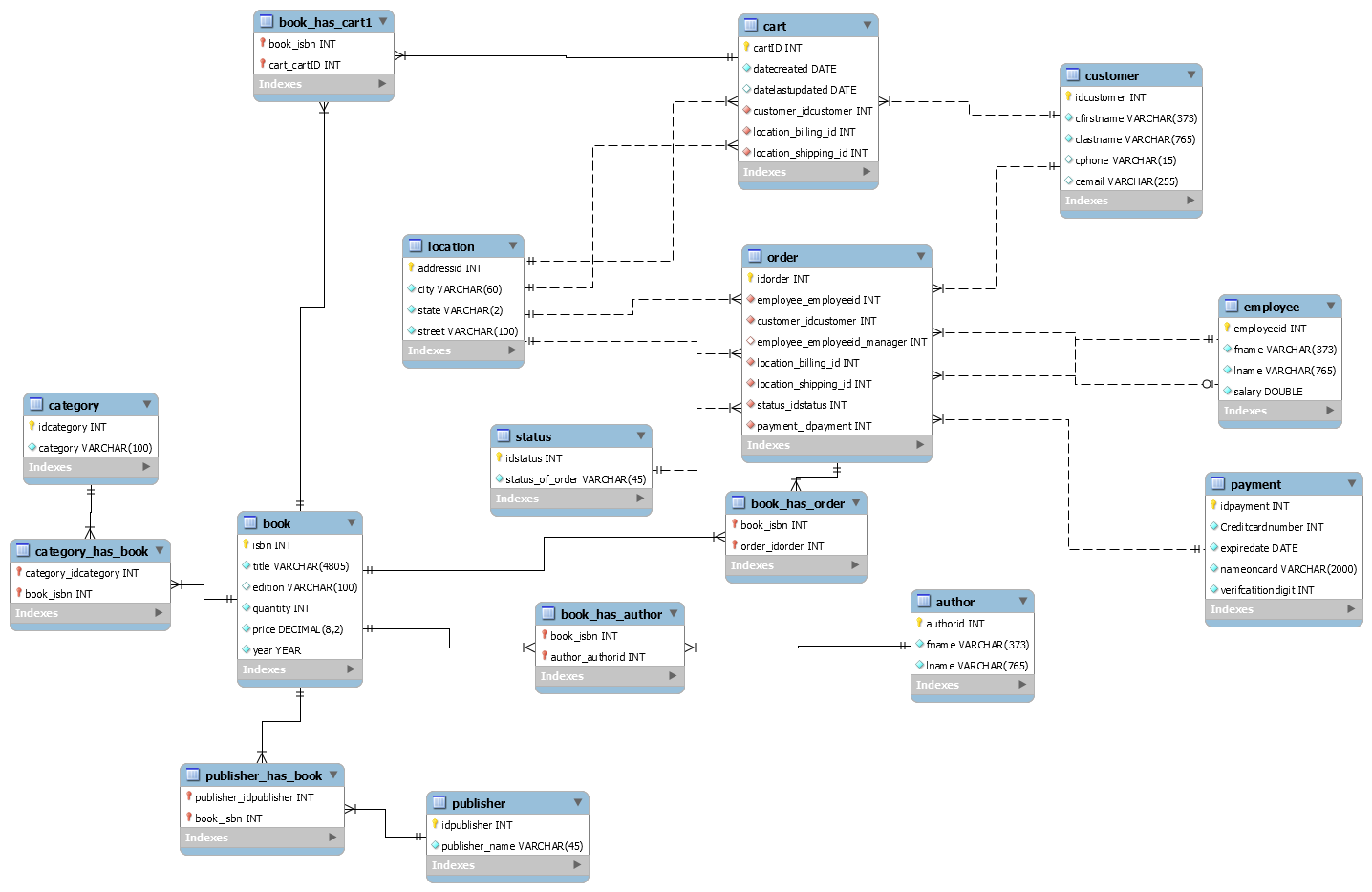
Business Rules:

New customers need to register first to get one account ID. The provided information includes: customer name, phone #, email address, and password. After registration, the customer will be assigned one account ID and he/she can login using account ID and password. One customer can only register one account and each account must belong to exact one customer.  
The bookstore keeps a large amount of books. Each book is identified by ISBN. For each book, the bookstore also needs to record its authors’ names, title, edition, year, category, publisher, quantity‐in‐stock, and price.  
One customer can place any number of orders. For each order, the bookstore needs to record who places this order, when, the order status, total price, shipping address, payment method, bill address, and ordered books. Note there is only one shipping address and one billing address for each order though the shipping address may not be the same as the billing address. Currently for payment method, it only accepts credit card, hence the bookstore needs to record credit card information.  
Customers can also manage their shopping carts. One customer can have any number of shopping carts. However, each shopping cart has exactly one customer. The shopping cart contains the following info: cart‐ID, name, date‐created, date‐last‐updated, books contained in this shopping cart, etc.   
The bookstore also has a number of employees which are identified by employee ID. The bookstore also needs to record employee’s name, address, salary. Some employees are ranked as manager.  
To better serve customers, each order will be assigned to one employee who will monitor and handle the order. For the order with total amount is more than $1,000 dollars, one manager should be assigned to.

EER Diagram:

First Diagram



Revised Diagram (It is zoomable)

Relational Schema:

First Schema

Order(OrderID, ShippingID, BillingID, CustomerID, Status, BookID)  
Customer(CustomerID, CustomerName, CustomerNumber, CustomerEmail)  
Cart(CardID, CustomerID, OrderID, Name, DateCreated, DateUpdated, BookID, Price)  
Employee(EmployeeID, Name, Address, Salary)  
Manager(EmployeeID)  
Location(AddressID, City, State, Zip)  
Book(ISSN, Title, Edition, Author, Category, Publisher, Quantity, Price)  
BookCart(CardID, ISSN)  
BookOrder(OrderID, ISSN)  
BookAuthor(ISSN, AuthorID)  
Author(AuthorID, Name)

Revised Schema



MySQL script

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='TRADITIONAL,ALLOW\_INVALID\_DATES';

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

USE `bookstore` ;

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

-- Table `bookstore`.`customer`

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

CREATE TABLE IF NOT EXISTS `bookstore`.`customer` (

`idcustomer` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`cfirstname` VARCHAR(373) NOT NULL COMMENT 'The longest first name is 373 characters long.',

`clastname` VARCHAR(765) NOT NULL COMMENT 'The longest last name extant is 764 characters long.',

`cphone` VARCHAR(15) NULL COMMENT '15 characters to account for 1 800 and international phone numbers.',

`cemail` VARCHAR(255) NULL COMMENT '255 is the longest an email can be. It is this long so the email isn\'t truncated.',

PRIMARY KEY (`idcustomer`))

ENGINE = InnoDB;

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

-- Table `bookstore`.`employee`

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

CREATE TABLE IF NOT EXISTS `bookstore`.`employee` (

`employeeid` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`fname` VARCHAR(373) NOT NULL,

`lname` VARCHAR(765) NOT NULL,

`salary` DOUBLE NOT NULL COMMENT 'Money should be really handled by decimal but a 1 cent difference in salary is inconsequential.',

PRIMARY KEY (`employeeid`))

ENGINE = InnoDB;

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

-- Table `bookstore`.`location`

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

CREATE TABLE IF NOT EXISTS `bookstore`.`location` (

`addressid` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`city` VARCHAR(60) NOT NULL COMMENT 'The longest city name in english is 58 letters.',

`state` VARCHAR(2) NOT NULL COMMENT 'The two letter abbreviation of the state.',

`street` VARCHAR(100) NOT NULL COMMENT 'The longest street name in english is 85 chacacters long. Addign 15 characters for street, avenue, boulevard etc.',

PRIMARY KEY (`addressid`))

ENGINE = InnoDB;

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

-- Table `bookstore`.`status`

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

CREATE TABLE IF NOT EXISTS `bookstore`.`status` (

`idstatus` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`status\_of\_order` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idstatus`),

UNIQUE INDEX `status\_of\_order\_UNIQUE` (`status\_of\_order` ASC))

ENGINE = InnoDB;

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

-- Table `bookstore`.`payment`

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

CREATE TABLE IF NOT EXISTS `bookstore`.`payment` (

`idpayment` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`Creditcardnumber` INT UNSIGNED NOT NULL,

`expiredate` DATE NOT NULL,

`nameoncard` VARCHAR(2000) NOT NULL,

`verifcatitiondigit` INT UNSIGNED NOT NULL,

PRIMARY KEY (`idpayment`))

ENGINE = InnoDB;

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

-- Table `bookstore`.`order`

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

CREATE TABLE IF NOT EXISTS `bookstore`.`order` (

`idorder` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`employee\_employeeid` INT UNSIGNED NOT NULL DEFAULT 1 COMMENT 'If the employee is deleted it will be set to default at int 1. This would be a row that holds orders until a program can divvy accumulated employeeless orders.',

`customer\_idcustomer` INT UNSIGNED NOT NULL,

`employee\_employeeid\_manager` INT UNSIGNED NULL COMMENT 'The relational database can not enforce the business rule that a order is given a manager. ',

`location\_billing\_id` INT UNSIGNED NOT NULL DEFAULT 1,

`location\_shipping\_id` INT UNSIGNED NOT NULL DEFAULT 1,

`status\_idstatus` INT UNSIGNED NOT NULL DEFAULT 1,

`payment\_idpayment` INT UNSIGNED NOT NULL,

PRIMARY KEY (`idorder`),

INDEX `fk\_order\_employee1\_idx` (`employee\_employeeid` ASC),

INDEX `fk\_order\_customer1\_idx` (`customer\_idcustomer` ASC),

INDEX `fk\_order\_employee2\_idx` (`employee\_employeeid\_manager` ASC),

INDEX `fk\_order\_location1\_idx` (`location\_billing\_id` ASC),

INDEX `fk\_order\_location2\_idx` (`location\_shipping\_id` ASC),

INDEX `fk\_order\_status1\_idx` (`status\_idstatus` ASC),

UNIQUE INDEX `employee\_employeeid\_UNIQUE` (`employee\_employeeid` ASC),

INDEX `fk\_order\_payment1\_idx` (`payment\_idpayment` ASC),

CONSTRAINT `fk\_order\_employee1`

FOREIGN KEY (`employee\_employeeid`)

REFERENCES `bookstore`.`employee` (`employeeid`)

ON DELETE restrict

ON UPDATE cascade, #Inno db does not allow default on delete. Thus we will restict delete and have a program delete the parent after the program updates all children

CONSTRAINT `fk\_order\_customer1`

FOREIGN KEY (`customer\_idcustomer`)

REFERENCES `bookstore`.`customer` (`idcustomer`)

ON DELETE cascade

ON UPDATE cascade,## Deleting the customer will also delete the order. Changeing the customer will change the customer stored in order

CONSTRAINT `fk\_order\_employee2`

FOREIGN KEY (`employee\_employeeid\_manager`)

REFERENCES `bookstore`.`employee` (`employeeid`)

ON DELETE set null

ON UPDATE cascade,## As we have to manually add managers to orders anyway when we delete we set it to null. When we update employee we also update employee in this table

CONSTRAINT `fk\_order\_location1`

FOREIGN KEY (`location\_billing\_id`)

REFERENCES `bookstore`.`location` (`addressid`)

ON DELETE restrict

ON UPDATE cascade,#We can not delete the location as we are billing it currently. We can update it though

CONSTRAINT `fk\_order\_location2`

FOREIGN KEY (`location\_shipping\_id`)

REFERENCES `bookstore`.`location` (`addressid`)

ON DELETE restrict

ON UPDATE cascade,#Restrict deleting this location as we are shipping it there

CONSTRAINT `fk\_order\_status1`

FOREIGN KEY (`status\_idstatus`)

REFERENCES `bookstore`.`status` (`idstatus`)

ON DELETE restrict

ON UPDATE cascade,#We do not allow deletion as there is important information we do not want to lose.

CONSTRAINT `fk\_order\_payment1`

FOREIGN KEY (`payment\_idpayment`)

REFERENCES `bookstore`.`payment` (`idpayment`)

ON DELETE restrict

ON UPDATE cascade)##We do not want to lose payment iformation. We wait until we have nw information before deleting it.

ENGINE = InnoDB;

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

-- Table `bookstore`.`book`

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

CREATE TABLE IF NOT EXISTS `bookstore`.`book` (

`isbn` INT UNSIGNED NOT NULL COMMENT 'Unsigned because there exists no isbn that is negative. Number constraint can not be give because isbn varys in length. Isbn is primary key because it is unique to every book.',

`title` VARCHAR(4805) NOT NULL COMMENT 'The reason for 4805 is because that is the longest book title extant.',

`edition` VARCHAR(100) NULL,

`quantity` INT NOT NULL,

`price` DECIMAL(8,2) NOT NULL COMMENT 'This is because double can\'t accuratly record monetary numbers. Ideally in a real bookstore price would also be held in the table linking book and order.',

`year` YEAR NOT NULL,

PRIMARY KEY (`isbn`))

ENGINE = InnoDB;

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

-- Table `bookstore`.`author`

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

CREATE TABLE IF NOT EXISTS `bookstore`.`author` (

`authorid` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`fname` VARCHAR(373) NOT NULL,

`lname` VARCHAR(765) NOT NULL,

PRIMARY KEY (`authorid`))

ENGINE = InnoDB;

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

-- Table `bookstore`.`cart`

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

CREATE TABLE IF NOT EXISTS `bookstore`.`cart` (

`cartID` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`datecreated` DATE NOT NULL,

`datelastupdated` DATE NULL,

`customer\_idcustomer` INT UNSIGNED NOT NULL,

`location\_billing\_id` INT UNSIGNED NOT NULL DEFAULT 1,

`location\_shipping\_id` INT UNSIGNED NOT NULL DEFAULT 1,

PRIMARY KEY (`cartID`),

INDEX `fk\_cart\_customer1\_idx` (`customer\_idcustomer` ASC),

INDEX `fk\_cart\_location1\_idx` (`location\_billing\_id` ASC),

INDEX `fk\_cart\_location2\_idx` (`location\_shipping\_id` ASC),

CONSTRAINT `fk\_cart\_customer1`

FOREIGN KEY (`customer\_idcustomer`)

REFERENCES `bookstore`.`customer` (`idcustomer`)

ON DELETE cascade

ON UPDATE cascade,## Deleting the customer will also delete the order. Changeing the customer will change the customer stored in cart

CONSTRAINT `fk\_cart\_location1`

FOREIGN KEY (`location\_billing\_id`)

REFERENCES `bookstore`.`location` (`addressid`)

ON DELETE restrict

ON UPDATE cascade,

CONSTRAINT `fk\_cart\_location2`

FOREIGN KEY (`location\_shipping\_id`)

REFERENCES `bookstore`.`location` (`addressid`)

ON DELETE restrict

ON UPDATE cascade)

ENGINE = InnoDB;

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

-- Table `bookstore`.`book\_has\_author`

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

CREATE TABLE IF NOT EXISTS `bookstore`.`book\_has\_author` (

`book\_isbn` INT UNSIGNED NOT NULL,

`author\_authorid` INT UNSIGNED NOT NULL,

PRIMARY KEY (`book\_isbn`, `author\_authorid`),

INDEX `fk\_book\_has\_author\_book1\_idx` (`book\_isbn` ASC),

INDEX `fk\_book\_has\_author\_author1\_idx` (`author\_authorid` ASC),

CONSTRAINT `fk\_book\_has\_author\_book1`

FOREIGN KEY (`book\_isbn`)

REFERENCES `bookstore`.`book` (`isbn`)

ON DELETE restrict

ON UPDATE cascade,

CONSTRAINT `fk\_book\_has\_author\_author1`

FOREIGN KEY (`author\_authorid`)

REFERENCES `bookstore`.`author` (`authorid`)

ON DELETE restrict

ON UPDATE cascade)

ENGINE = InnoDB;

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

-- Table `bookstore`.`book\_has\_order`

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

CREATE TABLE IF NOT EXISTS `bookstore`.`book\_has\_order` (

`book\_isbn` INT UNSIGNED NOT NULL,

`order\_idorder` INT UNSIGNED NOT NULL,

PRIMARY KEY (`book\_isbn`, `order\_idorder`),

INDEX `fk\_book\_has\_order\_order1\_idx` (`order\_idorder` ASC),

INDEX `fk\_book\_has\_order\_book1\_idx` (`book\_isbn` ASC),

CONSTRAINT `fk\_book\_has\_order\_book1`

FOREIGN KEY (`book\_isbn`)

REFERENCES `bookstore`.`book` (`isbn`)

ON DELETE restrict

ON UPDATE cascade,

CONSTRAINT `fk\_book\_has\_order\_order1`

FOREIGN KEY (`order\_idorder`)

REFERENCES `bookstore`.`order` (`idorder`)

ON DELETE cascade

ON UPDATE cascade)

ENGINE = InnoDB;

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

-- Table `bookstore`.`book\_has\_cart1`

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

CREATE TABLE IF NOT EXISTS `bookstore`.`book\_has\_cart1` (

`book\_isbn` INT UNSIGNED NOT NULL,

`cart\_cartID` INT UNSIGNED NOT NULL,

PRIMARY KEY (`book\_isbn`, `cart\_cartID`),

INDEX `fk\_book\_has\_cart1\_cart1\_idx` (`cart\_cartID` ASC),

INDEX `fk\_book\_has\_cart1\_book1\_idx` (`book\_isbn` ASC),

CONSTRAINT `fk\_book\_has\_cart1\_book1`

FOREIGN KEY (`book\_isbn`)

REFERENCES `bookstore`.`book` (`isbn`)

ON DELETE cascade

ON UPDATE cascade,##cart is not important enough to restrict deleting the book

CONSTRAINT `fk\_book\_has\_cart1\_cart1`

FOREIGN KEY (`cart\_cartID`)

REFERENCES `bookstore`.`cart` (`cartID`)

ON DELETE cascade

ON UPDATE cascade)

ENGINE = InnoDB;

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

-- Table `bookstore`.`category`

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

CREATE TABLE IF NOT EXISTS `bookstore`.`category` (

`idcategory` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`category` VARCHAR(100) NOT NULL,

PRIMARY KEY (`idcategory`))

ENGINE = InnoDB;

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

-- Table `bookstore`.`publisher`

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

CREATE TABLE IF NOT EXISTS `bookstore`.`publisher` (

`idpublisher` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

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

PRIMARY KEY (`idpublisher`))

ENGINE = InnoDB;

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

-- Table `bookstore`.`category\_has\_book`

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

CREATE TABLE IF NOT EXISTS `bookstore`.`category\_has\_book` (

`category\_idcategory` INT UNSIGNED NOT NULL,

`book\_isbn` INT UNSIGNED NOT NULL,

PRIMARY KEY (`category\_idcategory`, `book\_isbn`),

INDEX `fk\_category\_has\_book\_book1\_idx` (`book\_isbn` ASC),

INDEX `fk\_category\_has\_book\_category1\_idx` (`category\_idcategory` ASC),

CONSTRAINT `fk\_category\_has\_book\_category1`

FOREIGN KEY (`category\_idcategory`)

REFERENCES `bookstore`.`category` (`idcategory`)

ON DELETE restrict

ON UPDATE cascade,##Book is important enough to restrict deleting categories

CONSTRAINT `fk\_category\_has\_book\_book1`

FOREIGN KEY (`book\_isbn`)

REFERENCES `bookstore`.`book` (`isbn`)

ON DELETE cascade

ON UPDATE cascade)

ENGINE = InnoDB;

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

-- Table `bookstore`.`publisher\_has\_book`

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

CREATE TABLE IF NOT EXISTS `bookstore`.`publisher\_has\_book` (

`publisher\_idpublisher` INT UNSIGNED NOT NULL,

`book\_isbn` INT UNSIGNED NOT NULL,

PRIMARY KEY (`publisher\_idpublisher`, `book\_isbn`),

INDEX `fk\_publisher\_has\_book\_book1\_idx` (`book\_isbn` ASC),

INDEX `fk\_publisher\_has\_book\_publisher1\_idx` (`publisher\_idpublisher` ASC),

CONSTRAINT `fk\_publisher\_has\_book\_publisher1`

FOREIGN KEY (`publisher\_idpublisher`)

REFERENCES `bookstore`.`publisher` (`idpublisher`)

ON DELETE restrict

ON UPDATE cascade,##Book is important enough to restrict the deletion of it's publisher

CONSTRAINT `fk\_publisher\_has\_book\_book1`

FOREIGN KEY (`book\_isbn`)

REFERENCES `bookstore`.`book` (`isbn`)

ON DELETE cascade

ON UPDATE cascade)

ENGINE = InnoDB;

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

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

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

select \* from publisher;

INSERT INTO publisher (idpublisher, publisher\_name) VALUES (NULL, 'AAA'),(NULL, 'BBB'),(NULL, 'CCC'),(NULL, 'DDD');

select \* from publisher;

select \* from employee;

INSERT INTO employee (employeeid, fname, lname, salary) VALUES (null,'John','Jameson',50000),(null,'Alex','Johan',45000),(null,'Max','Von Hapsburg',50000),(null,'Jacob','Jingleheimer',500000),(null,'Jon','Wilson',55000);

select \* from publisher;

select \* from category;

INSERT INTO category (idcategory, category) VALUES (NULL, 'Fantasy'),(NULL, 'Nonfiction'),(NULL, 'Romance'),(NULL, 'Adventure');

select \* from book;

INSERT INTO `bookstore`.`book` (`isbn`, `title`, `edition`, `quantity`, `price`, `year`) VALUES ('12422', 'A tale of three and a half cities', 'first edition', '45', '400', 1996);

select \* from book;

INSERT INTO `bookstore`.`book` (`isbn`, `title`, `edition`, `quantity`, `price`, `year`) VALUES ('12434', 'Cat in the hat', 'first edition', '121', '40.50', 1996);

INSERT INTO `bookstore`.`book` (`isbn`, `title`, `edition`, `quantity`, `price`, `year`) VALUES ('58879', 'Romeo, Juliet, and Chuck Testa', 'Third Edition, Revised', '2000', '2000.14', 2016);

INSERT INTO `bookstore`.`book` (`isbn`, `title`, `edition`, `quantity`, `price`, `year`) VALUES ('49960', 'Famous Chicken and more chicken', 'Chicken Edition', '2', '1.99', 2015);

INSERT INTO `bookstore`.`book` (`isbn`, `title`, `edition`, `quantity`, `price`, `year`) VALUES ('30059', 'War and Peace', 'Pacifist Edition', '0', '1.21', 1914);

INSERT INTO `bookstore`.`book` (`isbn`, `title`, `edition`, `quantity`, `price`, `year`) VALUES ('06954', 'Peasant of the Necklace', 'Blatant Rip off Edition', '2', '19.99', 2004);

select \* from customer;

INSERT INTO `bookstore`.`customer` (`idcustomer`, `cfirstname`, `clastname`, `cphone`, `cemail`) VALUES (null, 'Charles ', 'Carmal', '7043434343', 'realJackBlack@aol.com');

INSERT INTO `bookstore`.`customer` (`idcustomer`, `cfirstname`, `clastname`, `cphone`, `cemail`) VALUES (null, 'Jimmy', 'Carter', '35422443', 'Jimmy@us.gov');

INSERT INTO `bookstore`.`customer` (`idcustomer`, `cfirstname`, `clastname`, `cphone`, `cemail`) VALUES (null, 'Ronald', 'Reagan', '30598585', 'RR@heaven.hv');

INSERT INTO `bookstore`.`customer` (`idcustomer`, `cfirstname`, `clastname`, `cphone`, `cemail`) VALUES (null, 'Gerald', 'Ford', '384868', 'GF@heaven.hv');

select \* from author;

INSERT INTO `bookstore`.`author` (`fname`, `lname`) VALUES ('Jacob', 'Lawrence');

INSERT INTO `bookstore`.`author` (`fname`, `lname`) VALUES ('Barry', 'Goldwater');

select \* from payment;

INSERT INTO `bookstore`.`payment` (`Creditcardnumber`, `expiredate`, `nameoncard`, `verifcatitiondigit`) VALUES ('234567', '2017-02-01', 'Jim Jacob', '453');

INSERT INTO `bookstore`.`payment` (`Creditcardnumber`, `expiredate`, `nameoncard`, `verifcatitiondigit`) VALUES ('345434', '2018-05-01', 'Alexander Bell', '123');

INSERT INTO `bookstore`.`payment` (`Creditcardnumber`, `expiredate`, `nameoncard`, `verifcatitiondigit`) VALUES ('315155', '4001-11-01', 'Zeus', '445');

select \* from location;

INSERT INTO `bookstore`.`location` (`city`, `state`, `street`) VALUES ('Charlotte', 'NC', 'Jacob St');

INSERT INTO `bookstore`.`location` (`city`, `state`, `street`) VALUES ('Rockhill', 'SC', 'Vierra St');

INSERT INTO `bookstore`.`location` (`city`, `state`, `street`) VALUES ('Concord', 'NC', 'Lexington Rd');

INSERT INTO `bookstore`.`location` (`city`, `state`, `street`) VALUES ('Lexington', 'NC', 'Mary Alexander Avenue');

select \* from category;

select \* from book;

select \* from category\_has\_book;

INSERT INTO `bookstore`.`category\_has\_book` (`category\_idcategory`, `book\_isbn`) VALUES ('1', '6954');

INSERT INTO `bookstore`.`category\_has\_book` (`category\_idcategory`, `book\_isbn`) VALUES ('2', '12434');

INSERT INTO `bookstore`.`category\_has\_book` (`category\_idcategory`, `book\_isbn`) VALUES ('1', '30059');

INSERT INTO `bookstore`.`category\_has\_book` (`category\_idcategory`, `book\_isbn`) VALUES ('2', '49960');

INSERT INTO `bookstore`.`category\_has\_book` (`category\_idcategory`, `book\_isbn`) VALUES ('3', '58879');

INSERT INTO `bookstore`.`category\_has\_book` (`category\_idcategory`, `book\_isbn`) VALUES ('1', '58879');

select \* from book;

select \* from publisher;

select \* from publisher\_has\_book;

INSERT INTO `bookstore`.`publisher\_has\_book` (`publisher\_idpublisher`, `book\_isbn`) VALUES ('1', '6954');

INSERT INTO `bookstore`.`publisher\_has\_book` (`publisher\_idpublisher`, `book\_isbn`) VALUES ('3', '12434');

INSERT INTO `bookstore`.`publisher\_has\_book` (`publisher\_idpublisher`, `book\_isbn`) VALUES ('2', '30059');

INSERT INTO `bookstore`.`publisher\_has\_book` (`publisher\_idpublisher`, `book\_isbn`) VALUES ('3', '49960');

INSERT INTO `bookstore`.`publisher\_has\_book` (`publisher\_idpublisher`, `book\_isbn`) VALUES ('2', '58879');

select \* from book;

select \* from author;

select \* from book\_has\_author;

INSERT INTO `bookstore`.`book\_has\_author` (`book\_isbn`, `author\_authorid`) VALUES ('6954', '1');

INSERT INTO `bookstore`.`book\_has\_author` (`book\_isbn`, `author\_authorid`) VALUES ('12434', '2');

INSERT INTO `bookstore`.`book\_has\_author` (`book\_isbn`, `author\_authorid`) VALUES ('30059', '2');

INSERT INTO `bookstore`.`book\_has\_author` (`book\_isbn`, `author\_authorid`) VALUES ('49960', '1');

INSERT INTO `bookstore`.`book\_has\_author` (`book\_isbn`, `author\_authorid`) VALUES ('58879', '2');

select \* from customer;

select \* from location;

select \* from cart;

INSERT INTO `bookstore`.`cart` (`datecreated`, `datelastupdated`, `customer\_idcustomer`, `location\_billing\_id`, `location\_shipping\_id`) VALUES ('2016-06-12', NULL, '6', '3', '3');

INSERT INTO `bookstore`.`cart` (`datecreated`, `customer\_idcustomer`, `location\_billing\_id`, `location\_shipping\_id`) VALUES ('1998-08-05', '4', '1', '1');

INSERT INTO `bookstore`.`cart` (`datecreated`, `customer\_idcustomer`, `location\_billing\_id`, `location\_shipping\_id`) VALUES ('1997-08-04', '5', '2', '2');

INSERT INTO `bookstore`.`cart` (`datecreated`, `customer\_idcustomer`, `location\_billing\_id`, `location\_shipping\_id`) VALUES ('2003-04-19', '7', '4', '4');

##We can update carts

UPDATE `bookstore`.`cart` SET `datelastupdated`='2016-11-11' WHERE `cartID`='3';

select \* from status;

INSERT INTO `bookstore`.`status` (`status\_of\_order`) VALUES ('Delivered');

INSERT INTO `bookstore`.`status` (`status\_of\_order`) VALUES ('Back Order');

INSERT INTO `bookstore`.`status` (`status\_of\_order`) VALUES ('Out of Stock');

INSERT INTO `bookstore`.`status` (`status\_of\_order`) VALUES ('Cancelled');

INSERT INTO `bookstore`.`status` (`status\_of\_order`) VALUES ('Alien Invasion -- ON HOLD');

select \* from `order`;

select \* from employee;

select \* from cart;

select \* from payment;

INSERT INTO `bookstore`.`order` (`employee\_employeeid`, `customer\_idcustomer`, `employee\_employeeid\_manager`, `location\_billing\_id`, `location\_shipping\_id`, `status\_idstatus`, `payment\_idpayment`) VALUES ('3', '7', '1', '3', '3', '3', '3');

INSERT INTO `bookstore`.`order` (`employee\_employeeid`, `customer\_idcustomer`, `location\_billing\_id`, `location\_shipping\_id`, `status\_idstatus`, `payment\_idpayment`) VALUES ('5', '5', '1', '1', '2', '2');

INSERT INTO `bookstore`.`order` (`employee\_employeeid`, `customer\_idcustomer`, `location\_billing\_id`, `location\_shipping\_id`, `status\_idstatus`, `payment\_idpayment`) VALUES ('2', '4', '2', '2', '4', '3');

INSERT INTO `bookstore`.`order` (`employee\_employeeid`, `customer\_idcustomer`, `location\_billing\_id`, `location\_shipping\_id`, `status\_idstatus`, `payment\_idpayment`) VALUES ('4', '6', '4', '4', '1', '1');

select \* from book\_has\_order;

INSERT INTO `bookstore`.`book\_has\_order` (`book\_isbn`, `order\_idorder`) VALUES ('6954', '3');

INSERT INTO `bookstore`.`book\_has\_order` (`book\_isbn`, `order\_idorder`) VALUES ('12434', '4');

INSERT INTO `bookstore`.`book\_has\_order` (`book\_isbn`, `order\_idorder`) VALUES ('30059', '1');

INSERT INTO `bookstore`.`book\_has\_order` (`book\_isbn`, `order\_idorder`) VALUES ('30059', '2');

#Able to show prices.

select \*

from book

where price>100;

#Shows quantity

select title, quantity

from book

where price > 20;

##List book information (e.g., title, author, price)

select \*

from book

left outer join book\_has\_author

on book.isbn = book\_has\_author.book\_isbn

left outer join author

on book\_has\_author.author\_authorid = author.authorid

where fname= 'jacob';

##List information about those orders assigned to him/her (Employee) in this case employee id number 3

select \*

from `order`

left outer join location

on location.addressid = `order`.location\_billing\_id and location.addressid = `order`.location\_shipping\_id

left outer join payment

on payment.idpayment = `order`.payment\_idpayment

left outer join `status`

on `status`.idstatus = `order`.status\_idstatus

where employee\_employeeid = 3;

## Shows that we can update order status

select \* from `order`;

update `order`

set status\_idstatus = 1

where idorder = 1;

select \* from `order`;

##Adding books to a shopping cart is as simple as inserting a relation into cart\_has\_book

##Creating new empty cart

Select \* from cart;

INSERT INTO `bookstore`.`cart` (`datecreated`, `customer\_idcustomer`, `location\_billing\_id`, `location\_shipping\_id`) VALUES ('2016-12-12', '1', '1', '1');

Select \* from cart;

## Merge shopping cart

## I would merge shopping carts in java. I would select all information from the row that would be absorbed. Store that in the java program. Delete the absorbed row. Insert any books that need to be inserted in the absorbing row.

## I would use java to create new orders from cart. I would select the cart id and insert a new order with the values from that cart and the cart has orders that it has.

##I would use a program like java to handle new user creation, login and updates.

## I would use a program to insert new books it would look the same as the inserts into book above.alter

##I would have a program monitor the price of orders so that managers would be assigned to orders there is no way that I know of to do that in a relational database.

---------------------------------------------------------------------------------------------------------------------------------------------------------------END OF SCRIPT-------------------------------------------------------------------

Conclusion and Future work

If this were a real life project I would have added many more features to reduce the amount of data corruption. For example if you make an order with a book in it and change the price of the book the price of the order changes. This should not happen. To get around this we should add a field in the order\_has\_book table that records the price of the book so we can change the price of books without fear of corrupting the price of orders placed. We also should have added a quantity to both cart has book and order has book. If we had expanded the role of manager we would have needed to shift the two relations between employee and order to three relations 1 between order and employee, 1 between order and manager and one between manager and employee. I think we did a good job looking up the maximum amount of character a certain field can be and don’t need much improvement there. We made price in book a decimal but left salary of the employee a double. If the company was sufficiently large a good amount of money could be lost, although the loss of a fraction of a cent may be trivial it should be fixed because of how easy this fix is.