**Week of 12 HW: Node.js & MySQL**

**Objectives**

* In this activity you will be creating an Amazon-like storefront using MySQL, the MySQL NPM Package, and the Prompt Package.
* The application will take in orders from customers and deplete from the store's total supplies.
* As a bonus: The application will additionally track these product sales across departments and provide a summary of the highest-grossing departments in the store.

**Submission Requirements:**

* When submitting this homework, use the normal GitHub and Heroku process.
* However, be sure to include screenshots and/or video of the working code in your README.md file of the GitHub Repo. (Research how to use README.md files if you are confused).

**Challenge #1: Customer View (Minimum Requirement)**

* Create a MySQL Database called Bamazon.
* Then create a Table inside of that database called Products.
* The products table should have each of the following columns:
  + ItemID (unique id for each product)
  + ProductName (Name of product)
  + DepartmentName
  + Price (cost to customer)
  + StockQuantity (how much of the product is available in stores)
* Populate this database with approximately 10 different products. (i.e. Insert "mock" data rows into this database and table).
* Then create a Node application called BamazonCustomer.js. Running this application will:
  + First Display All of the Items available for sale. This initial display, should include the ids, names, and prices of products for sale
  + Users should then be prompted with two messages. The first message should ask them the ID of the product they would like to buy. The second message should ask them how many of the product they would like to buy.
  + Once the customer has placed the order: Your application should...
    - Check if your store has enough quantity of the product to meet the customer's request. If not, you should respond to the user by saying: "Insufficient quantity" and prevent the order from going through.
    - If your store DOES have enough of the product to meet the customer's request, you should fulfill their order. This means that you should show them the total cost of their puchase. Then update the SQL database to reflect the remaining quantity.
* If this activity took you between 8-10 hours, then stop right here... Otherwise continue to the next set of challenges

**Challenge #2: Manager View (Next Level)**

* Create a new Node application called BamazonManager.js. Running this application will:
  + List a set of menu options: 1) View Products for Sale 2) View Low Inventory 3) Add to Inventory 4) Add New Product
  + If a manager selectis option 1 it should list all of the products available for sale: the item IDs, names, prices, and quantities.
  + If a manager selects option 2 it should list all items for which the quantity available in stores is less than 5.
  + If a manager selects option 3 it should provide the manager with the ability to "add more" of any item currently in the store.
  + If a manager selects option 4 it should provide the manager with the ability to add a completely new product to the store.
* If you've completed Challenge #2 and and have put in the hours you are willing to give this activity, then stop right here... Otherwise continue to the next and final challenge

**Challenge #3: Executive View (Final Level)**

* Create a new MySQL table called Departments. This MySQL table should have the following columns:
  + DepartmentID
  + DepartmentName
  + OverHeadCosts (A dummy number you set for each department)
  + TotalSales
* Modify the BamazonCustomer.js application such that customer transactions update not just the Products inventory but also calculates the total sales from each transaction (quantity \* price). Add the revenue from each transaction to the TotalSales for the related department.
* Create a new Node application called BamazonExecutive.js. Running this application will:
  + List a set of menu options: 1) View Product Sales by Department, 2) Create New Department
  + When an executive enters the View Product Sales option, they should given a summarized table which shows a format like the below:

| **DepartmentID** | **DepartmentName** | **OverHeadCosts** | **ProductSales** | **TotalProfit** |
| --- | --- | --- | --- | --- |
| 01 | Electronics | 10000 | 20000 | 10000 |
| 02 | Clothing | 60000 | 100000 | 40000 |

* + The TotalProfit should be calculated on the fly using the difference between OverheadCosts and ProductSales.TotalProfit should not be stored in any database. You should use a custom alias.
  + If this does not work, then feel free to go back and just add TotalProfit to the Departments table.
  + Hint: You will need to use joins to make this work.
  + Hint: You may need to look into grouping in MySQL.