COEN 178 Intro. To Database Systems Spring 2020

Project (200 pts)

Note: Requirements given below from 1.0 to 3.0 are for Single person teams.

Two (more than 2 people per team not accepted) people teams should, in addition to requirements from 1.0 to 3.0, should implement requirements in 3.1.

1.0 Introduction

You are commissioned to design and implement a **Sales Management System** for ***MasterOwl***, **a ComicBook** store that sells children's comic books online. The Sales Management System uses a Relational Database to maintain the sales and customer records and the support the queries.

You are required to implement the system using an Oracle database management system.

The store sells **comic books** and other items like **T-shirts** online. Comic books and T-shirts are considered **StoreItems**. A **comic book** may have 0 or more copies.

All **StoreItems** have an *ItemId* **(unique)** and *price***.** Each comic book has an *ISBN* (unique), *title, publishedDate* and *no. of copies*. A T-shirt will have a *size*.

A **Customer** can be a Regular Customer (*Custid, name, phone/email, address*) or a **Gold Customer**. A **Gold Customer**, in addition to having all the attributes of a customer, also has an additional attribute, *dateJoined*. A GoldCustomer pays an annual fee and does not have to pay the shipping fee on her/his orders.

A Customer (or a Goldcustomer) orders StoreItems and an **ItemOrder (Note: order by is a SQL key word)** contains the *orderid* (unique), *custid, itemid, date of order*, *no.ofItems*, *shippedDate*, *shippingFee*.

Each customer's order includes a flat rate of 5% tax (for all customers) on the order total.

GoldCustomers have no shipping fee and 10% discount on an order of $100.

1.1 Constraints to be enforced:

a) The custType of a customer can be of only one of the two values, namely, '**regular**', or'**gold**'.

b) Phone (or email) must be unique and not null.

c) The no.of copies of a ComicBook cannot be < 0.

d) The no. of copies of any book ordered cannot be more than the available no. of copies of that book.

e) The shippedDate cannot be less than the OrderedDate.

f) When a regular customer orders books, the shipping fee is $10. Before the books are shipped (ie. the shippedDate is null), if the custType of that customer changes to 'gold', then the shipping fee must be changed to 0 on all orders that are not shipped yet.

2.0 Functionality and Queries to be implemented

- Create a few customers (a mix of regular and gold customers), a few Comic Books, T-shirts, Customers etc.

a) Write a **PLSQL procedure**, let us call it, **addItemOrder()** that takes several parameters – *orderid, itemid, customerid, date ordered, number ordered and shipped date*). **Note**: You are free to add any other parameters).

The procedure must do the following:

- Check if the no.of books ordered is <= no. of copies available for that book. If not, take an appropriate action (for example, display an error message and exit).
- Check if the customer is regular or gold member. If gold member, make shipping fee 0. Otherwise, add a flat shipping fee of $10.00.
- Add the order into ItemOrders. Note: Make the shippedDate NULL (this can be changed later).
- Update the no. of copies in the ComicBook table.

b) Write (and test) a **Trigger** that does the following:

After the custType in Customer is updated to ‘gold’, then check the BookOrder table and if that customer has any orders pending (not shipped yet), set the *shippingFee* to 0.

c) Write a PLSQL procedure, **setShippingDate()**, that given an *orderid* and *shipping date* as parameters, sets the *shippingDate*.

d) Write a PLSQL function, **computeTotal()** that takes an *orderid*, computes the total for that order and returns the total. The function should consider the customer type, tax, shipping fee etc, to compute the total.

e) Write a PLSQL procedure **showItemOrders()** that takes a *customerid* and a *date* as parameters and displays the **details of each ItemOrder** after the date given as a parameter.
The details of each **ItemOrder** should include the details of **Customer**, details of Item **ordered**, and **payment** details.

Customer details : custid, name, phone, address

For each itemOrder, give the following details:

Orderid, itemid, item title (name), price of item, Date Ordered and shipped date.

Payment details: Total for all items, tax, shipping fee, any discount applied (in case of Gold Customers) and the grand total.

3.0 Deliverables

Deliverable 1 (25 pts) Due 21st May

A detailed, conceptual design using the **E-R model** should be included, showing the entities, relationships, cardinalities and integrity constraints. Any diagramming tool may be used, but a detailed legend to identify the notation used, should be included (you may be given recommendations for an automated tool).

Deliverable 2 (175 pts) Due 4th June

a) The process of translating the E-R model into a relational model should be clearly shown with the resulting tables. Clearly identify the primary keys and foreign keys**. (10 pts)**

b) The script files to create the tables using SQL and load the tables with data of your choice. The script files and program files to implement the queries that are necessary to offer the functionality required. **(40 pts)**

c) A spool file that clearly demonstrates the functionality implemented by showing the queries and the results. **(125 pts)**

3.1 Two People Teams

You must provide a GUI (Graphic User interface) to the Sales Management System and execute all the queries using a web page as the front end to the application. You must provide an order form to enter the details of an ItemOrder (as described in a) via the web form. The front-end GUI should connect to a PHP application and Oracle database on the server.

3.2 Tools and Technologies

You are required to use **Oracle** as the relational database. For the GUI, you must use **HTML** and **PHP** as the Web technologies. You are free to use any other additional tools you may be familiar with.