|  |
| --- |
|  |
| Retailer Database |
| Course Section: CS605.641.83  Summer, 2020 |
| Prepared by |
| **Benlong Huang** |
| **06/06/2020** |

|  |
| --- |
| Database Design Project Document |

**Table of Contents**

[1. Introduction 3](#_Toc516748910)

[1.1. Scope and Purpose of Document 3](#_Toc516748911)

[1.2. Project Objective 3](#_Toc516748912)

[2. System Requirements 4](#_Toc516748913)

[2.1 Hardware Requirements 4](#_Toc516748914)

[2.2 Software Requirements 4](#_Toc516748915)

[2.3 Functional Requirements 5](#_Toc516748916)

[2.4 Database Requirements 5](#_Toc516748917)

[3. Database Design Description 6](#_Toc516748918)

[3.1 Design Rationale 6](#_Toc516748919)

[3.2 E/R Model 6](#_Toc516748920)

[3.2.1 Entities 6](#_Toc516748921)

[3.2.2 Relationships 8](#_Toc516748922)

[3.2.3 E/R Diagram 9](#_Toc516748923)

[3.3 Relational Model 1](#_Toc516748924)0

[3.3.1 Data Dictionary 1](#_Toc516748925)0

[3.3.2 Integrity Rules 1](#_Toc516748926)2

[3.3.3 Operational Rules 1](#_Toc516748927)3

[3.3.4 Operations 1](#_Toc516748928)3

[3.4 Security 1](#_Toc516748929)3

[3.5 Database Backup and Recovery 1](#_Toc516748930)4

[3.6 Using Database Design or CASE Tool 1](#_Toc516748931)4

[3.7 Other Possible E/R Relationships 1](#_Toc516748932)4

[4. Implementation Description 1](#_Toc516748933)5

[4.1 Data Dictionary 1](#_Toc516748934)5

[4.2 Advanced Features 1](#_Toc516748935)7

[4.3 Queries 1](#_Toc516748936)8

[4.3.1 TOP 3 Customer 1](#_Toc516748937)8

[4.3.2 Yearly Order Of 2019 1](#_Toc516748938)8

[4.3.3 Customer Order Status 1](#_Toc516748939)8

[4.3.4 Order Subtotal 1](#_Toc516748940)8

[4.3.5 Second Highest Salary 1](#_Toc516748941)8

[4.3.6 Product Above Average Price 1](#_Toc516748942)8

[4.3.7 Favorite Payment Method 1](#_Toc516748943)8

[4.3.8 Summary Of Good Sales 1](#_Toc516748944)9

[5. CRUD Matrix 1](#_Toc516748945)9

[5.1 List of Entity Types 2](#_Toc516748946)0

[5.2 List of Functions 2](#_Toc516748947)0

[6. Concluding Remarks 2](#_Toc516748948)0

[Appendices 2](#_Toc516748949)1

[References 2](#_Toc516748950)7

# Introduction

Nowadays, the use of databases is adopted by companies in different industries such as technology, construction, manufacturing, healthcare, and retail. The needs of the database are growing rapidly especially in the retail industry because consumer purchase services and goods are changing every year. A great way to manage customer and product information is through the implementation of databases. Furthermore, databases can improve customer satisfaction since it can provide faster and convenient service in the companies’ daily operation; it can also increase business revenue by having access to various data types, to provide both overall and detailed analysis for future business strategy.

Although the retail companies have different company scale and product types, the expectation and requirement for a suitable database are roughly the same. The logic of database designing for retail companies shall match the common business logic and needs in this industry. For customer management, the database will store customer information tidy and up to date, which can maintain a strong customer relationship to provide better service and promote future sales easily. For product management, the database will log and monitor all transactions and show product storage. For supplier and vendor management, the database will provide an effective way to record each product resource and previous or on-going activities. For internal uses, a powerful database also provides various access authority to users from different levels for better internal operation and management. Having a solid and reliable database to manage and allocate resources would benefit greatly to the company not only in increasing revenue, but also in establishing and maintaining a healthy relationship with customers, partners, and staff.

This project is to design a database foundation that can be used for the retail companies from small to large size, which will heavily involve customer, transaction, product, and related information in the database frame. This project includes system requirements, database design, database implementation, and feature and function introduction.

## Scope and Purpose of Document

This document represents the result of the research and practice of database establishment for the companies in the retail industry, it also includes all work associated with preparation, design, and implementation of the database. This document and the corresponding data files and appendix are intended to enhance the understanding of the entire project. To meet the general business requirement for the companies in the retail industry, the system requirements will indicate the required hardware and software including but not limited to CPU, MySQL, and Ubuntu. Major entities within the system and the inner relationship amount will be shown by the visualization, database structure, and data dictionary. Moreover, a security backup plan is also included in this project as part of the support for recovering the database system if necessary.

The purpose and goal for this project are to illustrate a detailed plan for implement a database for the retail companies, which by building a searchable database, entity-relationship diagram (ERD), operational rules, and related elements to manage various resources and meet the market trend for business development.

## Project Objective

This database is designed fairly for company internal users for manage data and resources. The design and implementation are straightforward and clearly showed step by step in order for the companies to adapt. The main objectives of this project can be summarized as the following:

- Describe the preparation of the setting of the database frame.

- Describe queries for data sorting by following different requirements.

- Record system requirements that are matched to users' expectations.

- Use data visualization to perform data analysis and maintenance.

- Display detailed descriptions of the metadata for the model.

# System Requirements

**MySQL server:** Although it is not required, MySQL Workbench is designed to have either a remote or local MySQL server connection. For additional information about connecting to a MySQL server, see Chapter 5, Connections in MySQL Workbench. For additional information about installing a MySQL server, see Installing and Upgrading MySQL.

Data modeling does not require a MySQL server connection.

Some features take advantage of MySQL server features, and as such, they require more recent versions of MySQL Server. For example, the Performance Dashboard requires MySQL Server 5.6 or higher.

**Simultaneous client connections:** Opening a MySQL connection from the MySQL Workbench home page opens a new connection tab in MySQL Workbench for that connection. Each of these tabs requires two MySQL connections to perform basic tasks, such as schema discovery and SQL execution. Additionally, performing management related tasks, such as Server Status, requires two additional MySQL connections. Essentially, this means that each MySQL connection tab in MySQL Workbench requires four available connections to MySQL. For additional information about "Too many connections" related errors, see Too many connections.

This connection requirement doubles with each connection tab opened in MySQL Workbench, even if the two connection tabs point to the same MySQL server. SQL editor tabs share their connections, so having multiple SQL editor and SQL results tabs does not affect the number of required connections.

## Hardware Requirements

* CPU: Intel Core or Xeon 3GHz (or Dual Core 2GHz) or equal AMD CPU
* Cores: Single (Dual/Quad Core is recommended)
* RAM: 4 GB (6 GB recommended)
* Graphic Accelerators: nVidia or ATI with support of OpenGL 1.5 or higher
* Display Resolution: 1280×1024 is recommended, 1024×768 is minimum.

## Software Requirements

* Windows 7 (64-bit, Professional level or higher)
* Mac OS X 10.6.1+
* Ubuntu 9.10 (64bit)
* Ubuntu 8.04 (32bit/64bit)
* Windows XP SP3, Vista
* Mac OSX (10.5 and 10.6) Intel
* Ubuntu 8.04 (i386/x64)
* Ubuntu 9.04 (i386/x64)
* Fedora 11 (i386/x64)

## Functional Requirements

* SQL Development: Enables you to create and manage connections to database servers. As well as enabling you to configure connection parameters, MySQL Workbench provides the capability to execute SQL queries on the database connections using the built-in SQL Editor. This functionality replaces that previously provided by the Query Browser standalone application.
* Data Modeling: Enables you to create models of your database schema graphically, reverse and forward engineer between a schema and a live database, and edit all aspects of your database using the comprehensive Table Editor. The Table Editor provides easy-to-use facilities for editing Tables, Columns, Indexes, Triggers, Partitioning, Options, Inserts and Privileges, Routines and Views.
* Server Administration: Enables you to create and administer server instances.

## Database Requirements

MySQL Workbench, Version 8.0.20

# Database Design Description

This database is design for the most retail company, help them more efficiently maintain their products sales and order tracking system. Whole system essentially designed for these 3 types of users; Employee work in the front in charge of sales which is the sales department; Employee work in the middle in charge of tracking the orders which is order tracking department; And the employee work in the behind for supply chain department who in charge of checking the products and suppliers.

## Design Rationale

The business process of the retail should be customer come to the store, pick the product they want, find their own sales representative for further information and make a purchase. If the product that customer want is out of stock, then the sales representative work in the front should contact the supply chain department ask for product details and schedule a shipment with order tracking department.

## E/R Model

Based on the business process mentioned above. There are 13 tables in total in this E/R Model. Different users will in charge of different tables.

From the company perspective, firstly it need an EMPLOYEE table to record all the sales representative information and check who is responsible for each customer and order, after that need an OFFICE table to store each retail store’s location. Then need a table CUSTOMERS to record all customer's information. PRODUCT table which record all product information and SUPPLIER table which record all the supplier information. PRODUCT has many SUPPLIERS and SUPPLIER supplies many PRODUCT so there will be an intersection entity which is PRODUCT\_SUPPLIER to record where is each product come from. Also need a SHIPPER table to record all the shipper that works for the company. These are the basic table need for the service. After customer make the purchase, it will need further tables for orders.

Talk about the CUSTOMER and PRODUCT, then there has to be a ORDERS table to record all the information of the order itself like order\_date, shipped\_date, order\_status. Order status have many different types. So a weak entity table ORDER\_STATUS will show all the status will happened like Processed, Shipped, Delivered. To know the exact order detail, there will be a intersection entity between ORDER\_INFORMATION and PRODUCT table which is called ORDER\_INFORMATION include the product customer purchased, purchased amount and price.

Finally is the PAYMENT table which is record all the payment method, amount that customer purchased and Invoice. There are many payment method. So create another table called PAYMENT\_METHOD to list all the ways like credit card, cash, paypal, vemmo, etc. Another INVOICE table to record the invoice number and date.

In these 13 tables, the sales department will get the privilege to maintain table EMPLOYEE, CUSTOMER, ORDERS, PAYMENT, PAYMENT\_METHOD and INVOICE. Order tracking department will in charge of table ORDERS,ORDER\_STATUS and SHIPPER. Supply chain department mainly focus on table SUPPLIER, PRODUCT, PRODUCT\_SUPPLIER.

### Entities

There are 6 strong entities in this model, EMPLOYEE, OFFICE, CUSTOMER, SHIPPER, PRODUCT, SUPPLIER.

EMPLOYEE

* This table represent each employee has identified by its unique employee ID
* For each employee will save their personal information include names, phone number, email address
* Also shows the employee will report to their manager.

OFFICE

* This table represent each retail store has identified by its unique office ID
* For each retail store, the table will record their name and location

CUSTOMER

* This table represent each customer has identified by its unique customer ID
* For each customer will save their personal information include names, phone number, email address and points they earned
* For each customer it will show their own sales representative

SHIPPER

* This table represent each shipper has identified by its unique shipper ID
* For each shipper will save their personal information include names, phone number, email address

PRODUCT

* This table represent each product has identified by its unique product ID
* For each product the table will record the detail description, unit price and quantity in stock

SUPPLIER

* This table represent each supplier has identified by its unique supplier ID
* For each supplier the table will store their company name, contact information and address

There are 5 weak entities in this model, ORDERS, ORDER\_STATUS, PAYMENT, INVOICES, PAYMENT\_METHOD

ORDERS

* This table represent each order has identified by its unique order ID
* For each order, this table will show its order date, and ship date
* Each order will also include which customer order it, who is the sales representative, who is the shipper, what’s the order status, what’s the payment

ORDER\_STATUS

* This table represent each order status has identified by its unique order status ID
* Shows different order status that an order will have, like Processed, Delivered.

PAYMENT

* This table represent each payment has identified by its unique payment ID
* For each payment, the table will show which date the payments make, and the payment amount
* Will also reference other table to show which order does this payment belongs to, what’s the invoice number and what kind of payment method that customer use to purchase.

INVOICES

* This table represent each invoice has identified by its unique invoice ID
* For each invoice, the table will record the date that invoice been made and the invoice number.

PAYMENT\_METHOD

* This table represent each payment method has identified by its unique payment method ID
* Shows all the possible payment methods that each order is purchased by, like Cash, Credit card.

There are 2 intersection entities in this model, ORDER\_INFORMATION and PRODUCT\_SUPPLIER.

ORDER\_INFORMATION

* This table is an intersection entities between ORDERS and PRODUCT, combine ORDER ID and PRODUCT ID form a composite primary key
* This table shows the order details including product quantity and total price that each order been purchased

PRODUCT\_SUPPLIER

* This table is an intersection entities between Product and SUPPLIER, combine PRODUCT ID and SUPPLIER ID form a composite primary key
* This table indicate which supplier is each product come from, if there has further information, the note will also be provided

### Relationships

There are 4 non-identified relationship and 14 identified relationship

EMPLOYEE - OFFICE. There are two relationship between employee and office. One to many and one to zero or one. Indicate that an office need to have at least 1 or more employee, but each employee can only work in one office.

EMPLOYEE - EMPLOYEE. One to many relationship, indicate employee should report to their manager

EMPLOYEE - CUSTOMER. One to many relationship, an sales can server many customers, but a customer should have their sales representative once at a time.

EMPLOYEE - ORDERS. One to many relationship, each sales representative may in charge of many orders, many orders will go to zero or one sales representative.

CUSTOMER - ORDERS. One to many relationship, a customer can have 0 or more orders, and order belongs to only one customer.

ORDER\_STATUS - ORDERS. One to one relationship, an order will only have one order\_status once at a time, the status of an order either be not shipped or in process or delivered.

PAYMENT - ORDERS. One to one relationship, each order will have only one payment each time.

SHIPPER - ORDERS. One to many relationship, a shipper can have multiple orders to deliver. Many orders will assigned to one shipper.

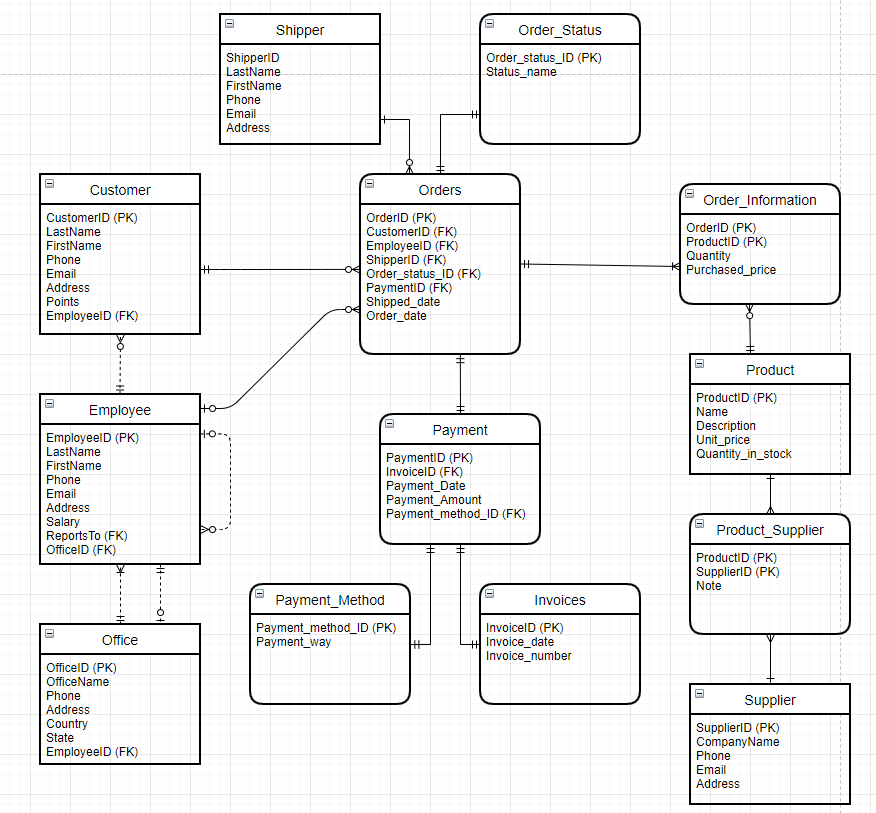
INVOICES - PAYMENT. One to one relationship, each payment will have only one invoice

PAYMENT\_METHOD - PAYMENT. One to one relationship, each payment will only have one type payment method, the customer either use cash or card or Paypal, etc.

ORDERS - ORDER\_INFORMATON - PRODUCT. Many to many relationship, so using intersection entity. An orders may contain many products, products may in many orders.

PRODUCT - PRODUCT\_SUPPLIER - SUPPLIER. Many to many relationship, so using intersection entity. A same product may come from many different suppliers, a supplier may offer many products.

### E/R Diagram



## Relational Model

### Data Dictionary

EMPLOYEE

| Column Name | Description | Data Type | Size | Constraint Type | Not Null? |
| --- | --- | --- | --- | --- | --- |
| EmployeeID | Employee ID Number | INT | 25 | Primary Key | Y |
| LastName | Employee last name | VARCHAR | 20 |  | Y |
| FirstName | Employee first name | VARCHAR | 10 |  | Y |
| Phone | Employee phone number | VARCHAR | 24 |  | Y |
| Email | Employee Email | VARCHAR | 45 |  | Y |
| Address | Employee address | VARCHAR | 45 |  | Y |
| Salary | Employee’s Salary | INT | 25 |  | Y |
| ReportsTo | Manager Social Security Number | INT | 25 | Foreign Key | Y |
| OfficeID | Which office does each employee work at | INT | 25 | Foreign Key | Y |

Customer

| Column Name | Description | Data Type | Size | Constraint Type | Not Null? |
| --- | --- | --- | --- | --- | --- |
| CustomerID | Customer ID number | INT | 25 | Primary Key | Y |
| LastName | Customer last name | VARCHAR | 20 |  | Y |
| FirstName | Customer first name | VARCHAR | 10 |  | Y |
| Phone | Customer phone number | INT | 24 |  | N |
| Email | Customer Email | VARCHAR | 45 |  | N |
| Address | Customer address | VARCHAR | 45 |  | N |
| Points | Customer reward points | INT | 25 |  | Y |
| EmployeeID | Sales representative in charge of each customer | INT | 25 | Foreign Key | N |

OFFICE

| Column Name | Description | Data Type | Size | Constraint Type | Not Null? |
| --- | --- | --- | --- | --- | --- |
| OfficeID | Office ID Number | INT | 25 | Primary Key | Y |
| OfficeName | Employee last name | VARCHAR | 45 |  | Y |
| Phone | Employee first name | VARCHAR | 24 |  | Y |
| Address | Employee phone number | VARCHAR | 45 |  | Y |
| Country | Employee Email | VARCHAR | 15 |  | Y |
| State | Employee address | VARCHAR | 15 |  | Y |
| EmployeeID | Employee works in this office | INT | 25 | Foreign Key | Y |

SHIPPER

| Column Name | Description | Data Type | Size | Constraint Type | Not Null? |
| --- | --- | --- | --- | --- | --- |
| ShipperID | Shipper ID Number | INT | 25 | Primary Key | Y |
| LastName | Shipper last name | VARCHAR | 20 |  | Y |
| FirstName | Shipper first name | VARCHAR | 10 |  | Y |
| Phone | Shipper phone number | VARCHAR | 24 |  | Y |
| Email | Shipper Email | VARCHAR | 45 |  | Y |
| Address | Shipper address | VARCHAR | 45 |  | Y |

ORDERS

| Column Name | Description | Data Type | Size | Constraint Type | Not Null? |
| --- | --- | --- | --- | --- | --- |
| OrderID | Order ID Number | INT | 25 | Primary Key | Y |
| CustomerID | Customer of this order | INT | 25 | Foreign Key | Y |
| EmployeeID | Sales representative of this order | INT | 25 | Foreign Key | Y |
| ShipperID | Shipper of the order | INT | 25 | Foreign Key | Y |
| Order\_status\_ID | Order status of the order | INT | 25 | Foreign Key | Y |
| Payment\_ID | Payment of the order | INT | 25 | Foreign Key | Y |
| Shipped\_date | The date that order been shipped | DATETIME | YYYY-MM-DD |  | Y |
| Order\_date | Order’s date | DATETIME | YYYY-MM-DD |  | Y |

ORDER\_STATUS

| Column Name | Description | Data Type | Size | Constraint Type | Not Null? |
| --- | --- | --- | --- | --- | --- |
| Order\_status\_ID | Order status ID number | INT | 25 | Primary Key | Y |
| Status\_name | Order status like, In process, delivered | VARCHAR | 45 |  | Y |

PAYMENT

| Column Name | Description | Data Type | Size | Constraint Type | Not Null? |
| --- | --- | --- | --- | --- | --- |
| PaymentID | Payment ID Number | INT | 25 | Primary Key | Y |
| InvoiceID | Invoice ID | INT | 25 | Foreign Key | Y |
| Payment\_date | Payment date | DATETIME | YYYY-MM-DD |  | Y |
| Payment\_amount | Payment amount | DECIMAL | (10,2) |  | Y |
| Payment\_method\_ID | Payment method | INT | 25 | Foreign Key | Y |

PAYMENT\_METHOD

| Column Name | Description | Data Type | Size | Constraint Type | Not Null? |
| --- | --- | --- | --- | --- | --- |
| Payment\_method\_ID | Payment method ID Number | INT | 25 | Primary Key | Y |
| Payment\_way | Payment method like cash, card | VARCHAR | 45 |  | Y |

INVOICES

| Column Name | Description | Data Type | Size | Constraint Type | Not Null? |
| --- | --- | --- | --- | --- | --- |
| InvoiceID | Invoice ID Number | INT | 25 | Primary Key | Y |
| PaymentID | Payment ID of this invoice | INT | 25 | Foreign Key | Y |
| Invoice\_date | Invoice date of the payment | DATETIME | YYYY-MM-DD |  | Y |
| Invoice\_number | Invoice number of the payment | VARCHAR | 45 |  | Y |

ORDER\_INFORMATION

| Column Name | Description | Data Type | Size | Constraint Type | Not Null? |
| --- | --- | --- | --- | --- | --- |
| OrderID | Order ID Number | INT | 25 | Primary Key | Y |
| ProductID | Product ID Number of this order | INT | 25 | Primary Key | Y |
| Quantity | Quantity ordered | INT | 25 |  | Y |
| Purchased\_price | Order price | DECIMAL | (10,2) |  | Y |

PRODUCT

| Column Name | Description | Data Type | Size | Constraint Type | Not Null? |
| --- | --- | --- | --- | --- | --- |
| ProductID | Product ID Number | INT | 25 | Primary Key | Y |
| Name | Product name | VARCHAR | 45 |  | Y |
| Description | Product description | TEXT | 50 |  | N |
| Unit\_price | Product unit price | DECIMAL | (7,2) |  | Y |
| Quantity\_in\_stock | Quantity in stock of the product | INT | 25 |  | Y |

PRODUCT\_SUPPLIER

| Column Name | Description | Data Type | Size | Constraint Type | Not Null? |
| --- | --- | --- | --- | --- | --- |
| ProductID | Product ID Number | INT | 25 | Primary Key | Y |
| SupplierID | Supplier ID Number | INT | 25 | Primary Key | Y |
| Note | Notes for some further instruction | TEXT | 50 |  | N |

SUPPLIER

| Column Name | Description | Data Type | Size | Constraint Type | Not Null? |
| --- | --- | --- | --- | --- | --- |
| SupplierID | Supplier ID Number | INT | 25 | Primary Key | Y |
| CompanyName | Supplier company name | VARCHAR | 45 |  | Y |
| Phone | Supplier company phone number | VARCHAR | 24 |  | Y |
| Address | Supplier company address | VARCHAR | 45 |  | N |

### Integrity Rules

The most frequent use data types in this system are INT, DECIMAL, VARCHAR, DATETIME and TEXT.

Most of the primary key and foreign key ID is the INT type allows input 25 valid values, Not Null and auto-increment. Detailed information like name, phone, address, city, state is VARCHAR type, but the valid values are quiet different, for example address is variable-length string of up to 45 characters, but country and state is variable-length string of up to 15 characters. Phone number would be also be VARCHAR type, since it sometimes may include punctuation marks like ‘(’ , ‘-’. For the DATETIME type like payment date, shipped date, order date, the system will only accept the form YYYY-MM-DD. For all the price like unit price of the product, total amount of payment paid by the customer will use DECIMAL(7,2) data type, allows the input be a decimal number with 2 decimal places. If there has any further explanation or description of the product or supplier, it will be TEXT format, allows the input string up to 50 characters.

About the referential integrity, started with the EMPLOYEE table, there are two foreign key in EMPLOYEE, one is referenced to itself, which is ReportTo, indicate which EMPLOYEE ID is each employee’s manager. Another one is OFFICE ID reference to the OFFICE table, shows that where did each employee work at. OFFICE table has one foreign key reference to EMPLOYEE table, shows every employee working at each office location. CUSTOMER table has one foreign key reference to EMPLOYEE table, indicate who is the sales representative of each customer. ORDER table is a complicated one since it’s the core of the business and related with all the service, so it has five foreign key. CUSTOMER ID reference to CUSTOMER table, gets the information of who order this, EMPLOYEE ID reference to EMPLOYEE table, indicate who is the sales representative that in charge of this order, SHIPPER ID reference to SHIPPER table, indicate the who is the shipper of this order, PAYMENT ID reference to PAYMENT table, record that all the payment information of this order, finally is the ORDER\_STATUS\_ID reference to ORDER\_STATUS table to keep track of the order. In the PAYMENT table, there are two foreign keys, separately record the payment information that is needed for each order, PAYMENT\_METHOD and INVOICES, so the PAYMENT\_METHOD\_ID is reference to PAYMENT\_METHOD table, check what customer use to pay, cash or card or Paypal. INVOICE ID reference to INVOICES table, indicate which invoice number belongs to each payment. In the intersection entity ORDER\_INFORMATION and PRODUCT\_SUPPLIERS, they both have two primary key combine together to form a composite primary key. In ORDER\_INFORMATION it’s ORDERID from order table and PRODUCT ID from PRODUCT table. In PRODUCT\_SUPPLIERS it’s PRODUCT ID from PRODUCT table and SUPPLIERID from SUPPLIER table.

### Operational Rules

There are three types of users for this system. The employee from sales department, order tracking department and supply chain department. The employee level is manager and normal employee, they will grant different privileges while using their account. Normal employee’s account will only able to use insert method to input the data. Manager account will have the access to use insert/delete/update/retrieve/execute, etc. Shipper can assigned multiple orders. In SUPPLIER table, the company name should be unique, won’t allow the employee enter two identical company name in the table. OFFICE table created with on Delete CASCADE constraints, when the office announced permanently closed and office ID be deleted in the OFFICE table every employee worked in that office will also be fired, so the employee information in the EMPLOYEE table will also be deleted.

### Operations

As I mentioned above, there are tree types of users for the system. For the sales representative they will using insert/delete/update/retrieve on the CUSTOMER table, ORDER table and PAYMENT table. When there is a customer coming to the store and make a purchase, each sales is responsible to insert or update the customer information, then add a new order in the order table, finally record the payment and invoice information in the system.

For the order tracking department employees they will in charge of assign and track the orders that need to be shipped. So maintain SHIPPER table, ORDER table and ORDER\_STATUS table is their job. Every time when there an order need to shipped, the employee will assign the order to an shipper, and during the shipping process, they need to frequently update the status of the order, for example in the beginning, the order status should be Processing, after the shipper get the product, order status should be on the way. Finally when the driver arrives, the status will be updated to delivered.

For the supply chain employee, they are responsible for PRODUCT table and SUPPLIER table, frequently check the potential supplier and clean out the current resource. So they need to frequently update the SUPPLIER table to add or delete the resource and update the new upcoming products list in the PRODUCT table.

## Security

This system will grant privileges to different level’s users for the data security. When a new employee is hired, firstly check which department he will be in. For example The supply chain department will only be able to see the PRODUCT, SUPPLIER and PRODUCT\_SUPPLIER table, preventing them to get the customer or payment information. Then check the level, if it’s manager level, then give them the privileges to use delete and update method in the system. The normal employee can only use insert method, preventing data loss like the new employee accidentally delete or update important date. If they insert something wrong, they have to let the manager know.

## Database Backup and Recovery

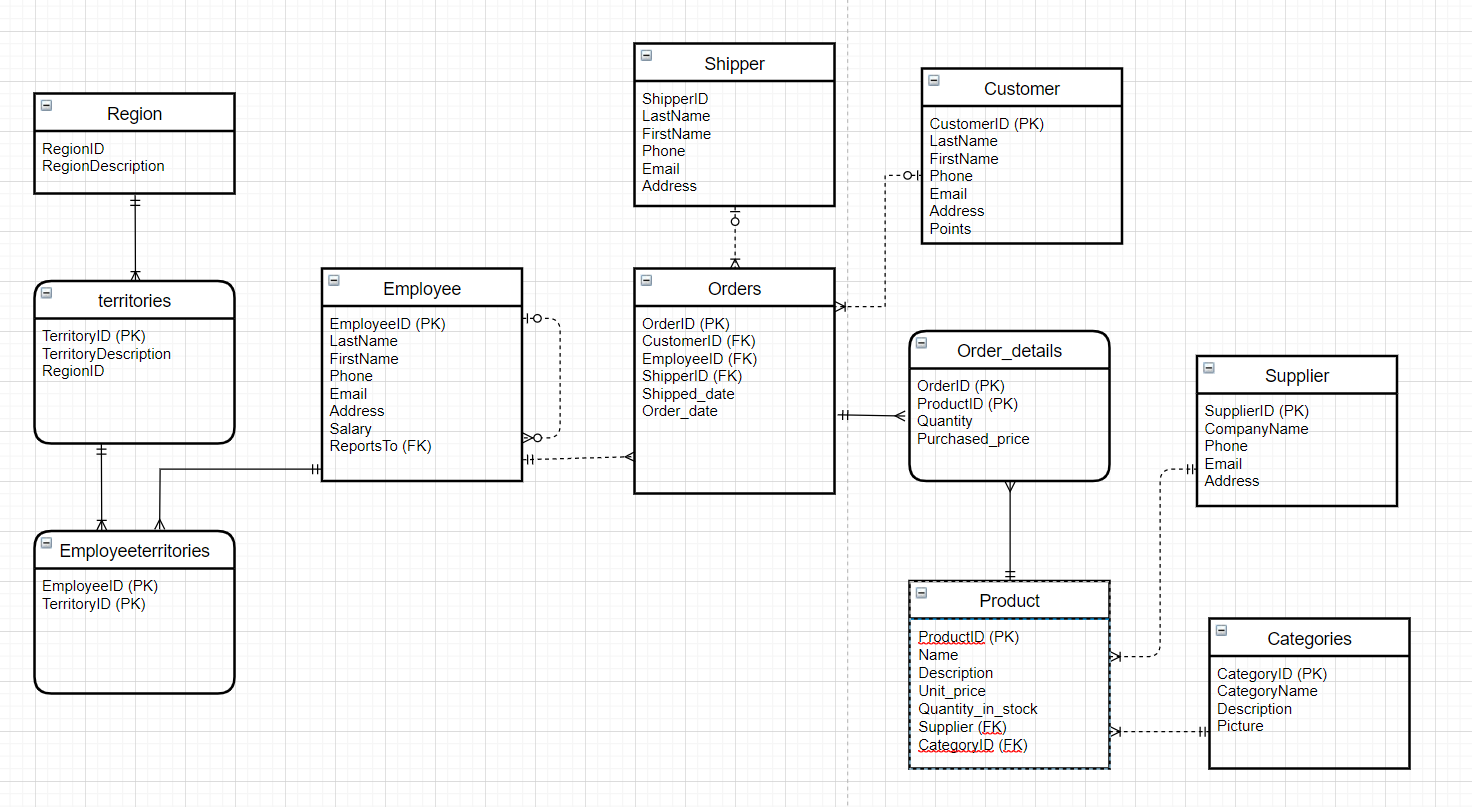
When using the Mysql Workbench, every time finished the editing(insert/delete/update/retrieve) of the system and forward engine. The system tool will automatically generate the SQL script. Save the script as a sql file for backup. If there are some data got accidentally delete, use the sql file to recover the system.

## Using Database Design or CASE Tool

The MySQL Workbench brings power through visualization to DBAs, developers and or data architects. t It's free, easy to install and easy to use. It allows individuals to design, model, generate, and manage databases in a single convenient software package. To be a bit more specific, a user can create simple to complex ER diagrams, forward and reverse engineering of physical database designs, Schema Synchronization and Comparison utilities, and DBDoc the point-and-click database documentation tool.

## Other Possible E/R Relationships

Here is an other possible E/R diagram



# Implementation Description

Using the Mysql workbench to implement all the tables and data, require

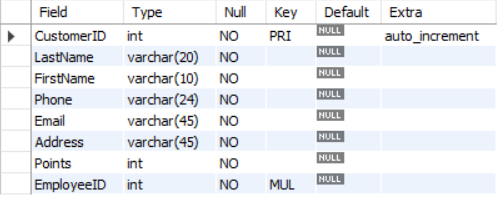
SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

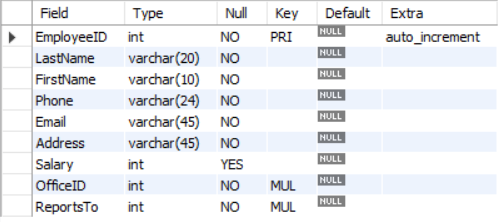
SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

## Data Dictionary

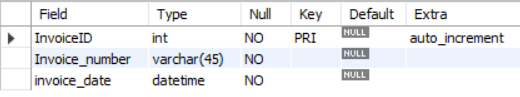
CUSTOMER



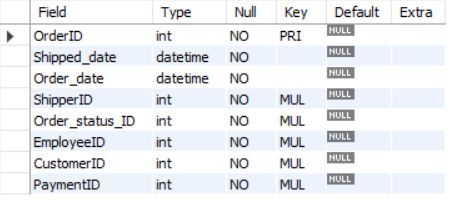
EMPLOYEE



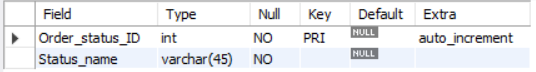
INVOICES



OFFICE

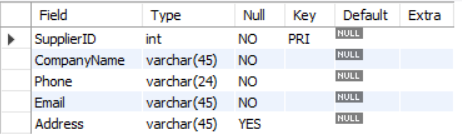
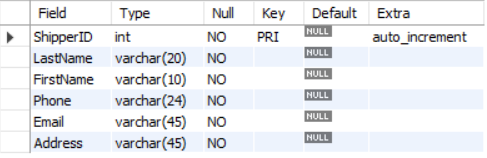
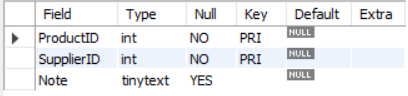
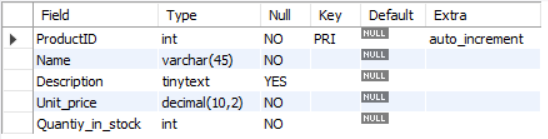
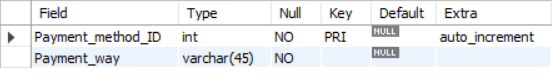
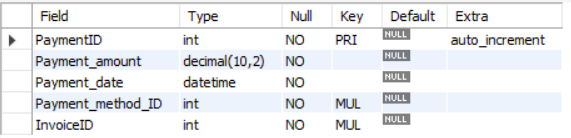
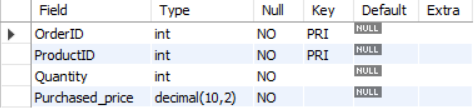
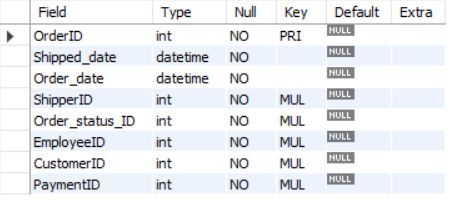


ORDER\_STATUS



ORDERS

ORDER\_INFORMATION  
  
PAYMENT  
  
PAYMENT\_METHOD  
  
PRODUCT  
  
PRODUCT\_SUPPLIER  
  
SHIPPER  
  
SUPPLIER



## Advanced Features

There is a trigger called decrements\_after\_purchase in the system use to maintain the product quantity. This will update the quantity\_in\_stock for each product in the PRODUCT table. Set with the ORDER\_INFORMATION table, the Quantity in this table will record how many does the customer purchased, after that use the quantity\_in\_stock minus the quantity customer purchase to get the NEW quantity\_in\_stock of that product.

There are three stored procedures in the system to help employee analyze the situation by the time

Sales by Year - Join the employee and order’s table, then group by, check all the order’s go through each employee.

Ten Most Expensive Products - Select the product table and sort the product based on their unit\_price, make it descending order and limit 10. Then we can always which 10 products are most expensive.

Customer Orders - Join customer and order’s table then group by, check how many orders did each customer made. Or simply sort the customer table by their reward points to check who has the highest reward points.

## Queries

The Database system can help retail store by generating queries to help analysis business like below

### TOP 3 Customers

.

SELECT \*

FROM Customer

ORDER BY Points DESC

LIMIT 3

### Customer Order Status

Same as above.

SELECT c.LastName, c.FirstName, o.OrderID, o.Order\_date, os.Status\_name AS status

FROM Orders o

JOIN Customer c

ON o.CustomerID = c.CustomerID

JOIN Order\_Status os

ON o.Order\_status\_ID = os.Order\_status\_ID

### Yearly Order of 2019

Same as above.

SELECT c.LastName, c.FirstName o.OrderID, o.CustomerID, o.EmployeeID, o.Order\_date

FROM Orders o

JOIN Customer c

ON o.CustomerID = c.CustomerID

WHERE o.Order\_date BETWEEN ‘2019-01-01’ AND ‘2019-12-31’

### Order Subtotal

SELECT sum(Purchased\_price) AS Order\_Subtotal

FROM Order\_Information

### Second highest Salary

SELECT EmployeeID, LastName, FirstName, Salary

FROM EMPLOYEE

ORDER BY Salary DESC

LIMIT 1 OFFSET 1

### Product Above Average Price

SELECT ProductID, Name,Unit\_price

FROM Product

WHERE Unit\_price < (SELECT AVG (Unit\_price) FROM Product)

### Favorite Payment method

SELECT pm.Payment\_way, COUNT(pm.Payment\_method\_ID) AS Times

FROM Payment p

JOIN Payment\_Method pm

ON p.Payment\_method\_ID = pm.Payment\_method\_ID

GROUP BY p.Payment\_method\_ID

### Summary of Good Sales

SELECT EmployeeID,COUNT(orderID)

FROM Orders

GROUP BY EmployeeID

ORDER BY COUNT(OrderID) DESC

LIMIT 1

# CRUD Matrix

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Function/Entity Interaction | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | E10 | E11 | E12 | E13 |
| F1 | CRUD |  |  |  |  |  |  |  |  |  |  |  |  |
| F2 |  | CRUD |  |  |  |  |  |  |  |  |  |  |  |
| F3 |  |  | CRUD |  |  |  |  |  |  |  |  |  |  |
| F4 |  |  |  | CRUD |  |  |  |  |  |  |  |  |  |
| F5 |  |  |  |  | CRUD |  |  |  |  |  |  |  |  |
| F6 |  |  |  |  |  | CRUD |  |  |  |  |  |  |  |
| F7 |  |  |  |  |  |  | CRUD |  |  |  |  |  |  |
| F8 |  |  |  |  |  |  |  | CRUD |  |  |  |  |  |
| F9 |  |  |  |  |  |  |  |  | CRUD |  |  |  |  |
| F10 |  |  |  |  |  |  |  |  |  | CRUD |  |  |  |
| F11 |  |  |  |  |  |  |  |  |  |  | CRUD |  |  |
| F12 |  |  |  |  |  |  |  |  |  |  |  | CRUD |  |
| F13 |  |  |  |  |  |  |  |  |  |  |  |  | CRUD |
| F14 |  |  |  | CRUD |  |  |  |  |  |  |  |  |  |

## List of Entity Types

E1: OFFICE

E2: EMPLOYEE

E3: CUSTOMER

E4: SHIPPER

E5: ORDERS

E6: ORDER\_STATUS

E7: PAYMENT

E8: PAYMENT\_METHOD

E9: INVOICES

E10: ORDER\_INFORMATION

E11: PRODUCT

E12: PRODUCT\_SUPPLIER

E13: SUPPLIER

## List of Functions

F1: insert/update/delete/retrieve an office

F2: insert/update/delete/retrieve an employee

F3: insert/update/delete/retrieve a customer

F4: insert/update/delete/retrieve a shipper

F5: insert/update/delete/retrieve an order

F6: insert/update/delete/retrieve an order status

F7: insert/update/delete/retrieve a payment

F8: insert/update/delete/retrieve a payment method

F9: insert/update/delete/retrieve an invoice

F10: get the order details

F11: insert/update/delete/retrieve a product

F12: get the note of product from which supplier

F13: insert/update/delete/retrieve a supplier

F14: assign an order to a shipper

# Concluding Remarks

I learned a lot through this database project including considering multiple possible ER diagram. I realize that the database design doesn’t have an absolute answer, the designer need to based on the business rule and real data situation to consider a relatively better design. The strength of this system is it covers the whole business process, have very clear business logic, let every department user know what they specifically need to do, what’s their role in this system. The thing I need to improve might be some detailed table. If I have more time, I’ll add more tables to clarify each table, like the employee should have department table or territory table to show what their strength and role. For the product table I can add a category table to more clearly arrange all the product. Make the overall system more mature for company to use.

Appendices

Data security are only performed on grant privileges based on the user level. Don’t have any other way to protect the system. Data needs to be processed in a way that ensures nothing is lost. Even in a nonrelational database or in a distributed cluster, there must be ACID guarantees both across the database and throughout the cluster. The challenge for fully-transactional databases is performance. ACID costs time. A good database is one that can give you both superior performance and transactional guarantees.

Appendix A - DDL, INSERT, SELECT Statements

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

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

-- Schema mydb

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

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

-- Schema mydb

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

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;

USE `mydb` ;

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

-- Table `mydb`.`Office`

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

CREATE TABLE IF NOT EXISTS `mydb`.`Office` (

`OfficeID` INT NOT NULL AUTO\_INCREMENT,

`OfficeName` VARCHAR(45) NOT NULL,

`Phone` VARCHAR(24) NOT NULL,

`Address` VARCHAR(45) NOT NULL,

`Country` VARCHAR(15) NOT NULL,

`State` VARCHAR(15) NOT NULL,

`EmployeeID` INT NOT NULL,

PRIMARY KEY (`OfficeID`),

INDEX `fk\_Office\_Employee1\_idx` (`EmployeeID` ASC) VISIBLE,

CONSTRAINT `fk\_Office\_Employee1`

FOREIGN KEY (`EmployeeID`)

REFERENCES `mydb`.`Employee` (`EmployeeID`)

ON DELETE CASCADE

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `mydb`.`Employee`

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

CREATE TABLE IF NOT EXISTS `mydb`.`Employee` (

`EmployeeID` INT NOT NULL AUTO\_INCREMENT,

`LastName` VARCHAR(20) NOT NULL,

`FirstName` VARCHAR(10) NOT NULL,

`Phone` VARCHAR(24) NOT NULL,

`Email` VARCHAR(45) NOT NULL,

`Address` VARCHAR(45) NOT NULL,

`Salary` INT NULL,

`OfficeID` INT NOT NULL,

`ReportsTo` INT NOT NULL,

PRIMARY KEY (`EmployeeID`),

INDEX `fk\_Employee\_Office\_idx` (`OfficeID` ASC) VISIBLE,

INDEX `fk\_Employee\_Employee1\_idx` (`ReportsTo` ASC) VISIBLE,

CONSTRAINT `fk\_Employee\_Office`

FOREIGN KEY (`OfficeID`)

REFERENCES `mydb`.`Office` (`OfficeID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Employee\_Employee1`

FOREIGN KEY (`ReportsTo`)

REFERENCES `mydb`.`Employee` (`EmployeeID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `mydb`.`Customer`

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

CREATE TABLE IF NOT EXISTS `mydb`.`Customer` (

`CustomerID` INT NOT NULL AUTO\_INCREMENT,

`LastName` VARCHAR(20) NOT NULL,

`FirstName` VARCHAR(10) NOT NULL,

`Phone` VARCHAR(24) NOT NULL,

`Email` VARCHAR(45) NOT NULL,

`Address` VARCHAR(45) NOT NULL,

`Points` INT NOT NULL,

`EmployeeID` INT NOT NULL,

PRIMARY KEY (`CustomerID`),

INDEX `fk\_Customer\_Employee1\_idx` (`EmployeeID` ASC) VISIBLE,

CONSTRAINT `fk\_Customer\_Employee1`

FOREIGN KEY (`EmployeeID`)

REFERENCES `mydb`.`Employee` (`EmployeeID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `mydb`.`Shipper`

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

CREATE TABLE IF NOT EXISTS `mydb`.`Shipper` (

`ShipperID` INT NOT NULL AUTO\_INCREMENT,

`LastName` VARCHAR(20) NOT NULL,

`FirstName` VARCHAR(10) NOT NULL,

`Phone` VARCHAR(24) NOT NULL,

`Email` VARCHAR(45) NOT NULL,

`Address` VARCHAR(45) NOT NULL,

PRIMARY KEY (`ShipperID`))

ENGINE = InnoDB;

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

-- Table `mydb`.`Order\_Status`

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

CREATE TABLE IF NOT EXISTS `mydb`.`Order\_Status` (

`Order\_status\_ID` INT NOT NULL AUTO\_INCREMENT,

`Status\_name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`Order\_status\_ID`))

ENGINE = InnoDB;

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

-- Table `mydb`.`Payment\_method`

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

CREATE TABLE IF NOT EXISTS `mydb`.`Payment\_method` (

`Payment\_method\_ID` INT NOT NULL AUTO\_INCREMENT,

`Payment\_way` VARCHAR(45) NOT NULL,

PRIMARY KEY (`Payment\_method\_ID`))

ENGINE = InnoDB;

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

-- Table `mydb`.`Invoices`

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

CREATE TABLE IF NOT EXISTS `mydb`.`Invoices` (

`InvoiceID` INT NOT NULL AUTO\_INCREMENT,

`Invoice\_number` VARCHAR(45) NOT NULL,

`invoice\_date` DATETIME NOT NULL,

PRIMARY KEY (`InvoiceID`))

ENGINE = InnoDB;

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

-- Table `mydb`.`Payment`

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

CREATE TABLE IF NOT EXISTS `mydb`.`Payment` (

`PaymentID` INT NOT NULL AUTO\_INCREMENT,

`Payment\_amount` DECIMAL(10,2) NOT NULL,

`Payment\_date` DATETIME NOT NULL,

`Payment\_method\_ID` INT NOT NULL,

`InvoiceID` INT NOT NULL,

PRIMARY KEY (`PaymentID`),

INDEX `fk\_Payment\_Payment\_method1\_idx` (`Payment\_method\_ID` ASC) VISIBLE,

INDEX `fk\_Payment\_Invoices1\_idx` (`InvoiceID` ASC) VISIBLE,

CONSTRAINT `fk\_Payment\_Payment\_method1`

FOREIGN KEY (`Payment\_method\_ID`)

REFERENCES `mydb`.`Payment\_method` (`Payment\_method\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Payment\_Invoices1`

FOREIGN KEY (`InvoiceID`)

REFERENCES `mydb`.`Invoices` (`InvoiceID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `mydb`.`Orders`

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

CREATE TABLE IF NOT EXISTS `mydb`.`Orders` (

`OrderID` INT NOT NULL,

`Shipped\_date` DATETIME NOT NULL,

`Order\_date` DATETIME NOT NULL,

`ShipperID` INT NOT NULL,

`Order\_status\_ID` INT NOT NULL,

`EmployeeID` INT NOT NULL,

`CustomerID` INT NOT NULL,

`PaymentID` INT NOT NULL,

PRIMARY KEY (`OrderID`),

INDEX `fk\_Orders\_Shipper1\_idx` (`ShipperID` ASC) VISIBLE,

INDEX `fk\_Orders\_Order\_Status1\_idx` (`Order\_status\_ID` ASC) VISIBLE,

INDEX `fk\_Orders\_Employee1\_idx` (`EmployeeID` ASC) VISIBLE,

INDEX `fk\_Orders\_Customer1\_idx` (`CustomerID` ASC) VISIBLE,

INDEX `fk\_Orders\_Payment1\_idx` (`PaymentID` ASC) VISIBLE,

CONSTRAINT `fk\_Orders\_Shipper1`

FOREIGN KEY (`ShipperID`)

REFERENCES `mydb`.`Shipper` (`ShipperID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Orders\_Order\_Status1`

FOREIGN KEY (`Order\_status\_ID`)

REFERENCES `mydb`.`Order\_Status` (`Order\_status\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Orders\_Employee1`

FOREIGN KEY (`EmployeeID`)

REFERENCES `mydb`.`Employee` (`EmployeeID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Orders\_Customer1`

FOREIGN KEY (`CustomerID`)

REFERENCES `mydb`.`Customer` (`CustomerID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Orders\_Payment1`

FOREIGN KEY (`PaymentID`)

REFERENCES `mydb`.`Payment` (`PaymentID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `mydb`.`Product`

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

CREATE TABLE IF NOT EXISTS `mydb`.`Product` (

`ProductID` INT NOT NULL AUTO\_INCREMENT,

`Name` VARCHAR(45) NOT NULL,

`Description` TEXT(50) NULL,

`Unit\_price` DECIMAL(10,2) NOT NULL,

`Quantiy\_in\_stock` INT NOT NULL,

PRIMARY KEY (`ProductID`))

ENGINE = InnoDB;

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

-- Table `mydb`.`Supplier`

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

CREATE TABLE IF NOT EXISTS `mydb`.`Supplier` (

`SupplierID` INT NOT NULL,

`CompanyName` VARCHAR(45) NOT NULL,

`Phone` VARCHAR(24) NOT NULL,

`Email` VARCHAR(45) NOT NULL,

`Address` VARCHAR(45) NULL,

PRIMARY KEY (`SupplierID`))

ENGINE = InnoDB;

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

-- Table `mydb`.`Orders\_information`

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

CREATE TABLE IF NOT EXISTS `mydb`.`Orders\_information` (

`OrderID` INT NOT NULL,

`ProductID` INT NOT NULL,

`Quantity` INT NOT NULL,

`Purchased\_price` DECIMAL(10,2) NOT NULL,

PRIMARY KEY (`OrderID`, `ProductID`),

INDEX `fk\_Orders\_has\_Product\_Product1\_idx` (`ProductID` ASC) VISIBLE,

INDEX `fk\_Orders\_has\_Product\_Orders1\_idx` (`OrderID` ASC) VISIBLE,

CONSTRAINT `fk\_Orders\_has\_Product\_Orders1`

FOREIGN KEY (`OrderID`)

REFERENCES `mydb`.`Orders` (`OrderID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Orders\_has\_Product\_Product1`

FOREIGN KEY (`ProductID`)

REFERENCES `mydb`.`Product` (`ProductID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `mydb`.`Product\_Supplier`

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

CREATE TABLE IF NOT EXISTS `mydb`.`Product\_Supplier` (

`ProductID` INT NOT NULL,

`SupplierID` INT NOT NULL,

`Note` TEXT(50) NULL,

PRIMARY KEY (`ProductID`, `SupplierID`),

INDEX `fk\_Product\_has\_Supplier\_Supplier1\_idx` (`SupplierID` ASC) VISIBLE,

INDEX `fk\_Product\_has\_Supplier\_Product1\_idx` (`ProductID` ASC) VISIBLE,

CONSTRAINT `fk\_Product\_has\_Supplier\_Product1`

FOREIGN KEY (`ProductID`)

REFERENCES `mydb`.`Product` (`ProductID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Product\_has\_Supplier\_Supplier1`

FOREIGN KEY (`SupplierID`)

REFERENCES `mydb`.`Supplier` (`SupplierID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

Appendix B - Data Dictionary Index

Every table sorted by ascending order based on their tables’ unique ID, normally the primary key.

References

<https://dev.mysql.com/doc/workbench/en/wb-requirements.html>

<https://www.databasejournal.com/features/mysql/article.php/3918776/MySQL-Workbench---Top-13-things-you-should-know.htm#:~:text=The%20MySQL%20Workbench%20brings%20power,a%20single%20convenient%20software%20package.>

<https://www3.ntu.edu.sg/home/ehchua/programming/sql/MySQL_Beginner.html#zz-3.4>

<https://jaxenter.com/overcome-5-common-database-challenges-149559.html>