**Estimating**

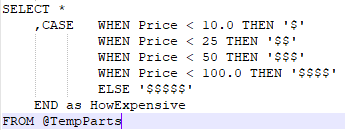
Basic requirement – Email shopping cart contents to user entered email address.

1. Add Email button to the bottom of the shopping cart page. Button should be standard grey button.
   1. 0.5 hours. Assuming standard grey button is no style or an existing style that exists in the CSS. This is adding html to an existing page for a simple input. Extra time as customer will probably change where they want the button several times.
2. The button should not display if their cart is empty.
   1. 1.0 hour. Assuming the cart is an object that already exists with a count property as most e-commerce sites display your cart with a number in the header on every page. Need to develop a method that checks the cart’s count property and set the button’s visible property to false.
3. When the email button is clicked – it should open a modal window that accepts an email address, validates that it is valid and send the shopping cart contents as an email. Allow for multiple emails separated by a comma or semicolon. If the email address is invalid handle gracefully.
   1. 5 hours. Generate a new simple page with a textbox and submit button. Creating a check for valid email is difficult because of the multitude of formats they could be in. A regular expression that validates <something>@<something>.<something> works well for a simple check.
   2. To incorporate multiple emails, split the string into an array on commas and semicolons and run the regex against each member of the array.
   3. If commas or semicolons are valid characters in an email address, will need to develop a method to only split the string on those characters after finding an @ character before it. This estimate does not include having to create this method.
4. You can leverage the existing email for the order confirmation but update it as needed to work for this purpose.
   1. 3 hours. Create an overload to the email method with an additional variable that specifies how the method was called. When no value or confirm is passed, use the regular email. If Print is passed, use the new formatting we are creating.
5. The email should also include the cart contents in an additional format for our Quick Order page. This is a simple list of items in the following format: {Qty}, {PartNumber}. The customer can easily copy this data from their email and paste it into another form on our site to quickly add all these items to their shopping cart.
   1. 5 hours. Assuming the email is html and not plaintext, create an html list in the email then loop through the cart object, printing {Qty} , {PartNumber} as a list item. After loop close the html list.

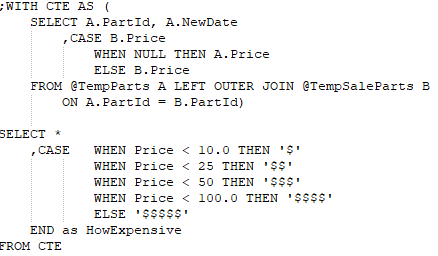
**Debugging**

1. What steps would you take to troubleshoot/solve this error
   1. The error says “Subquery return more than 1 value.” My first look would be to look at AmazonPartRepository.Select and see what query is run. Then looking at that query, look for any subqueries that are only expecting one result. If database is available, I would separately query the parts table on ID 706121S to see what rows exist for that part and what could be returned.
2. Where do you think the problem is located?
   1. If the query has been working until this point, the problem is most likely with the data that the query is running against. A non-enforced unique constraint could be being violated. The query could have assumptions in it that are not true or once were true but are not anymore. If the query has multiple parameters, it could be a problem with user input, or the way the data was entered or processed before being handed to the query.
3. Do you have any ideas what that problem may be?
   1. Without knowing what the code or database, my first thought is a problem with the data the Parts table. Secondly would be with user input or the processing of that input before using it in the query. Lastly there could be a problem with the query itself, making false assumptions in its original programming, or those assumptions have changed.

**T-SQL**

1. SELECT TOP 4 \* FROM @TempParts ORDER BY NewDate Desc
2. SELECT TOP 10 \* FROM @TempParts ORDER BY Price Asc
3. SELECT TOP 5 \* FROM @TempParts ORDER BY NEWID()
4. 

**Part 2**

1. 
2. 