Documentation

HP

2023

# **Data Analysis**

When I was observing the spreadsheet, I noticed many rows had no values captured and there was redundancy within the data, in order to be able to reorganize the spreadsheet data into a database that supports an online grocery business as they expand their offerings, I proceeded to reorganize the data into relational tables using the modeling tool in MySQL workbench, but firstly I needed to find out entities that needed to be store and how will these entities be related in the database.

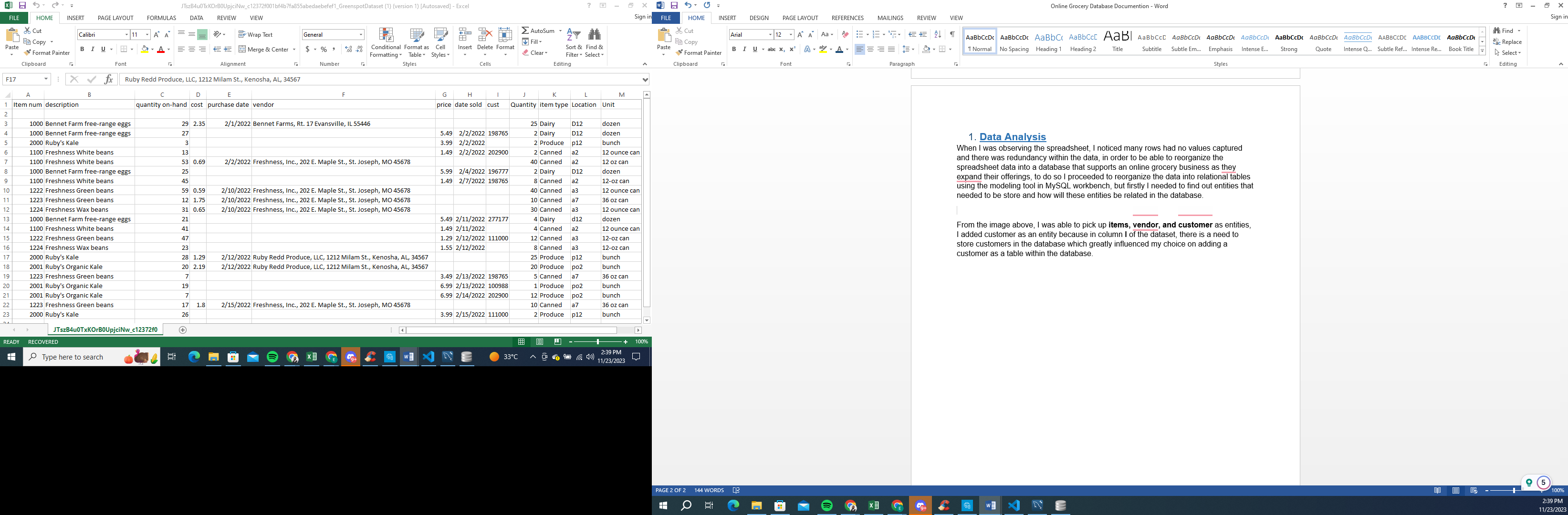


Figure 1.

From the image above, I was able to pick up **item, vendor, order, order\_item, transfer, inventory, location, delivery, deliveryDetail, and customer** as entities, I added customer as an entity because in column **I** of the dataset, there is a need to store customers’ information in the database which greatly influenced my choice on adding a customer as a table within the database. Afterwards, I needed to find out how these entities would be related within the database to create a scalable relational database which I managed to accomplish according to the following:

## ITEM

Item is the main table or starting point for designing a database system, as I observed the spreadsheet we could notice some attributes that should be grouped for the item entity such as:

* ItemNum: This should be a unique number and the primary identifier (natural key) of the entity. I used an **INTEGER** type for this attribute.
* ItemDescription: A more detailed description of the item. I used **VARCHAR(2000)** datatype.
* ItemType: The item’s type. I used **VARCHAR(100)** datatype.

## LOCATION

This entity represents places where inventory is located. The online grocery store has many locations and each location has one or more inventory to facilitate the delivery of items to customers.

* LocationID: This attribute should be a unique ID number and the primary identifier (Surrogate primary key) of the entity. I used **INTEGER** datatype.
* LocationName: This attribute stores the name of the location. I used **VARCHAR(100)** datatype.
* LocationAddress: This attribute stores the full address, I used **VARCHAR(200)** datatype. The locationAddress could be broken into several attributes, for simplicity purposes I decided to leave it as it is.

## VENDOR

Online grocery purchases items from vendors, a vendor table is needed to store information about vendors.

* VendorID: This attribute should be a unique ID and the primary identifier (surrogate primary key) of the entity. I used an **INTEGER** datatype.
* VendorName: The vendor’s name. I used an **VARCHAR(100)** datatype
* VendorAddress: The vendor’s address. I used **VARCHAR(200)** datatype. The vendor’s address could be broken into several attributes but I kept it simple.
* Price: This attribute stores information about the price of of items. I used **DECIMAL(10,2)** datatype.

## ORDER & ORDER\_ITEM

When online grocery purchase items from provider, they include information about the quantities that need to be delivered, the date sold. This information is store in the order and order\_item entities:

**ORDER**

* OrderID: This attribute should be used as a unique ID number and surrogate primary key of the entity. I used an **INTEGER** datatype.
* Cost: This attribute stores information about the cost of procurement of items. I used **DECIMAL(10,2)** datatype.
* Date\_sold: The date when items have been sold to the company. I used **DATE** datatype.
* Quantity: The amount of specific items ordered for the company. I used **INTEGER** datatype.
* VendorID: This attributes should be used as a **foreign key** to create a relationship between a vendor and orders, and assign orders to a vendor.

**ORDER\_ITEM**

* Order\_itemID: This attribute should be used as a unique ID number and surrogate primary key of the entity. I used an I**NTEGER** datatype.
* OrderID: This attribute should be used as a foreign key to create a relationship between the **ORDER** table and the **ORDER\_ITEM** junction table.
* ItemNum: This attribute should be used as a foreign key to create a relationship between the **ITEM** table and the **ORDER\_ITEM** junction table.
* ExpectedDate: The date when the item should arrive at the company’s hand. I used **DATE** datatype.
* ActualDate: The date when the items were received by the company’s hand. I used a **DATE** datatype.

## INVENTORY AND TRANSFER

The inventory entity provides information on the quantity of available items. Each item may be stored in several inventories, and each inventory may contain many different items. This gave me a many-to-many relationship, to avoid multiple entries a junction table **TRANSFER** needed to be created to sustain the lowest level of normalization.

**INVENTORY**

* InventoryID: This attribute should be used as a unique ID number and the surrogate primary key. I used an **INTEGER** datatype.
* Quanty\_on\_hand: The quantity on hand of items. I used an **INTEGER** datatype.
* InvLocation: The location of items within the inventory. I used **VARCHAR(50)** datatype.
* Unit: The name of the unit where the item is categorized. I used an **VARCHAR(50)** datatype.
* LocationID: This attributes should be used as a **foreign key** to create a relationship between an inventory and physical locations.

**TRANSFER**

* TransferID: This attribute should be used as a unique ID number and the surrogate primary key. I used an **INTEGER** datatype.
* TransferQuantity: The amount of a specific item has been sent to the inventory. I used an **INTEGER** datatype.
* ReceivedDate: The date when the item arrived at the inventory. I used **DATE** datatype.
* ItemNum: This attribute should be used as a foreign key to create a relationship between the **ITEM** table and the **TRANSFER** junction table.
* InventoryID: This attribute should be used as a foreign key to create a relationship between the **INVENTORY** table and the **TRANSFER** junction table.

## CUSTOMER

Online grocery sells items to customers, so it is needed to store information about customers.

* CustomerID: This attribute should be used as a unique ID number and the surrogate primary key. I used an **INTEGER** datatype.
* CustNum: This attribute should be used as a unique number and the natural candidate key. I used an **INTEGER** datatype.
* CustFirstName: This attribute should store the customer’s first name. I used a **VARCHAR(100)** datatype.
* CustLastName: This attribute should store the customer’s last name. I used a **VARCHAR(100)** datatype.
* CustPhoneNum: This attribute should store the customer’s phone number. I used an **INTEGER** datatype.
* CustAddress: The customer’s full address. I used a **VARCHAR(200)** datatype.

## DELIVERY AND DELIVERYDETAIL

Once the online grocery sells items to customers, it may include different items from different inventory locations, depending on the item’s availability and inventory’s proximity based on the customer’s address.

**DELIVERY**

* DeliveryID: This attribute should be used as a unique ID number and the surrogate primary key. I used an **INTEGER** datatype.
* CustomerID: This attribute should be used as a foreign key to create a relationship between the **CUSTOMER** table and the **DELIVERY** table.
* PurchaseDate: The date when a customer purchased items. I used a **DATE** datatype.

**DELEVERYDETAIL**

* DeliveryDetailID: This attribute should be used as a unique ID number and the surrogate primary key. I used an **INTEGER** datatype.
* DeliveryQuantity: The amount of items to be delivered to the customer. I used an **INTEGER** datatype.
* ExpectedDate: The date when the items should arrive at the customer’s address. I used a **DATE** datatype.
* ActualDate: This date when the items were delivered. I used a **DATE** datatype.
* DeliveryID: This attribute should be used as a foreign key to create a relationship between the **DELIVERY** table and the **DELIVERYDETAIL** junctiontable.
* InventoryID: This attribute should be used as a foreign key to create a relationship between the **INVENTORY** table and the **DELIVERYDETAIL** junctiontable.

## RELATIONSHIP BETWEEN ENTITIES

**VENDOR AND ORDER**

Each order is assigned to one vendor, not all vendors may have orders. A one to many (1:N) relationship needs to be established.

**ORDER AND ORDERITEM**

Each orderItem is assigned to one order, each order must have at least one or many orderItems. A one to many (1:N) relationship needs to be established.

**ORDERITEM AND ITEM**

Each orderItem must be associated with an item and each item may be included in one or many orders. A one-to-many relationship needs to be established.

**CUSTOMER AND DELIVERY**

Each delivery is assigned to a customer, each customer must have at least one order. A one to many (1:N) relationship needs to be established.

**DELIVERY AND DELIVERYDETAIL**

Each delivery must have at least one deliveryDetail, and each deliveryDetail belongs to one and only one delivery. This is a one to many relationship.

**INVENTORY AND DELIVERYDETAIL**

Each deliveryDetail is associated with an inventory, each inventory may have zero, one or many deliveryDetails. A one to many relationship needs to be established.

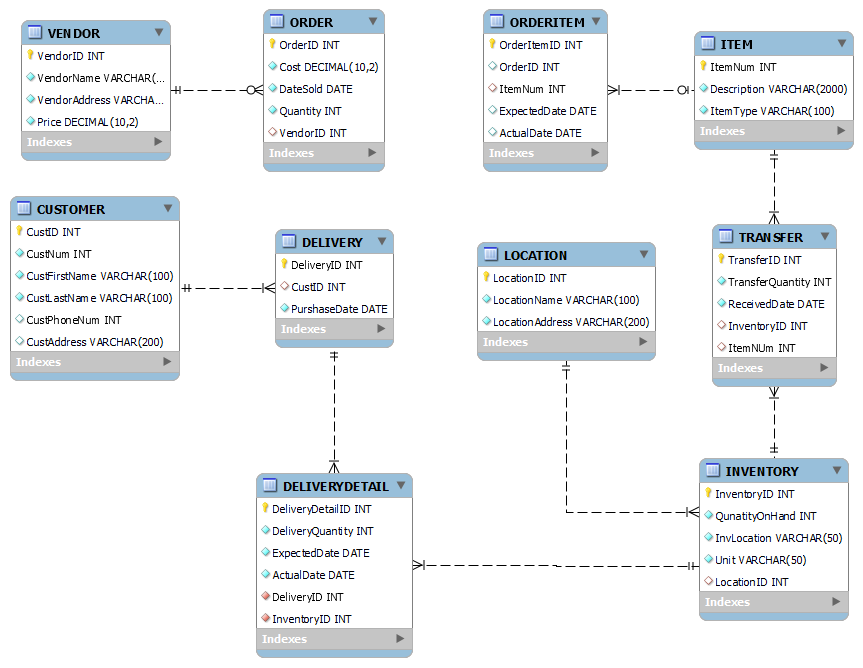
**ITEM AND INVENTORY**

Each item may be transferred to many inventory and each inventory may receive one or many items. A transfer table was created to resolve many to many relationship between ITEM and INVENTORY

**LOCATION AND INVENTORY**

Each location can have one or more inventories, an inventory must be assigned to one location. This is a one-to-many relationship.

## MODEL



*P.S: This is my first time after a long time project based on databases so I am open to improvements and constructive critics.*

## SQL SCRIPT

-- 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`.`VENDOR`

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

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

`VendorID` INT NOT NULL AUTO\_INCREMENT,

`VendorName` VARCHAR(100) NOT NULL,

`VendorAddress` VARCHAR(200) NOT NULL,

`Price` DECIMAL(10,2) NOT NULL,

PRIMARY KEY (`VendorID`))

ENGINE = InnoDB;

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

-- Table `mydb`.`ORDER`

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

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

`OrderID` INT NOT NULL AUTO\_INCREMENT,

`Cost` DECIMAL(10,2) NOT NULL,

`DateSold` DATE NOT NULL,

`Quantity` INT NOT NULL,

`VendorID` INT NULL,

PRIMARY KEY (`OrderID`),

INDEX `OrderID` () VISIBLE,

INDEX `VendorID\_idx` (`VendorID` ASC) VISIBLE,

CONSTRAINT `VendorID`

FOREIGN KEY (`VendorID`)

REFERENCES `mydb`.`VENDOR` (`VendorID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

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

-- Table `mydb`.`ITEM`

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

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

`ItemNum` INT NOT NULL AUTO\_INCREMENT,

`Description` VARCHAR(2000) NOT NULL,

`ItemType` VARCHAR(100) NOT NULL,

PRIMARY KEY (`ItemNum`))

ENGINE = InnoDB;

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

-- Table `mydb`.`ORDERITEM`

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

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

`OrderItemID` INT NOT NULL AUTO\_INCREMENT,

`OrderID` INT NULL,

`ItemNum` INT NULL,

`ExpectedDate` DATE NULL,

`ActualDate` DATE NULL,

PRIMARY KEY (`OrderItemID`),

INDEX `ItemNum\_idx` (`ItemNum` ASC) VISIBLE,

CONSTRAINT `ItemNum`

FOREIGN KEY (`ItemNum`)

REFERENCES `mydb`.`ITEM` (`ItemNum`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

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

-- Table `mydb`.`LOCATION`

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

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

`LocationID` INT NOT NULL AUTO\_INCREMENT,

`LocationName` VARCHAR(100) NOT NULL,

`LocationAddress` VARCHAR(200) NOT NULL,

PRIMARY KEY (`LocationID`))

ENGINE = InnoDB;

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

-- Table `mydb`.`INVENTORY`

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

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

`InventoryID` INT NOT NULL AUTO\_INCREMENT,

`QunatityOnHand` INT NOT NULL,

`InvLocation` VARCHAR(50) NOT NULL,

`Unit` VARCHAR(50) NOT NULL,

`LocationID` INT NULL,

PRIMARY KEY (`InventoryID`),

INDEX `LocationID\_idx` (`LocationID` ASC) VISIBLE,

CONSTRAINT `LocationID`

FOREIGN KEY (`LocationID`)

REFERENCES `mydb`.`LOCATION` (`LocationID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

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

-- Table `mydb`.`TRANSFER`

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

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

`TransferID` INT NOT NULL AUTO\_INCREMENT,

`TransferQuantity` INT NOT NULL,

`ReceivedDate` DATE NOT NULL,

`InventoryID` INT NULL,

`ItemNUm` INT NULL,

PRIMARY KEY (`TransferID`),

INDEX `ItemNum\_idx` (`ItemNUm` ASC) VISIBLE,

INDEX `InventoryID\_idx` (`InventoryID` ASC) VISIBLE,

CONSTRAINT `ItemNum`

FOREIGN KEY (`ItemNUm`)

REFERENCES `mydb`.`ITEM` (`ItemNum`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `InventoryID`

FOREIGN KEY (`InventoryID`)

REFERENCES `mydb`.`INVENTORY` (`InventoryID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

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

-- Table `mydb`.`CUSTOMER`

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

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

`CustID` INT NOT NULL AUTO\_INCREMENT,

`CustNum` INT NOT NULL,

`CustFirstName` VARCHAR(100) NOT NULL,

`CustLastName` VARCHAR(100) NOT NULL,

`CustPhoneNum` INT NULL,

`CustAddress` VARCHAR(200) NULL,

PRIMARY KEY (`CustID`),

UNIQUE INDEX `CustNum\_UNIQUE` (`CustNum` ASC) VISIBLE,

UNIQUE INDEX `CustPhoneNumb\_UNIQUE` (`CustPhoneNum` ASC) VISIBLE)

ENGINE = InnoDB;

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

-- Table `mydb`.`DELIVERY`

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

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

`DeliveryID` INT NOT NULL AUTO\_INCREMENT,

`CustID` INT NULL,

`PurshaseDate` DATE NOT NULL,

PRIMARY KEY (`DeliveryID`),

INDEX `CustID\_idx` (`CustID` ASC) VISIBLE,

CONSTRAINT `CustID`

FOREIGN KEY (`CustID`)

REFERENCES `mydb`.`CUSTOMER` (`CustID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

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

-- Table `mydb`.`DELIVERYDETAIL`

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

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

`DeliveryDetailID` INT NOT NULL AUTO\_INCREMENT,

`DeliveryQuantity` INT NOT NULL,

`ExpectedDate` DATE NOT NULL,

`ActualDate` DATE NOT NULL,

`DeliveryID` INT NOT NULL,

`InventoryID` INT NOT NULL,

PRIMARY KEY (`DeliveryDetailID`),

INDEX `DeliveryID\_idx` (`DeliveryID` ASC) VISIBLE,

INDEX `InventoryID\_idx` (`InventoryID` ASC) VISIBLE,

CONSTRAINT `DeliveryID`

FOREIGN KEY (`DeliveryID`)

REFERENCES `mydb`.`DELIVERY` (`DeliveryID`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `InventoryID`

FOREIGN KEY (`InventoryID`)

REFERENCES `mydb`.`INVENTORY` (`InventoryID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;