
THE UNIVERSITY OF MELBOURNE

# ISYS90086

# Data Warehousing

**Summer Semester 2019**

## Assignment 1-Data Warehouse Design


| | |
|---|---|
| **Dinghao Yong** | **868878** |
| **Min Xue** | **897082** |

## Content

# 1. Executive Summary

Fantastic Fireworks is a firework selling company established in 1991 by Nancy Lightening with four stores in the capital city in Australia. With the continuous expansion of business and rapid growth of revenue，the owner tends to optimize the business further more in terms of opening times of stores, location of products and sites selection of new stores. The data warehouse which was used widely for the last few decades had been proved effectively and efficiently in accessing data and producing business analysis. The reasonable choice of architecture and modelling make contributions for the end-user to query in the data warehouse so as to support reporting and decision making. This document provides details in why and how to build a data warehouse for the owner of the company. A star schema is chosen as the design for the data warehouse which is implemented by using MYSQL. The data warehouse is created based on two existing information systems consisted of the inventory system and sales system and will be considered respectively.

The central purpose of this document is to introduce the importance and feasibility of establishing the data warehouse for the Fireworks, which will be realized through following parts. Firstly, some related information around the data warehouse will be explained from several perspectives, including the general concept, the necessity and benefits for using data warehouse. The following part is about the design of the data warehouse; moreover, different aspects of the dimensional modelling are demonstrated in detail. The next part is about how to use the data warehouse to address some specific problems about which the owner is concerned. The processes of querying the data warehouse to retrieve data are presented; besides, they are easily understood and can be implemented by the user. The last part is comprised of several appendixes, involving the data dictionary, the SQL statements and the work breakdown.

# 2. Introduction to data warehousing

This section provides details about the purpose, audience, limitations and the evolution of document.

## Definition

A data warehouse is not a product or a data model which simply copy the transactional data for operations, but a single repository where stores a collection of data from multiple sources, whether they are historical or current, and to support users for better decision making. As stated by Inmon in 1992, it carries characteristics as subject-oriented, integrated, consistent, time-variant and non-volatile. [1]

The data warehouse is a database but it is more an information database than an operational database. It focuses more on producing strategic information and supporting decisional process rather than handling day to day operations. The process of handling data in the data warehouse involves a couple of functions like data extraction, transformation, storage, loading and delivery. Hence, a combination of multiple technologies is required to support those functions which includes data modelling, data acquisition, data analysis and etc. [2]

## Necessity

As the expansion of corporations and accumulation of data stored in recent years, it becomes much more important for the business executives to use data available from various sources to make strategic decisions so as to improve the competitiveness of the company. The existing operational system is designed for processing daily routines which involves amounts of write operations for putting data into the database. However, we care more about extracting strategic information out of the database which is just the opposite of what the operational system meant to do. Far more than recording an order or processing a claim, managers focus more on key factors which influence the enterprise competitiveness. Information relates to customers' preferences, sales performance and quality of service is essential for making business decisions which can be integrated together as strategic information. The urgent need for those strategic data motivates the development of the data warehouse and the establishment of the data warehouse helps executives to make reasonable decisions, set business goals and control results.

The data warehouse provides integrated data which is associated with business process and can be easily accessed and queried by user without IT background. The flexibility and interactivity make the data warehouse even more attractive and compelling.

## Benefits

One of the most significant benefits of the data warehouse is the integration of historical data which is composed of both past and current data. Different types of business events are measured as diverse facts which then described by multiple dimensions. One or more hierarchies are embedded in dimension tables which form the foundation of multidimensional analysis; that is to say, the data can be played through drilling down, rolling up, slicing or dicing to analyse business processes from many perspectives. [3]

Another obvious advantage is the flexibility and accessibility for users to query the database and get requested data. The structure of the star schema simplifies the complexity of the data warehouse to a large extent by minimizing the quantity of tables which means the reduction of

joins operations. Hence, even users without background in IT field can easily query the database and get the data for analysis. Furthermore, using star schema for organizing data ensures the effectiveness of managing and controlling data and makes it easily to understand and retrieve.

In a word, the adaption of data warehouse contributes to the analysis of the past, the assessment of current status and the forecast for the future development.

# 3. Design of the data warehouse

In this section, we will discuss the design of the date warehouse. A datagram is provided to show the overall structure of the design. Details of the fact tables and dimension tables are also introduced and justified.

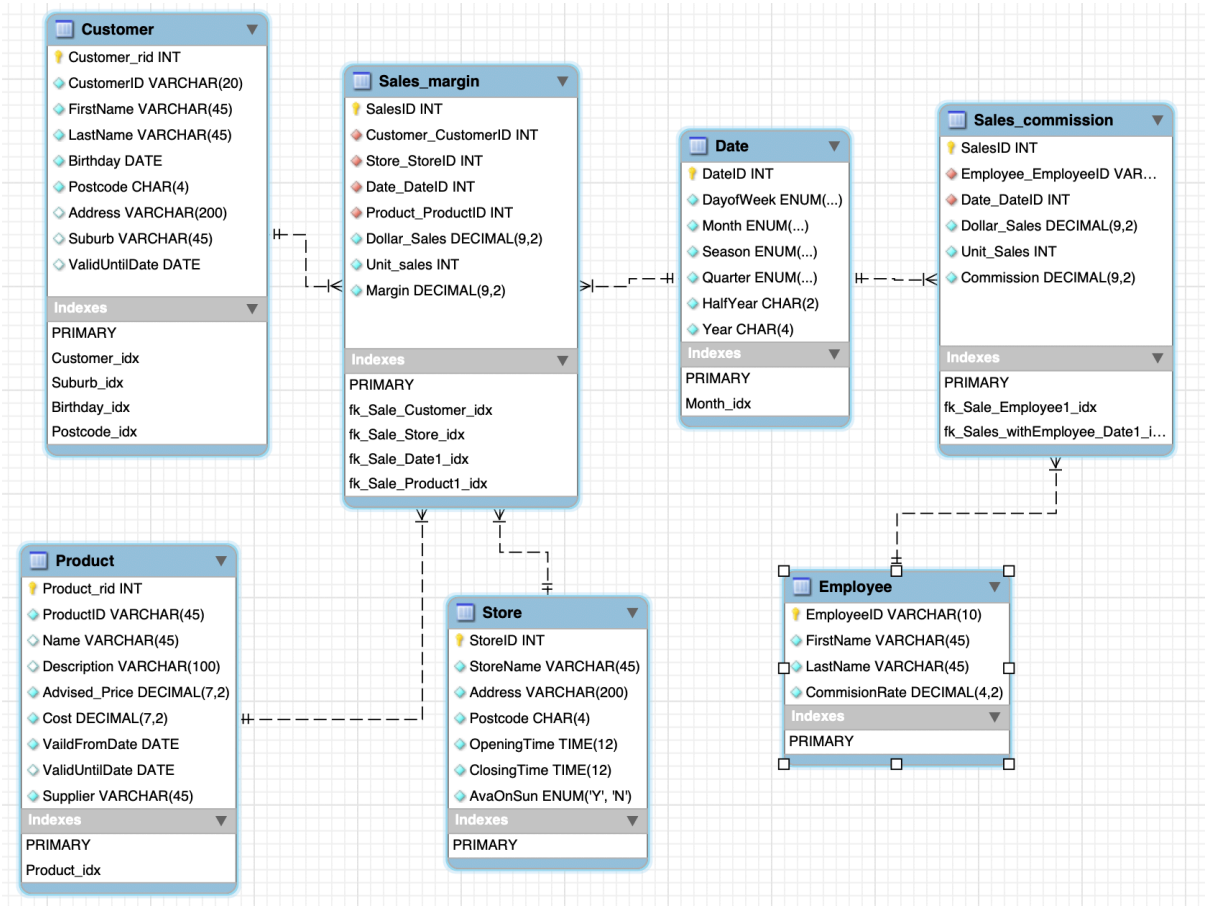
## Business Case Introduction and Analysis

In this case, the business owner wants to expand her business of Fantastic Fireworks, which already has four stores in different cities. The main process of the business is purchasing fireworks from supplier and selling them to customers. Two information systems have already been built to manage the business, including a sales system and an inventory system.

The business owner is thinking about having more stores, expanding floor area, adjusting opening time and finding out the key products in her stores. In order to expand her business in a suitable way, she wants to find valuable information from the recordings in the past years to support the decisions she will make.

For this purpose, a data warehouse is needed. The data warehouse should be able to provide the business owner the information she wants, including useful information about customers, products, employees, time periods and stores. Also, all the data required in the data warehouse should be able to be acquired from the two existing information systems.

## Overall Design




The design can be divided into two parts, a *Sales_margin* transaction model and an *Sales_commission* transaction model. The *Sales_margin* transaction model includes the fact table *Sales_margin* and the *Sales_commission* transaction model includes the fact table *Sales_commission*. Generally, a row in the table *Sales_margin* is a record of selling products to a customer and a row in the table *Sales_commission* is a record of the commission for an employee in one sale. Data in the fact table *Sales_margin* have four dimensions, which are corresponding to five dimensions *tables, Date, Customer, Product* and *Store.* As for data in the *Sales_commission,* they have two dimensions, *Date* and *Employee*.

## Grain of fact tables and the other details

A row in the fact table *Sales_margin* represents one sale of one product to a customer. To be specific, it represents that one particular customer buys some amounts of one particular product in a particular store in a particular day. Besides the four dimensions, the quantity of the products in the sale (*unit_sales*), the total sales amount (*dollar_sales)*, and the total profit (*margin*) are included in this fact table. *Margin* equals to the difference between total sales amount and the total cost. Indexes are added on the four foreign keys to help with data retrieving.

A row in the table *Sales_commission* represents one sale which causes one commission. To be specific, it represents that one employee brings some amount dollar sales to the shop in

one particular day. Besides the two dimensions, *Date* and *Employee*, the unit sales (*unit_sales*), the total sales amount (*dollar_sales)* and the commission earned by the employee (*commission)*are included in this fact table. *Commission* equals to *dollar_sales* multiplies commission rate of the corresponding employee. Indexes are added on the two foreign keys to help with data retrieving.

## Dimensions and the details of dimension tables

The *Sales_margin* part and the *Sales_commission* part of this design share one dimension, *Date*. Besides *date*, the *Sales_margin* facts have other three dimensions, *customer*, *product* and *store*, and the *Sales_commission* facts have another dimension, *employee*.

The *Customer* dimension is one of the dimensions of *Sales_margin* facts. In the dimension table, basic information of the customers is recorded including their names, addresses and IDs in existing sales system. As required, we should be able to aggregate customer reports into age groups – 30 years and under, 31-50, and over 50. In this case, birthdays are needed to calculate their age group. Also, postcodes and suburbs are needed as it is important to know the suburb in which they lived at the time of a sale. Customers may move houses and change the addresses, so we should add a surrogate key to show the changes and a *validUntilDate* to record when the address in this row becomes invalid. Every time a customer changes his address, there will be one new row in this table. In order to retrieving data, indexes are added in three columns, including *customer_id*, birthday, postcode and suburb.

The *Product* dimension is one of the dimensions of *Sales_margin* facts. In this dimension table, basic information of the products is recorded, including their IDs in existing systems, the advised price, the description and the first valid date of the price. As required, cost of one product is the cost of this product on the most recent purchasing from the supplier. In this case, the cost of one product may be changed, so a surrogate key is added to manage the changes and two date, *ValidFromDate* and *ValidUntilDate*, are added to record when the cost price becomes valid or invalid. Every time the cost of one product is changed, there will be one new row in this table. In order to retrieving data, indexes are added in the column *product_id*.

The *Employee* dimension is one of the dimensions of *Sales_commision* facts. As the business owner requires, she wants to know the performance of each employee. For each sale, we should record who sells the product to the customer. This dimension can help with deciding who are the key employees. In this dimension table, first name and last name of the employees are recorded. The commission rates of employees are also included. Although the commission rate of an employee may be changed, the system will edit the row and modify the rate in that row. If an employee leaves the store, the row corresponding to him/her will be deleted.

The *Date* dimension is one dimension of both the fact tables. As the business owner requires, she wants to analyse the business in various time periods. This dimension table decides which time periods one particular date belongs to. For a particular date, we need to know it is in

which day of a week, which month, which quarter, which season and which year. Indexes are added in the column month to help with the data retrieving.

The *Store* dimension is a dimension of *Sales_margin* fact table. As the business owner requires, she wants to analyse the performance of each store. For a particular store, we need to know its address, opening time and closing time. We also need to record whether it opens on Sunday as only one of the existing stores opens on Sunday.

# 4. Address business problems

In this section, solutions to five key business problems are discussed. For each problem, we will identify the data that need to be retrieved, the main process of the solution, explanation of each steps and the final table that will be presented.

## Who are the key customers?

| Table Name | Retrieved Data |
|---|---|
| Sales_margin | Customer_CustomerID, Date_DateID, Unit_Sales, Dollar_ Sales, Margin |
| Customer | CustomerID, Birthday, Postcode |
| Date | DateID, Year, Month, Quarter |

Table 4.1 Retrieved Data for deciding key customers

The process to present the key customers:

1. **Decide the time period for analysing and select the dates**
   Select the rows in the table Date according to the time period the business leader would like to choose. For example, when using the first quarter of 2018, select the rows in the table *Date*, in which '*Quarter*' is 'Q1' and '*Year*' is '2018'.
2. **Select the sales data in that time period**
   According to the result of the step1, select the rows from the Table *Sales_margin*, in which *Date_DateID* are from the result of step1.
3. **Group sales data and calculate the required information for each customer in various time periods**
   Group the sales data by customers. When grouping, for each customer in the result of step2, calculate the total dollar sales in that time period by summing up *Dollar_sales*, calculate the margin by summing up *margin* and calculate the unit sales by summing up *Unit_sales*. Then we get a customer performance table.
4. **Sort the customer performance table**
   Sort the result table of Step3 by one of the three numbers, total unit sales, total dollar sales or total margin, in descending order. The top customers in the result table of this step are the key customers.
5. **Find out where key customers live and which age groups they are in**
   Join the customer table to the result of step4. Add a new column *Age_group* for each customer and decide their age groups according to the date of their birth. Group the customers by the suburb or their age group and find out which suburbs or which age groups have more key customers.

The final presented table:

| CustomerID | Unit Sales | Dollar Sales | Total Margin | Suburb | Age_group |
|---|---|---|---|---|---|
| | | | | | |

## Which products are the most profitable?

| Table Name | Retrieved Data |
|---|---|
| Sales_margin | Product_ProductID, Date_DateID, Unit_sales, Dollar_ sales, margin |

| | |
|---|---|
| Date | DateID, Year, Month, Season |
| Product | ProductID, Description |

Table 4.2 Retrieved Data for deciding key products

The process to present the key customers:

1. **Decide the time period of the data for analysing the key products**
   Select the rows in the table Date according to the time period the business leader would like to choose. For example, when using the spring of 2018, select the rows in the table *Date*, in which '*Season*' is 'Spring' and '*Year*' is '2018'.
2. **Select the sales data in that time period**
   According to the result of the step1, select the rows from the Table *Sales_margin*, in which *Date_DateID* are from the result of step1.
3. **Group sales data and calculate the required information for each product**
   Group the sales data by products. When grouping, for each product in the result of step2, calculate the total dollar sales, the total margin and the total unit sales by summing up the data.
4. **Sort the product performance table**
   Sort the result table of Step3 by one of the three numbers, unit sales, total dollar sales or total margin, in descending order. The top products in the result of this step are the key product.

The final presented table:

| ProductID | Unit Sales | Dollar Sales | Margin |
|---|---|---|---|
| | | | |

## Which store location is the most profitable?

| Table Name | Retrieved Data |
|---|---|
| Sales_margin | Store_StoreID, Date_DateID, Unit_sales, Dollar_ sales, margin |
| Date | DateID, Year, Month |
| Store | StoreID, StoreName |

Table 4.3 Retrieved Data for deciding the most profitable store

The process to present the key customers:

1. **Decide the time period of the data for analysing the key products**
   Select the rows in the table Date according to the time period the business leader would like to choose. For example, when using the first month of 2018, select the rows in the table **Date**, in which '*month*' is 'Jan' and '*Year*' is '2018'.
2. **Select the sales data in that time period**
   According to the result of the step1, select the rows from the Table *Sales_margin*, in which *Date_DateID* are from the result of step1.
3. **Group sales data and calculate the required information in each store**
   Group the sales data by stores. When grouping, for each store in the result of step2, calculate the total dollar sales, total margin and the total unit sales by summing up the data.
4. **Sort the store performance table**
   Sort the result table of Step3 by one of the three numbers, unit sales, total dollar sales or total margin, in descending order. The first store in the result of the sorting is the most profitable store.

The final presented table:

| StoreID | Unit Sales | Dollar Sales | Margin |
|---|---|---|---|
| | | | |

## Which time periods are the most profitable?

| Table Name | Retrieved Data |
|---|---|
| Sales_margin | StoreID, DateID, Unit_sales, Dollar_ sales, margin |
| Date | DateID, Year, Month, DayofWeek |

Table 4.4 Retrieved Data for deciding the most profitable time periods

The process to present the key customers:

1. **Join the table *Date* into the table *Sales_margin***
   Join the table *Date* into the table *Sales*, with the date type 'Month', 'Year' and 'DayOfWeek'. Then group all the sales data by the chosen time period. For example, if the business leader wants to compare different months, group the table *Sales_margin* by 'Month'
2. **Group all the sales data by time period and calculate the required information in each time period**
   Group all the sales data by the chosen time period. For example, if the business leader wants to compare different months, group the sales table by 'Month'. When grouping, for each time period (for example, Jan, Feb, …Dec), calculate the total dollar sales, total margin and the total quantity by summing up the data. Calculate the unit sales, which equals to the total quantity divides the total sales.
3. **Sort the time period performance table**
   Sort the result table of Step2 by one of the three numbers, unit sales, total dollar sales or total margin in descending order. The top time periods in the result of the sorting is the most profitable time period.

The final presented table:

| Month/Year/DayofWeek | Unit Sales | Total Sales | Total Margin |
|---|---|---|---|
| | | | |

## Who are the Key employees?

| Table Name | Retrieved Data |
|---|---|
| Sales_employee | Employee_EmployeeID, Date_DateID, Unit_sales, Dollar_ sales, Commission |
| Date | DateID, Year, Month, Quarter |
| Employee | EmployeeID, LastName, FirstName |

Table 4.5 Retrieved Data for deciding key products

The process to present the key employees:

1. **Decide the time period of the data for analysing the key employees**
   Select the rows in the table Date according to the time period the business leader would like to choose. For example, when using the first quarter of 2018, select the rows in the table *Date*, in which 'Quarter' is 'Q1' and 'Year' is '2018'.

2. **Select the sales data in that time period**
   According to the result of the step1, select the rows from the Table Sales_employee, in which Date are from the result of step1.
3. **Group selected sales data and calculate the required information for every employee**
   Group the selected sales data by employees. When grouping, for each employee in the result of step2, calculate the total dollar sales by summing up *Dollar_sales*, calculate the margin by summing up *Margin* and calculate the total unit sales by summing up *Unit_sales*.
4. **Sort the employee performance table**
   Sort the result table of Step3 by one of the three numbers, unit sales, dollar sales, margin or total commission, in descending order. The top employees in the result of this step are the key product.

The final presented table:

| EmployeeID | Unit Sales | Total Sales | Total Margin | Total Commission |
|---|---|---|---|---|
| | | | | |

Words count:3258

# Appendix 1- Data Dictionary

Sales_margin Fact Table

| Attribute | Description | Source |
|---|---|---|
| SalesID | Unique identifier of an order | A surrogate key generated by the data warehouse load process |
| Customer_CustomerID | The identifier of each customer consisted of digits and characters | The CustomerID attribute of Customer table stored in the sales system |
| Store_StoreID | Unique identifier of a store | The StoreID attribute of Store table stored in the sales system |
| Date_DateID | Unique identifier of a date which is described differently as daily, monthly, seasonally, quarterly and yearly | A surrogate key generated by the data warehouse load process |
| Product_ProductID | The identifier of each product consisted of digits, characters and symbols | The ProductID attribute of Product Price List table stored in the sales system |
| Dollar_Sales | The total price of each kind of product for each sale | Generated by the product of Quantity and unit price values stored in the Sales Item table |
| Unit_Sales | The total amount of each kind of product for each sale | The Quantity attribute of Sales Item table stored in the sales system |
| Margin | The profit for selling each type of product for each sale | The differences between the Dollar_Sales values and Cost values |
| fk_Sale_Customer_idx | The index on the foreign key Customer_CustomerID | System generated index |
| fk_Sale_Store_idx | The index on the foreign key Store_StoreID | System generated index |
| k_Sale_Date1_idx | The index on the foreign key Date_DateID | System generated index |
| fk_Sale_Product1_idx | The index on the foreign key Product_ProductID | System generated index |

Customer Dimension Table

| Attribute | Description | Source |
|---|---|---|
| Customer_rid | The numeric identifier of each customer which is unique | A surrogate key generated by the data warehouse load process |
| CustmoerID | The identifier of each customer consisted of digits and characters | The CustomerID attribute of Customer table stored in the sales system |
| FirstName | The first name of each customer | The Name attribute of Customer table stored in the |

| | | |
|---|---|---|
| | | sales system which is transformed into the first name of each customer |
| LastName | The last name of each customer | The Name attribute of Customer table stored in the sales system which is transformed into the last name of each customer |
| Birthday | The birthday of each customer which is presented as date and used for customer grouping based on age | The Date of Birth attribute of Customer table stored in the sales system |
| Address | The address of each customer | Extracted from the system which stores the customers' personal information |
| Postcosde | The postcode of every customer's living address | The Postcode attribute of Customer table stored in the sales system |
| Suburb | The suburb in which each customer lives | Extracted from the system which stores the customers' personal information |
| ValidUntilDate | The period of validity of each customer's data | The Valid until date attribute of Customer table stored in the sales system |
| Customer_idx | The index on the key CustomerID | System generated index |
| Suburb_idx | The index on the key Suburb | System generated index |
| Birthday_idx | The index on the key Birthday | System generated index |
| Postcode_idx | The index on the key Postcode | System generated index |

Date Dimension Table

| **Attribute** | **Description** | **Source** |
|---|---|---|
| DateID | Unique identifier of a date which is described differently as daily, monthly, seasonally, quarterly and yearly | A surrogate key generated by the data warehouse load process |
| DayofWeek | The weekday of each sale | Aggregated based on the analysis needs |
| Month | The current month of each sale | Aggregated based on the analysis needs |
| Season | The current season of each sale | Aggregated based on the analysis needs |
| Quarter | The current quarter of each sale | Aggregated based on the analysis needs |
| HalfYear | Whether the sale belongs to the first half of the year or the latter half of the year | Aggregated based on the analysis needs |
| Year | The current year of each sale | Aggregated based on the analysis needs |

Product Dimension Table

| Attribute | Description | Source |
|---|---|---|
| Product_rid | The numeric identifier of each product which is unique | A surrogate key generated by the data warehouse load process |
| ProductID | The identifier of each product consisted of digits, characters and symbols | The PartNumber attribute of Product table stored in the inventory system |
| Name | The name of each kind of product | Extracted from the system which stores the products' information |
| Description | The description of each product, involving the appearance, size, colour and etc | The Description attribute of Product table stored in the inventory system |
| Advised_Price | The selling price of each product | The Price attribute of Product table stored in the inventory system |
| Cost | The cost of each product which may change over time | The Cost per item attribute of Product Order table stored in the inventory system |
| ValidFromDate | The valid-from date of each product's data | The ValidFrom attribute of Product table stored in the inventory system |
| ValidUntilDate | The period of validity of each product's data | Added when the data of the product is updated |
| Supplier | The name of the supplier of the product | Extracted from the system which stores suppliers' relevant information |
| PRIMARY | The index on the primary key Product_rid | System generated index |

Store Dimension Table

| Attribute | Description | Source |
|---|---|---|
| StoreID | Unique identifier of each store | The StoreID attribute of Store table stored in the sales system |
| StoreName | The name of each store | Extracted from the system which stores the customers' personal information |
| Address | The address of each store | The Address attribute of Store table stored in the sales system which is transformed without the postcode |

| | | |
|---|---|---|
| Postcode | The postcode of each store's address | The data extracted from the Address attribute of Store table in the sales system |
| OpeningTime | The opening time of each store | Extracted from the system which records stores' relevant information |
| ClosingTime | The clsoing time of each store | Extracted from the system which records stores' relevant information |
| AvaOnSun | Whether the store is opened on Sunday or not | Extracted from the system which records stores' relevant information |

Employee Dimension Table

| **Attribute** | **Description** | **Source** |
|---|---|---|
| EmployeeID | Unique identifier of each employee | The ID attribute of Sales Person table stored in the sales system |
| FirstName | The first name of each employee | The Name attribute of Sales Person table stored in the sales system which is transformed into the first name of each employee |
| LastName | The last name of each employee | The Name attribute of Sales Person table stored in the sales system which is transformed into the last name of each employee |
| CommissionRate | The commission rate of each sales person for each sale | The Commission rate attribute of Sales Person table stored in the sales system |

Sales_commission Fact Table

| **Attribute** | **Description** | **Source** |
|---|---|---|
| SalesID | Unique identifier of an inventory | A surrogate key generated by the data warehouse load process |
| Employee_EmployeeID | Unique identifier of an employee | The ID attribute of Sales Person table stored in the sales system |
| Date_DateID | Unique identifier of a date | A surrogate key generated by the data warehouse load process |
| Dollar_Sales | The total price of each kind of product for each sale | Generated by the product of Quantity and Unit Price values stored in the Sales Item table |

| | | |
|---|---|---|
| Unit_Sales | The total amount of each kind of product for each sale | The Quantity attribute of Sales Item table stored in the sales system |
| Commission | The amount of commission that an employee earns for different time periods | Generated by the product of Dollar sales and commission values of the employee stored in the Sales Item table |
| fk_Sale_Employee1_idx | The index on the foreign key EmployeeID | System generated index |
| fk_Sales_withEmployee_Date1_idx | The index on the foreign key DateID | System generated index |

## Appendix 2- SQL

1. CREATE TABLE IF NOT EXISTS `mydb`.`Sales_margin` (
   `SalesID` INT NOT NULL,
   `Customer_CustomerID` INT NOT NULL,
   `Store_StoreID` INT NOT NULL,
   `Date_DateID` INT NOT NULL,
   `Product_ProductID` INT NOT NULL,
   `Dollar_Sales` DECIMAL(9,2) NOT NULL,
   `Unit_sales` INT NOT NULL,
   `Margin` DECIMAL(9,2) NOT NULL,
   PRIMARY KEY (`SalesID`),
   CONSTRAINT `fk_Sale_Customer0`
   FOREIGN KEY (`Customer_CustomerID`)
   REFERENCES `mydb`.`Customer` (`CustomerID`)
   ON DELETE NO ACTION
   ON UPDATE NO ACTION,
   CONSTRAINT `fk_Sale_Location10`
   FOREIGN KEY (`Store_StoreID`)
   REFERENCES `mydb`.`Store` (`StoreID`)
   ON DELETE NO ACTION
   ON UPDATE NO ACTION,
   CONSTRAINT `fk_Sale_Date10`
   FOREIGN KEY (`Date_DateID`)
   REFERENCES `mydb`.`Date` (`DateID`)
   ON DELETE NO ACTION
   ON UPDATE NO ACTION,
   CONSTRAINT `fk_Sale_Product10`
   FOREIGN KEY (`Product_ProductID`)
   REFERENCES `mydb`.`Product` (`ProductID`)
   ON DELETE NO ACTION

```
ON UPDATE NO ACTION)

CREATE INDEX `fk_Sale_Customer_idx` USING BTREE ON
`mydb`.`Sales_margin` (`Customer_CustomerID` ASC);

CREATE INDEX `fk_Sale_Store_idx` USING BTREE ON `mydb`.`Sales_margin`
(`Store_StoreID` ASC);

CREATE INDEX `fk_Sale_Date1_idx` USING BTREE ON `mydb`.`Sales_margin`
(`Date_DateID` ASC);

CREATE INDEX `fk_Sale_Product1_idx` USING BTREE ON
`mydb`.`Sales_margin` (`Product_ProductID` ASC);
```

2. 
```
CREATE TABLE IF NOT EXISTS `mydb`.`Customer` (
`Customer_rid` INT NOT NULL AUTO_INCREMENT,
`CustomerID` VARCHAR(20) NOT NULL,
`FirstName` VARCHAR(45) NOT NULL,
`LastName` VARCHAR(45) NOT NULL,
`Birthday` DATE NOT NULL,
`Postcode` CHAR(4) NOT NULL,
`Address` VARCHAR(200) NULL,
`Suburb` VARCHAR(45) NULL,
`ValidUntilDate` DATE NULL,
PRIMARY KEY (`Customer_rid`))

CREATE INDEX `Customer_idx` USING BTREE ON `mydb`.`Customer`
(`CustomerID` ASC);

CREATE INDEX `Suburb_idx` USING BTREE ON `mydb`.`Customer` (`Suburb`
ASC);

CREATE INDEX `Birthday_idx` USING BTREE ON `mydb`.`Customer`
(`Birthday` ASC);

CREATE INDEX `Postcode_idx` ON `mydb`.`Customer` (`Postcode` ASC);
```

3. 
```
CREATE TABLE IF NOT EXISTS `mydb`.`Date` (
`DateID` INT NOT NULL,
`DayofWeek` ENUM('Mon', 'Tue', 'Wed', 'Thur', 'Fri', 'Sat', 'Sun') NOT NULL,
`Month` ENUM('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sept', 'Oct', 'Nov',
'Dec') NOT NULL,
`Season` ENUM('Spring', 'Summer', 'Fall', 'Winter') NOT NULL,
`Quarter` ENUM('Q1', 'Q2', 'Q3', 'Q4') NOT NULL,
`HalfYear` CHAR(2) NOT NULL,
```

```
`Year` CHAR(4) NOT NULL,
PRIMARY KEY (`DateID`))

CREATE INDEX `Month_idx` USING BTREE ON `mydb`.`Date` (`Month` ASC);
```

4. 
```
CREATE TABLE IF NOT EXISTS `mydb`.`Product` (
`Product_rid` INT NOT NULL AUTO_INCREMENT,
`ProductID` VARCHAR(45) NOT NULL,
`Name` VARCHAR(45) NULL,
`Description` VARCHAR(100) NULL,
`Advised_Price` DECIMAL(7,2) NOT NULL,
`Cost` DECIMAL(7,2) NOT NULL,
`VaildFromDate` DATE NOT NULL,
`ValidUntilDate` DATE NULL,
`Supplier` VARCHAR(45) NOT NULL,
PRIMARY KEY (`Product_rid`))

CREATE INDEX `Product_idx` USING BTREE ON `mydb`.`Product` (`ProductID` ASC);
```

5. 
```
CREATE TABLE IF NOT EXISTS `mydb`.`Store` (
`StoreID` INT NOT NULL,
`StoreName` VARCHAR(45) NOT NULL,
`Address` VARCHAR(200) NOT NULL,
`Postcode` CHAR(4) NOT NULL,
`OpeningTime` TIME(12) NOT NULL,
`ClosingTime` TIME(12) NOT NULL,
`AvaOnSun` ENUM('Y', 'N') NOT NULL,
PRIMARY KEY (`StoreID`))
```

6. 
```
CREATE TABLE IF NOT EXISTS `mydb`.`Sales_commission` (
`SalesID` INT NOT NULL,
`Employee_EmployeeID` VARCHAR(10) NOT NULL,
`Date_DateID` INT NOT NULL,
`Dollar_Sales` DECIMAL(9,2) NOT NULL,
`Unit_Sales` INT NOT NULL,
`Commission` DECIMAL(9,2) NOT NULL,
PRIMARY KEY (`SalesID`),
CONSTRAINT `fk_Sale_Employee1`
FOREIGN KEY (`Employee_EmployeeID`)
REFERENCES `mydb`.`Employee` (`EmployeeID`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_Sales_withEmployee_Date1`
```

```
FOREIGN KEY (`Date_DateID`)
REFERENCES `mydb`.`Date` (`DateID`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)

CREATE INDEX `fk_Sale_Employee1_idx` USING BTREE ON
`mydb`.`Sales_commission` (`Employee_EmployeeID` ASC);

CREATE INDEX `fk_Sales_withEmployee_Date1_idx` ON
`mydb`.`Sales_commission` (`Date_DateID` ASC);
```

7. 
```
CREATE TABLE IF NOT EXISTS `mydb`.`Employee` (
`EmployeeID` VARCHAR(10) NOT NULL,
`FirstName` VARCHAR(45) NOT NULL,
`LastName` VARCHAR(45) NOT NULL,
`CommisionRate` DECIMAL(4,2) NOT NULL,
PRIMARY KEY (`EmployeeID`))
```

## Appendix 3- Work Breakdown

Both members participate in the design of the data warehouse and the accomplishment of the report.

To be more specific, Dinghao Yong (868878) completes following parts:

1. Design of the data warehouse and Draw the model on the workbench;
2. Implement the data warehouse to address business problems;
3. Complete the report in terms of the design and problem-solving parts.

Min Xue (897082) finishes tasks below:

1. Design of the data warehouse and Draw the model on the workbench;
2. Discuss about the implementation of the data warehouse;
3. Complete the report in terms of executive summary, introduction of the data warehousing and appendixes.

## References

[1] Golfarelli and Rizzi, “Data Warehouse Design: Modern Principles and Methodologies,” *Amazon*. [Online]. Available: https://www.amazon.com/Data-Warehouse-Design-Principles-Methodologies/dp/0071610391. [Accessed: 11-Jan-2019].

[2] P. Ponniah, *Data warehousing fundamentals: a comprehensive guide for IT professionals*. San Franciso, CA: Wiley, 2011.

[3] D. L. Moody and M. A. R. Kortink, “From ER Models to Dimensional Models: Bridging the Gap between OLTP and OLAP Design, Part I,” *From ER Models to Dimensional Models: Bridging the Gap between OLTP and OLAP Design, Part I*, 2003.