***Turn the Auto Rental Invoice System into a Database Connected Program***

The previous assignment had you create a program that can calculate the charge for renting a vehicle from the WSU motor pool. In this assignment you take that prior work and extend that program such that it can store records of data, which are invoices for the rental of a WSU vehicle. This is an intermediate step in the functionality of this program.

***Requirements:  
  
Part 1: Make the database table for inserts and create the webpage that can drop the rows into the database table***

1. Add a table to your database called VehicleRentals. Details of each rental invoice will be saved as a new row in this table. Start with a primary key field called RentalID which has the datatype integer. Make the RentalID field a primary key that auto-increments.

2. Add other fields as necessary to record the details of the rental invoice, refer to prior versions of your project with datatables. One of the goals of the project is to total up number of reservations, miles driven, etc by vehicle model (ie van, vs. sedan) so include a VehicleModel field to record the category of vehicle. The scenario is that you are inserting a row of data that records the invoice for the vehicle rental, when it is returned. The dollar invoice is based on the daily cost for the vehicle, mileage charge, and damages.

3. After the table is created, add the webpage so that fields map up to the columns in the database table. You can improve your prior version of the project, but for learning starting a new project is recommended. We learn by typing.

4. Write parameterized SQL code that takes the values from the ASP.NET web page and inserts them into a new row saved into the SQL Server database VehicleRentals.

5. After the record is saved to the database, show the one row of data in a gridview on the ASP.NET webpage. Here is some similar code which can pull the most recent customer added to a customers table.

USE [yourdatabase];

SELECT TOP 1 \*

from [dbo].[ VehicleRentals]

ORDER BY RentalID DESC

**Part 2 - Updating the Vehicle Rental System to Update the Transaction and Master Data Records**

When the business transaction is recorded, it often needs to be updated when new information is collected. Similarly transaction data will be used to update columns in related tables which are hybrid dimension data. For example they have mostly dimension data (vehicle ID, description for example) but also have cumulative measures (running totals and metrics).  
  
For example an insurance employee's performance record is changed to reflect price quotes offered (for example the quota may be 25 price quotes per day), the inventory levels for the stock receive is increased after the received delivery, the data records for the factory machines that completed the production are changed to reflect the throughput, and the inventory for the products shipped to customers is decreased by the amount pick/packed/shipped to the customer.

In the current scenario of a rented vehicle, while the transaction (rental) record is created, or updated, it is also common to update the information for the vehicle being rented (such as perhaps recording the additional miles on the engine - which is useful for auto maintenance, etc.).

In this assignment you add the functionality of updating one rental to close it out, and to update group totals for the vehicle type after a vehicle rental process is completed.

In the previous assignment you created a program that can calculate the price of a vehicle rental (cars, vans, trucks, etc.) and now we save the transaction record into a cloud-based SQL Server database table. This operation adds rows of transaction data to a fact table which are used to store records of transactions with primarily measures (numbers and dates and identifiers, not textual descriptors). While inserting records into fact tables, it is common to update a few columns of measures in the master records that are stored in dimension tables - related to the business process. While fact tables hold mostly numbers, dimension tables (master data) hold mostly descriptors, words used to categorize the facts.

Hybrid Dimension tables (master data) add some measure columns (such as progress to a goal or running totals to the textual descriptor data. It is often common to add some summary measures to the master data in the dimension tables such as to add the quantity in stock to the product record, the total dollar sales to a customer record, the year to date productivity of an employee, and the bank balance of a customer. While the data can be summarized for the master data artifact (product, customer, employee) by analyzing the fact tables, it is also very common to place summary measures in the master data record for quick viewing. We use this approach in the customer-sales demonstration in class and here in then vehicle rentals-vehicle scenario of this assignment. After a sale it is common to update customer records (to show the increased business with that customer), after a work week it is common to update the employee record (to show the number of work processes completed such as the number of purchase orders completed by a buyer), or to update the machine records (to show the results of a recent completed production run).

This project is a blueprint for the foundation of your final project - inserting records into some tables, updating others.

**Update specifics**:

1. In your SQL Server database, using SQL Server Management Studio (SSMS) add a dimension table called VehicleCategory (or similar) with the following fields; ModelID#, VehicleModel, NumberTimesRented, NumberDaysRented, NumberMilesRented, RevenueGenerated.

2. The WSU motor pool rents three types of vehicles a) 4-door sedan, 8 passenger van, and 3/4 ton pick-up truck. In SSMS right click your tables and select edit table and add the 3 rows of data. Place zeros in each of the measure fields (NumberTimesRented, NumberDaysRented, NumberMilesRented, RevenueGenerated).

3. After each rental invoice is completed, you will update one row of the VehicleCategory table such as when a sedan is returned, the measure fields (NumberTimesRented, NumberDaysRented, NumberMilesRented, RevenueGenerated) for the sedans is updated.

In Visual Studio create a ‘called’ procedure such as Private sub UpdateVehicle() that is called from the aforementioned procedure that inserts a row. Add a call to this procedure right after the insert command is executed in the try catch block of your insert command. You do not add a second button click procedure to update the vehicleCategory table.

So an approximation of the code would look like  
  
Private Sub Button1\_click()  
…  
try

If con.state = closed then con.open   
 insertCommand.executenonquery  
 Call UpdateVehicle()   
Catch  
Finally  
 con.close  
end try

Idea for extra functionality to get the full 10 points or to flush out the project

a) here we update totals for the vehicle category. You could create a vehicles dimension table that held the details for each fleet in the vehicle. For example WSU has 20 sedans. Each could have its own master data and other fields that could be updated after each rental return and invoice.  
  
b) if you now have detailed information about each vehicleID (by VIN#) you could have fields such as # miles to oil change, # miles to tune up, etc. to keep track of scheduled maintenance and raise a flag of caution when the vehicle is overdue to have its tires rotated for example.

c) the customers of the WSU motor pool are frequent flyers meaning they rent often, you could add records for these departments and metrics to track how often they rent vehicles, etc.

Use prior assignment instructions for submission. Again add extra functionality to earn a higher grade.