Planning doc + sql code

**Database Design**

**Describe the enterprise:**

* Inventory Control management DATABASE PROJECT

**Introduction and description**

* A database design student is tasked with creating a database for an inventory control management database. I chose inventory control management as my final project because I use to volunteer at a food pantry that was fairly new, at the time and there was always a problem with managing the inventory of stock coming in and out from different suppliers and maintaining a consistent inventory or having a reliable volunteers and workers. That and I had a friend who wanted to start her own business.
* So, my goal is to emulate an inventory control management database to better understand how to maintain a proper variety of required items, increase inventory turnover, reduce, and maintain optimize inventory and safety stock levels, obtain low raw material prices, reduce storage cost, reduce insurance cost, and reduce taxes.

**What functions should the system perform? For example, inventory control, billing, ordering, etc.**

* The system should be able to:
  + Searching for employees, customers, inventory stock, order details, and expenses
  + Inserting new employees, customer, inventory stock, order details
  + Deleting old employees, customer, inventory stock, order details
* Customers
  + Can order many items from the inventory stock
  + Has a total to their purchase
* Employees
  + Has the department they work
  + They can have many tasks in the same department (floor runner+stocker)
* Inventory
  + Can have many categories of type of inventory (health, food, etc)
  + Amount of available stock
  + Amount of stock that needs to be shipped out
* Order details
  + Has the stock from the inventory the customer requests
  + Has the price of each stock item from inventory
  + Taxes of their state and item
  + Has only one customer per order

Where did you get the idea for this project? Did you make it up, get it from work, or find it in a book? Please site your sources.

* Database management final project from professor Weiping Zhang.

Entity Relationship Design:

Describe your entities. Be sure to define the meaning of each attribute. You must describe the “role” each attribute will play in your table (i.e. what is it and who will use it). You must have enough entities (at least 10 entities).

* Customer 🡪 to identify and keep track of potential customers information
  + Lastname
  + Firstname
  + Address
  + Email
  + Phone number
  + Customer ID

CUSTOMER

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CustomerID | FirstName | LastName | Address | Email | PhoneNumber |
| 1 | John | Doe | 123 Freedom Drive | [Abc@gmail.com](mailto:Abc@gmail.com) | (347) 555-5555 |

* Employee🡪 to identify and keep track of employee information
  + First name
  + Last name
  + Email
  + employee ID(PK)
  + DepartmentID(FK)

EMPLOYEE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EmployeeID | FirstName | LastName | Email | DepartmentID |
| 2 | June | Due | [def@gmail.com](mailto:def@gmail.com) | 1 |

* Departments
  + DepartmentID(PK)
  + Department

DEPARTMENT

|  |  |
| --- | --- |
| DepartmentID | Department |
| 1  2  3  4  5  6 | Stocker  Deliver  Accounting  Technology  Management  CustomerService |

* bridge tables
  + Inventory turnover: orderedDetailsXinventory(CK)
  + manages: employeeXinventory(CK)

INVENTORY TURNOVER

|  |  |  |
| --- | --- | --- |
| OrderDetailID | InventoryID | ReplacementRatio |
| 12 | 23 | 12% |

MANAGES

|  |  |  |
| --- | --- | --- |
| EmployeeID | InventoryID | JobTask |
| 1 | 12 | Stocking |

* inventory
  + inventoryID(PK)
  + manufacturer
  + amount in stock
  + purchasePrice
  + SoldPrice
  + OrderDetailsID (FK)

INVENTORY

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| InventoryID | Manufacturer | Stock | PurchasedPrice | SoldPrice | OrderDetailsID |
| 1 | Rutgers | 5 | $5.00 | $14000.00 | 2 |

* Order details
  + OrderDetailID(CK)
  + InventoryID(CK)
  + Taxes (some could have tax some none)
  + insurance cost
  + customer(FK)

ORDER DETAILS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| OrderID | InventoryID | TotalPrice | Taxes | Insurance | CustomerID |
| 123 | 2 | $5000 | 15% | $10.00 | 2 |

Describe your relationship 1:1, 1:N, N:M relationship& minmax cardinality

|  |  |  |
| --- | --- | --- |
| Relationships | 1:1, 1:N, N:M | Min and Max Cardinality |
| EMPLOYEE to DEPARTMENT | 1:N | Employee: min1 max many  Department: min1 max 1 |
| DEPARTMENT to MANAGES | 1:N | Department: min 1 max 1  Manages: min 1 max many |
| MANAGES to INVENTORY | 1:N | Manages: min 1 max 1  Inventory: min 1 max many |
| CUSTOMER to ORDERDETAILS | 1:1 | Customer: min 1 max 1  Orderdetails: min 1 max 1 |
| ORDERDETAILS to INVETORYTURNOVER | 1:N | Orderdetails 1 min 1 max  InvetoryTurnOver min 1 max many |
| INVENTORYTURNOVER to INVENTORY | 1:1 | InventoryTurnOver min 1 max 1  Inventory min 1 max 1 |
| ORDERDETAILS to INVENTORY | 1:N | OrderDetail min 1 max 1  Inventory min 1 max many |

Weak Entities and Strong Entities

|  |  |
| --- | --- |
| Weak Entities | Strong Entities |
| Manages  OrderDetails | Employee  Departments  InventoryTurnOver  Customer  Inventory |

ID-Dependent or Non-ID-Dependent

|  |  |
| --- | --- |
| ID-Dependent | Non-ID-Dependent |
| Manages  OrderDetails | Employee  Departments  InventoryTurnOver  Customer  Inventory |

Draw the E-R Diagram by using either MySQL tool or Online tool (e.g., draw.io)

1. Database Implementation (4\* 6 = 24%)

Implement your tables in MySQL. Your set of queries must include queries that represent.

1. Insert, update, delete, truncate
2. Multi table queries – Joins (left outer join and right outer join)
3. Built-in Aggregate Functions: sum(), max(), min()….
4. Where, Group By, Having, Order by, Between,

You must implement your queries using MySQL

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_implemented SQL

-- 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='TRADITIONAL,ALLOW\_INVALID\_DATES';

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

-- Schema InventoryManagement

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

-- For database final project!

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

-- Schema InventoryManagement

--

-- For database final project!

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

CREATE SCHEMA IF NOT EXISTS `InventoryManagement` DEFAULT CHARACTER SET utf8 COLLATE utf8\_bin ;

USE `InventoryManagement` ;

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

-- Table `InventoryManagement`.`Customer`

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

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

`CustomerID` INT NOT NULL AUTO\_INCREMENT,

`FirstName` VARCHAR(45) NOT NULL,

`LastName` VARCHAR(45) NOT NULL,

`Address` VARCHAR(45) NOT NULL,

`Email` VARCHAR(45) NOT NULL,

`PhoneNumber` INT NOT NULL,

PRIMARY KEY (`CustomerID`),

UNIQUE INDEX `CustomerID\_UNIQUE` (`CustomerID` ASC),

UNIQUE INDEX `Customercol\_UNIQUE` (`PhoneNumber` ASC),

UNIQUE INDEX `Email\_UNIQUE` (`Email` ASC))

ENGINE = InnoDB;

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

-- Table `InventoryManagement`.`Departments`

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

CREATE TABLE IF NOT EXISTS `InventoryManagement`.`Departments` (

`idDepartments` INT NOT NULL AUTO\_INCREMENT,

`Department` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idDepartments`),

UNIQUE INDEX `idDepartments\_UNIQUE` (`idDepartments` ASC))

ENGINE = InnoDB;

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

-- Table `InventoryManagement`.`Employee`

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

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

`idEmployee` INT NOT NULL AUTO\_INCREMENT,

`FirstName` VARCHAR(45) NOT NULL,

`LastName` VARCHAR(45) NOT NULL,

`Departments\_idDepartments` INT NOT NULL,

`email` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idEmployee`),

UNIQUE INDEX `idEmployee\_UNIQUE` (`idEmployee` ASC),

INDEX `fk\_Employee\_Departments1\_idx` (`Departments\_idDepartments` ASC),

CONSTRAINT `fk\_Employee\_Departments1`

FOREIGN KEY (`Departments\_idDepartments`)

REFERENCES `InventoryManagement`.`Departments` (`idDepartments`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `InventoryManagement`.`OrderDetails`

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

CREATE TABLE IF NOT EXISTS `InventoryManagement`.`OrderDetails` (

`OrderDetails\_itemID` INT NOT NULL,

`Customer\_CustomerID` INT NOT NULL,

`TotalPrice` FLOAT NOT NULL,

`taxes` FLOAT NOT NULL,

`insuranceCost` FLOAT NOT NULL,

PRIMARY KEY (`OrderDetails\_itemID`),

INDEX `fk\_OrderDetails\_Customer1\_idx` (`Customer\_CustomerID` ASC),

CONSTRAINT `fk\_OrderDetails\_Customer1`

FOREIGN KEY (`Customer\_CustomerID`)

REFERENCES `InventoryManagement`.`Customer` (`CustomerID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `InventoryManagement`.`Inventory`

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

CREATE TABLE IF NOT EXISTS `InventoryManagement`.`Inventory` (

`Inventory\_itemID` INT NOT NULL AUTO\_INCREMENT,

`OrderDetails\_ItemID` INT NOT NULL,

`Manufacture` VARCHAR(45) NOT NULL,

`stock` INT NOT NULL,

`purchasedPrice` FLOAT NOT NULL,

`soldPrice` FLOAT NOT NULL,

PRIMARY KEY (`Inventory\_itemID`),

INDEX `fk\_Inventory\_OrderDetails1\_idx` (`OrderDetails\_ItemID` ASC),

CONSTRAINT `fk\_Inventory\_OrderDetails1`

FOREIGN KEY (`OrderDetails\_ItemID`)

REFERENCES `InventoryManagement`.`OrderDetails` (`OrderDetails\_itemID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `InventoryManagement`.`Manages`

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

CREATE TABLE IF NOT EXISTS `InventoryManagement`.`Manages` (

`Inventory\_itemID` INT NOT NULL,

`Departments\_idDepartments` INT NOT NULL,

`JobTask` VARCHAR(45) NOT NULL,

INDEX `fk\_Works\_Inventory1\_idx` (`Inventory\_itemID` ASC),

INDEX `fk\_Works\_Departments1\_idx` (`Departments\_idDepartments` ASC),

PRIMARY KEY (`Inventory\_itemID`, `Departments\_idDepartments`),

CONSTRAINT `fk\_Works\_Inventory1`

FOREIGN KEY (`Inventory\_itemID`)

REFERENCES `InventoryManagement`.`Inventory` (`Inventory\_itemID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Works\_Departments1`

FOREIGN KEY (`Departments\_idDepartments`)

REFERENCES `InventoryManagement`.`Departments` (`idDepartments`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `InventoryManagement`.`inventoryTurnOver`

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

CREATE TABLE IF NOT EXISTS `InventoryManagement`.`inventoryTurnOver` (

`OrderDetails\_OrderDetails\_itemID` INT NOT NULL,

`ReplacementRatio` FLOAT NOT NULL,

`Inventory\_Inventory\_itemID` INT NOT NULL,

INDEX `fk\_inventoryTurnOver\_OrderDetails1\_idx` (`OrderDetails\_OrderDetails\_itemID` ASC),

INDEX `fk\_inventoryTurnOver\_Inventory1\_idx` (`Inventory\_Inventory\_itemID` ASC),

PRIMARY KEY (`OrderDetails\_OrderDetails\_itemID`, `Inventory\_Inventory\_itemID`),

CONSTRAINT `fk\_inventoryTurnOver\_OrderDetails1`

FOREIGN KEY (`OrderDetails\_OrderDetails\_itemID`)

REFERENCES `InventoryManagement`.`OrderDetails` (`OrderDetails\_itemID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_inventoryTurnOver\_Inventory1`

FOREIGN KEY (`Inventory\_Inventory\_itemID`)

REFERENCES `InventoryManagement`.`Inventory` (`Inventory\_itemID`)

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;

What to submit?

1. E-R Diagram
2. Entire MySQL database (\*.sql)