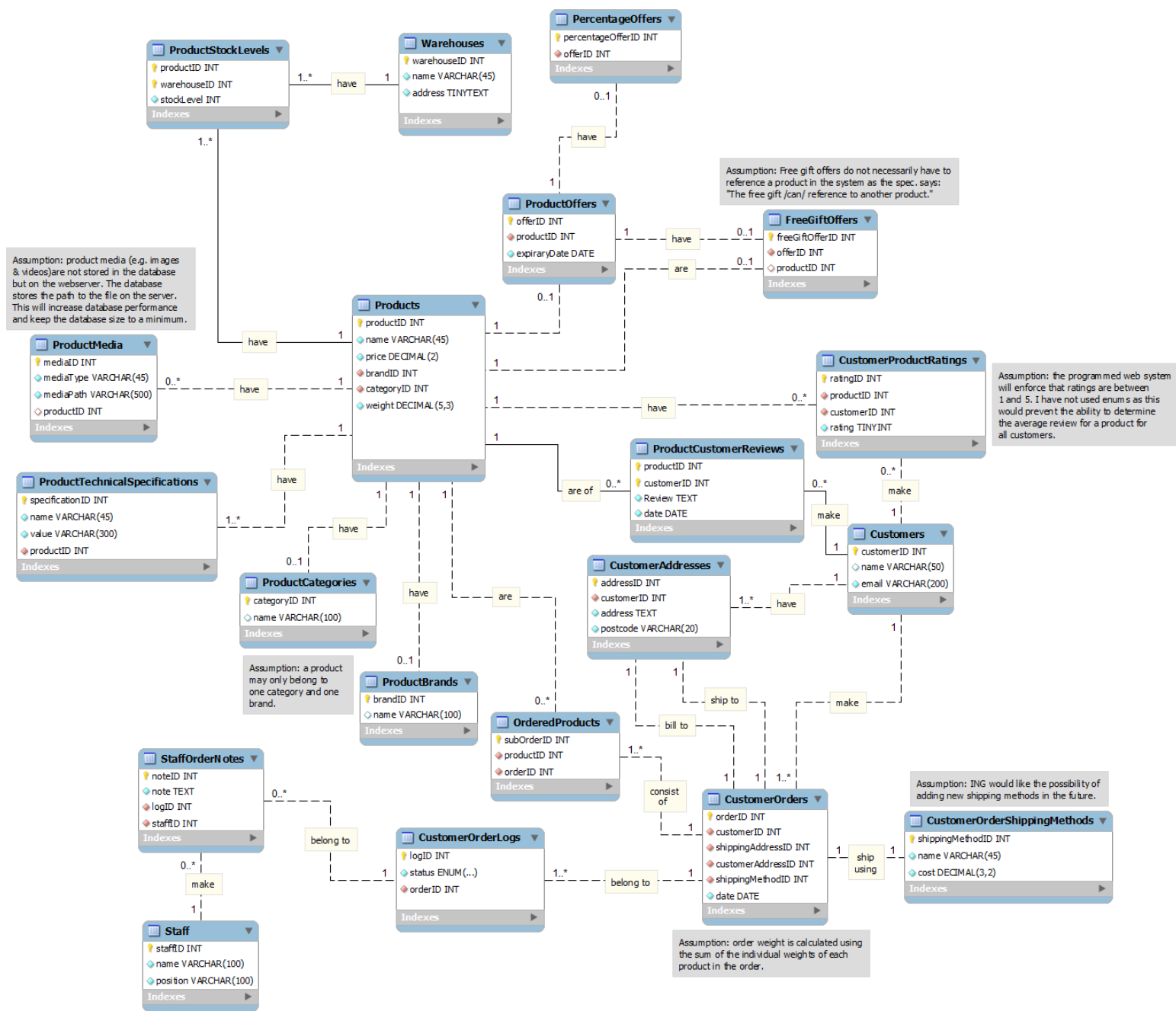


# Danny King: Advanced Databases: ING Online Web Store

## Question 1

Based on the problem scenario, draw an EER diagram to construct a logical data model of the system using UML notation.

**Please note: If this diagram is too small, you can see alternatively the attached 'question1.pdf'.**



15/01/2010

## Question 2

Based on the logical model you have constructed, write the SQL DDL to implement the physical database design.

```
CREATE SCHEMA `ING`;
```

```
CREATE TABLE IF NOT EXISTS `ING`.`ProductBrands` (  
  `brandID` INT NOT NULL AUTO_INCREMENT ,  
  `name` VARCHAR(100) NULL ,  
  PRIMARY KEY (`brandID`) ,  
  UNIQUE INDEX `brandID_UNIQUE` (`brandID` ASC) )  
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `ING`.`ProductCategories` (  
  `categoryID` INT NOT NULL AUTO_INCREMENT ,  
  `name` VARCHAR(100) NULL ,  
  PRIMARY KEY (`categoryID`) ,  
  UNIQUE INDEX `categoryID_UNIQUE` (`categoryID` ASC) )  
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `ING`.`Products` (  
  `productID` INT NOT NULL AUTO_INCREMENT ,  
  `name` VARCHAR(45) NOT NULL ,  
  `price` DECIMAL(2) NOT NULL ,  
  `brandID` INT NOT NULL ,  
  `categoryID` INT NOT NULL ,  
  `weight` DECIMAL(5,3) NOT NULL ,  
  PRIMARY KEY (`productID`) ,  
  INDEX `fk_Products_ProductBrands` (`brandID` ASC) ,  
  INDEX `fk_Products_ProductCategories2` (`categoryID` ASC) ,  
  UNIQUE INDEX `productID_UNIQUE` (`productID` ASC) ,  
  CONSTRAINT `fk_Products_ProductBrands`  
    FOREIGN KEY (`brandID` )  
    REFERENCES `ING`.`ProductBrands` (`brandID` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `fk_Products_ProductCategories2`  
    FOREIGN KEY (`categoryID` )  
    REFERENCES `ING`.`ProductCategories` (`categoryID` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `ING`.`Warehouses` (  
  `warehouseID` INT NOT NULL AUTO_INCREMENT ,  
  `name` VARCHAR(45) NOT NULL ,  
  `address` TINYTEXT NOT NULL ,  
  PRIMARY KEY (`warehouseID`) ,  
  UNIQUE INDEX `warehouseID_UNIQUE` (`warehouseID` ASC) )  
ENGINE = InnoDB;
```

15/01/2010

```
CREATE TABLE IF NOT EXISTS `ING`.`ProductStockLevels` (  
  `productID` INT NOT NULL ,  
  `warehouseID` INT NOT NULL ,  
  `stockLevel` INT NOT NULL ,  
  PRIMARY KEY (`productID`, `warehouseID`) ,  
  INDEX `fk_ProductStockLevels_Products` (`productID` ASC) ,  
  INDEX `fk_ProductStockLevels_Warehouses` (`warehouseID` ASC) ,  
  UNIQUE INDEX `productID_UNIQUE` (`productID` ASC) ,  
  UNIQUE INDEX `warehouseID_UNIQUE` (`warehouseID` ASC) ,  
  CONSTRAINT `fk_ProductStockLevels_Products`  
    FOREIGN KEY (`productID` )  
    REFERENCES `ING`.`Products` (`productID` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `fk_ProductStockLevels_Warehouses`  
    FOREIGN KEY (`warehouseID` )  
    REFERENCES `ING`.`Warehouses` (`warehouseID` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `ING`.`ProductOffers` (  
  `offerID` INT NOT NULL AUTO_INCREMENT ,  
  `productID` INT NOT NULL ,  
  `expiryDate` DATE NOT NULL ,  
  PRIMARY KEY (`offerID`) ,  
  INDEX `fk_ProductOffers_Products` (`productID` ASC) ,  
  UNIQUE INDEX `offerID_UNIQUE` (`offerID` ASC) ,  
  CONSTRAINT `fk_ProductOffers_Products`  
    FOREIGN KEY (`productID` )  
    REFERENCES `ING`.`Products` (`productID` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `ING`.`PercentageOffers` (  
  `percentageOfferID` INT NOT NULL AUTO_INCREMENT ,  
  `offerID` INT NOT NULL ,  
  PRIMARY KEY (`percentageOfferID`) ,  
  INDEX `fk_PercentageOffers_ProductOffers` (`offerID` ASC) ,  
  UNIQUE INDEX `percentageOfferID_UNIQUE` (`percentageOfferID` ASC) ,  
  CONSTRAINT `fk_PercentageOffers_ProductOffers`  
    FOREIGN KEY (`offerID` )  
    REFERENCES `ING`.`ProductOffers` (`offerID` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

15/01/2010

```
CREATE TABLE IF NOT EXISTS `ING`.`FreeGiftOffers` (  
  `freeGiftOfferID` INT NOT NULL AUTO_INCREMENT ,  
  `offerID` INT NOT NULL ,  
  `productID` INT NULL ,  
  PRIMARY KEY (`freeGiftOfferID`) ,  
  INDEX `fk_FreeGiftOffers_ProductOffers` (`offerID` ASC) ,  
  INDEX `fk_FreeGiftOffers_Products` (`productID` ASC) ,  
  UNIQUE INDEX `freeGiftOfferID_UNIQUE` (`freeGiftOfferID` ASC) ,  
  CONSTRAINT `fk_FreeGiftOffers_ProductOffers`  
    FOREIGN KEY (`offerID` )  
    REFERENCES `ING`.`ProductOffers` (`offerID` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `fk_FreeGiftOffers_Products`  
    FOREIGN KEY (`productID` )  
    REFERENCES `ING`.`Products` (`productID` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `ING`.`ProductMedia` (  
  `mediaID` INT NOT NULL AUTO_INCREMENT ,  
  `mediaType` VARCHAR(45) NOT NULL ,  
  `mediaPath` VARCHAR(500) NOT NULL ,  
  `productID` INT NULL ,  
  PRIMARY KEY (`mediaID`) ,  
  INDEX `fk_ProductMedia_Products` (`productID` ASC) ,  
  UNIQUE INDEX `mediaID_UNIQUE` (`mediaID` ASC) ,  
  CONSTRAINT `fk_ProductMedia_Products`  
    FOREIGN KEY (`productID` )  
    REFERENCES `ING`.`Products` (`productID` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `ING`.`ProductTechnicalSpecifications` (  
  `specificationID` INT NOT NULL AUTO_INCREMENT ,  
  `name` VARCHAR(45) NOT NULL ,  
  `value` VARCHAR(300) NOT NULL ,  
  `productID` INT NOT NULL ,  
  PRIMARY KEY (`specificationID`) ,  
  INDEX `fk_ProductTechnicalSpecifications_Products` (`productID` ASC) ,  
  UNIQUE INDEX `specificationID_UNIQUE` (`specificationID` ASC) ,  
  CONSTRAINT `fk_ProductTechnicalSpecifications_Products`  
    FOREIGN KEY (`productID` )  
    REFERENCES `ING`.`Products` (`productID` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

15/01/2010

```
CREATE TABLE IF NOT EXISTS `ING`.`Customers` (  
  `customerID` INT NOT NULL AUTO_INCREMENT ,  
  `name` VARCHAR(50) NULL ,  
  `email` VARCHAR(200) NOT NULL ,  
  PRIMARY KEY (`customerID`) ,  
  UNIQUE INDEX `customerID_UNIQUE` (`customerID` ASC) )  
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `ING`.`CustomerAddresses` (  
  `addressID` INT NOT NULL AUTO_INCREMENT ,  
  `customerID` INT NOT NULL ,  
  `address` TEXT NOT NULL ,  
  `postcode` VARCHAR(20) NOT NULL ,  
  PRIMARY KEY (`addressID`) ,  
  INDEX `fk_CustomerAddresses_Customers` (`customerID` ASC) ,  
  UNIQUE INDEX `addressID_UNIQUE` (`addressID` ASC) ,  
  CONSTRAINT `fk_CustomerAddresses_Customers`  
    FOREIGN KEY (`customerID` )  
    REFERENCES `ING`.`Customers` (`customerID` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `ING`.`CustomerOrderShippingMethods` (  
  `shippingMethodID` INT NOT NULL AUTO_INCREMENT ,  
  `name` VARCHAR(45) NOT NULL ,  
  `cost` DECIMAL(3,2) NOT NULL ,  
  PRIMARY KEY (`shippingMethodID`) ,  
  UNIQUE INDEX `shippingMethodID_UNIQUE` (`shippingMethodID` ASC) )  
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `ING`.`CustomerOrders` (  
  `orderID` INT NOT NULL AUTO_INCREMENT ,  
  `customerID` INT NOT NULL ,  
  `shippingAddressID` INT NOT NULL ,  
  `customerAddressID` INT NOT NULL ,  
  `shippingMethodID` INT NOT NULL ,  
  `date` DATE NOT NULL ,  
  PRIMARY KEY (`orderID`) ,  
  INDEX `fk_CustomerOrders_Customers` (`customerID` ASC) ,  
  INDEX `fk_CustomerOrders_CustomerAddresses` (`shippingAddressID` ASC) ,  
  INDEX `fk_CustomerOrders_CustomerAddresses1` (`customerAddressID` ASC) ,  
  INDEX `fk_CustomerOrders_CustomerOrderShippingMethod` (`shippingMethodID` ASC) ,  
  UNIQUE INDEX `orderID_UNIQUE` (`orderID` ASC) ,  
  CONSTRAINT `fk_CustomerOrders_Customers`  
    FOREIGN KEY (`customerID` )  
    REFERENCES `ING`.`Customers` (`customerID` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `fk_CustomerOrders_CustomerAddresses`  
    FOREIGN KEY (`shippingAddressID` )
```

15/01/2010

```
REFERENCES `ING`.`CustomerAddresses` (`addressID` )
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_CustomerOrders_CustomerAddresses1`
FOREIGN KEY (`customerAddressID` )
REFERENCES `ING`.`CustomerAddresses` (`addressID` )
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_CustomerOrders_CustomerOrderShippingMethod`
FOREIGN KEY (`shippingMethodID` )
REFERENCES `ING`.`CustomerOrderShippingMethods` (`shippingMethodID` )
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `ING`.`CustomerProductRatings` (
`ratingID` INT NOT NULL AUTO_INCREMENT ,
`productID` INT NOT NULL ,
`customerID` INT NOT NULL ,
`rating` TINYINT UNSIGNED NOT NULL ,
PRIMARY KEY (`ratingID` ) ,
INDEX `fk_CustomerProductRating_Products` (`productID` ASC) ,
INDEX `fk_CustomerProductRating_Customers` (`customerID` ASC) ,
UNIQUE INDEX `ratingID_UNIQUE` (`ratingID` ASC) ,
CONSTRAINT `fk_CustomerProductRating_Products`
FOREIGN KEY (`productID` )
REFERENCES `ING`.`Products` (`productID` )
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_CustomerProductRating_Customers`
FOREIGN KEY (`customerID` )
REFERENCES `ING`.`Customers` (`customerID` )
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `ING`.`CustomerOrderLogs` (
`logID` INT NOT NULL AUTO_INCREMENT ,
`status` ENUM('PENDING','PROCESSED','DISPATCHED','RETURNED','DELIVERED') NOT NULL ,
`orderID` INT NOT NULL ,
PRIMARY KEY (`logID` ) ,
INDEX `fk_CustomerOrderLog_CustomerOrders` (`orderID` ASC) ,
UNIQUE INDEX `logID_UNIQUE` (`logID` ASC) ,
CONSTRAINT `fk_CustomerOrderLog_CustomerOrders`
FOREIGN KEY (`orderID` )
REFERENCES `ING`.`CustomerOrders` (`orderID` )
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

15/01/2010

```
CREATE TABLE IF NOT EXISTS `ING`.`Staff` (  
  `staffID` INT NOT NULL AUTO_INCREMENT ,  
  `name` VARCHAR(100) NOT NULL ,  
  `position` VARCHAR(100) NOT NULL ,  
  PRIMARY KEY (`staffID`) ,  
  UNIQUE INDEX `staffID_UNIQUE` (`staffID` ASC) )  
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `ING`.`StaffOrderNotes` (  
  `noteID` INT NOT NULL AUTO_INCREMENT ,  
  `note` TEXT NOT NULL ,  
  `logID` INT NOT NULL ,  
  `staffID` INT NOT NULL ,  
  PRIMARY KEY (`noteID`) ,  
  INDEX `fk_StaffOrderNotes_CustomerOrderLog` (`logID` ASC) ,  
  INDEX `fk_StaffOrderNotes_Staff` (`staffID` ASC) ,  
  UNIQUE INDEX `noteID_UNIQUE` (`noteID` ASC) ,  
  CONSTRAINT `fk_StaffOrderNotes_CustomerOrderLog`  
    FOREIGN KEY (`logID` )  
    REFERENCES `ING`.`CustomerOrderLogs` (`logID` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `fk_StaffOrderNotes_Staff`  
    FOREIGN KEY (`staffID` )  
    REFERENCES `ING`.`Staff` (`staffID` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `ING`.`ProductCustomerReviews` (  
  `productID` INT NOT NULL ,  
  `customerID` INT NOT NULL ,  
  `Review` TEXT NOT NULL ,  
  `date` DATE NOT NULL ,  
  PRIMARY KEY (`productID`, `customerID`) ,  
  INDEX `fk_Products_has_Customers_Products` (`productID` ASC) ,  
  INDEX `fk_Products_has_Customers_Customers` (`customerID` ASC) ,  
  UNIQUE INDEX `productID_UNIQUE` (`productID` ASC) ,  
  UNIQUE INDEX `customerID_UNIQUE` (`customerID` ASC) ,  
  CONSTRAINT `fk_Products_has_Customers_Products`  
    FOREIGN KEY (`productID` )  
    REFERENCES `ING`.`Products` (`productID` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `fk_Products_has_Customers_Customers`  
    FOREIGN KEY (`customerID` )  
    REFERENCES `ING`.`Customers` (`customerID` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION);
```

15/01/2010

```
CREATE TABLE IF NOT EXISTS `ING`.`OrderedProducts` (  
  `subOrderID` INT NOT NULL AUTO_INCREMENT,  
  `productID` INT NOT NULL ,  
  `orderID` INT NOT NULL ,  
  PRIMARY KEY (`subOrderID`),  
  INDEX `fk_table1_Products` (`productID` ASC) ,  
  INDEX `fk_orderedProducts_CustomerOrders` (`orderID` ASC) ,  
  UNIQUE INDEX `subOrderID_UNIQUE` (`subOrderID` ASC) ,  
  CONSTRAINT `fk_table1_Products`  
    FOREIGN KEY (`productID` )  
    REFERENCES `ING`.`Products` (`productID` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `fk_orderedProducts_CustomerOrders`  
    FOREIGN KEY (`orderID` )  
    REFERENCES `ING`.`CustomerOrders` (`orderID` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

### Question 3

Based on the DDL model you have implemented, answer the following queries in SQL.

**I have used joins rather than sub-queries where possible for portability and efficiency reasons.**

1. Retrieve the total numbers of reviews with four stars or above rating for product ID '1'.

```
SELECT count(*)  
FROM CustomerProductRatings  
WHERE rating > 4 AND productID=1;
```

2. Retrieve the reviews from the last seven days for product id 'A01223'.

```
SELECT review  
FROM ProductCustomerReviews  
WHERE date BETWEEN CURDATE()-7 AND CURDATE() AND productID = 1;
```

**Or if the dates are worked out programmatically:**

```
SELECT Review  
FROM ProductCustomerReviews  
WHERE date BETWEEN '2009-01-00' AND '2009-01-07' AND productID = 1;
```

**3. Retrieve the orders with the 'PENDING' status and sorted by order date.**

```
SELECT *
FROM CustomerOrders
LEFT JOIN CustomerOrderLogs ON CustomerOrderLogs.orderID = CustomerOrders.orderID
WHERE CustomerOrderLogs.status = "PENDING"
ORDER BY CustomerOrders.date ASC;
```

**4. Retrieve the customer IDs with no orders made.**

```
SELECT Customers.customerID
FROM Customers
LEFT JOIN CustomerOrders ON CustomerOrders.customerID = Customers.customerID
WHERE CustomerOrders.customerID IS NULL;
```

**5. Retrieve the details of the product categories with more than 10 types of products.**

```
SELECT ProductCategories.categoryID, ProductCategories.name, count(*) as
numberOfProductsInCategory
FROM ProductCategories
LEFT JOIN Products ON Products.categoryID = ProductCategories.categoryID
GROUP BY ProductCategories.categoryID
HAVING numberOfProductsInCategory > 10;
```

**6. Retrieve the product IDs with the stock level lower than 20 in the warehouse 'Durham'.**

```
SELECT *
FROM Products
LEFT JOIN ProductStockLevels on Products.productID = ProductStockLevels.productID
LEFT JOIN Warehouses on ProductStockLevels.warehouseID = Warehouses.warehouseID
GROUP BY Products.productID
HAVING ProductStockLevels.stockLevel < 20 AND Warehouses.name = 'Durham';
```

**7. Retrieve the product IDs with no media gallery.**

```
SELECT Products.productID
FROM Products
LEFT JOIN ProductMedia ON ProductMedia.productID= Products.productID
WHERE ProductMedia.productID IS NULL;
```

**8. Retrieve the total number of orders with the Second class shipping method.**

```
SELECT count(*) AS numberOfSecondClassOrders
FROM CustomerOrders
LEFT JOIN CustomerOrderShippingMethods ON CustomerOrders.shippingMethodID =
CustomerOrderShippingMethods.shippingMethodID
GROUP BY CustomerOrderShippingMethods.name
HAVING CustomerOrderShippingMethods.name = "Second class";
```

15/01/2010

9. Retrieve the order details made by customer ID '1' sorted by order date. I wasn't sure if the question was asking to just display the product IDs or if it was asking to show the details of the products too, so I have written a query for each:

**9-a. Just displaying the product IDs.**

```
SELECT *
FROM CustomerOrders
LEFT JOIN orderedProducts on CustomerOrders.orderID = orderedProducts.orderID
HAVING CustomerOrders.customerID = 1
ORDER BY CustomerOrders.date ASC;
```

**9-b. Displaying the details of the products ordered too.**

```
SELECT *
FROM CustomerOrders
LEFT JOIN orderedProducts on CustomerOrders.orderID = orderedProducts.orderID
LEFT JOIN Products on orderedProducts.productID = Products.productID
HAVING CustomerOrders.customerID = 1
ORDER BY CustomerOrders.date ASC;
```

10. Add a new order note 'Credit card expired' to order ID '1'.

```
INSERT INTO StaffOrderNotes
VALUES
(
    NULL,
    'Credit card expired',
    (SELECT logID FROM CustomerOrderLogs WHERE orderID = 1), 1
)
```

15/01/2010

## Question 4

The system also provides recommendation of products to the customer who is viewing the product. Write a stored procedure or function for the following query: providing a product ID and Customer ID, retrieve the details of the top five other products the other customers have bought in the same order.

```
DROP PROCEDURE IF EXISTS recommendation;
DELIMITER $$
CREATE PROCEDURE recommendation(IN product INT, IN customer INT)
BEGIN
    DECLARE no_more_rows BOOL DEFAULT FALSE;
    DECLARE currentProductID INT;
    DECLARE currentProductName VARCHAR(45);
    DECLARE currentProductPrice decimal(2,0);
    DECLARE temp_customerID INT;
    DECLARE temp_timesOrdered INT;

    DECLARE orders CURSOR FOR
        SELECT productID, customerID, COUNT(productID) AS timesOrdered
           FROM CustomerOrders
          LEFT JOIN orderedProducts ON CustomerOrders.orderID = orderedProducts.orderID
          GROUP BY ProductID
          HAVING CustomerOrders.customerID != customer AND orderedProducts.productID != product
          ORDER BY timesOrdered DESC
          LIMIT 5;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET no_more_rows = TRUE;

    CREATE TABLE IF NOT EXISTS ProcedureOutput(ID INT, name VARCHAR(45), price decimal(2,0));
    DELETE FROM ProcedureOutput;
    OPEN orders;
    loop1: LOOP
        FETCH orders
           INTO currentProductID, temp_customerID, temp_timesOrdered;
        SELECT name, price FROM Products WHERE productID = currentProductID INTO
currentProductName, currentProductPrice;
        IF no_more_rows THEN
            CLOSE orders;
            LEAVE loop1;
        END IF;
        INSERT INTO ProcedureOutput VALUES (currentProductID, currentProductName,
currentProductPrice);
    END LOOP loop1;

    SELECT * FROM ProcedureOutput;
    DROP TABLE ProcedureOutput;
END $$
DELIMITER ;
```