01-11 21:31:57
Features: GROUP_BY
Success!
DAG 0:
Name: PigLatin:dimproductdata-0_scope-0
ApplicationId: job_1673464324994_0003
TotalLaunchedTasks: 2
FileBytesRead: 1019
FileBytesWritten: 955
HdfsBytesRead: 1821
HdfsBytesWritten: 2113
SpillableMemoryManager spill count: 0
Bags proactively spilled: 0
ENG
21:59
11/01/2023


maria_dev@sandbox-hdp:~ - Sh
localhost:4200
DAG Plan:
Tez vertex scope-22 -> Tez vertex scope-23,
Tez vertex scope-23
Vertex Stats:
VertexId Parallelism TotalTasks InputRecords ReduceInputRecords OutputRecords FileBytesRead FileBytesWritten HdfsBytesRead HdfsBytesWritten Alias FeatureO
utputs
scope-22 1 1 22 0 22 64 955 1821 0 dimproductdata,produc
tbytype
scope-23 1 1 0 22 4 955 0 0 2113 GROUP_BY h
dfs://sandbox-hdp.hortonworks.com:8020/tmp/temp-606541396/tmp-946666102,
Input(s):
Successfully read 22 records (1821 bytes) from: "/tmp/ABC_Warehouse/dimproductforhive.txt"
Output(s):
Successfully stored 4 records (2113 bytes) in: "hdfs://sandbox-hdp.hortonworks.com:8020/tmp/temp-606541396/tmp-946666102"
2023-01-11 21:31:57,256 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input files to process : 1
2023-01-11 21:31:57,256 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(IMAGING,{(TOW222,HOYA 40.5mm CP Filter - Slim,IMAGING,HOYA 40.5mm CP Filter - Slim,TOSHIBA),(TOMCC,Hoya 46mm Slim PL-CIR (Circular Polarising) Filter,IMAGING,HOYA 46mm CP Filter - Slim,Hoya),(SO9999,Hoya Revo 52mm SMC UV Filter,IMAGING,Hoya Revo 52mm SMC UV Filter,Hoya),(SEN222,Hoya 37S-HOY 37MM SKYLIGHT FILTER Hoya,IMAGING,37MM SKYLIGHT FILTER Hoya,SENNHEISER)})
(ACCESSORY,{(SO6677,Manfrotto MT057C3 Carbon Fibre 3 Section Geared,ACCESSORY,Carbon Fibre 3 Section Geared,SONY),(CANI999,Rycote 55410 Full Windshield Kit,ACCESSORY,Full Windshield Kit,Rycote),(JV66622,Rycote 55412 Full Windshield Kit,ACCESSORY,Full Windshield Kit,Rycote),(JVRRRR2,Rycote 55314 Full Windshield Kit,ACCESSORY,Full Windshield Kit,Rycote),(SAMrr22,Rycote 55430 Full Windshield Kit,ACCESSORY,Full Windshield Kit,Rycote),(SEN23322,Manfrotto MN1004BAC Master Light Stand,ACCESSORY,Master Light Stand,SENNHEISER),(SO777W,Rycote 55438 Full Windshield Kit,ACCESSORY,Full Windshield Kit,Rycote),(SO12211,Verbatim 43441 Light Scribe CD spindle,ACCESSORY,Light Scribe CD spindle,MSCS),(SOY992,Rycote 55439 Full Windshield Kit,ACCESSORY,Full Windshield Kit,Rycote),(SUM33444,Rycote 55384 Full Windshield Kit,ACCESSORY,Full Windshield Kit,Rycote),(TO2333,Rycote 55417 Full Windshield Kit,ACCESSORY,Full Windshield Kit,Rycote),(TOHDCC,Rycote 55409 Full Windshield Kit,ACCESSORY,Full Windshield Kit,Rycote)})
(CAMCORDER,{(SOL2222,Rycote 41128 Invision Video Hotshoe,CAMCORDER,Invision Video Hotshoe,MSCS),(JV2222,Rycote 37705 Portable Recorder Suspension,CAMCORDER,Portable Recorder Suspension,JVC),(CO2J111,Rycote 41127 Invision Video Hotshoe,CAMCORDER,Invision Video Hotshoe,MSCS),(CO8211,Rycote 41126 Invision Video Hotshoe,CAMCORDER,Invision Video Hotshoe,TOSHIBA),(MS7771,Rycote 41118 Portable Recorder Suspension,CAMCORDER,Portable Recorder Suspension,MSCS)})
(producttype,{(sku,productname,producttype,description,brand)})
2023-01-11 21:31:57,497 [main] INFO org.apache.pig.Main - Pig script completed in 54 seconds and 462 milliseconds (54462 ms)
2023-01-11 21:31:57,505 [main] INFO org.apache.pig.backend.hadoop.executionengine.tez.TezLauncher - Shutting down thread pool
2023-01-11 21:31:57,528 [shutdown-hook-0] INFO org.apache.pig.backend.hadoop.executionengine.tez.TezSessionManager - Shutting down Tez session org.apache.tez.client.TezClient@345507e6
2023-01-11 21:31:57,528 [shutdown-hook-0] INFO org.apache.tez.client.TezClient - Shutting down Tez Session, sessionName=PigLatin:dimproductdata, applicationId=application_1673464324994_0003
[maria_dev@sandbox-hdp ~]$
ENG
22:00
11/01/2023

13.Learnings and Understandings

I have learnt the process of creating a database, its dimension and fact tables in a relational database management system in our case SQL Server Management Studio (SSMS). I can now design the database schema to meet the requirements of the data and its usage. I also learnt how to import data into the database using the import export wizard, which is a crucial step in populating the tables. I also learnt the importance of using a staging area and use of lookup tables. I understood the importance of SSMS in managing and querying databases, it's a powerful tool that allows us to easily view and manipulate the data stored in the database.

Additionally, I have also learned about the Hadoop ecosystem and its components. I learnt about HDFS, which is a distributed file system that allows for storing large amounts of data across multiple machines. I gained insights about Hive and Pig, which are data processing tools that can be used to analyze and query large data sets. Hive allows us to use SQL-like language to query data, it's similar to using SQL in a traditional relational database. Pig, on the other hand, uses Pig Latin, which is a high-level programming language specifically designed for data processing. With Pig, we can write programs to process and analyze data in a more flexible way than Hive.

Overall, I have gained a solid understanding of the various technologies and tools used for managing and analyzing large data sets in a distributed environment like Hadoop ecosystem.

14.Conclusion

Data warehousing is a technique that allows organizations to analyze their business functions to make better decisions. It separates data analysis and query processes from transactional processes, increasing analytical power without disrupting transactional systems. Hadoop is an open-source framework that enables the storage and processing of large data sets across multiple machines. It includes components like HDFS and tools like Hive and Pig for analyzing and querying data. These features make Hadoop a valuable platform for managing and analyzing large data sets in a cost-effective way. The use of SSMS and Hadoop in ABC warehouse has been proven effective in managing inventory and orders. By using the data analysis capabilities of these tools, ABC warehouse has been able to gain a deeper understanding of its processes and make better decisions. SSMS provides an easy-to-use interface for querying and analyzing data, while Hadoop enables efficient processing of large datasets. Together, these tools have improved the operations and customer service of ABC warehouse. The implementation of SSMS and Hadoop has been a successful step for ABC warehouse, making it more efficient and data-driven in decision making.