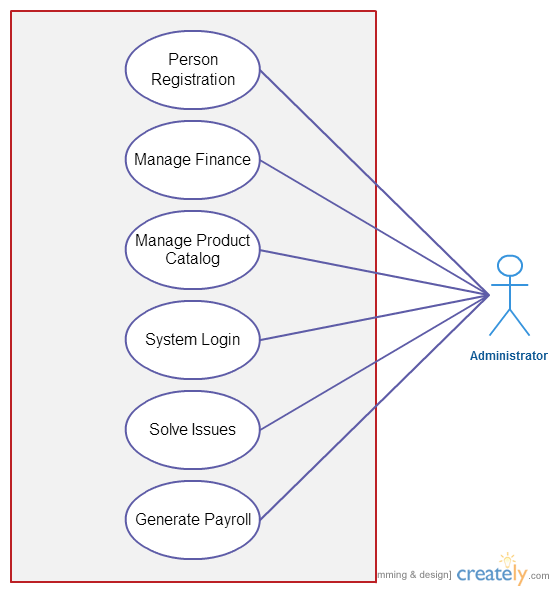
# Ecommerce Database for Barnes and Noble Site

INFO 6210 Final Project Database Design  
Vayalandi Danisha Kannan

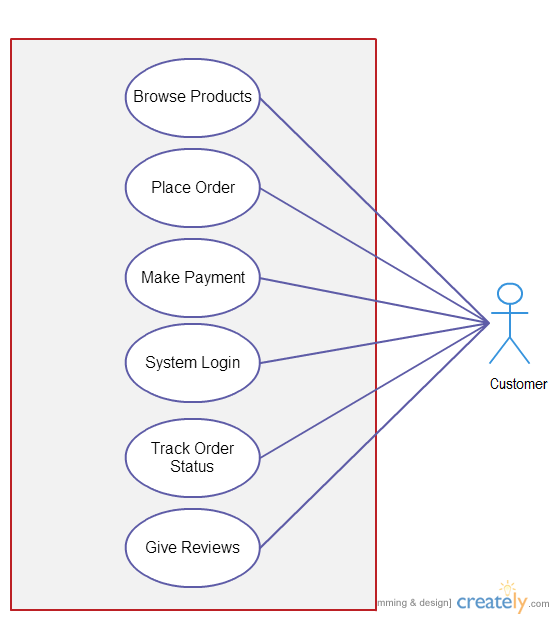
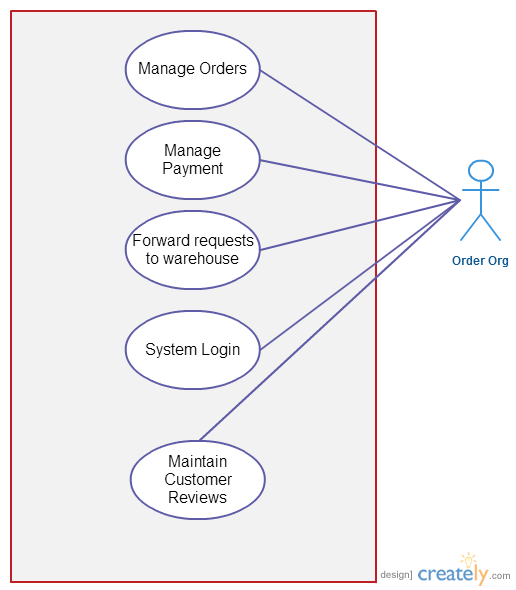
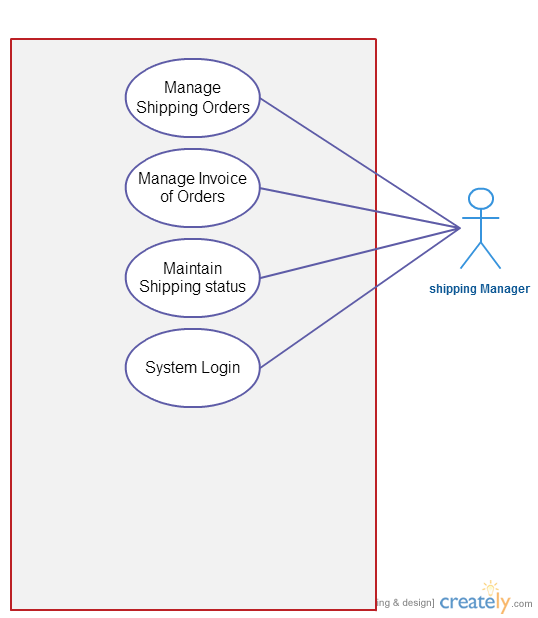
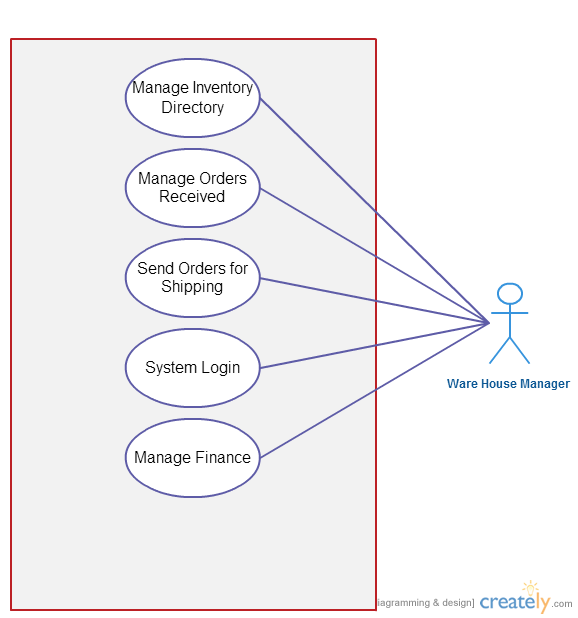
|  |  |  |
| --- | --- | --- |
| Serial No | Name | Page Number |
| 1 | Abstract | 2 |
| 2 | Introduction | 3 |
| 3 | Roles and Tasks | 4 |
| 4 | Entity Relationship Diagram | 10 |
| 5 | Normalization | 15 |
| 6 | Views | 16 |
| 7 | Database And Application Security | 18 |
| 8 | Procedures | 22 |
| 9 | Triggers | 25 |

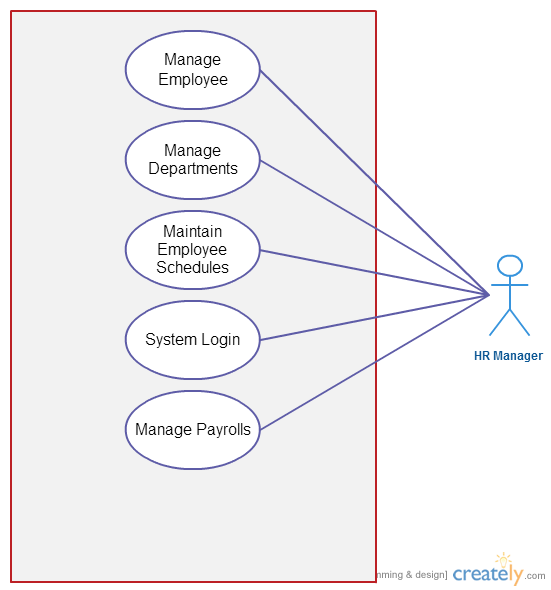
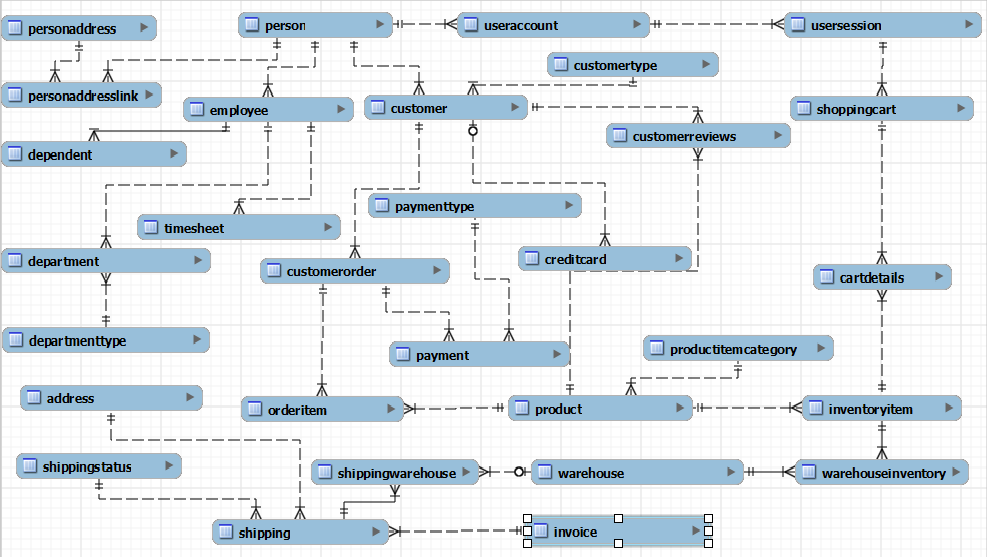
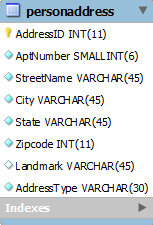
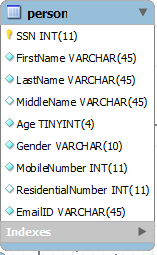
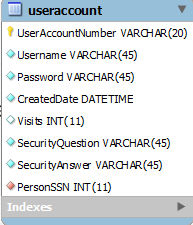
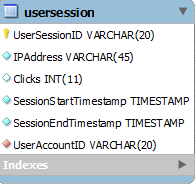
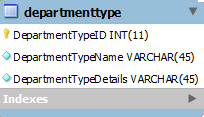
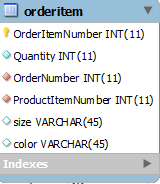
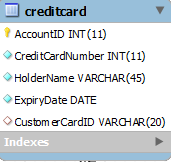
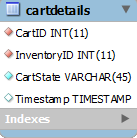
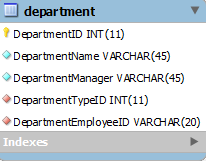
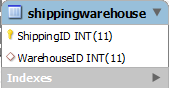
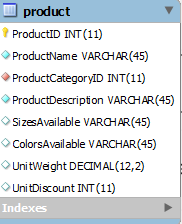
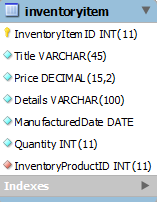
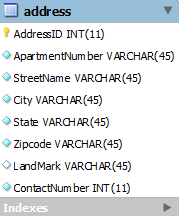
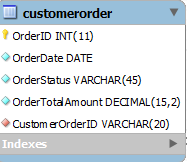
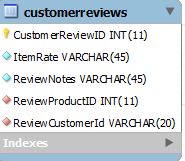
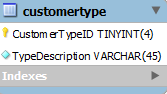
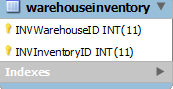
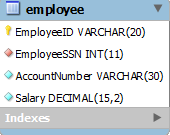
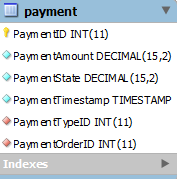
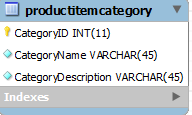
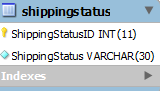
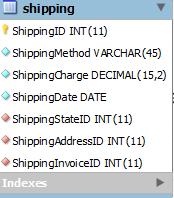
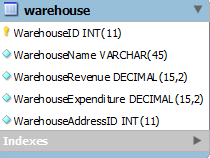
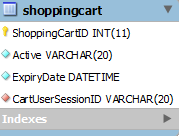
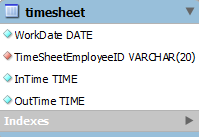
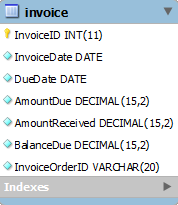
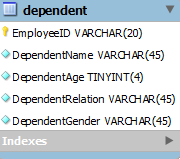
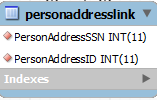
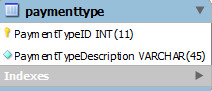
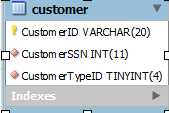
ABSTRACT

The Barnes and Noble Site is an e-commerce site which sells books and other stationary products. They provide services where in a customer can place order online and pay it online. The respective product would be delivered to the customer in a certain period of time. The various entities involved in the system are the employees, customer, warehouse manager, shipping manager, HR manager and the administrator who manages everything. There are different organizations which would take care of certain responsibilities. The main goal is to provide high quality, time efficient and cost effective product and product delivery system to the customers.  
  
Once the order is placed by the customer the order status would be mentioned as ‘pending’, the order request would be forwarded to the respective warehouses where the ordered order-items would be available. These order-items would be then forwarded to the shipping organization. The shipping organization has the shipping address and they provide the shipping status to the customer as well, through which they can track their order.  
  
The employee details are also managed by the admin, where in the employee details, their work schedules, pay details, dependent details and their department are all known. These details would help us to manage the staff of the enterprise efficiently.

1. Introduction  
   1. Document Purpose  
        
      The purpose of this document is to describe the business requirements of an application completely and accurately. The above application is developed for Barnes and Noble site to serve the purpose of online shopping system. The system enables an efficient shopping environment to search for products according to the customer’s needs and place an order for them. The system provides user accounts for both customers as well as the employees of the enterprise.
   2. Intended Audience  
        
      The main intended audience for this document are the business owners of the proposed system. The intended audience for this system are the Enterprise Administrators, Order Organization admins, Warehouse Managers, Shipping Managers, HR Department and the Customers. Each user has specified roles which are discussed in detail in the following sections.
   3. Rationale  
        
      The application would benefit the owners as the profit would be earned from the proper delivery of the ordered products to the customer. The application also serves as an interface between the different organizations as well as the customer and the enterprise.
2. Roles and Tasks  
   1. Actors and Roles  
        
      Administrator:  
       The person in charge of all the enterprise transactions as well as the major functionalities of various organizations.  
        
      

Customer:

  
  
  
  
  
  
  
  
  
  
  
Order Organization:  
  
  
  
  
  
  
  
  
  
  
  
  
Shipping Manager:  
  
  
  
  
  
  
  
  
  
  
  
Warehouse Manager:  
  


HR Manager:  
  
  
  
  
  
  
  
  
  
  
  
  
  
3. Entity Relationship Diagram  
  
This diagram represents the relationship between the different business entities in the database and description of their attributes. The metadata definitions would also be given later.  
  
 3.1 Data Architecture  
 The entities present in the data model are designed such that the data integrity is maintained and the business critical data is easily retrievable. The Entity Relationship diagram represents a detailed overview of the database created.  
  
  
  
  
  
  
  
  
  
3.2 Table Definitions  
  
 The metadata definitions are also given to ease the transition to the physical schema implementation phase.  
  
   “  
  
    
   
     
  
     
  
     
  
     
  
  
     
   
    
  
    
  
   

3.3 Key Entities and Relationships  
  
Person: This entity contains information about the two main user types included in the enterprise which are employee or customer. The Person, Employee and the Customer tables follow the specialization constraint. It includes common information like name, address, age and contact number.  
  
Employee: It includes the unique information other than the person table which includes the employee Id, department, time schedule, dependents and the salary.  
  
Timesheet: This entity gives detail about the employees working schedule according to which they’ll decide the salary of the employee.  
  
Customer: It includes unique information about customer other than the person table which includes the customer ID and the customer type.  
  
Customer Order: This entity includes information about the orders placed by the customer like the order date, order ID, order status which would help the customer to know at what state the order has reached, order total amount and a reference key to link each customer with the order.  
  
Customer Reviews: Each customer is allowed to give reviews about the products present in the product catalog. These review information is stored in the customer reviews table.  
  
User Account: This entity stores the user account information of each and every person entity formed in the system. It mainly includes the username, password and the security question and answers.  
  
User Session: This entity helps to store every sessions accessed by the person entity in the system. It consists of info like the unique session ID, IP address and the session start and end time.  
  
Shopping Cart: For every session used by a customer, the customer is provided with a shopping cart where in the products to be purchased are added. This entity maintains details regarding that action and has attributes like the shopping card id, active or inactive, expiry date of the cart and link between the cart and the user session.  
  
Person Address: This entity stores address information about each person. A person can store as many address details as one wants to.   
  
Credit Card: This entity stores information about the credit card information about the customer so that it would be easy to trace the payment details from the customer to the bank through which the payment is being done.  
  
Order Item: It includes details of all the items placed in the customer order which includes the quantity of products placed.  
  
Warehouse: The products would be ordered from certain warehouses and the warehouse details would be stored in this entity. There can be many warehouses based on different addresses.  
  
Shipping: The shipping of the product details would be known from this entity as well the other entities associated with it like the address entity which gives details about the shipping address and the shipping status entity which gives details about the shipping status of the order.  
  
Department: There are many department in the enterprise and this entity gives info about each one and their managers.  
  
Invoice: This entity gives a detailed bill of the order placed by the customer to the various other entities.  
  
Product and Product Category: These both entities give details about the products and their types.  
  
  
  
4. Normalization  
  
 To confirm that data is consistent across the various entities that store address information for each customer and employee, the address has been kept as a separate entity so that its information remains consistent for each person entity. There are many other cases where such changes are made, to maintain the consistency of the data throughout. The EER model mentioned above is the normalized one. Some tables were not in normalized form and they were normalized till 3NF forms.   
  
Those table details are mentioned below:  
  
Person:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SSN | First name | Last name | Age | Gender | Contact number | Address1 | Address2 |

The above table on normalization is represented as:  
Person:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SSN | First Name | Last Name | Middle Name | Age | Gender | Mobile Number | Email |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| User Account ID | Username | Password | Created Date | Visits | Security Question | Security Answer | Person SSN |

The same normalization techniques has been used to attain the following normalized relations. They are as follows:  
  
Customer:

|  |  |  |
| --- | --- | --- |
| Customer ID | Customer SSN | Customer Type ID |

Customer Type:

|  |  |
| --- | --- |
| Customer Type ID | Type Description |

5. Views  
  
There can be many views based on the different users using the application. An example of these views for Customer and admin is given as follows:  
  
Customer View:-  
  
CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `customer\_order\_view` AS

select

`customer`.`CustomerID` AS `Customer ID`,

`customerorder`.`OrderID` AS `Order ID`,

`customerorder`.`OrderDate` AS `Order Date`,

`customerorder`.`OrderStatus` AS `Order Status`,

`customerorder`.`OrderTotalAmount` AS `Total Amount`,

`invoice`.`DueDate` AS `Due Date`

from

((`customer`

join `invoice`)

join `customerorder`)

where

((`customer`.`CustomerID` = `customerorder`.`CustomerOrderID`)

and (`invoice`.`InvoiceOrderID` = `customerorder`.`OrderID`))  
  
  
  
  
  
  
  
  
Employee Manager View:-  
  
CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `hr\_view` AS

select

`employee`.`EmployeeSSN` AS `Employee SSN`,

`employee`.`EmployeeID` AS `Employee ID`,

`person`.`FirstName` AS `First Name`,

`person`.`LastName` AS `Last Name`,

`person`.`Age` AS `Age`,

`person`.`Gender` AS `Gender`,

`person`.`MobileNumber` AS `Contact Number`,

`department`.`DepartmentName` AS `Department Name`,

`useraccount`.`Username` AS `UserName`,

`useraccount`.`Password` AS `Password`

from

(((`employee`

join `useraccount`)

join `department`)

join `person`)

where

((`employee`.`EmployeeSSN` = `useraccount`.`PersonSSN`)

and (`department`.`DepartmentEmployeeID` = `employee`.`EmployeeID`)

and (`person`.`SSN` = `employee`.`EmployeeSSN`))  
  
  
  
  
  
  
  
6. Database and Application Security  
  
 As the proposed database contains various types of data which is shared across various functional roles and with some of the data shared publicly with online customers, different MySQL users were created.   
  
The accounts created were:   
 a. Administrator  
 b. Customer  
 c. Order Org   
 d. Warehouse Org  
 e. Shipping Org  
 f. Human Resources Org  
  
 Grants were considered carefully, based on the review of tables and views necessary to support functions required of each type of account. A list of grants for each account is provided below.  
  
Administrator:  
  
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin';

GRANT ALL ON ecommercedb.\* TO 'admin'@'localhost';

GRANT SELECT ON ecommercedb.\* TO 'admin'@'localhost';

grant update on ecommercedb.employee to 'admin'@'localhost';

grant update on ecommercedb.person to 'admin'@'localhost';

grant update on ecommercedb.product to 'admin'@'localhost';

grant update on ecommercedb.productitemcategory to 'admin'@'localhost';

grant update on ecommercedb.department to 'admin'@'localhost';

grant update on ecommercedb.departmenttype to 'admin'@'localhost';

grant insert on ecommercedb.department to 'admin'@'localhost';

grant insert on ecommercedb.departmenttype to 'admin'@'localhost';

grant insert on ecommercedb.product to 'admin'@'localhost';

grant insert on ecommercedb.productitemcategory to 'admin'@'localhost';

GRANT USAGE ON \*.\* TO 'admin'@'localhost' WITH MAX\_QUERIES\_PER\_HOUR 90;

Customer:   
  
CREATE USER 'customer'@'localhost' IDENTIFIED BY 'customer’;

GRANT ALL ON ecommercedb.\* TO 'customer'@'localhost';

GRANT SELECT ON ecommercedb.customer TO ' customer '@'localhost';

GRANT SELECT ON ecommercedb.customerorder TO ' customer '@'localhost';

GRANT SELECT ON ecommercedb.orderItem TO ' customer '@'localhost';

GRANT SELECT ON ecommercedb.payment TO ' customer '@'localhost';

grant update on ecommercedb.customer to ' customer '@'localhost';

grant update, insert on ecommercedb.customerorder to ' customer '@'localhost';

grant insert on ecommercedb.creditcard to 'admin'@'localhost';

grant insert on ecommercedb.customer to 'admin'@'localhost';

grant insert on ecommercedb.customerreviews to 'admin'@'localhost';

grant update on ecommercedb.orderItem to ' customer '@'localhost';

GRANT USAGE ON \*.\* TO ' customer '@'localhost' WITH MAX\_QUERIES\_PER\_HOUR 90;

Order Org:

CREATE USER 'orderorg'@'localhost' IDENTIFIED BY ' orderorg’;

GRANT ALL ON ecommercedb.\* TO ' orderorg '@'localhost';

GRANT SELECT ON ecommercedb.customer TO ' orderorg '@'localhost';

GRANT SELECT ON ecommercedb.customerorder TO ' orderorg '@'localhost';

GRANT SELECT ON ecommercedb.orderItem TO ' orderorg '@'localhost';

GRANT SELECT ON ecommercedb.payment TO ' orderorg '@'localhost';

grant update, insert on ecommercedb.product to ' orderorg '@'localhost';

grant update, insert on ecommercedb.productcategory to ' orderorg '@'localhost';

grant insert on ecommercedb.creditcard to ' invoice '@'localhost';

grant update on ecommercedb.orderItem to ' orderorg '@'localhost';

GRANT USAGE ON \*.\* TO ' orderorg '@'localhost' WITH MAX\_QUERIES\_PER\_HOUR 90;

Shipping Org:  
  
CREATE USER 'shipping'@'localhost' IDENTIFIED BY 'shipping';

GRANT ALL ON ecommercedb.shipping TO 'shipping'@'localhost';

GRANT SELECT ON ecommercedb.shippingstatus TO 'shipping'@'localhost';

GRANT SELECT ON ecommercedb.customerorder TO 'shipping'@'localhost';

GRANT SELECT ON ecommercedb.inventoryitem TO 'shipping'@'localhost';

GRANT UPDATE, insert on ecommercedb.shipping TO 'shipping'@'localhost';

GRANT INSERT ON ecommercedb.shippingstatus TO 'shipping'@'localhost';

GRANT USAGE ON \*.\* TO 'warehouse'@'localhost' WITH MAX\_QUERIES\_PER\_HOUR 90;

Warehouse Org:  
  
CREATE USER 'warehouse'@'localhost' IDENTIFIED BY 'warehouse';

GRANT ALL ON ecommercedb.shippingwarehouse TO 'warehouse'@'localhost';

GRANT SELECT ON ecommercedb.shippingstatus TO 'warehouse'@'localhost';

GRANT SELECT ON ecommercedb.customerorder TO 'warehouse'@'localhost';

GRANT SELECT ON ecommercedb.inventoryitem TO 'warehouse'@'localhost';

GRANT UPDATE ON ecommercedb.inventoryitem TO 'warehouse'@'localhost';

GRANT INSERT ON ecommercedb.inventoryitem TO 'warehouse'@'localhost';

GRANT USAGE ON \*.\* TO 'warehouse'@'localhost' WITH MAX\_QUERIES\_PER\_HOUR 90;

HR Org:  
  
CREATE USER 'HR'@'localhost' IDENTIFIED BY 'HR';

GRANT ALL ON ecommercedb.employee TO 'HR'@'localhost';

GRANT SELECT ON ecommercedb.person TO 'HR'@'localhost';

GRANT SELECT ON ecommercedb.useraccount TO 'HR'@'localhost';

GRANT SELECT ON ecommercedb.department TO 'HR'@'localhost';

GRANT SELECT ON ecommercedb.dependent TO 'HR'@'localhost';

GRANT UPDATE ON ecommercedb.employee TO 'HR'@'localhost';

GRANT USAGE ON \*.\* TO 'HR'@'localhost' WITH MAX\_QUERIES\_PER\_HOUR 90;

7. Procedures  
  
  
Two procedures are created to carry out some pre-defined activity for regular use. In the first procedure, orderDetails we receive the orderId as a parameter and return the order details which includes the order status, total amount and due date.

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `orderDetails\_procedure`(IN orderID int)

BEGIN

SET @sql= Concat('Select

`customer`.`CustomerID` AS `Customer ID`,

`customerorder`.`OrderDate` AS `Order Date`,

`customerorder`.`OrderStatus` AS `Order Status`,

`customerorder`.`OrderTotalAmount` AS `Total Amount`,

`invoice`.`DueDate` AS `Due Date`

from

((`customer`

join `invoice`)

join `customerorder`)

where

((`customer`.`CustomerID` = `customerorder`.`CustomerOrderID`)

and (`invoice`.`InvoiceOrderID` = ',orderID,'))');

PREPARE stmt FROM @sql;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

END  
  
  
In the second procedure named as orderItemDetails, we receive the orderID as parameter and return all the details regarding the order items included in the given order.   
The SQL code for it as follows:

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `orderItemDetails\_procedure`(IN orderID int(11))

BEGIN

SET @sql= Concat('Select

OrderItemNumber as `Item Number`,

ProductName as `Product Name`,

Quantity as `Quantity`,

ProductItemNumber as `Product Code`,

ProductItemCategory.CategoryName as `Product Category`,

size as `Size`,

color as `Color`,

UnitWeight as `Weight`,

UnitDiscount as `Discount`

from

(`orderitem`

join `product`), `ProductItemCategory`

where

((`orderitem`.`ProductItemNumber` = `product`.`ProductID`)

and (`orderitem`.`OrderNumber` = ',orderID,')

and (product.ProductCategoryID = ProductItemCategory.CategoryID))');

PREPARE stmt FROM @sql;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

END

8. Triggers  
  
  
A trigger has been used on after insert check on the CustomerOrder table, where in if the order amount is more than 400$ then the customer type would be changed to privileged customer.  
  
The SQL code for the same is given below:

DELIMITER $$

USE `ecommercedb`$$

CREATE

DEFINER=`root`@`localhost`

TRIGGER `ecommercedb`.`checkCustomerorder`

AFTER INSERT ON `ecommercedb`.`customerorder`

FOR EACH ROW

-- Edit trigger body code below this line. Do not edit lines above this one

BEGIN

IF new.OrderTotalAmount> 400 THEN

UPDATE customer SET CustomerTypeID=2

WHERE customer.CustomerID= new.CustomerOrderID;

END IF;

END$$