Assignment 4

Creating and Using a RESTful Web-Service

Objective

The purpose of this assignment is to give you practice at creating and calling a RESTful web-service. This web-service will mimic an on-line shopping website. This web-service will be used/driven from the client application (a standalone application or a web application). The client will feed (insert, update, delete and get) data into/from the *shopping data model*. You can choose either MySQL or MSSQLServer as your database engine to house the data. Please make sure that you populate the data from my sample tables into your database for demo-day (D-Day) [see the Shopping Data Model at the end of this assignment]. Make sure that this is the **only data** in your database on D-Day …

The requirements given below will describe the behaviour of this client/web-service pair. I have supplied you with a mock-up of a UI for the client – since the main thrust of this assignment is not how well you define a UI, you can stick with my mock-up if you like. If you have some other ideas for the UI (that align with my client dataflow concept) then feel free to implement.

This is a fairly large undertaking for an assignment and I'd like you to work on it in teams of (up to) 4 people … I really think that there is enough content here to keep that many people busy – if you are going to create a good solution …

UI Mock-Up

Below are three screens which I see can make up a (somewhat) effective user interface for this shopping experience … As you can see the flow of the user experience is to :

- Be introduced to the web site and be asked what they are there for (SCREEN #1)
- Once they select their purpose they are taken to SCREEN #2 where they are asked to specify information (see rules and requirements below)
    - In this screen they can "GO BACK" to SCREEN #1
    - "EXECUTE" their action from SCREEN #1 with the data from SCREEN #2
    - Leave the website
    - Note that you would probably also be best to dedicate some area in this screen for some error reporting back to the user
- SCREEN #3 is the general output screen
    - If they are querying data, this is the screen that will show them their results
    - In my mock-up of SCREEN #3 – I have specified what the purchase order (P.O) will look like and what information needs to be present
    - If they have inserted / updated / deleted any information – then this screen will inform them that they were successful (or not)

CRAZY MELVIN'S SHOPPING EMPORIUM

Here at Crazy Melvin's we believe in selling things cheap !! That's why our User Interface is cheap !

Use the buttons below to tell me what you'd like to do here at Crazy Melvin's ??

Search

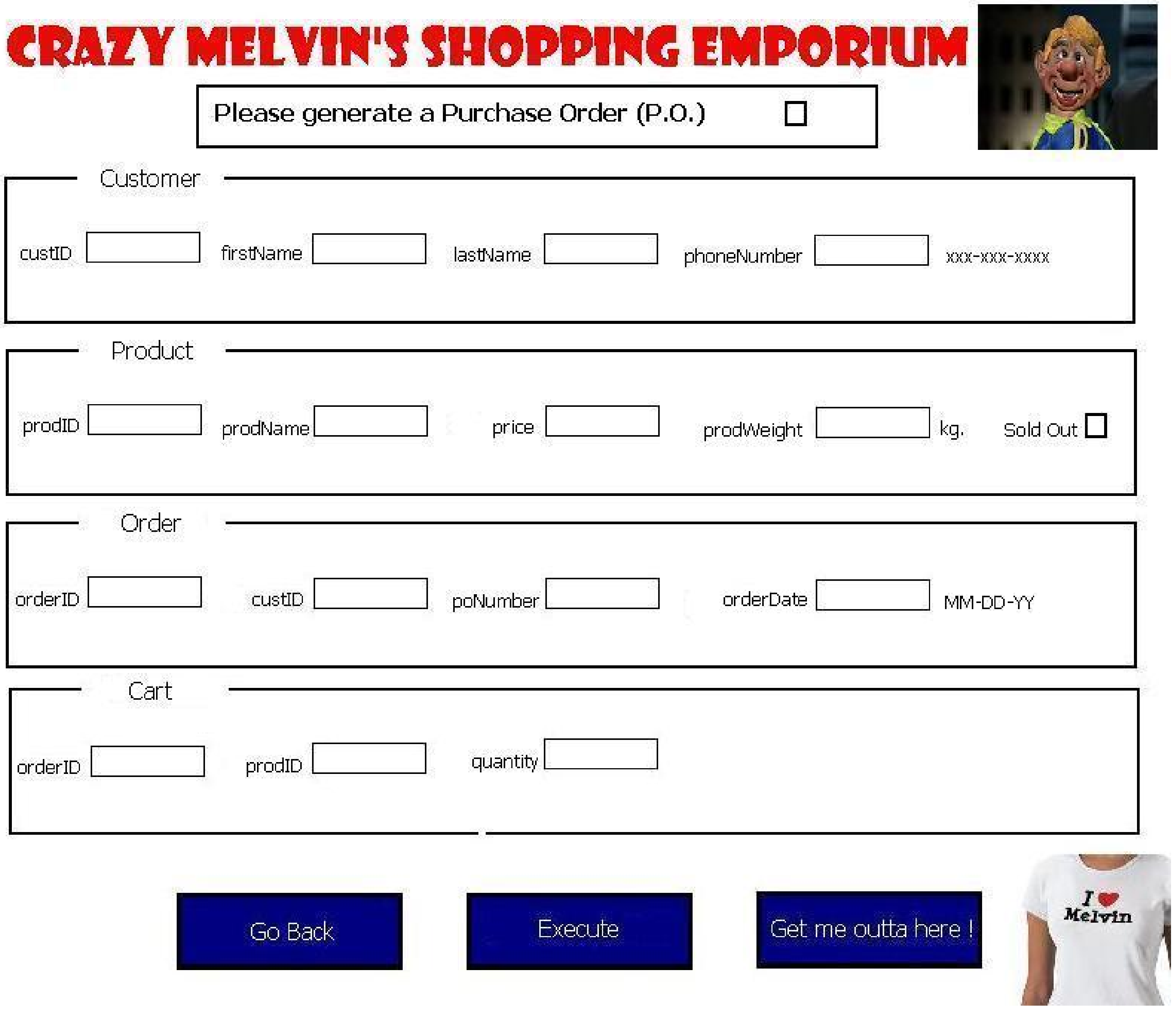
Insert some Stuff

Update some Stuff

Delete some Stuff

Get me outta here !

<<< SCREEN #1



CRAZY MELVIN'S SHOPPING EMPORIUM

Please generate a Purchase Order (P.O.) ☐

Customer

custID		firstName		lastName		phoneNumber		xxx-xxx-xxxx

Product

prodID		prodName		price		prodWeight		kg.	Sold Out	☐

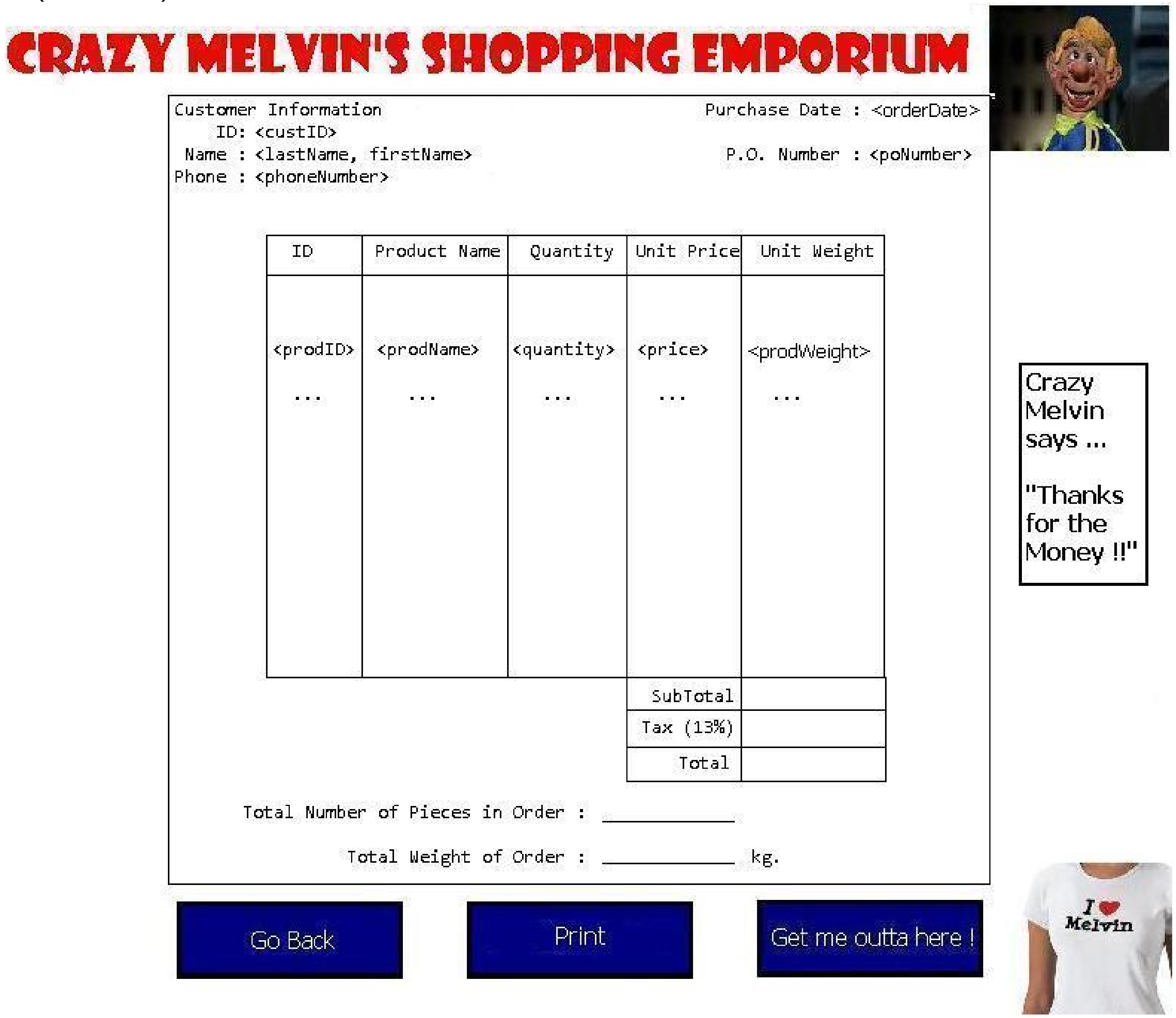
Order

orderID		custID		poNumber		orderDate		MM-DD-YY

Cart

orderID		prodID		quantity	

<<< SCREEN #2



CRAZY MELVIN'S SHOPPING EMPORIUM

Customer Information
ID: <custID>
Name : <lastName, firstName>
Phone : <phoneNumber>

Purchase Date : <orderDate>
P.O. Number : <poNumber>

ID	Product Name	Quantity	Unit Price	Unit Weight
<prodID>	<prodName>	<quantity>	<price>	<prodWeight>
...	...	...	...	...
			SubTotal	
			Tax (13%)	
			Total	

Total Number of Pieces in Order : ____________

Total Weight of Order : ____________ kg.

Crazy Melvin says ...

"Thanks for the Money !!"

<<< SCREEN #3

Architecture of Client and Web-Service

The only hard and fast rules around your solution's architecture are that you need to make the client a stand-alone application or a web application. And as well, your web-service needs to be a RESTful one. Beyond that, the architecture is up to you …

What I mean by this may be found in your answers to the following questions :

- Will the client or the web-service be responsible for doing the calculations for sub-total, tax, total, totalNumberOfPieces and the totalWeightOfOrder?
- How will you get the potentially multiple rows for a customer's query back from the web-service – all at once? one at a time?
- How will you handle a customer query across multiple tables in the database as far as your URL entry-point into the web-service goes?
    - E.g. how will you handle a call to the web-service looking for information about a customer with a lastName of "Smith" and an order date of "10-11-11"
- Will the client or the web-service be responsible to doing user input validation?
    - Is some validation better suited to be done on the client side?
    - While other validation better suited on the server side?
- How will you communicate errors back from your web-service to your client? [Since it isn't a SOAP web-service – you can't throw SOAP Faults …]
- Since the database has some column's that limit the value's width – should the client or the web-service do that? Does the user need to be informed?

Shopping Data Model

The following database schema lays out the 4 database tables used in this solution. I have indicated the primary (PK) and foreign keys (FK) for you as well – and in some cases, I have told you how to generate the values for certain columns. In the case of the suggested data types – I have used common dbase types …

Customer			
Column Name	Suggested Data Type	Description / Purpose	Relationship Information
custID	int	Unique customer identifier	Should be auto-generated as increasing value, is PK
firstName	nvarchar[50]	Customer's first name	
lastName	nvarchar[50]	Customer's last name	
phoneNumber	nvarchar[12]	Main customer contact number	Must be stored in format XXX-XXX-XXXX

Product			
Column Name	Suggested Data Type	Description / Purpose	Relationship Information
prodID	int	Unique product identifier	Should be auto-generated as increasing value (PK)
prodName	nvarchar[100]	Product's name	
price	float	Unit price for the product (for 1 product)	
prodWeight	float	The weight (in kilograms) for the product	Has to assume the user entered in kg.
inStock	?? (I'll leave this to you)	Indicates whether a product is in stock	

Order			
Column Name	Suggested Data Type	Description / Purpose	Relationship Information
orderID	int	Unique order identifier	Should be auto-generated as increasing value (PK)
custID	int	Customer that this order belongs to	FK to Customer table
poNumber	nvarchar[30]	A user driven P.O. number	Although it has “number” in its name an acceptable value could be “XYZ-33-091111”
orderDate	?? (I’ll leave this to you)	Date that order was created	However you store it – the customer will input in MM-DD-YY format and that is the format you need to output it in as well

Cart			
Column Name	Suggested Data Type	Description / Purpose	Relationship Information
orderID	int	Order that this cart belongs to	FK to Order table
prodID	int	Specific product in the cart	FK to Product Table
quantity	int	The number of the products in the cart	

Notes :

- As you can see, the Cart database table has no cartID acting as a unique key … but as you know, you need to define a key for this table … I’ll leave it to you to figure out what the PK is on the Cart table … ☺
- Further on in this Assignment write-up you will find the default values that I want you to load into your tables for demo purposes – remember that this should be the **only data** in your database!

Requirements

Basically the purpose of the web-service is to merely be a "mostly-dumb" front end to the database tables. I say "mostly-dumb" because for a number of the RESTful verbs (INSERT, UPDATE, DELETE) the mapping of incoming user information to database action is straightforward (effectively a pass-through type of strategy). The only time that the web-service really needs to apply much logic is in the GET verb's (a query) case – when a user can ask for information across multiple tables.

Below is the set of requirements, constraints and guidelines for logic and processing that both the client and/or web-service needs to do

1. The Customer and Product tables are independent of each other
   - **<u>The UI</u>** is not to allow data from the Customer **<u>and</u>** the Product areas to be input for searching, inserting, updating and deleting
   - Behind the scenes (on the web-service side) it very well may be the case that joins of customer and product are required, but they are not to be allowed through the UI!!
   - Do not allow the user to do this – tell them they are in error.
2. INSERT, UPDATE and DELETE commands can only be performed on a single table
   - That is, if a user wants to insert some Customer and Order information – they will need to do that through 2 uses of SCREEN #2
   - You will need to apply some intelligence when performing DELETEs
     i. If the user wants to delete an order – what does that imply? That also means that the corresponding Cart rows will need to be deleted
     ii. If the user wishes to delete a customer – what can that imply? It doesn't make good business sense – but in this data model, deleting a customer also deletes all orders and therefore carts associated with that customer
     iii. If the user wishes to delete a product, then all rows in the Cart table containing that product are also to be removed
   - When doing INSERTs the "id" value is not allowed as entry for the user – as it will be an auto generated number with the DBase table
3. You have a special condition to consider with the inStock value of the Product table
   - Let's say a customer adds 5 of one product to their cart
   - Then after the fact another potential user updates the Product table to indicate that that product is now out of stock
   - Then when the original customer asks for their Purchase Order (through the check box at the top of SCREEN #2) and you generate the order, your solution should tell the user that the product is out of stock
     i. So don't just look at the quantity value in the Cart table
     ii. An out of stock product will also affect the subTotal, taxes, total, totalNumberOfPieces and the totalWeightOfOrder
4. The P.O Generation option should only be visible on SCREEN #2 if the user has selected "Search" as their SCREEN #1 action
5. In your design – ensure that calling the web-service / entry points into web-service are only accessed through pure URL notation
   - Although we have seen that RESTful web-services do and can support the query-string idea on the URL – do not use query strings in this solution and design
   - For example to GET details about a Customer with the last name "Smith" the following URL would be used
     http://myservice.com/Customer/Smith
   - Instead of this URL
     http://myservice.com/Customer?lastName=Smith
6. When asking for a P.O to be generated, only the following fields can be used (in any combination) to build the query
   - orderID, custID, lastName, firstName, poNumber and/or orderDate

7. Not all columns in each table are mandatory – here is a list of mandatory columns on a per table basis)
    - Customer : custID, lastName, phoneNumber
    - Product : prodID, prodName, price, prodWeight, inStock
    - Order : orderID, custID, orderDate
    - Cart : orderID, prodID, quantity
8. *I reserve the right to add more information here (in case I forgot anything and it comes to me …)*

Important to Note

- What you need to realize is that Crazy Mel's UI is not intended to be a direct mapping to the underlying database structure in the case of SEARCHES (an HTTP GET) from the database
    - In the case of the INSERT (an HTTP POST), UPDATE (an HTTP PUT) and DELETE (an HTTP DELETE) Crazy Mel's interface is a 1:1 mapping to the underlying tables (as per requirement #2 above)
- In the case of using the UI for performing database searches, you will need to apply some intelligence on your web-service side to determine which database tables you will need to call on in order to satisfy the search that the user is trying to do
    - As well, I ask you through your responsible web-service design that you think what value-add can be done in performing the search
    - In order to put you on the right train-of-thought … let me run through some examples :

1. The user specifies custID in the Order area of the UI and a prodID in the Cart area of the UI … what is the user expecting? What tables do you need to use in your query?
    - The user wants to know "Has this customer (given by custID) ever ordered this product (given by prodID)?"
    - Behind the scenes in your web-service you will need to join the Order with the Cart table to see where this occurred and return the intersection
    - Now what data would actually be in your result set? Perhaps orderID, custID, prodID, quantity, poNumber, orderDate (i.e. all columns from the Order and Cart tables for the intersection rows)
    - Would you think to also throw in the name (last, first) of the customer? What about the productName? Would this be a nice thing (value add) for your user? You could argue that in this query, the user knew the custID and prodID – so it would be safe to say that the user knows the names
2. The user specifies custID in the Order area of the UI and a prodName in the Product area of the UI … what is the user expecting? What tables do you need to use in your query?
    - The user wants to know "Has this customer (given by custID) every ordered this product (given by prodName)?"
    - Behind the scenes in your web-service you will need to join the Order and Product with the Cart table to see where this occurred and return the intersection
    - Now what data would actually be in your result set? Perhaps orderID, custID, prodName, quantity, poNumber, orderDate (i.e. columns from the Order, Product and Cart tables for the intersection rows)
    - Would you think to also throw in the prodID for the product that they named? They clearly didn't use it in their query – so they probably don't know it … it might be nice to tell them …
3. The user specifies an orderDate in the Order area of the UI … what is the user expecting? What tables do you need to use in your query?
    - The user wants to know "What orders were taken on this date?"
    - Behind the scenes in your web-service you will need to query the Order table to display information about orders from that date.
    - Now what data would actually be in your result set? Perhaps orderID, custID, poNumber and orderDate (i.e. columns from the Order table)
    - Would you think to also throw in the customerName (last, first) for each custID named? This might be a nice thing to do …

- These examples are meant to get you to think about your web-service's resultSet strategy --- what data will you and won't you include. It is your decision – do what makes sense, but just make a conscious decision about your strategy and implement it consistently across all possible queries.

Data to Populate

I realize that I have left some database column data-types up to you to design – and for the purposes of the data in the following tables, I have provided some values for those columns.

- In the case of the "inStock" column for the Product table a "YES" value means it is in stock and a "NO" value means that it is not in stock.
- In the case of the "orderDate" field in the Order table, a value of 2011-09-15 means September 15, 2011 and would be output as 09-15-11

Customer			
custID	firstName	lastName	phoneNumber
1	Joe	Bzolay	555-555-1212
2	Nancy	Finklbaum	555-235-4578
3	Henry	Svitzinski	555-326-8456

Product				
prodID	prodName	price	prodWeight	inStock
1	Grapple Grommet	0.02	0.005	YES
2	Wandoozals	2.35	0.532	YES
3	Kardoofals	8.75	5.650	NO

Order			
orderID	custID	orderDate (MM-DD-YY)	poNumber
1	1	2011-09-15	GRAP-09-2011-001
2	1	2011-09-30	GRAP-09-2011-056
3	3	2011-10-05	

Cart		
orderID	prodID	quantity
1	1	500
1	2	1000
2	3	10
3	1	75
3	2	15
3	3	5

Please submit your project (in ZIP format) as well as your detached/zipped database files to the appropriate *eConestoga Dropbox* by 11:59pm on **November 23, 2014**– be prepared to **demo** your team's application **in class the next day**.