ISYS90086
Data Warehousing

Assignment 1

Jie Niu (890649)
Chiyu Chen (901265)

Content

1. Executive Summary

There is a Melbourne-located winery that sells 3 types (7 brands) of wine. The managing director of this winery is looking for further sales growth, he believes that the winery has to make a huge improvement on the business management and the comprehension of the wine industry trend. The data warehouse can provide high-quality and consistent data from various data sources. Managers can quickly access many data sources and a mass of data in regard of history records in data warehouse. This can help managers no longer has to lean upon limited data and make business decisions intuitively. This report aims to explain the necessity and feasibility of building a data warehouse through this case. Based on the winery's original production system and merchant sales system, a snowflake schema was selected to establish a suitable data warehouse for Overhill winery.

This report elaborates the establishment of the winery data warehouse in five parts. The executive summary is the first part, which briefly introduces key information and suggestions, and lists the report structure. The next part is the relevant content of the data warehouse, which covers the specific concepts, necessity and benefits of the data warehouse for the enterprise. After that, the third part is to detail the considerations when designing this data warehouse. This report applies a snowflake schema data warehouse for this winery and explained the content and details of related tables in the data warehouse. The main content of the fourth part is to present how to address a series of issues from a data warehousing view. At the end of this report, there are three appendixes to show the data dictionary, SQL statements and the detailed contribution of each team member.

2. Introduction to data warehousing

Definition

Data warehouse is a subject-oriented, integrated, relatively stable, data collection that reflects historical changes. A data warehouse is not simply a collection of information from different databases. It is a large-scale application system. The data warehouse is based on a large-scale management system, and in this system stores comprehensive data obtained from all the business databases of the enterprise. Through these comprehensive data, users are provided with useful information after processing, and users can analyze it (Golfarelli,2009).

The data warehouse can be understood in two aspects. The data in the data warehouse is integrated into multiple heterogeneous data sources. The integrated data is grouped according to the theme. The data placed in the data warehouse is generally not modified and contains historical data (N. Nataraj,2014).

On the other hand, the data warehouse is an analysis-oriented data processing method, which focuses on supporting decision-making rather than dealing with daily business.

Necessity

With the development of digital technologies and the expansion of demand for data in recent years, it is much more difficult to manage data than before. The main purpose of business database is designed for storing data and basic functions. It can be applied for some business analysis but requires many readjustments for that. Besides, it still has many drawbacks such as complicated data structures, data mess, shortage of history and slowness for large-scale queries. Gradually, the common business database no longer meets managers' needs. This is why data warehousing was created. Data warehousing is analysis-oriented and focuses on solving the drawbacks of analysis by business database. For more than storing data and recording history, managers are eager to pick the key information and extract them for decision-making. Those needs for strategic data information motivate the establishment of data warehousing in a great extent (Qin, 2012).

Additionally, the granularity of operations is on all integrated data level. It can be accessed easily and provides fast queries for their users. That makes data warehousing more attractive and popular in business analysis.

Benefits

The major benefit of data warehousing is to help directors and managers to make strategic decisions significantly. A data warehousing can provide enhanced business intelligence. It is not necessary to make decision intuitively for them. Data warehousing

can be used in several business steps such as financial management, sales and other business processes (Donald P, 1999).

Another significant strength is to improve the quality and consistency of data. To establish a data warehousing, the data from many other data source systems need to be transferred to a uniform data format. Due to the standardization of data from different departments, data warehousing provides more accurate data and accuracy is the basis of business strategic decisions.

3. Design of the data warehouse

In this part, this business case will be introduced and analyzed briefly, and we will discuss detailly how we design the data warehouse. Additionally, the fact table and dimensional tables are introduced separately below.

Business case introduction and analysis

Overhill Winery is a medium-sized and Melbourne-located winery. There are seven types of wines in total. The winery is in charge of planting grapes, producing wines and selling them. The wine products are sold locally in Melbourne area, interstate markets in the rest parts of Australia and internationally especially in the UK. The managing director is seeking more further growth based on the grammatic growth in the past three years.

Currently, there are two separate information systems applied in the business, one is to manage production and the other one is for sales to wine merchants. However, that causes a range of issues about different data formats and database systems.

In order to keep a rapid growth, the managing director has to make decisions based on the extracted key information instead of experience and intuition. For this purpose, a data warehouse is needed to integrate all different data from two systems and to offer valuable information about customers, sales agents and sales time periods and markets.

Overall Design

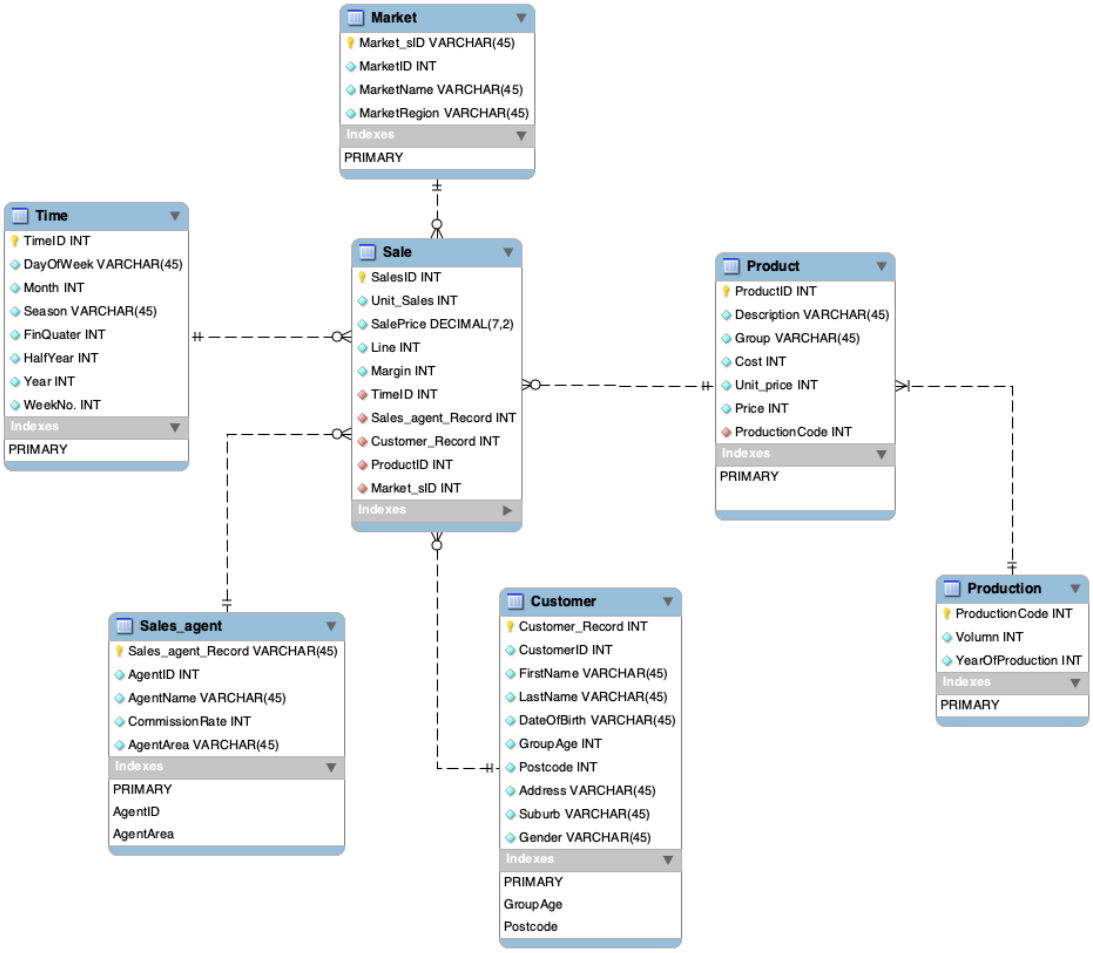

Figure 3.1 design of dimension model

As shown in figure 3.1, the Overhill winery's data warehouse is based on the snowflake schema to support Sane Yardman. He is able to better understand sales trends and solve problems facing the company. The data warehouse is composed of 7 tables. At the center is a *Sale* fact table, and 5 dimension tables connected to it, which are called *Market*, *Customer*,*Time*, *Sales_agent*, and *Product.* There is also a dimension table named *production* connected to the product dimension table. The details about Overhill winery's data warehouse and the relationship between the tables are described below.

Fact table

At the center of the chart is the fact table called *Sale*, which is the center around other dimensions. The *Sale fact* table mainly contains sales data of Overhill winery, which is also the business process that Overhill winery focuses on.

There are 10 attributes in the *Sale* table. For the Sale fact table, *SalesID* is the only primary key. A row in the *Sale* table indicates that a specific customer or agent bought a specific product within a specific time. These products sold to specific customers or agents are sold to a specific market for Overhill winery. This corresponds to 5 foreign keys in the table, namely *CustomerRecord*, *AgentRecord*, *TimeID*, *ProductID*, and *MarketID*. The remaining other attributes in the *Sale* fact table are measures. The sales volume of the product is represented by the *Unit_Sales* attribute, *SalePrice* indicates the total cost of the customer in the transaction, *Line* indicates the order line, and *Margin* is referring to the profit from this transaction.

Dimension tables

The *Customer* dimension table is used to record the specific information of customers in the transaction.. The basic information of customers is recorded in this table, such as *CustomerID*, *Firstname*, *Lastname*, and *addresses*. Key customers can be analyzed from the following aspects. First of all, the proportion of products purchased by men and women should be a direction of analyzing the main customers, so the *gender* attribute is added. At the same time, the age range of the customer is also a direction that should be paid attention to. We use the *DateOfBirth* and *GroupAge* attributes to describe it. The value of Group Age is divided into three intervals less than 30 years old, between 30 and 60 years old, and over 60 years old. It is obtained by subtracting the value of the *DateOfBirth* attribute from the value of the time in the *Sale fact* table. Finally, the area where customers live is also a direction that needs attention, by adding *Postcode* and *Suburb.* Because the customer's information may change, it is important to keep both new data from customer changes and historical data of the customer, so add *Customer_Recorder* as the surrogate key, and it is also the primary key of the Customer dimension table. For search convenience, set up *CustomerID*, *GroupAge* and *Postcode* as indexes.

The *Time dimension* table is one of the dimensions of *Sale fact*. The problem that companies need to solve is What time is the most profitable. So, for a specific date, it needs to be analyzed from the perspective of the day of the week, which month, which quarter, which fiscal quarter, and the year it belongs to. Which week of the year it is, whether it belongs to the first half or the second half. Therefore, a total of 8 attribute implementations have been added, Where as the primary key of the *Time* dimension table is *TimeID*, followed by *DayOfWeek*, *Month*, *Season*, *FinQuate*r, *Year*, *WeekNo*, and *HalfYear*. The 7 attributes correspond to the above description.

The *Sales_agent* dimension table is one of the dimensions of *Sale fact*. Sales agents are the main sales objects of Overhill winery. One issue that also needs to be analyzed is Who are the key sales agents. So, the basic information of the agents recorded in the table includes *AgentID*, *Agent'Name*, and *CommissionRate*. At the same time, we can analyze the importance of agents from different regions by adding the *AgentArea* attribute. For the change of Agent, we also need to record new data and historical

data, so *Sales_agent_record* is set as the surrogate key, and it is also the primary key. For quick retrieval, we set *AgentID* and *AgentArea* as indexes.

The *Market* dimension table is used to record in which region the product is sold in a specific transaction. This table records the *MarketID*, the *MarketName* and the *MarketRegion*. As the director requires, this dimension table shows which market each unit sales and dollar sales belong to for every month. Market records may change in the future, so there should be a surrogate key called *Market_sID*to present changes. For retrieving concern, the index is added in the *MarketID*.

The dimensional table of *Product* is to record all the product relevant information. In this table, basic information of product is recorded, covering their original id, descriptions, which group they belong to, the cost to produce them, unit price which is also called advised price and the real price for sale. According to the real sale price and cost, the margin for each product can be calculated. In order to retrieve, the index is added to the column *ProductID*.

The *Production* dimension table is to record the information about producing wine process. The *ProductionCode* is labelled as the primary key in this table. It has two attributes, the *Volume* is to describe the yearly yield of wines, the *YearOfProduction* is to record the year of each production. The *Production* dimension table has a hierarchical relationship with *Product* dimension table. The quantity of production must be always greater than the quantity of product. So, the *ProductionCode* in this table is a FK (foreign key) to the dimensional table of *Product*. For retrieving purpose, the index in this table is added in the *ProductionCode.*

4. Address business problems

Which products are the most profitable?

Table	Retrieval Attributes
Sale	Unit_Sales, SalePrice,Margin,TimeID,ProductID
Time	TimeID, Year, Season, Month
Product	ProductID, Cost, Description, Group

Table 4.1

The steps for grouping most profitable product:

1. **Decide the time period**
   Choose a time period that decision maker would like to check and select the corresponding time period on *Time* table. For instance, if the director wants to check the sales on April of 2019, then select rows where 'Month' = '4' and 'Year' = '2019'.
2. **Select the information of sales during that time period**
   Based on the result of the last step, choose the corresponding rows of *Sale* fact table, where TimeID equals the results of the last step.
3. **Group by product and aggregate the relative sale information**
   Based on the result of step2, group all information by ProductID and sum up all required sale information respectively. Therefore, that forms a new table that can be called *Key_Products*. After that, each product has its own sale information such as unit sales, dollar sales and margin.
4. **Sort sales information**
   After grouping them by ProductID, sort them in descending order by any one of them, unit sales, dollar sales or margin. Then the top one is what the director looks for on that order.
5. **Find out more information about the key product**
   Join *Key_Product* with *Product* table, then we will get the detailed wine type (called Group *Product* table) and the base product (called Description in *Product* table) for the key product.

The final provided information as below:

Base product	Wine type	Unit sales	Dollar sales	Cost	Margin

Who are the key customers?

Table	Retrieval Attributes
Sale	Unit_Sales, SalePrice,Margin,TimeID,Customer_Record,ProductID
Costumer	CustomerID,GroupAge,Postcode
Time	TimeID, Year, Season
Product	ProductID, Cost, Description, Group

Table 4.2

The steps for grouping key customers:

1. **Decide the time period and check the productID for each product.**
   Choose a time period that the decision-maker would like to check and select the corresponding time period on the *Time* dimension table. Taking the 'winter of 2012' for an example, select the 'Year' = '2012' and 'Season' = 'winter' in the *Time* table.
2. **Select the sales information during that period.**
   Based on the results of step1, select the corresponding rows of *Sale*, where TimeID equals the results of that in *Time* table. After that, select the productID which the director is looking for.
3. **Group by customers and calculate the relative sale information**
   Based on the results of step2, group all data by customers. For each customer, calculate the unit sales, dollar sales, cost and margin by summing up Unit_Sales, SalePrice, Cost and Margin respectively in that time period, which is a new table called *Key_Customers.*
4. **Sort the results from the last step.**
   Sort the results from step3 in descending order by any one of them, unit sales, dollar sales or margin. The first customer could be considered as key customers.
5. **Find out the information about key customers for each product.**
   Join Key_Customers with *Customer* table and *Product* table. Then, group them by GroupAge or Postcode so that we can reckon that the rough age and place of the key customers for each product.

The final provided information as below:

CustomerID	Description (wine type)	Group (base product)	Unit Sales	Dollar sales	Cost	Margin	GroupAge	Postcode

Which market is the most profitable?

Table	Retrieval Attributes
Sale	TimeID, Unit_Sales, SalePrice
Time	TimeID, Month, Year
Market	Market_sID, MarketID, MarketName

Table 4.3

The steps for grouping the most profitable market:

1. **Decided to year**
   Assuming that the previous year is 2019, select all TimeIDs with 'Year' = '2019' from the *Time* dimension table as the new set.
2. **Add month condition**
   Add a new condition of Month = 1 to the set returned in step 1, to obtain a new TimeID set.
3. **Select the sales information during that period**
   Connect the result in Step 2 with the *Sale* fact table and filter out some rows from the Sale fact table according to the result in Step 2. Then group by MarketID.
4. **Calculate unit sales and dollar sales in the same market for one month in the same year**
   In the result of step 3, based on the same Market_sID, the values of the Unit_Sales and SalePrice attributes of the same Market_sID are summed.
5. **Add the corresponding Market Name**
   According to the result of step 4 and the Market_ID, join the *Market* dimension table to the result of step 4, and get the MarketName.
6. **Find the most profitable market**
   The Unit_Sales or SalePrice attributes of the table generated in step 5 are sorted from large to small (the summed values). The result is called the *mpm* table, and the most profitable market is obtained. If you want to change the year and month, change it in the first or second step and search again.

The final search result should be displayed as:

MarketID	MarketName	Year	Month	Unit_Sales	SalePrice

Which time periods are the most profitable?

Table	Retrieval Attributes
Sale	TimeID, Unit_Sales, SalePrice, Margin
Time	TimeID, Month, Season, FinQuater, Year, WeekNo

Table 4.4

The steps to find the most profitable time period:

1. **Join Time dimension table to Sale fact table**
   According to the TimeID, join the *Time* dimension table to the *Sale* fact table, so that you can get the values of Month, Season, FinQuater, Year, WeekNo in the Time table.

2. **Group and merge based on time period**
   Group according to the retrieved time period. Here it is assumed that the retrieved time period is month. In the result obtained in step 1, the three columns of Unit_Sales, SalePrice, and Margin with the same Month attribute value are accumulated.
3. **Find the most profitable time period**
   According to the result of step 2, select any one of Unit_Sales, SalePrice, and Margin to sort from the largest to the smallest (the summed values), and we will get the new table as *the most profitable time period* table, and we will get the most profitable time period . If you need to retrieve other time periods, you only need to group them according to different time periods (Month, Season, FinQuater, Year, WeekNo) in step 2.

The final search result should be displayed as:

WeekNo /Month/Season/FinQuater/Year	Unit_Sales	SalePrice	Margin

Who are the key sales agents?

Table	Retrieval Attributes
Sale	Sales_agent_Record，Margin,
Sales_agent	Sales_agent_Record, AgentID, AgentName, CommissionRate

Table 4.5

The steps for grouping the Key sales agents:

1. **Join Sales_agent dimension table to Sale fact table**
   According to the same Sales_agent_Record, join the *Sales_agent* dimension table to the *Sale* Fact table, you can get the corresponding AgentID, AgentName, and CommissionRate values from the *Sales_agent* dimension table.
2. **Group and merge based on AgentID**
   According to the result of step 1, the results of step 1 are grouped according to the same AgentID value. And add the value of Margin according to the same AgentID.
3. **Find out the information about key sale agents**
   Based on the result of step 2, sort the values according to the Margin property from large to small, and get *Key sale agents* table. We can retrieve the required Key sale agents.

The final search result should be displayed as:

AgentID	AgentName	CommissionRate	Margin

Words count:3125

Appendix 1 – Data Dictionary

Sale Fact Table

Attribute	Description	Source
SalesID	A distinctive identification code for each sale	The Sales Order attribute in original sale system
Unit_Sales	The number of each product of each sale	The quantity of product in original sale system
SalePrice	The total amount of money a customer needs to spend on a product for each sale	The sale price in original sale system
Line	One transaction identifies one customer bought different product	The line attribute in original sale system
Margin	The profit made on a product in a sale	The Margin arrtibue in Sale table, the value is SalePrice minus Unit_Sales times Cost
TimeID	A unique identification code for a time	The TimeID attribute in Sale fact table, from Time dimension table.
Sales_agent_Record	A unique identification code r for an agent	The Sales_agent_Record is from Sales_agent dimensional table.
Customer_Record	A unique identification code for a customer	The Customer_Record in attribute Sale fact table, from Customer dimensional table.
ProductID	A distinctive identification code for a product	The ProductID attribute is from Product dimensional table.
Market_sID	A unique identification code for a market	The Market_sID is a surrogate is from Market dimension table.

Customer dimension table

Attribute	Description	Source
Customer_Record	Customer's surrogate key to prevent situations where the client changes information	The Customer_Record attribute in Customer table, data is automatically generated when loaded.
CustomerID	A unique identification code for a client	The CustomerID is from the original sales system.
FirstName	The FirstName of each client	The FirstName attribute is from original sales system.
LastName	The LastName of each client	The LastName attribute is from the original sales system.

DateOfBirth	The birthday of each client	The Date of Birth attribute is from the original sales system.
GroupAge	The age group of each client	The Group Age attribute is from the original sales system.
Postcode	The postcode of each client's address	The Postcode attribute is from the original sales system.
Address	The address of each client	The Address attribute is from the original sales system.
Suburb	The suburb of each client's address	The Suburb is from the original sales system.
Gender	The gender of each client's address	The Gender attribute is from the original sales system.

Product dimension table

Attribute	Description	Source
ProductID	A distinctive identification code for each product	The ProductID is from original production system.
Description	The specific brand of each product	The Description is from original production system.
Group	The specific types of each product	The Group is from original production system.
Cost	The specific cost of each product	The Cost is from original production system.
Unit_price	Recommended retail price for each product	The Unit_price is from original sales system.
Price	The actual retail price of each product	The Price is from the original sales system.
ProductionCode	A unique identification code for each production lot	The ProductionCode is an attribute of The Product dimension table, from The Production dimension table

Production dimension table

Attribute	Description	Source
ProductionCode	A unique identification code for each production lot	The ProductionCode is from original production system.
Volumn	The amount of each production	The Volumn is from original production system.
YearOfProduction	Year record of each production	The YearOfProduction is from original production system.

Time dimension table

Attributes	Description	Source
TimeID	The distinctive identifier for time	A distinctive number for data warehousing
DayOfWeek	The weekday for each sale	Aggregated for each transaction in weekdays
Month	The month for each sale	Aggregated for each transaction in months
Season	The season for each sale	Aggregated for each transaction in seasons
FinQuater	The financial quarter for each	Aggregated for each transaction in financial quaters
HalfYear	Categorize sale in half year	Aggregated for each transaction in half of year
Year	The calendar year for each sale	Aggregated for each transaction in years
WeekNo.	Measure sales weekly	Aggregated for each transaction in weeks

Sales_agent dimension table

Attributes	Description	Source
Sales_agent_Record	A numeric identifier for each change of agent	A surrogate key for each change of agent table
AgentID	A distinctive identifier for each agent	A distinct number generated in the original database
AgentName	The name of each agent	Extracted from the product system
CommissionRate	The commission rate for each agent	Gained form each transaction for each agent
AgentArea	The area where agent sales	Extracted from the product system about agent information

Market dimension table

Attributes	Description	Source
Market_sID	A numeric identifier for each change of market	A surrogate key for each change of market table
MarketID	A numeric identifier for each market	Extracted from the merchant sales system
MarketName	The name of each market	Extracted from merchant sales system about market information
MarketRegion	The region where each market is	Extracted from merchant sales system about market geographic information

Appendix 2 – SQL

1.

```
CREATE TABLE IF NOT EXISTS `mydb`.`Customer` (
  `Customer_Record` INT NOT NULL,
  `CustomerID` INT NOT NULL,
  `FirstName` VARCHAR(45) NOT NULL,
  `LastName` VARCHAR(45) NOT NULL,
  `DateOfBirth` VARCHAR(45) NOT NULL,
  `GroupAge` INT NOT NULL,
  `Postcode` INT NOT NULL,
  `Address` VARCHAR(45) NOT NULL,
  `Suburb` VARCHAR(45) NOT NULL,
  `Gender` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Customer_Record`))
ENGINE = InnoDB;

CREATE INDEX `GroupAge` ON `mydb`.`Customer` () VISIBLE;

CREATE INDEX `Postcode` ON `mydb`.`Customer` () VISIBLE;
```

2.

```
CREATE TABLE IF NOT EXISTS `mydb`.`Time` (
  `TimeID` INT NOT NULL,
  `DayOfWeek` VARCHAR(45) NOT NULL,
  `Month` INT NOT NULL,
  `Season` VARCHAR(45) NOT NULL,
  `FinQuater` INT NOT NULL,
  `HalfYear` INT NOT NULL,
  `Year` INT NOT NULL,
```

```
    `WeekNo.` INT NOT NULL,
    PRIMARY KEY (`TimeID`))
ENGINE = InnoDB;
```

3.

```
CREATE TABLE IF NOT EXISTS `mydb`.`Sales_agent` (
    `Sales_agent_Record` VARCHAR(45) NOT NULL,
    `AgentID` INT NOT NULL,
    `AgentName` VARCHAR(45) NOT NULL,
    `CommissionRate` INT NOT NULL,
    `AgentArea` VARCHAR(45) NOT NULL,
    PRIMARY KEY (`Sales_agent_Record`))
ENGINE = InnoDB;

CREATE INDEX `AgentID` ON `mydb`.`Sales_agent` () VISIBLE;

CREATE INDEX `AgentArea` ON `mydb`.`Sales_agent` () VISIBLE;
```

4.

```
CREATE TABLE IF NOT EXISTS `mydb`.`Market` (
    `MarketID` INT NOT NULL,
    `MarketName` VARCHAR(45) NOT NULL,
    `MarketRegion` VARCHAR(45) NOT NULL,
    PRIMARY KEY (`MarketID`))
ENGINE = InnoDB;
```

5.

```
CREATE TABLE IF NOT EXISTS `mydb`.`Production` (
    `ProductionCode` INT NOT NULL,
    `Volumn` INT NOT NULL,
    `YearOfProduction` INT NOT NULL,
    PRIMARY KEY (`ProductionCode`))
ENGINE = InnoDB;
```

6.

```
CREATE TABLE IF NOT EXISTS `mydb`.`Product` (
    `ProductID` INT NOT NULL,
    `Description` VARCHAR(45) NOT NULL,
    `Group` VARCHAR(45) NOT NULL,
    `Cost` INT NOT NULL,
    `Unit_price` INT NOT NULL,
    `Price` INT NOT NULL,
    `ProductionCode` INT NOT NULL,
    PRIMARY KEY (`ProductID`),
```

```
  CONSTRAINT `fk_product_production1`
    FOREIGN KEY (`ProductionCode`)
    REFERENCES `mydb`.`Production` (`ProductionCode`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE INDEX ON `mydb`.`Product` (`ProductionCode` ASC) VISIBLE;
```

7.

```
CREATE TABLE IF NOT EXISTS `mydb`.`Sale` (
  `SalesID` INT NOT NULL,
  `Unit_Sales` INT NOT NULL,
  `SalePrice` DECIMAL(7,2) NOT NULL,
  `Line` INT NOT NULL,
  `Margin` INT NOT NULL,
  `TimeID` INT NOT NULL,
  `Sales_agent_Record` INT NOT NULL,
  `Customer_Record` INT NOT NULL,
  `ProductID` INT NOT NULL,
  `MarketID` INT NOT NULL,
  PRIMARY KEY (`SalesID`),
  CONSTRAINT `fk_Sale_time`
    FOREIGN KEY (`TimeID`)
    REFERENCES `mydb`.`Time` (`TimeID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Sale_Sales_agent1`
    FOREIGN KEY (`Sales_agent_Record`)
    REFERENCES `mydb`.`Sales_agent` (`AgentID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Sale_customer1`
    FOREIGN KEY (`Customer_Record`)
    REFERENCES `mydb`.`Customer` (`CustomerID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Sale_product1`
    FOREIGN KEY (`ProductID`)
    REFERENCES `mydb`.`Product` (`ProductID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Sale_market1`
    FOREIGN KEY (`MarketID`)
```

```
    REFERENCES `mydb`.`Market` (`MarketID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE INDEX `fk_Sale_time_idx` ON `mydb`.`Sale` (`TimeID` ASC) VISIBLE;

CREATE INDEX `fk_Sale_Sales_agent1_idx` ON `mydb`.`Sale`
(`Sales_agent_Record` ASC) VISIBLE;

CREATE INDEX `fk_Sale_customer1_idx` ON `mydb`.`Sale` (`Customer_Record`
ASC) VISIBLE;

CREATE INDEX `fk_Sale_product1_idx` ON `mydb`.`Sale` (`ProductID` ASC)
VISIBLE;

CREATE INDEX `fk_Sale_market1_idx` ON `mydb`.`Sale` (`MarketID` ASC)
VISIBLE;
```

Appendix 3 – Work Breakdown

Both team members participate in this assignment and design the dimension model.
To be more specific:
Jie Niu (890649) completes the following tasks:
1. Design the dimension model and draw it on the MySQL workbench.
2. Complete the summary and data warehousing definition part of this report.
3. Justify the dimension tables.
4. Address the case- relative problems.
5. Accomplish the data dictionary and SQL statement.
Chiyu Chen (901265) completes the following parts:
1. Design the dimension model and draw it on the MySQL workbench.
2. Complete the basic conceptual part of this report.
3. Justify the dimension tables.
4. Address the case- relative problems.
5. Accomplish the data dictionary.

Reference:

[1] Donald, P. B., & G, K.T. (1999). Enhancing DataQuality in DataWarehouse Environments. *COMMUNICATIONS OF THE ACM*, 42(1), 73-78. Retrieved from https://www.researchgate.net/profile/Giri_Tayi/publication/220421583_Enhancing_Data_Quality_in_Data_Warehouse_Environments/links/00b7d52f8f94119f4f000000.pdf

[2] Nataraj,N. , & Nataraj, R.V. Analysis of ETL Process in Data Warehouse. *International Journal of Engineering Research and General Science*, 2(6), 279-282. Retrieved from http://ijergs.org/files/documents/ANALYSIS-34.pdf

[3] Qin, H., Jin, X., & Zhang, X. Research on Extract, Transform and Load (ETL) in Land and Resources Star Schema Data Warehouse. *Proceedings of the 2012 Fifth International Symposium on Computational Intelligence and Design*, 12(1), 120-123. Retrieved from https://doi.org/10.1109/ISCID.2012.38