B9DA102

Data Storage Solutions for Data Analytics

CA1 Assessment Tasks

Submitted By

Shekhar Ramesh Vasudev

10383982

Contents

1. <u>Introduction</u>

The North Wind Database contains data about a company 'North Wind Traders', a company that exports and imports food items around the globe. The database contains information about the company's Employees, Products, Suppliers, Customers and Orders etc.

For the requirement of this assignment, a Business Driver will be selected from the operational database (Northwind Database). For which a Data warehouse will be designed and implemented to extract, transform and load (ETL process) necessary data from the Operational Data Base into the Data Warehouse using **Microsoft SQL Server Management Studio 2017**.

Business Intelligence is the use of tools and methodologies that help companies access the data from their data bases or data warehouses to prepare it for analysis, develop and queries on the data to generate reports, dashboards and data visualizations to deliver analytical findings to the decision makers of the company. For the Business Intelligence aspect of the assignment **Tableau** is used for applying data analysis techniques to support the Decision-making process of the Company.

For the final part of the assignment **Neo4j** is used to implement the Northwind E-R model as a graph database. Using Cypher (Neo4j query language) to retrieve information and analyze the North wind graph database based on the selected business driver.

The assignment can be broken down into the following sections:

- Part 1: Business Driver
- Part 2: Data Modelling
- Part 3: Implement Tables and ETL Procedures
- Part 4: Reporting and analysis
- Part 5: Modelling and Graph Data Models
- Part 6: Graph information retrieval and analysis

2. Part 1: Business Driver

The subject area chosen for analysis is **Inventory Management**.

This Business driver will be based around the following tables of the North Wind Database

- Products
- Orders
- Order list
- Suppliers

2.1 Reason for selecting the Business Driver

Inventory management is the management of inventory and stock. It includes aspects such as controlling and overseeing ordering inventory, storage of inventory, and controlling the amount of product available for sale.

Inventory management helps the organization maintain the right amount of stock in the warehouse. Inadequate inventory of products will result in the loss of sales and eventually drive customers to another supplier. Excess Inventory may result in profit losses due to expiry of the product, occupy storage space, damages caused due to storage etc.

Having the right quantity, in the right place at the right time and at the right cost will ensure that the products will be delivered on the promised delivery dates. Having inadequate inventory of the products will lead to a poor reputation of customer service.

2.2 Identify Key Stakeholders

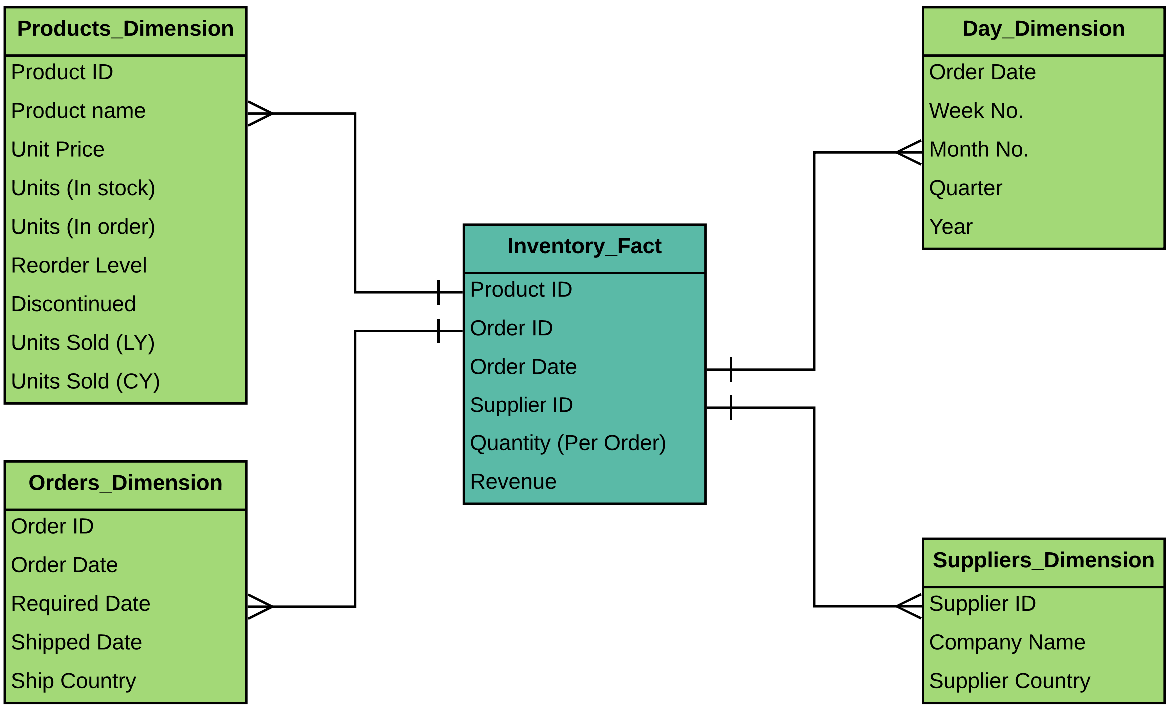
The Organization (North Wind Traders), Suppliers, Customers and the Employees are the Key Stakeholders who will be affected.

2.3 Vision and Goals of the Data Warehouse

- Compare the units sold Last year (1996-97) and Current year (1997-98) with respect to the reorder level of products.
- To find evidence of any dead stock (discontinued products) in the inventory.
- To find when was the discontinued product last ordered.
- What is the amount spent on storing unnecessary stock?
- To find if the products are restocked efficiently when the units in stock are low.

3. Part 2: Data Modelling

Implementing the star schema, we can create a Data Warehouse (Data Mart) to evaluate the efficiency of the Inventory management in North Wind traders using the following layout.



3.1 Reasons for the Design

The star schema is one of the simplest Data warehouse schema. It consists of one central fact table from which the four dimension tables branch out to resemble a star shape. The Fact tables are in third normal form (3NF) and the dimension tables are de-normalized.

A fact table contains the numeric measures produced by an operational measurement event. A fact table row corresponds to a measurement event and vice versa. The fact table contains the foreign keys to dimension tables and fact’s data on a detailed level.

The attributes of the dimension table explain the dimensional value. The primary keys of each dimension tables are the part of the composite primary key of the fact table.

Main characteristics of a star schema:

- Star schema is widely used in data warehouse implementations and is supported by almost all Business Intelligence tools.

- Business users can interpret the data with ease as Star schemas follows a simple structure.

- The star schema has fewer tables and clear join paths which results in great query effectiveness.

- The star schema reduces the time required to load the data into the Data warehouse, initially we define and separate the facts and dimensions into different tables which result in loading the data faster.

4. Part 3: Implement tables and ETL procedures

The star schema designed for the Data warehouse can be created using MS SQL Management Studio 2017. Initially the necessary tables are created and then using ETL we can move the data from the operational Database (North wind) to the Data warehouse (Inventory DW). The following are the Tables in the Inventory DW:

- Inventory_Fact
- Order_Dimension
- Product_Dimension
- Supplier_Dimension
- Day_dimension

The Extract, Transform, and Load (ETL) system of the Data Warehouse environment consists of a work area, instantiated data structures, and a set of processes. The ETL system bridges the operational database and the Data warehouse.

- Initially the data is extracted from the operational database into the DW. Extraction means reading and understanding the source data and copying data into the ETL system. The primary mission of the ETL system is to deliver the dimension and fact tables.
- After the Extraction, the data goes through various transformations such as cleaning the data, combining data from various sources, and de-duplicating data.
- Final step of the ETL process is the physical structuring and loading of data into the presentation area's target dimensional models.

The following are the table creation and ETL procedure scripts for the Data warehouse

Inventory_fact Table Creation

```
-- This script creates a data warehouse with the name "Inventory_dw" and
populates the necessary columns in it

CREATE DATABASE Inventory_dw -- Creating the Inventory Data warehouse

USE [Inventory_dw] -- Using the created data ware house
GO

BEGIN TRY
    DROP TABLE [dbo].[Inventory_fact] -- Deleting the table if it already exists
END TRY

BEGIN CATCH
    /*No Action*/
```

```
END CATCH

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-- Populating the Inventory_fact table with the column names mentioned below

CREATE TABLE [dbo].[Inventory_fact](
    [ProductID] [int] NOT NULL,
    [OrderID] [int] NOT NULL,
    [OrderDate] [datetime] NOT NULL,
    [SupplierID] [int] NOT NULL,
    [QuantityPerOrder] [smallint] NULL,
    [Revenue] [money] NOT NULL,
     CONSTRAINT [IX_Inventory_fact] UNIQUE NONCLUSTERED
(
    [OrderID] ASC,
    [ProductID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
```

Order_Dimension Table Creation

```
-- This script creates an order_dimension table in the Inventory data warehouse

DROP TABLE [Inventory_dw].dbo.order_dimension; -- Delete the table if it already
exists

-- Populate the order_dimension table with the column names mentioned below

CREATE TABLE [Inventory_dw].dbo.order_dimension (
    OrderID INT NOT NULL,
    OrderDate DATETIME NOT NULL,
    RequiredDate DATETIME NOT NULL,
    ShippedDate DATETIME NOT NULL,
    ShipCountry NVARCHAR(15) NULL,
PRIMARY KEY CLUSTERED
(
    [OrderID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Product_Dimension Table Creation

```
-- This script creates the product_dimension table in the Inventory data
warehouse

DROP TABLE [Inventory_dw].dbo.product_dimension; -- Delete the table if it
already exists

-- Populate the product_dimension table with the column names mentioned below

CREATE TABLE [Inventory_dw].dbo.product_dimension (
    ProductID INT NOT NULL,
    ProductName NVARCHAR(40) NOT NULL,
    Unitprice MONEY,
    UnitsInStock int NULL,
    UnitsOnOrder int NULL,
    ReorderLevel int NULL,
    Discontinued BIT NOT NULL,
    UnitsSoldLY int NULL,
    UnitsSoldCY int NULL,
PRIMARY KEY CLUSTERED
(
    [ProductID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Supplier_Dimension Table Creation

```
-- This script creates the supplier_dimension table in the Inventory data
warehouse

DROP TABLE [Inventory_dw].dbo.supplier_dimension; -- Delete the table if it
already exists

-- Populate the supplier_dimension table with the column names mentioned below

CREATE TABLE [Inventory_dw].dbo.supplier_dimension (
    SupplierID INT NOT NULL,
    CompanyName NVARCHAR(40) NOT NULL ,
    SupplierCountry NVARCHAR(15) NULL,
PRIMARY KEY CLUSTERED
(
    [SupplierID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Day_Dimension Table Creation

```
-- This script creates a calender that can split the data by the
date,week,month,quarter,year etc.

USE [Inventory_dw]; -- Using the Inventory Data Warehouse

BEGIN TRY
    DROP TABLE [dbo].[day_dimension]
END TRY

BEGIN CATCH
    /*No Action*/
END CATCH

/*********************************************************************************
**/

CREATE TABLE    [dbo].[day_dimension]
    (   [DateKey] INT primary key,
        [Date] DATETIME,
        [FullDateUK] CHAR(10), -- Date in dd-MM-yyyy format
        [FullDateUSA] CHAR(10),-- Date in MM-dd-yyyy format
        [DayOfMonth] INT, -- Field will hold day number of Month
        [DaySuffix] VARCHAR(4), -- Apply suffix as 1st, 2nd ,3rd etc
        [DayName] VARCHAR(9), -- Contains name of the day, Sunday, Monday
        [DayOfWeekUSA] INT,-- First Day Sunday=1 and Saturday=7
        [DayOfWeekUK] INT,-- First Day Monday=1 and Sunday=7
        [DayOfWeekInMonth] INT, --1st Monday or 2nd Monday in Month
        [DayOfWeekInYear] INT,
        [DayOfQuarter] INT,
        [DayOfYear] INT,
        [WeekOfMonth] INT,-- Week Number of Month
        [WeekOfQuarter] INT, --Week Number of the Quarter
        [WeekOfYear] INT,--Week Number of the Year
        [Month] INT, --Number of the Month 1 to 12
        [MonthName] VARCHAR(9),--January, February etc
        [MonthOfQuarter] INT,-- Month Number belongs to Quarter
        [Quarter] INT,
        [QuarterName] VARCHAR(9),--First,Second..
```

```
         [Year] INT,-- Year value of Date stored in Row
         [YearName] CHAR(7), --CY 2012,CY 2013
         [MonthYear] CHAR(10), --Jan-2013,Feb-2013
         [MMYYYY] INT,
         [FirstDayOfMonth] DATE,
         [LastDayOfMonth] DATE,
         [FirstDayOfQuarter] DATE,
         [LastDayOfQuarter] DATE,
         [FirstDayOfYear] DATE,
         [LastDayOfYear] DATE,
         [IsHolidayUSA] BIT,-- Flag 1=National Holiday, 0-No National Holiday
         [IsWeekday] BIT,-- 0=Week End ,1=Week Day
         [HolidayUSA] VARCHAR(50),--Name of Holiday in US
         [IsHolidayUK] BIT Null,-- Flag 1=National Holiday, 0-No National Holiday
         [HolidayUK] VARCHAR(50) Null --Name of Holiday in UK
    )
GO

/********************************************************************************
************/
--Specify Start Date and End date here
--Value of Start Date Must be Less than Your End Date

DECLARE @StartDate DATETIME = '01/01/1990' --Starting value of Date Range
DECLARE @EndDate DATETIME = '01/01/2015' --End Value of Date Range

--Temporary Variables To Hold the Values During Processing of Each Date of Year
DECLARE
    @DayOfWeekInMonth INT,
    @DayOfWeekInYear INT,
    @DayOfQuarter INT,
    @WeekOfMonth INT,
    @CurrentYear INT,
    @CurrentMonth INT,
    @CurrentQuarter INT

/*Table Data type to store the day of week count for the month and year*/
DECLARE @DayOfWeek TABLE (DOW INT, MonthCount INT, QuarterCount INT, YearCount
INT)

INSERT INTO @DayOfWeek VALUES (1, 0, 0, 0)
INSERT INTO @DayOfWeek VALUES (2, 0, 0, 0)
INSERT INTO @DayOfWeek VALUES (3, 0, 0, 0)
INSERT INTO @DayOfWeek VALUES (4, 0, 0, 0)
INSERT INTO @DayOfWeek VALUES (5, 0, 0, 0)
INSERT INTO @DayOfWeek VALUES (6, 0, 0, 0)
INSERT INTO @DayOfWeek VALUES (7, 0, 0, 0)
```

```
--Extract and assign various parts of Values from Current Date to Variable

DECLARE @CurrentDate AS DATETIME = @StartDate
SET @CurrentMonth = DATEPART(MM, @CurrentDate)
SET @CurrentYear = DATEPART(YY, @CurrentDate)
SET @CurrentQuarter = DATEPART(QQ, @CurrentDate)

/********************************************************************************
************/
--Proceed only if Start Date(Current date ) is less than End date you specified
above

WHILE @CurrentDate < @EndDate
BEGIN

/*Begin day of week logic*/

         /*Check for Change in Month of the Current date if Month changed then
          Change variable value*/
    IF @CurrentMonth != DATEPART(MM, @CurrentDate)
    BEGIN
        UPDATE @DayOfWeek
        SET MonthCount = 0
        SET @CurrentMonth = DATEPART(MM, @CurrentDate)
    END

        /* Check for Change in Quarter of the Current date if Quarter changed
then change
         Variable value*/

    IF @CurrentQuarter != DATEPART(QQ, @CurrentDate)
    BEGIN
        UPDATE @DayOfWeek
        SET QuarterCount = 0
        SET @CurrentQuarter = DATEPART(QQ, @CurrentDate)
    END

        /* Check for Change in Year of the Current date if Year changed then
change
         Variable value*/


    IF @CurrentYear != DATEPART(YY, @CurrentDate)
    BEGIN
        UPDATE @DayOfWeek
        SET YearCount = 0
        SET @CurrentYear = DATEPART(YY, @CurrentDate)
    END
```

```
        -- Set values in table data type created above from variables

    UPDATE @DayOfWeek
    SET
        MonthCount = MonthCount + 1,
        QuarterCount = QuarterCount + 1,
        YearCount = YearCount + 1
    WHERE DOW = DATEPART(DW, @CurrentDate)

    SELECT
        @DayOfWeekInMonth = MonthCount,
        @DayOfQuarter = QuarterCount,
        @DayOfWeekInYear = YearCount
    FROM @DayOfWeek
    WHERE DOW = DATEPART(DW, @CurrentDate)

/*End day of week logic*/


/* Populate Your Dimension Table with values*/

    INSERT INTO [dbo].[day_dimension]
    SELECT

        CONVERT (char(8),@CurrentDate,112) as DateKey,
        @CurrentDate AS Date,
        CONVERT (char(10),@CurrentDate,103) as FullDateUK,
        CONVERT (char(10),@CurrentDate,101) as FullDateUSA,
        DATEPART(DD, @CurrentDate) AS DayOfMonth,
        --Apply Suffix values like 1st, 2nd 3rd etc..
        CASE
            WHEN DATEPART(DD,@CurrentDate) IN (11,12,13)
            THEN CAST(DATEPART(DD,@CurrentDate) AS VARCHAR) + 'th'
            WHEN RIGHT(DATEPART(DD,@CurrentDate),1) = 1
            THEN CAST(DATEPART(DD,@CurrentDate) AS VARCHAR) + 'st'
            WHEN RIGHT(DATEPART(DD,@CurrentDate),1) = 2
            THEN CAST(DATEPART(DD,@CurrentDate) AS VARCHAR) + 'nd'
            WHEN RIGHT(DATEPART(DD,@CurrentDate),1) = 3
            THEN CAST(DATEPART(DD,@CurrentDate) AS VARCHAR) + 'rd'
            ELSE CAST(DATEPART(DD,@CurrentDate) AS VARCHAR) + 'th'
            END AS DaySuffix,

        DATENAME(DW, @CurrentDate) AS DayName,
        DATEPART(DW, @CurrentDate) AS DayOfWeekUSA,

        -- check for day of week as Per US and change it as per UK format
        CASE DATEPART(DW, @CurrentDate)
```

```
            WHEN 1 THEN 7
            WHEN 2 THEN 1
            WHEN 3 THEN 2
            WHEN 4 THEN 3
            WHEN 5 THEN 4
            WHEN 6 THEN 5
            WHEN 7 THEN 6
            END
            AS DayOfWeekUK,

        @DayOfWeekInMonth AS DayOfWeekInMonth,
        @DayOfWeekInYear AS DayOfWeekInYear,
        @DayOfQuarter AS DayOfQuarter,
        DATEPART(DY, @CurrentDate) AS DayOfYear,
        DATEPART(WW, @CurrentDate) + 1 - DATEPART(WW, CONVERT(VARCHAR,
        DATEPART(MM, @CurrentDate)) + '/1/' + CONVERT(VARCHAR,
        DATEPART(YY, @CurrentDate))) AS WeekOfMonth,
        (DATEDIFF(DD, DATEADD(QQ, DATEDIFF(QQ, 0, @CurrentDate), 0),
        @CurrentDate) / 7) + 1 AS WeekOfQuarter,
        DATEPART(WW, @CurrentDate) AS WeekOfYear,
        DATEPART(MM, @CurrentDate) AS Month,
        DATENAME(MM, @CurrentDate) AS MonthName,
        CASE
            WHEN DATEPART(MM, @CurrentDate) IN (1, 4, 7, 10) THEN 1
            WHEN DATEPART(MM, @CurrentDate) IN (2, 5, 8, 11) THEN 2
            WHEN DATEPART(MM, @CurrentDate) IN (3, 6, 9, 12) THEN 3
            END AS MonthOfQuarter,
        DATEPART(QQ, @CurrentDate) AS Quarter,
        CASE DATEPART(QQ, @CurrentDate)
            WHEN 1 THEN 'First'
            WHEN 2 THEN 'Second'
            WHEN 3 THEN 'Third'
            WHEN 4 THEN 'Fourth'
            END AS QuarterName,
        DATEPART(YEAR, @CurrentDate) AS Year,
        'CY ' + CONVERT(VARCHAR, DATEPART(YEAR, @CurrentDate)) AS YearName,
        LEFT(DATENAME(MM, @CurrentDate), 3) + '-' + CONVERT(VARCHAR,
        DATEPART(YY, @CurrentDate)) AS MonthYear,
        RIGHT('0' + CONVERT(VARCHAR, DATEPART(MM, @CurrentDate)),2) +
        CONVERT(VARCHAR, DATEPART(YY, @CurrentDate)) AS MMYYYY,
        CONVERT(DATETIME, CONVERT(DATE, DATEADD(DD, - (DATEPART(DD,
        @CurrentDate) - 1), @CurrentDate))) AS FirstDayOfMonth,
        CONVERT(DATETIME, CONVERT(DATE, DATEADD(DD, - (DATEPART(DD,
        (DATEADD(MM, 1, @CurrentDate)))), DATEADD(MM, 1,
        @CurrentDate)))) AS LastDayOfMonth,
        DATEADD(QQ, DATEDIFF(QQ, 0, @CurrentDate), 0) AS FirstDayOfQuarter,
        DATEADD(QQ, DATEDIFF(QQ, -1, @CurrentDate), -1) AS LastDayOfQuarter,
        CONVERT(DATETIME, '01/01/' + CONVERT(VARCHAR, DATEPART(YY,
        @CurrentDate))) AS FirstDayOfYear,
        CONVERT(DATETIME, '12/31/' + CONVERT(VARCHAR, DATEPART(YY,
        @CurrentDate))) AS LastDayOfYear,
        NULL AS IsHolidayUSA,
        CASE DATEPART(DW, @CurrentDate)
            WHEN 1 THEN 0
            WHEN 2 THEN 1
            WHEN 3 THEN 1
            WHEN 4 THEN 1
            WHEN 5 THEN 1
            WHEN 6 THEN 1
            WHEN 7 THEN 0
            END AS IsWeekday,
        NULL AS HolidayUSA, Null, Null

    SET @CurrentDate = DATEADD(DD, 1, @CurrentDate)
END
```

Inventory_Fact ETL Procedure

```
-- This script Extracts, Transforms and Loads the data from the Northwind
Database into the Inventory_fact table in the data warehouse

USE [Northwind] -- Using the Northwind database
GO

DELETE FROM [Inventory_dw].[dbo].Inventory_fact -- Delete any exisiting data in
the table


INSERT INTO [Inventory_dw].[dbo].Inventory_fact -- Assign the columns to insert
the data
        (ProductID,
        OrderID,
        OrderDate,
        SupplierID,
        QuantityPerOrder,
        Revenue
        )
SELECT distinct -- Extract the data from the Northwind database
    p.ProductID,
    o.OrderID,
    o.OrderDate,
    p.SupplierID,
    SUM(od.Quantity) AS QuantityPerOrder,
    SUM(od.UnitPrice * od.Quantity) AS Revenue
```

```
FROM [dbo].[Products] AS p
JOIN [dbo].[Order Details] AS od
    ON p.ProductID = od.ProductID
JOIN [Orders] AS o ON od.OrderID = o.OrderID
GROUP BY p.ProductID,o.OrderDate,o.OrderID,p.SupplierID
ORDER BY p.ProductID;


-- Check the table is populated correctly

SELECT * FROM dbo.Inventory_fact
ORDER BY ProductID
```

Order_Dimension ETL Procedure

```
-- This script Extracts, Transforms and Loads the data from the Northwind
Database into the order_dimension table in the data warehouse

USE [Northwind] -- Using the Northwind database
GO

DELETE FROM [Inventory_dw].[dbo].order_dimension -- Delete any exisiting data in
the table


INSERT INTO [Inventory_dw].[dbo].order_dimension -- Assign the columns to insert
the data
        (OrderID,
        OrderDate,
        RequiredDate,
        ShippedDate,
        ShipCountry
        )
SELECT DISTINCT -- Extract the data from the Northwind database
    o.OrderID,
    o.OrderDate,
    o.RequiredDate,
    o.ShippedDate,
    o.ShipCountry
FROM [dbo].[Orders] AS o;

-- Check if the table is populated properly

SELECT * FROM [Inventory_dw].dbo.order_dimension;
```

Product_Dimension ETL Procedure

```
-- This script Extracts, Transforms and Loads the data from the Northwind
Database into the product_dimension table in the data warehouse

USE [Northwind] -- Using the Northwind database
GO

DELETE FROM [Inventory_dw].[dbo].product_dimension -- Delete any exisiting data
in the table

INSERT INTO [Inventory_dw].[dbo].product_dimension -- Assign the columns to
insert the data
        (ProductID,
        ProductName,
        Unitprice,
        UnitsInStock,
        UnitsOnOrder,
        ReorderLevel,
        Discontinued,
        UnitsSoldLY,
        UnitsSoldCY
        )
SELECT DISTINCT -- Extract the data from the Northwind database
    p.ProductID,
    p.ProductName,
    p.UnitPrice AS Unitprice,
    p.UnitsInStock,
    p.UnitsOnOrder,
    p.ReorderLevel,
    p.Discontinued,
(SELECT SUM(od.Quantity)
FROM    Orders o
        left join [Order Details] od on od.OrderID = o.OrderID
WHERE   OrderDate >= '1996-07-01' AND
        OrderDate   <= '1997-06-30'
        AND od.ProductID = p.ProductID
        GROUP BY od.ProductID
            ),
(SELECT  SUM(od.Quantity)
FROM    Orders o
        left join [Order Details] od on od.OrderID = o.OrderID
WHERE   OrderDate >= '1997-07-01' AND
        OrderDate   <= '1998-06-30'
        AND od.ProductID = p.ProductID
        GROUP BY od.ProductID
            )
FROM [dbo].[Products] AS p;
```

```
-- Check if the table is populated properly

SELECT * FROM [Inventory_dw].dbo.product_dimension;
```

Supplier_Dimension ETL Procedure

```
-- This script Extracts, Transforms and Loads the data from the Northwind
Database into the supplier_dimension table in the data warehouse

USE [Northwind] -- Using the Northwind database
GO

DELETE FROM [Inventory_dw].[dbo].supplier_dimension -- Delete any exisiting data
in the table

INSERT INTO [Inventory_dw].[dbo].supplier_dimension -- Assign the columns to
insert the data
        (SupplierID,
        CompanyName,
        SupplierCountry
        )
SELECT DISTINCT -- Extract the data from the Northwind database
    s.SupplierID,
    s.CompanyName,
    s.Country AS SupplierCountry
FROM [dbo].[Suppliers] AS s;

-- Checking if the table is populated properly

SELECT * FROM [Inventory_dw].dbo.supplier_dimension;
```

4.1 Why MS SQL Server DW Solutions

Microsoft SQL developed by Microsoft is a widely used RDBMS (Relational Database Management System). As a Database server, its primary function is to store and access the data on request of other applications. Using Microsoft SQL server database, we can efficiently manage our server's performance without compromising its recoverability and availability.

Advantages of MS SQL:

- **Ease of Installation:** Updates are detected and downloaded by the setup wizard.
- **Improved Security:** Only authorized personnel can gain access to the database.
- **Improved performance:** Since MS SQL has a built-in data compression feature with encryption it won't need to modify programs to encrypt the data which enhances performance with respect to data collection.
- **Additional packages:** SQL server includes other tools such as data management, data mining and disk partitioning

Guidelines for using MS SQL Server Database:

- **Maintaining a standardized environment:** Maintaining consistency throughout the SQL server's configuration will minimize the load of the operations.
- **Using a Dedicated SQL server:** For better efficiency and uninterrupted functioning it is advisable no to run multiple server instances on the same server
- **Managing log and data files:** Create your own log files with sufficient space to avoid AUTOGROW (should be enabled) events.
- **Least security principle:** A user is given only required security permissions i.e. a user is given his SA account with a strong password, a developer isn't given administrative permission.

Microsoft currently offers Azure SQL Data Warehouse which is a new warehousing application.

4.2 Comparing MS solution with a vendor offering competitive data warehousing solution

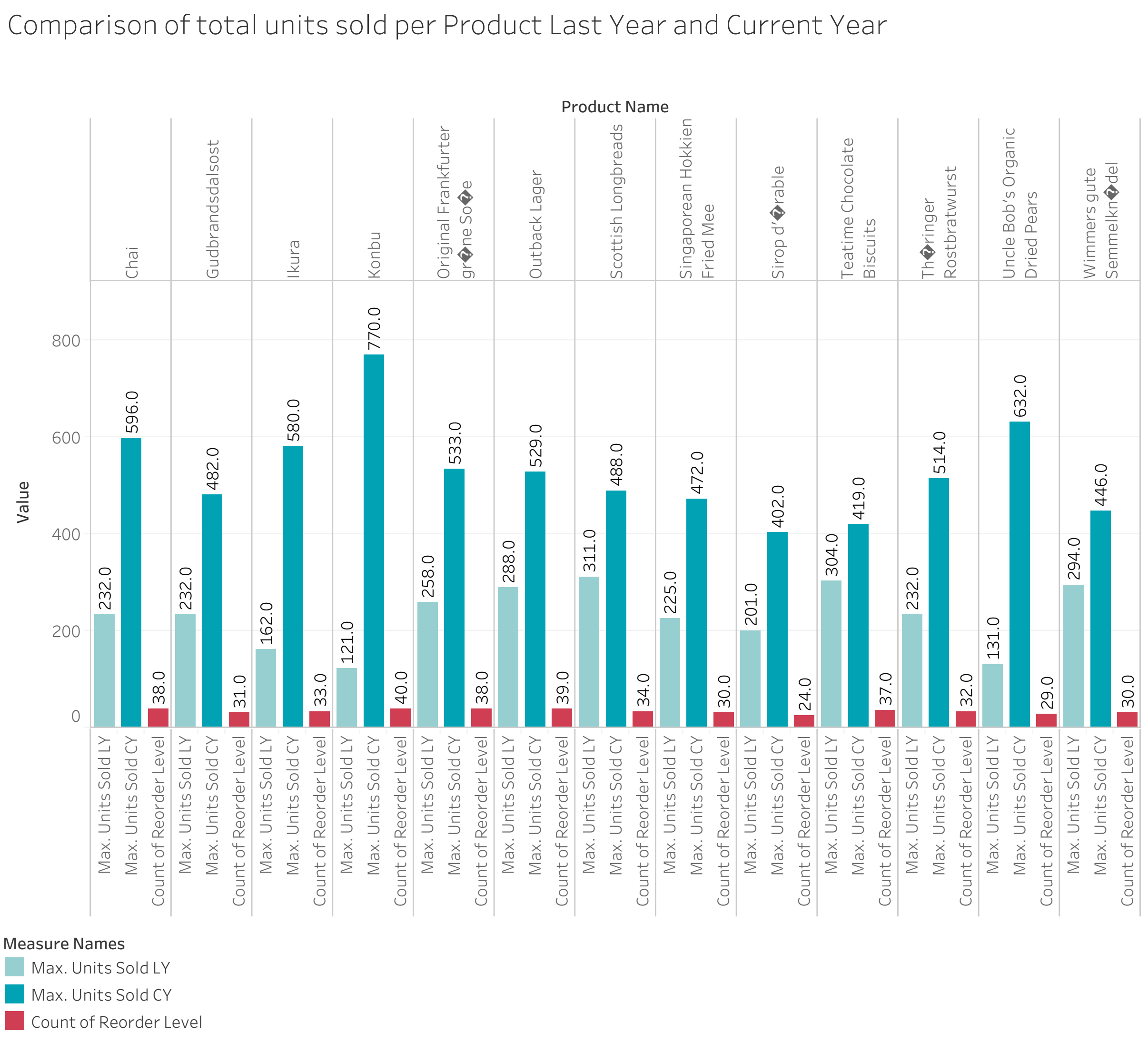
According to the Gartner Magic Quadrant for Data Management solutions for Analytics (Feb 2017 edition) **Oracle** are the market leaders. This comparison will be between Oracle Data warehouse and Microsoft SQL Solutions.

Oracle Data Warehouse	Microsoft SQL server
Oracle is suited for very large databases and provides scalability and performance with larger data resulting in better user query load.	SQL Server delivers large amounts of data by moving querying logic into stored procedures and views to enhance performance.

Pros	
• Manages large data sizes efficiently in terms of performance, high availability and manageability perspective which is accomplished through Oracle Partitioning functionality. • Uses Oracle In-Memory functionality to support for dual-format architecture. • Reduces storage footprint and increases performance including query performance using multiple compression capabilities.	• Using the correct hardware and configuration SQL server can be very powerful. • SQL is an open source solution, it can handle large number of request and obtain results at great speeds. • SQL server Management studio manages databases with great flexibility. • Provides Hands on training to users.
Cons	
• Limitation of 8 columns on the storage indexes. • Addition features would increase the licensing cost.	• Importing Spread sheets using SSIS, SQL server populates the initial few rows and generates an error. • SQL server is very strict with database object names.

5. **Part 4: Reporting and Analysis**

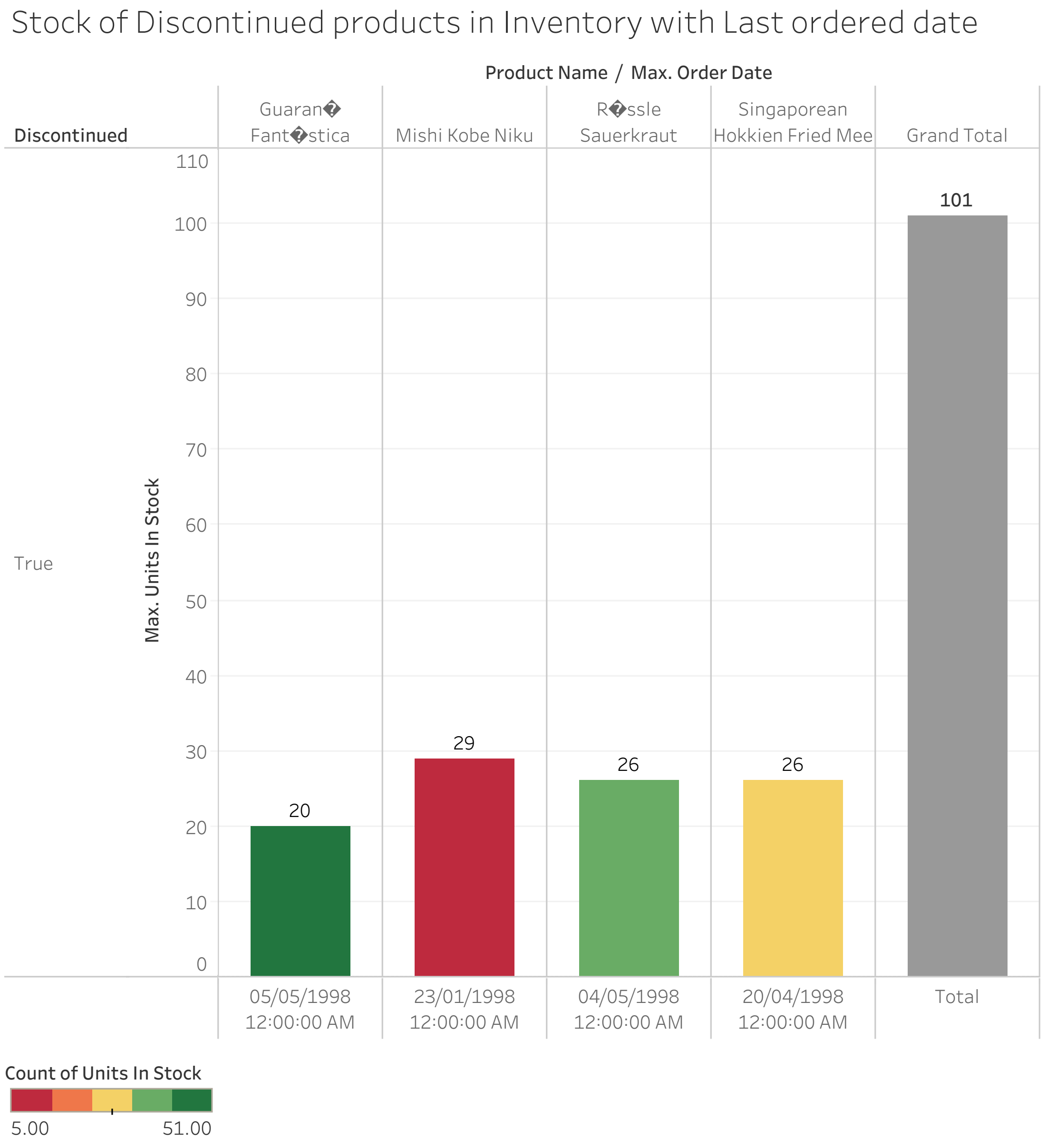
Using Tableau (Business Intelligence Tool) we can produce reports and analyze the efficiency of the Inventory management in North Wind Traders.



Comparing the number of units sold Last year (July 1996 – June 1997) and the units sold current year (July 1997 – June 1999) we can see that the demand for the products like Chai, Gudbrandsdalsost, Ikura, Konbu, Outback Lager, Sirop d'rable, Rostbratwurst, Uncle Bob's Organic Dried Pears have had a substantial increase (almost double) in demand when compared to the previous year.

Since the comparison of the units sold is for a specific time frame we can eliminate the presence of seasonal goods.

The Company has a fixed Reorder level, i.e. when the Units in stock falls below the Reorder level the company will order the required stock from the suppliers. Even though there was an increase in demand for these products the reorder level remained the same, which means that the company had to reorder the products more frequently. Resulting in a time delay to restock the products which in turn will affect the timely delivery.



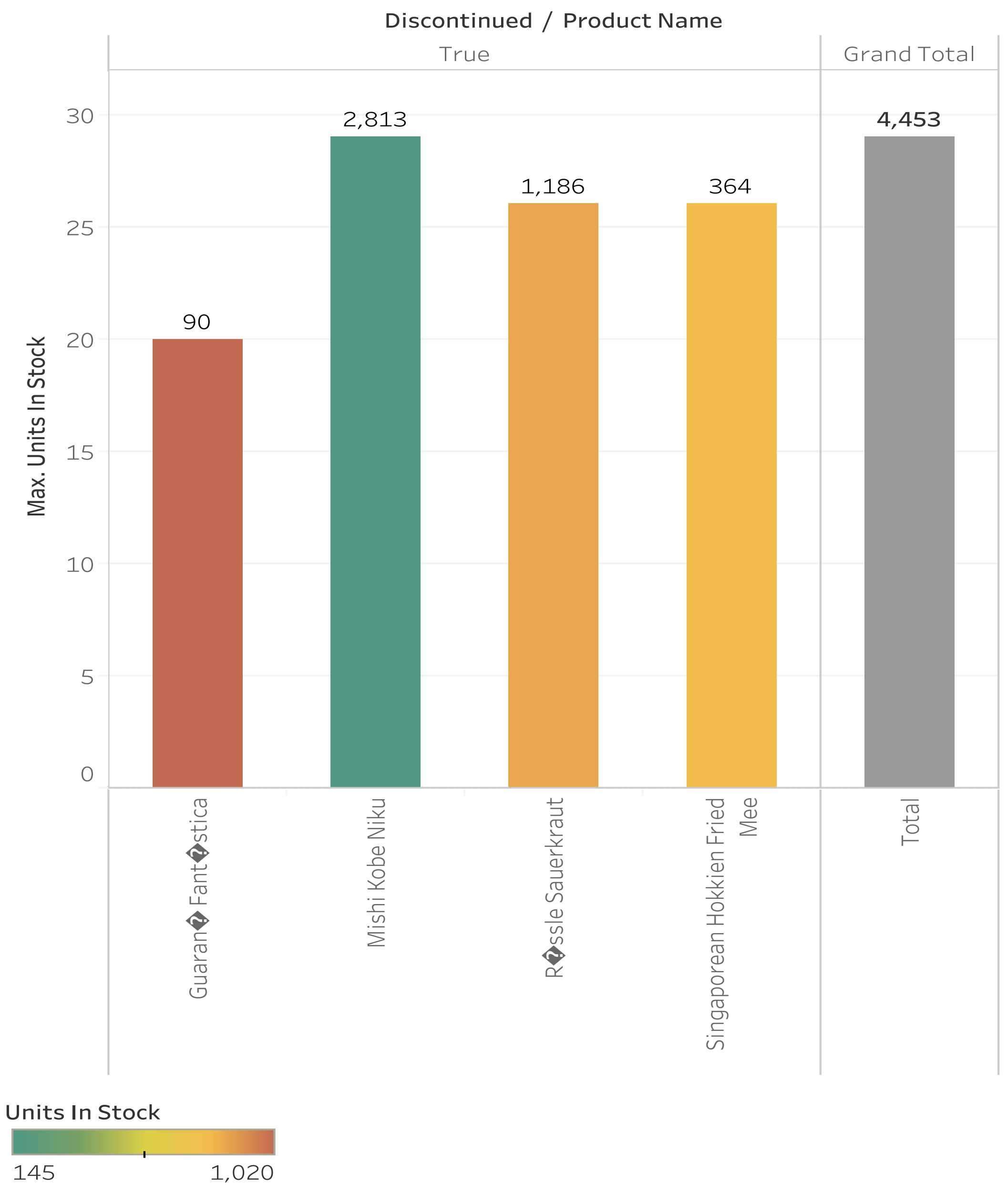
From the Bar chart, it is evident that there are quite a few discontinued products in the inventory. The Bar chart explains how many units of the discontinued products are there in the Inventory and when was the product last ordered (To give us an idea how long were the products in the Inventory for). There is a total of 101 units of Discontinued products in the Inventory.

There are 29 units of Mishi Kobe Niku in the inventory, the last time the product was ordered on 23rd January 1998. 26 Units of Singaporean Hokkien Fried Mee, last ordered on 20th April

1998, 26 units of Rossle Sauerkraut, last ordered on 4th May 1998 and 20 units of Guaran Fantastica which was last ordered on 5th May 1998.

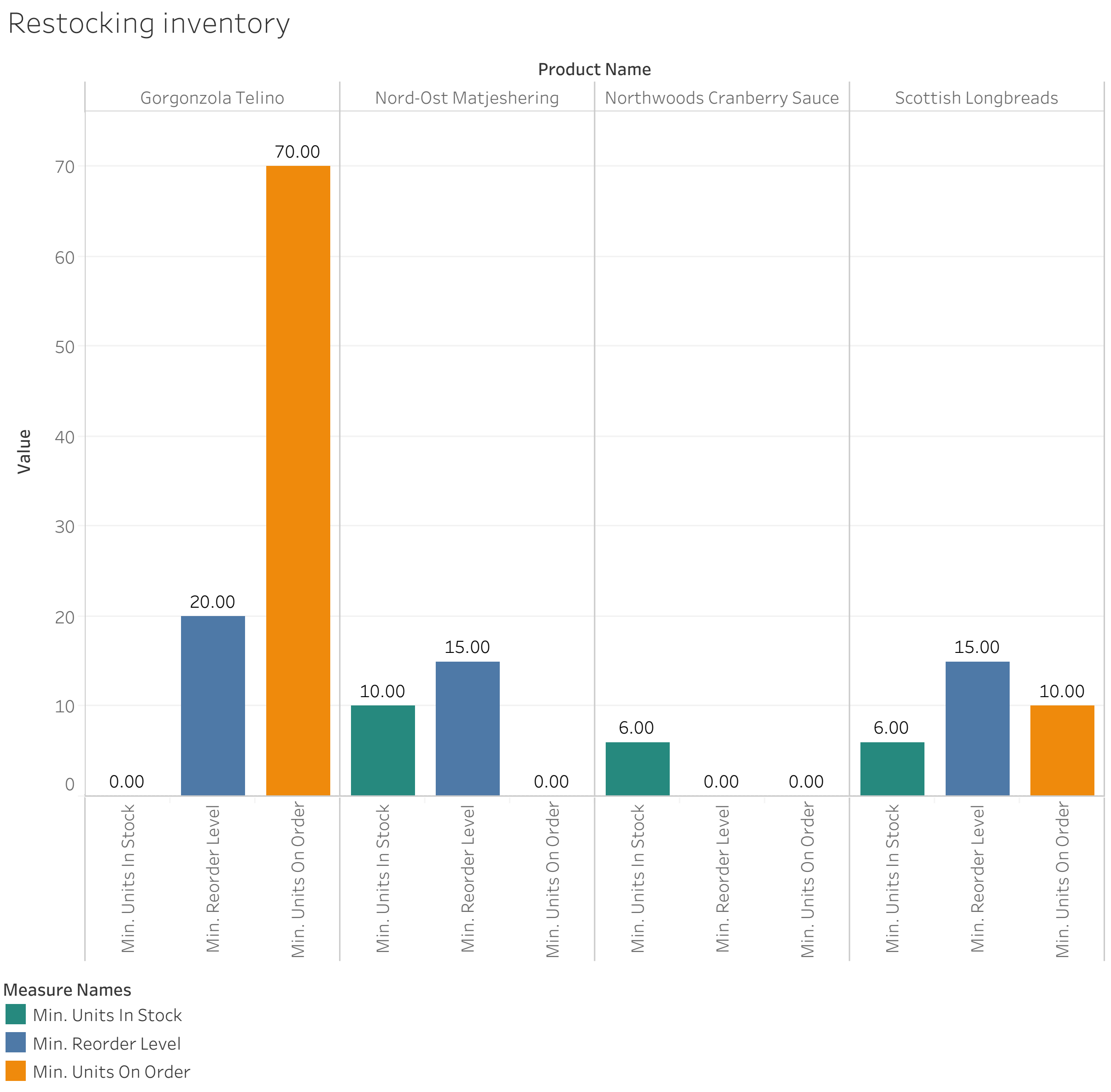
These 4 Products can be classified as dead stock, which means that they won't be made available for sale. These products should be cleared from the inventory at clearance rates which a loss incurred to the company. Even storing the discontinued products is a recurring cost to the company.

Amount spend on storing Discontinued products



A total of **$4453** worth of Discontinued products (101 units) are stored in the Inventory.

There are 29 units of Mishi Kobe Niku in the inventory, which has total unit cost of $2813. 26 Units of Singaporean Hokkien Fried Mee in the inventory, which has total unit cost of $ 364, 26 units of Rossle Sauerkraut in the inventory, which has total unit cost of $1186 and 20 units of Guaran Fantastica in the inventory, which has total unit cost of $ 90.



From the Bar chart, it is evident that for a few products aren't efficiently restocked based on the reorder level.

The product Gorgonzola Telino has a Reorder level of 20, and the Units in stock are 0. Even though the supplier has been notified about the restocking of the inventory, the customers will have to wait for the products which will drive them to other vendors.

The product Nord-Ost Matjeshering has units of stock below the Reorder level.

The product Northernwoods Cranberry Sauce doesn’t have a Reorder level and are stocked based on the demand. It may lead to delay in the delivery of the product.

The product Scottish long breads have a low reorder level and units or order which means that the frequency of ordering the product from the supplier will be comparative high.

5.1 Why Tableau

I have chosen Tableau as the Business Intelligence Tool for this part of the assignment. Being one among the three market leaders **Tableau** offers a very interactive and intuitive visualization experience to the business users. Using Tableau business users can access, prepare and analyze data without coding. Tableau aimed at making analytical workflow experience easier for users and providing users with great power to explore and find insights in data. The main advantages of Tableau are as follows:

- Gold standard for intuitive exploration: The core product strength of Tableau is its intuitive interactive visualization and analytical dash boarding capabilities on any type of data source.
- Focus on customer experience and success: Tableau offers its customers various learning opportunities such as online tutorials, webinars and hands on classroom based training to help educate and empower the users.
- Expanding deployment and standardization rates: Some organizations use centralized teams and others prefer decentralized analysis by business users which results in expanding the deployment of Tableau and becoming a BI and analytics standard.
- Flexible deployment options: Tableau can be deployed by cloud or on premises. It provides deployment options for virtual machines such as AWS and Microsoft Azure and support for Google Cloud platform.

6. Part 5: Modelling and Graph Data Models

6.1 Limitations of ER and star schema dimensional models

The **Entity-Relationship** (E-R) modelling also known as the Top down approach is a graphical technique used to convert the requirements of the system into graphical representations to make it understandable. The E-R model is a conceptual data model that views the real world as entities and relationships.

Limitations of E-R model:
- No exact format is followed by Industries for developing an E-R diagram
- E-R model does not model a Business, it models the micro relationships between data.
- E-R models are highly variable in structure i.e. each data warehouse needs custom handwritten and tuned SQL codes. It means that the model is vulnerable to change.
- The E-R model is popular for high-level design.

The **Star schema** is one of the simplest Data warehouse schema. It consists of one central fact table from which the four dimension tables branch out to resemble a star shape. The Fact tables are in third normal form (3NF) and the dimension tables are de-normalized.

Limitations of Star schema:
- Data integrity is not enforced when compared to OLTP databases.
- Star schema is not flexible inters of analytical application.
- Due to the simplicity of the Star schema, it doesn't support many to many relationships between business entities.

6.2 Implementing the Northwind ER model as a graph on Neo4j.

A graph database is an online database management system with Create, Read, Update, and Delete (CRUD) operations working on a graph model. The Graph database consists of two elements:

- Node: Represents an entity.
- Relationship: How two nodes are related.

Advantages of using a Graph database:
- Performance: Graph DBs effectively stores the data relationships, hence their performance stays constant even with larger sizes of data.
- Flexibility: Graph DBs are flexible when expanding a data model in requirement with a business need.
- Agility: Graph DBs are equipped for frictionless development and graceful systems maintenance.

Creating the Northwind ER model as a graph using Neo4j, the ideal method is into import each of the table into csv format, since the files are available on the neo4j URL we can use it.

```
// Creating the nodes

// Create products

LOAD CSV WITH HEADERS FROM "http://data.neo4j.com/northwind/products.csv" AS
row
CREATE (n:Product)
SET n = row,
  n.unitPrice = toFloat(row.unitPrice),
  n.unitsInStock = toInteger(row.unitsInStock), n.unitsOnOrder =
toInteger(row.unitsOnOrder),
  n.reorderLevel = toInteger(row.reorderLevel), n.discontinued = (row.discontinued <>
"0")

// Create categories

LOAD CSV WITH HEADERS FROM "http://data.neo4j.com/northwind/categories.csv"
AS row
CREATE (n:Category)
SET n = row

// Create suppliers

LOAD CSV WITH HEADERS FROM "http://data.neo4j.com/northwind/suppliers.csv"
AS row
CREATE (n:Supplier)
SET n = row

// Create customers

LOAD CSV WITH HEADERS FROM " http://data.neo4j.com/northwind/customers.csv"
AS row
CREATE (n:Customer)
SET n = row

// Create employees

LOAD CSV WITH HEADERS FROM " http://data.neo4j.com/northwind/employees.csv"
AS row
CREATE (n:Employee)
SET n = row

// Create Orders
```

```
LOAD CSV WITH HEADERS FROM " http://data.neo4j.com/northwind/orders.csv" AS
row
CREATE (n:Order)
SET n = row


// Create Order details

LOAD CSV WITH HEADERS FROM " http://data.neo4j.com/northwind/order-
details.csv" AS row
CREATE (n:OrderDetails)
SET n = row

-----------------------------------------------------------

// Create Indexes on the created nodes

CREATE INDEX ON :Product(productID);

CREATE INDEX ON :Product(productName);

CREATE INDEX ON :Category(categoryID);

CREATE INDEX ON :Employee(employeeID);

CREATE INDEX ON :Supplier(supplierID);

CREATE INDEX ON :Customer(customerID);

CREATE INDEX ON :Customer(customerName);

-----------------------------------------------------------

// Creating Relationships between Products and Category

MATCH (p:Product),(c:Category)
WHERE p.categoryID = c.categoryID
CREATE (p)-[:PART_OF]->(c)

// Creating Relationships between Product and Supplier

MATCH (p:Product),(s:Supplier)
WHERE p.supplierID = s.supplierID
CREATE (s)-[:SUPPLIES]->(p)

// Creating Relationships between Product and Order Details
```

```
MATCH (p:Product), (od:OrderDetails)
WHERE p.productID = od.productID
CREATE (od)-[details:ORDERS]->(p)

// Creating Relationships between Employee and Order

MATCH (e:Employee),(o:Order)
WHERE e.employeeID = o.employeeID
CREATE (e)-[:COLLECTS]->(o)
// Creating Relationships between Customer and Order

MATCH (c:Customer),(o:Order)
WHERE c.customerID = o.customerID
CREATE (s)-[:COLLECTS]->(p)

// Creating Relationships between Order details and Order

MATCH (od:OrderDetails), (o:Order)
WHERE od.orderID = o.orderID
CREATE (od)-[:PART_OF]->(o)
```

6.3 Comparing the codes of Neo4j and SQL

- Getting a list of product categories by each supplier

```
MATCH (s:Supplier)-->(:Product)-->(c:Category)
RETURN s.companyName as Company, collect(distinct c.categoryName) as Categories
```

On Neo4j the code is very concise, and loads relative faster than SQL.

```
SELECT DISTINCT s.CompanyName,c.CategoryName
FROM Suppliers AS s
JOIN Products AS p ON s.SupplierID=p.SupplierID
JOIN Categories AS c ON p.CategoryID = c.CategoryID;
```

The Join commands in SQL makes the code complex and compared to Neo4j SQL take longer time to load.

- Getting a list of suppliers who supply 'Produce' products.

```
MATCH (c:Category {categoryName:"Produce"})<--(:Product)<--(s:Supplier)
 RETURN DISTINCT s.companyName as ProduceSuppliers
```

On Neo4j, the filter is in the join statement. Quicker execution time.

```
SELECT DISTINCT s.CompanyName
FROM Suppliers AS s
JOIN Products AS p ON s.SupplierID=p.SupplierID
JOIN Categories AS c ON p.CategoryID = c.CategoryID
WHERE CategoryName='Produce';
```

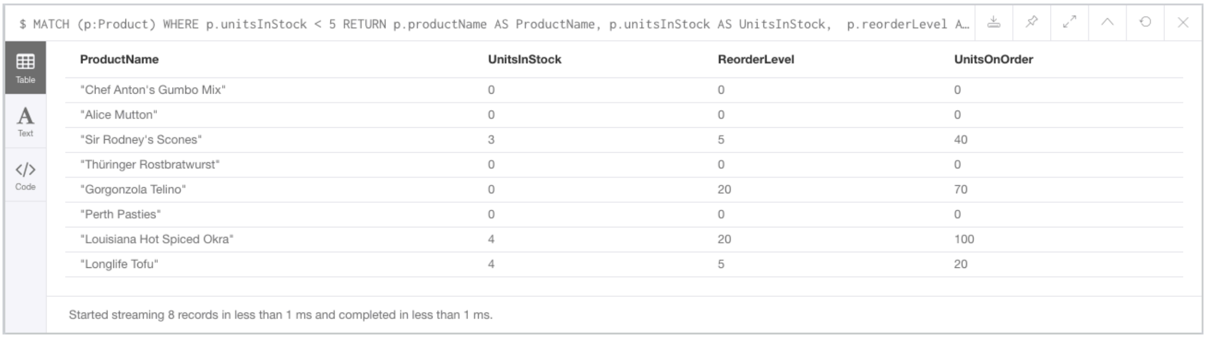
In SQL language, filter WHERE is used at the end of the query. Longer Execution time.

7. **Part 6: Graph information retrieval and analysis**

Using Cypher, Neo4j's querying language we can run a few analyses based on the selected Business driver (Inventory Management). Cypher is a declarative, SQL-inspired language for describing patterns in graphs visually using asci-art syntax.

- Getting the list of products which has less than 5 units in stock along with their reorder level and Units on order

```
MATCH (p:Product)
WHERE p.unitsInStock < 5
RETURN p.productName AS ProductName, p.unitsInStock AS UnitsInStock,
p.reorderLevel AS ReorderLevel,p.unitsOnOrder AS UnitsOnOrder;
```



```
$ MATCH (p:Product) WHERE p.unitsInStock < 5 RETURN p.productName AS ProductName, p.unitsInStock AS UnitsInStock,  p.reorderLevel A…
```

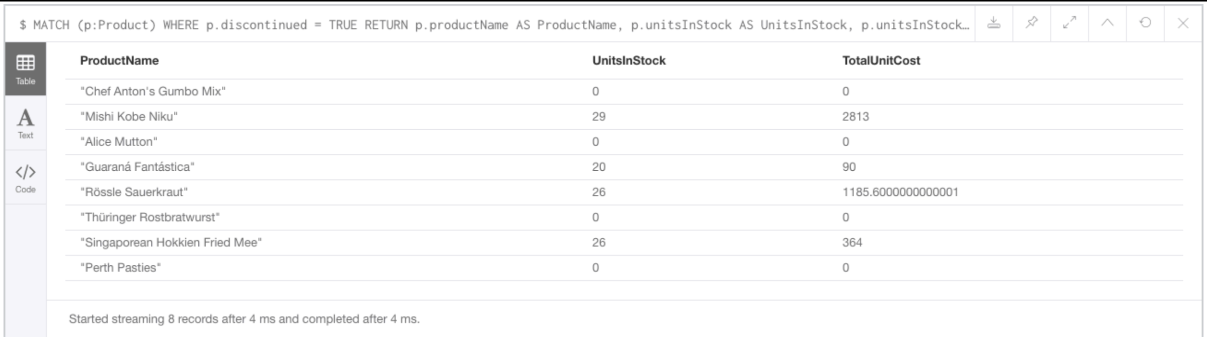
ProductName	UnitsInStock	ReorderLevel	UnitsOnOrder
"Chef Anton's Gumbo Mix"	0	0	0
"Alice Mutton"	0	0	0
"Sir Rodney's Scones"	3	5	40
"Thüringer Rostbratwurst"	0	0	0
"Gorgonzola Telino"	0	20	70
"Perth Pasties"	0	0	0
"Louisiana Hot Spiced Okra"	4	20	100
"Longlife Tofu"	4	5	20

Started streaming 8 records in less than 1 ms and completed in less than 1 ms.

The result shows that 8 records were found in less than 1ms. The products 'Longlife tofu' and 'Louisiana Hot Spiced Okra' have 4 units in stock, since the units in stock is low the products have been ordered from supplier as shown on the Units in order. 4 products have 0 units in stock and 0 units in order which suggests that the product is no longer in sale.

- Getting the list of discontinued products along with their units in stock and the total amount spent on the inventory of those products

```
MATCH (p:Product)
WHERE p.discontinued = TRUE
RETURN p.productName AS ProductName, p.unitsInStock AS UnitsInStock,
p.unitsInStock*p.unitPrice AS TotalUnitCost;
```



```
$ MATCH (p:Product) WHERE p.discontinued = TRUE RETURN p.productName AS ProductName, p.unitsInStock AS UnitsInStock, p.unitsInStock…
```

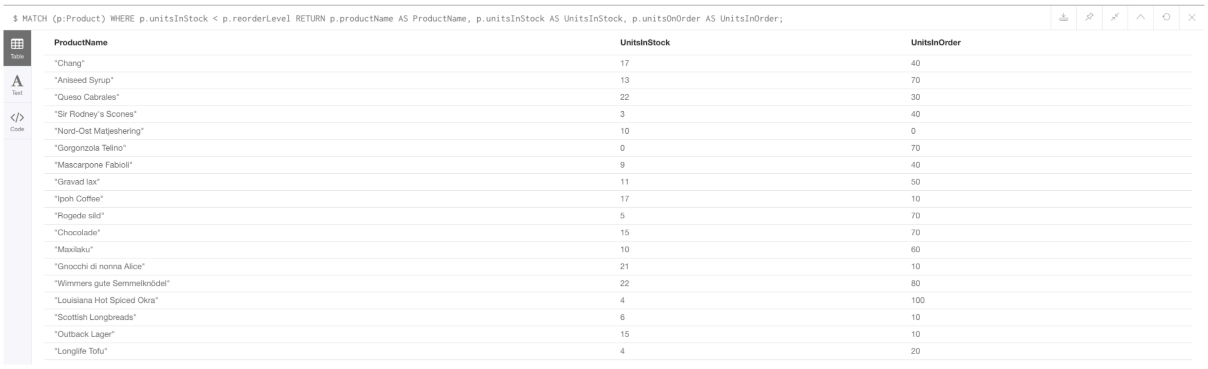
ProductName	UnitsInStock	TotalUnitCost
"Chef Anton's Gumbo Mix"	0	0
"Mishi Kobe Niku"	29	2813
"Alice Mutton"	0	0
"Guaraná Fantástica"	20	90
"Rössle Sauerkraut"	26	1185.6000000000001
"Thüringer Rostbratwurst"	0	0
"Singaporean Hokkien Fried Mee"	26	364
"Perth Pasties"	0	0

Started streaming 8 records after 4 ms and completed after 4 ms.

The result shows that there are 8 products which were discontinued, out of which 4 products (Mishi Kobe Niku, Guarana Fantastica, Rossle Sauerkraut, Singaporean Hokkien Fried) have 29, 20, 26 and 26 units in stock with an inventory cost of $2813, $90, $1186 and $364.

- Getting a list of products which have their Units in stock lower than the reorderlevel along with their Units in order.

```
MATCH (p:Product)
WHERE p.unitsInStock < p.reorderLevel
RETURN p.productName AS ProductName, p.unitsInStock AS UnitsInStock,
p.unitsOnOrder AS UnitsInOrder;
```



```
$ MATCH (p:Product) WHERE p.unitsInStock < p.reorderLevel RETURN p.productName AS ProductName, p.unitsInStock AS UnitsInStock, p.unitsOnOrder AS UnitsInOrder;
```

ProductName	UnitsInStock	UnitsInOrder
"Chang"	17	40
"Aniseed Syrup"	13	70
"Queso Cabrales"	22	30
"Sir Rodney's Scones"	3	40
"Nord-Ost Matjeshering"	10	0
"Gorgonzola Telino"	0	70
"Mascarpone Fabioli"	9	40
"Gravad lax"	11	50
"Ipoh Coffee"	17	10
"Rogede sild"	5	70
"Chocolade"	15	70
"Maxilaku"	10	60
"Gnocchi di nonna Alice"	21	10
"Wimmers gute Semmelknödel"	22	80
"Louisiana Hot Spiced Okra"	4	100
"Scottish Longbreads"	6	10
"Outback Lager"	15	10
"Longlife Tofu"	4	20

The result shows that there are 18 products which has the units in stock lower than the reorder level that means the supplier must be notified to restock the inventory. Out of the 18 products, 17 have been reordered except for 'Nord-Ost Matjeshering'.

7.1 The role of graph databases in Hadoop-Spark Ecosystem

- Neo4j had made new contributions to the Hadoop Ecosystem enabling graph analytic capabilities for Spark, resulting in Cypher being an in-memory analytic engine.

- With the release of Cypher for Apache Spark (CAPS) Language toolkit a combination that will help Bid data analyst use graph and graph algorithms and widening the connections in their data. Spark Joins Neo4j, SAP HANA, Redis and AgensGraph in supporting Cypher expand its reach as **openCypher**.

- Data scientist using Spark couldn’t use graph pattern matching till now, with the use of Cypher for Apache Spark the data scientists can iterate data easier and connect adjacent data sources to their graph applications more efficiently.

- Cypher for Apache Spark enables the data scientists to return graphs along with the queries allowing users to chain queries together with in-memory Spark-based graph representations between steps which lets Spark users work on graph analytics with ease and directly connected with their Hadoop environment.

8. **Reference**

Introduction

- Stedman, Craig, and Ed Burns. "What Is Business Intelligence (BI)." *SearchBusinessAnalytics*, Margaret Rouse, 1 Aug. 2017, searchbusinessanalytics.techtarget.com/definition/business-intelligence-BI. (Accessed on 12th April 2018)

Part 1: Business Driver

- TradeGecko. "What Is Inventory Management?" *What Is Inventory Management? - Know the Basics | TradeGecko*, www.tradegecko.com/learning-center/what-is-inventory-management. (Accessed on 12th April 2018)
- "What Is Inventory Control & Why Is It So Important?" *Handshake Blog*, www.handshake.com/blog/what-is-inventory-control/. (Accessed on 12th April 2018)

Part 2: Data Modelling

- "Star Schema." *Data Warehouse Schema Architecture - Star Schema*, datawarehouse4u.info/Data-warehouse-schema-architecture-star-schema.html. (Accessed on 12th April 2018)

Part 3: Implement Tables and ETL Procedures

- "Extract, Transformation, and Load System." *The Data Warehouse Toolkit: the Definitive Guide to Dimensional Modeling*, by Ralph Kimball and Margy Ross, Wiley, 2013, pp. 19–21.
- Babar, Shane. "Microsoft SQL Server Database Advantages And Best Practices." *QuickStart Technology Training*, 18 Jan. 2018, www.quickstart.com/blog/post/microsoft-sql-server-database-advantages-and-best-practices/. (Accessed on 17th April 2018)
- "Oracle Data Warehouse vs Microsoft SQL Server." *TrustRadius*, www.trustradius.com/compare-products/oracle-data-warehouse-vs-sql-server. (Accessed on 17th April 2018)

Part 4: Reporting and analysis

- "Tableau." Magic Quadrant for Business Intelligence and Analytics Platform 2017(pdf), pp. 46.

Part 5: Modelling and Graph Data Models

- Thakur, Dinesh. " What Is ER-Model?Advantages and Disadvantages of E-R Model. ." *Computer Notes*, ecomputernotes.com/fundamental/what-is-a-database/advantages-and-disadvantages-of-e-r-model. (Accessed on 28th April 2018)
- Kimball, Ralph. "A Dimensional Modeling Manifesto." *Kimball Group*, 26 Jan. 2016, www.kimballgroup.com/1997/08/a-dimensional-modeling-manifesto/. (Accessed on 28th April 2018)
- S, Vithal. "Data Warehouse Star Schema Model and Design." *DWgeek.com*, 15 Mar. 2018, dwgeek.com/star-schema-model-data-darehouse.html/.

Part 6: Graph information retrieval and analysis

- "Why Graph Databases?" *Neo4j Graph Database Platform*, neo4j.com/why-graph-databases/. (Accessed on 29th April)

- Smith, Tom. “How Neo4j Is Making Graph Technology More Accessible.” *Dzone.com*, D-Zone, 24 Oct. 2017, dzone.com/articles/how-neo4j-is-making-graph-technology-more-accessib. (Accessed on 29th April)