Fall 11-1-2006

Tuition Reduction Incentive Program Application (COMPUSCRIP)

Steve Williams
Dakota State University

DAKOTA STATE UNIVERSITY

MSIS
PROJECT APPROVAL FORM
(Form #3)

Student Name: Steve Williams

Master's Project Title:

TUITION REDUCTION INCENTIVE PROGRAM APPLICATION (COMPUSCRIP)

Signature Approval:

Faculty supervisor: _____ Date: 12/6/2006

Committee member: _____ Date: 12/6/2006

Committee member: _____ Date: 12/6/2006

Original to Office of Graduate Studies and Research
Acid-free copies with written reports to library

12/6/2006

TUITION REDUCTION INCENTIVE PROGRAM APPLICATION (COMPUSCRIP)

A graduate project submitted to Dakota State University in partial fulfillment of the requirements for the degree of

Master of Science

in

Information Systems

November 2006

By
Steve Williams

Project Committee:

Dr. Sreekanth Malladi
Dr. Xinwen Fu
Dr. Joshua Pauli

ACKNOWLEDGMENT

I would like to thank Lisa Vande Kamp and Kristin Mulder with Sioux Falls Christian for providing me assistance in learning the Sioux Falls Christian TRIP program. Their valuable assistance, enthusiasm, and patience helped make this project possible. I hope the resulting CompuSCRIP application will exceed expectations of both Sioux Falls Christian as well as the families that use CompuSCRIP.

I extend my appreciation to Dr. Sreekanth Malladi for his guidance in helping me begin to learn the technology used to build CompuSCRIP. I appreciate Dr. Malladi's encouragement during this process. I also thank Dr. Xinwen Fu and Dr. Josh Pauli for agreeing to serve as committee members for this project.

A special appreciation goes to my incredible daughters for the sacrifices they have made over the years in support of my efforts. I am so grateful to both of them.

Finally, and most importantly, I thank my wife Jenna for her patience, encouragement, and loving support in my pursuit of a Master of Science in Information Systems. It is hard to believe this would have ever been possible without the level of support she has shown me. Thank you Jenna for standing by me and being actively involved and interested in helping me achieve my goal.

ABSTRACT

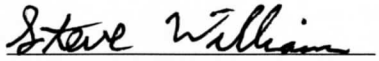
Often, private organizations such as schools, clubs, and non-profits operate on modest budgets in order to keep tuition, dues, or other types of fees associated with their organization affordable for its members. An option to help supplement tuition costs or fees is for the organization to partner with local businesses that will provide a "kick back" to the organization or members for purchasing debit cards from the businesses. The standard/generic term for this type of program is SCRIP. SCRIP is "a Latin term that means anything used instead of money" (United Scrip). This paper will describe a plan to create a web application for Sioux Falls Christian to help manage their SCRIP program. This paper will also compare and contrast the new web-based application created as part of this project with the previous semi-manual process used to manage the SCRIP program.

DECLARATION

I hereby certify that this project constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I declare that the project describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,



<Student name>

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Description of Typical SCRIP/TRIP Programs

Currently many private schools have a program called SCRIP that help parents supplement tuition costs by selling them debit cards from local participating businesses such as grocery stores and restaurants. Businesses agree to sell private schools these debit cards at a discounted price, but still honor the face value for the actual purchase. The difference between the discounted price and the face value of the debit cards is deposited into a SCRIP account for each family (maintained by the school). The balances in a family's individual SCRIP account accumulate over time as they continue to order/purchase debit cards from participating businesses. Periodically, perhaps four times per year, the balance of these individual SCRIP accounts is credited to the respective family's school tuition account, thereby reducing the total amount of tuition owed by parents. Grandparents, aunts, uncles, etc. can also participate in the SCRIP program and donate their earned funds to a child's tuition account. The benefit to the participating businesses of course is increased exposure in the community and potentially increased sales. SCRIP is typically a successful program for schools for two reasons. First, "the participating retailers are stores where your supporters already shop all the time" (Miller, Sullivan). Second, "supporters can help your school without spending an extra dime." (Miller, Sullivan).

An example of how profitable a SCRIP program can be is demonstrated by the following example. Consider a family that spends about $100 weekly on groceries or $5000 yearly. If the nonprofit gets a 5% discount on debit cards from that grocery store and then sells it at face value to the family, the non-profit would make $250 for the year. (Scrip.Net).

The project described in this document entails creating a web-based tool, called CompuSCRIP, that will be created for and utilized by Sioux Falls Christian (SFC). It will give families the ability to place SCRIP orders on-line. (SCRIP is the term typically used to describe this type of tuition assistance program, but SFC calls their SCRIP program "TRIP" which stands for Tuition Reduction Incentive Program.). CompuSCRIP will also allow participating families the ability to keep track of historical data, such as their total purchases and total TRIP savings during a given time period. In turn, the school will be able to use the same web application to compile and process the TRIP orders in a more automated environment. Using CompuSCRIP, the school will also have access to historical reports that will help them better manage the TRIP program as well as a systemic process for communicating with their TRIP participants.

Background of the Problem

SFC provides education for pre-school through 12th grade. There are 350 families whose children attend the school. One campus, in the center of Sioux Falls, services pre-school through 4th grade while the other campus, on the south end of town, provides for 5th through 12th grade. There is one part time staff person that works as the TRIP administrator for SFC. The program has been very successful, and since its creation, it has grown to over 600 participating families.

The TRIP administrator currently utilizes a client-based software called MANNAger. It is installed on a single PC and is specifically designed to help the administrator manage TRIP family accounts.

Statement of Problem

Because of the popularity and growth of the program, the existing semi-manual process, using MANNAger client-based software, no longer meets the needs of the school. The existing software does not leverage the benefits of an Internet-based application which can help decentralize the workload and improve communication. The current process requires parents to fill out paper order forms in order to request gift cards through the TRIP program. In turn they of course need to physically deliver the form, with payment, to the Administrator each week. The administrator then has to key each of these orders into the MANNAger software in order to total the number of orders for each participating merchant.

Objectives of the Project

This project entails creating a web application called CompuSCRIP. Customers will be able to log into CompuSCRIP at www.CompuSCRIP.org via their Web browser and an Internet connection. Once logged in, they will be able to make their gift card selections on-line and submit them to the TRIP administrator electronically. Likewise, the administrator will see the submitted orders in CompuScrip and be able to process them accordingly. Both the administrator and customers will have access to standard reports that will help them track current and historical orders made using CompuSCRIP.

The creation of the CompuSCRIP application will help streamline the existing process and facilitate better communication between TRIP customers and the TRIP administrator.

Project Scope

The CompuSCRIP application will be developed using the Asp.Net framework with a MS SQL Server backend database that will be relational and normalized appropriately. The application code and database will be housed with a service provider by the name of Lunarpages. Lunarpages has been selected because of their affordability, stability, security, and performance. For more information on Lunarpages, the company website can be found at http://www.lunarpages.com/index.php. Sioux Falls Christian will then place a link to the CompuSCRIP application on their website www.SiouxFallsChristian.org.

CHAPTER 2

TRIP PROCESS OVERVIEW (PRIOR TO COMPUSCRIP)

Phase 1 (Creating the Weekly Order Form)

After the administrator creates a family account in the MANNAger software, the family can begin placing TRIP orders. This is primarily a manual process. Each Monday, the TRIP administrator uses the MANNAger software to print out address-sized labels from the family account database created within the MANNAger software. The label includes the account number, family's name, date, current account balance, previous balance, a "Hold for pickup" message (this indicates the parent will come to the campus to pick up the SCRIP order in person), and the name of the campus where the parent will pick up the order. The two campuses are Sneve (preschool through 3rd grade) or Charger (4^{th} through 12^{th} grade). The label with this information is placed on a business-sized envelope and physically handed to the parent. The envelope has roughly 100 business names preprinted on the outside of the envelope that have agreed to participate in TRIP. Next to each business name are the denomination options of the debit cards provided by that business.

Phase 2 (Placing the TRIP Order)

The family takes the envelope home. During the weekend, they will determine which merchants they wish to purchase debit cards from and the specific number and denomination amounts they wish to purchase. They will check the appropriate boxes on

the envelope, enter the corresponding denomination amounts, add the total amount, and finally place a check for the total inside the envelope (see Figure 1). The following Monday, the parent will take the completed order form envelope and drop it off at the Administration office of either campus, Sneve or Charger. If the parent is dropping the order off at the Sneve campus the deadline is on Monday by 1:00. If the order is brought to the Charger campus, it is due by 9:30 AM on Tuesday.

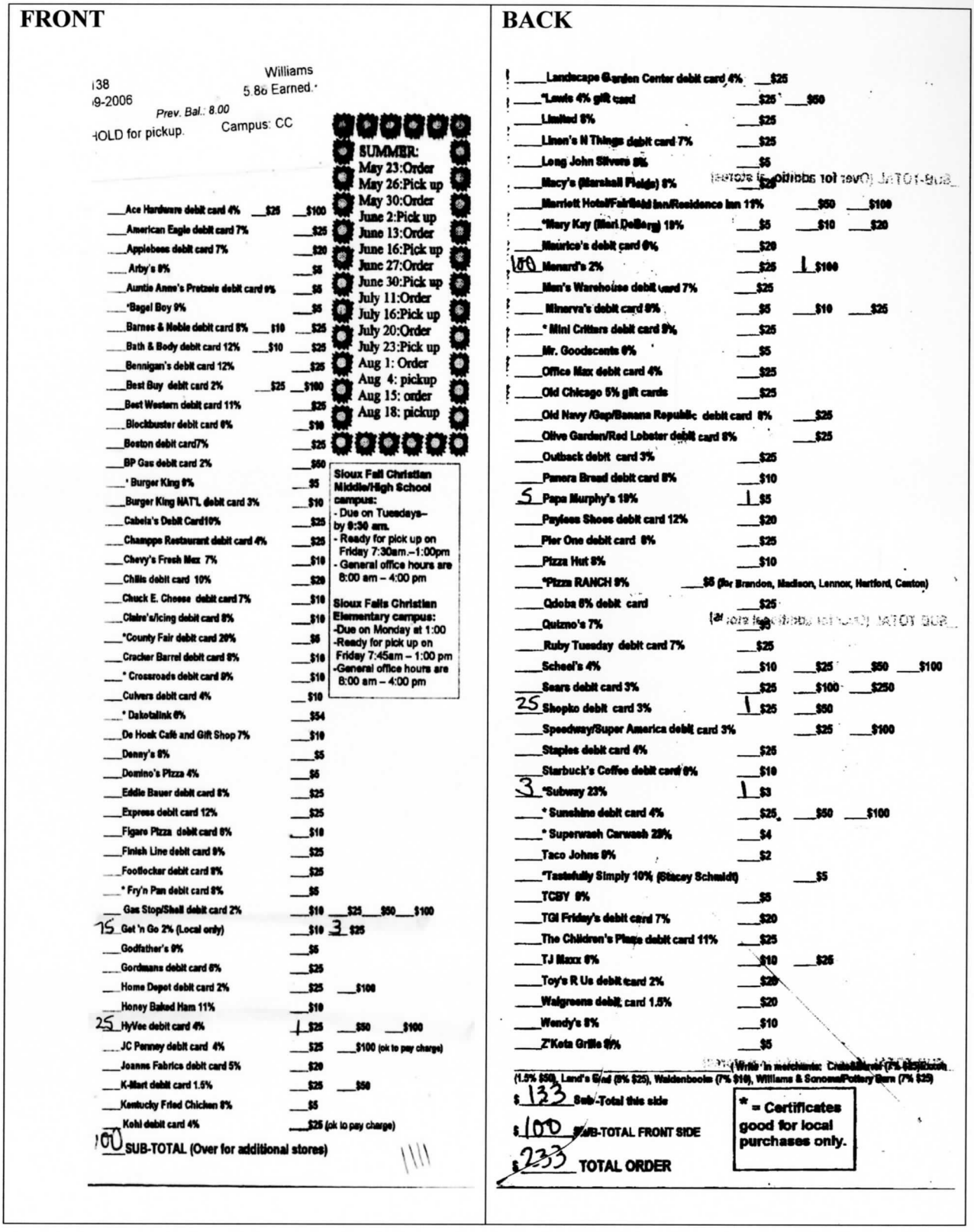


Figure 1. TRIP Order Envelope

Phase 3 (Processing TRIP Orders)

On Monday afternoon, the TRIP administrator will take each envelope, re-calculate the totals, and ensure the check has been filled out correctly. At that time, she will manually take each family's order for the week and individually enter the number and their choice of denominations for each business into the family account in the MANNAger software. On an average, there are 128 envelopes/orders to process each week and take an average of four hours to enter the orders into the MANNAger application. When all the orders have been manually entered, the TRIP administrator will print a report, through MANNAger, that displays the total number of gift cards to be ordered from each of the businesses. She will then call each business and tell them the total number and denominations of gift cards that are needed for that week. On Tuesday and Wednesday, she will drive to each of the businesses and pick up that week's orders. Because the debit cards are "same as cash", it is far too risky to ask the businesses to physically mail them to the school each week.

Phase 4 (Preparation for Order Pick-up)

On Thursdays, the order form envelopes that were turned in on Monday are filled with the respective debit cards for each of the businesses that were selected on the envelope. On Friday mornings, the administrator will print new address-sized labels (described previously in Phase 1) for each family, attach them to more pre-printed order forms envelopes, and paperclip the new week's envelope to the previous week's envelope that contains the ordered debit cards. Of course, the final step is when the parent goes to school on Friday to pick up their gift card orders for that week along with the attached

"clean" order form envelope for any orders they may wish to place the following Monday. On the following Monday, the process begins again with Phase 1.

Review of Process Gaps Prior to Use of CompuSCRIP

1. The administrator must spend an average of four hours per week manually entering the week's orders from families.
2. The administrator must make weekly calls to the businesses to give them the orders for the week.
3. At least 100 new TRIP order envelopes need to be pre-printed each week to give to TRIP families.
4. Labels generated by the MANNAger software must be manually placed on each pre-printed envelope.
5. It is difficult to add businesses or additional information on the order form due to lack of space.
6. In order to fit all current business names and information on the order form, the font needs to be very small which can be difficult to read for some TRIP participants.
7. TRIP users need to spend time multiplying/adding their sub totals/grand totals on the order envelope, which creates more opportunity for human error.
8. The availability for TRIP customers to see their transaction history is limited and reliant upon the administrator to provide them the information verbally or by email.

CHAPTER 3

SYSTEM DESIGN (COMPUSCRIP)

CompuSCRIP - Overview

This chapter describes the new method of processing TRIP orders as well as provides information regarding CompuSCRIP's user interface and functionality.

As with any application being developed, it is important to create software that is easy to learn regardless of the skill level of the end user.

> "The interface **design** (Fraternali, 1999) characteristic of a web application can be represented by its: *structure* describes the organization of the information space presented by a web application; *navigation* enables moving through the information space presented by the web application; *presentation* describes the interaction styles used to present the information and behavior of the web application. Usability is affected by the type of task and its complexity, the interaction style used to perform the actions and the design of the interface. All these characteristics directly affect the learnability, efficiency of use and subject satisfaction usability attributes of web application's usability." (Bruno, Tam, and Thorn November 2005)

The objective of this project is to provide not only added features and functionalities over the existing TRIP process, but to also provide an easy to use and easy to learn application. It is intended that the tool will reduce and eventually eliminate the need for the administrator to manually enter individual TRIP orders into the existing client software tool (MANNAger). Additional features will be added to CompuSCRIP beyond this Senior project phase, with the intent of eventually eliminating the need for the current MANNAger software. Those additional features will take an extended period of time to plan and develop so are considered out of scope for this Senior project.

Membership and Roles

"Along with the increased importance of web applications, the negative impact of security flaws in such applications has grown as well. Vulnerabilities that may lead to the compromise of sensitive information are being reported continuously, and the costs of the resulting damages are increasing. The main reasons for this phenomenon are time and financial constraints, limited programming skills, and lack of security awareness on part of the developers." (Jovanovic, Kruegel, Kirda, June 2006).

In order to put a level of control in place to protect data within the CompuSCRIP application, the forms mode of authentication within the ASP.Net framework is utilized. In addition, corresponding roles have been created to determine whether the individual logged into CompuSCRIP has access to view specific forms and pages. The roles also determine whether a user is allowed to make updates to certain pages or whether they will be view-only. The following role types are utilized to provide for authentication:

Customer Level 1- Allows the user view-only access to one or two pages until the TRIP administrator has approved their request to be added as a fully participating customer in the TRIP program. This level is considered a "Guest Access" role.

Customer Level 2- Allows a customer full access to place TRIP orders, make modifications to their family profile, and run any reports specific to their account.

Administrator Level 1 - Allows user view-only access to one or two pages until the Administrator Level 3 has approved their request to be added to a higher level administrative role. This level is considered an "Administrator Guest Access" role. The initial administrator will be approved and added by the CompuSCRIP Web Master.

Administrator Level 2 – Allows the user to modify their CompuSCRIP profile page, access any administrator pages in view-only mode and will allow them to run administrator-specific reports. This type of role could be granted for staff members of the school who are interested in viewing reports only (i.e. bookkeeper, accountant, etc.).

Administrator Level 3 – Full access to manage any profile on the system for the school, ability to send communications or set alerts on the CompuSCRIP application for customers, and run any detailed or summary transaction reports at the Customer or Administrator level.

Customer Enrollment Functionality

The new process involves prospective TRIP program customers to create an account in CompuSCRIP. The customer will navigate to the application by opening an Internet Explorer browser window and entering the URL www.CompuSCRIP.org. The customer will select the Login hyperlink from the page which will drive them to the Login form (figure 2).

Figure 2. Log In Form

The user will select the "Create New User" hyperlink located on the form, which will in turn, display a new page (shown in Figure 3) for them to enter additional user information.

Figure 3. Sign Up for Your New Account

The User Name field is a free form field allowing the user to type in a User Name of their choice followed by a corresponding password. If the user enters a User Name already assigned to a different user in the database, an error message is displayed asking them to choose a different User Name.

The E-mail field is also a free form field allowing the user to enter in an email address of their choice. If the user attempts to enter an email address already assigned to an existing CompuSCRIP user account, an error message will be displayed asking for a different email address.

The Security Question is a free form field where the user may type a question that only they would be able to answer. The Security Answer field is also a free form field

that allows the user to type the answer to the Security Question. These fields will of course be utilized to allow the customer to retrieve their User Name and Password in the event they have forgotten it.

After all fields have been successfully entered, the customer will select the "Create User" button. If a new user account was successfully created and stored in the database, the user will receive a new page (shown in Figure 4) indicating the account was successfully created.

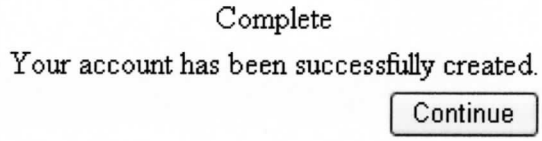


Figure 4. Account Created Message

In the initial phase, the user will automatically be assigned a Customer Level 2 role allowing them to make online orders and generate standard account activity reports. In the initial phase, the Administrator will work with the developer to add a TRIP account number to the user's account. In this way, when the user makes on-line orders, the funds will be recorded toward the proper TRIP account number associated with Sioux Falls Christian. In later phases, there will be customer profile screens that the TRIP administrator will utilize to maintain user account data (account number, contact information, etc.).

After successfully creating the account, the user will be logged into the system and, based on the Customer role, will automatically be navigated to the Customer Home page which will be described and viewed later in this paper.

TRIP Administrator Access

In this first phase of the project, the TRIP administrator also utilizes the Customer Enrollment functionality described in the previous section to create the initial Administrator account. The CompuSCRIP application manager will assign this new account to the Level 3 Administrator role described in the Membership and Roles section of this document. This role allows the Administrator full access to all of the Customer and Administrator screens within CompuSCRIP. In later phases of the project, screens and functionality will be created to allow the CompuSCRIP application manager to approve or deny a request for a new Administrator account as well as assign specific administrator roles (Level 1, 2, or 3).

The Administrator, because of the assigned role, is systemically driven to the Sponsor home page when signing into the CompuSCRIP application.

Sponsor Home Page

The CompuSCRIP administrator will maintain the content of the Sponsor Home page (shown in Figure 5). It will be viewable only by users with Administrator Level roles. This screen will be used by the CompuSCRIP developer to communicate with the TRIP Administrator. It can be utilized to notify the TRIP Administrator of upcoming CompuSCRIP system changes such as new pages/functionality, planned outages for application maintenance, etc.

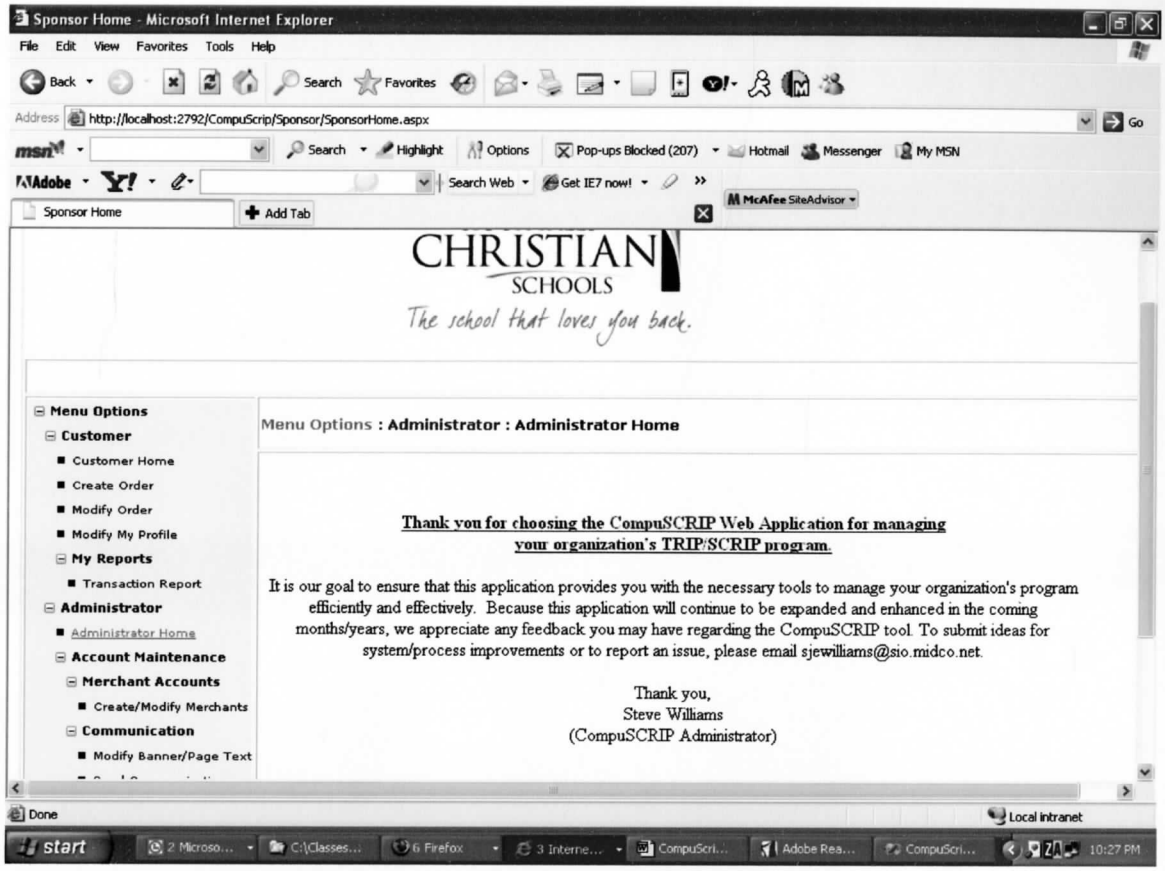


Figure 5. Administrator Home Page

Merchant Account Administration

The ability to create and modify merchant accounts is provided for in a single aspx page that utilizes the robust .Net Formview component.

In order for customers to begin ordering gift cards from merchants through CompuSCRIP, the TRIP Administrator must create the merchant accounts within the application. After signing in, the TRIP Administrator will select the Create/Modify Merchants option in the navigation pane under the Account Maintenance/Merchant

Accounts parent menu. This link will display the Merchant Administration page (shown in Figure 6).

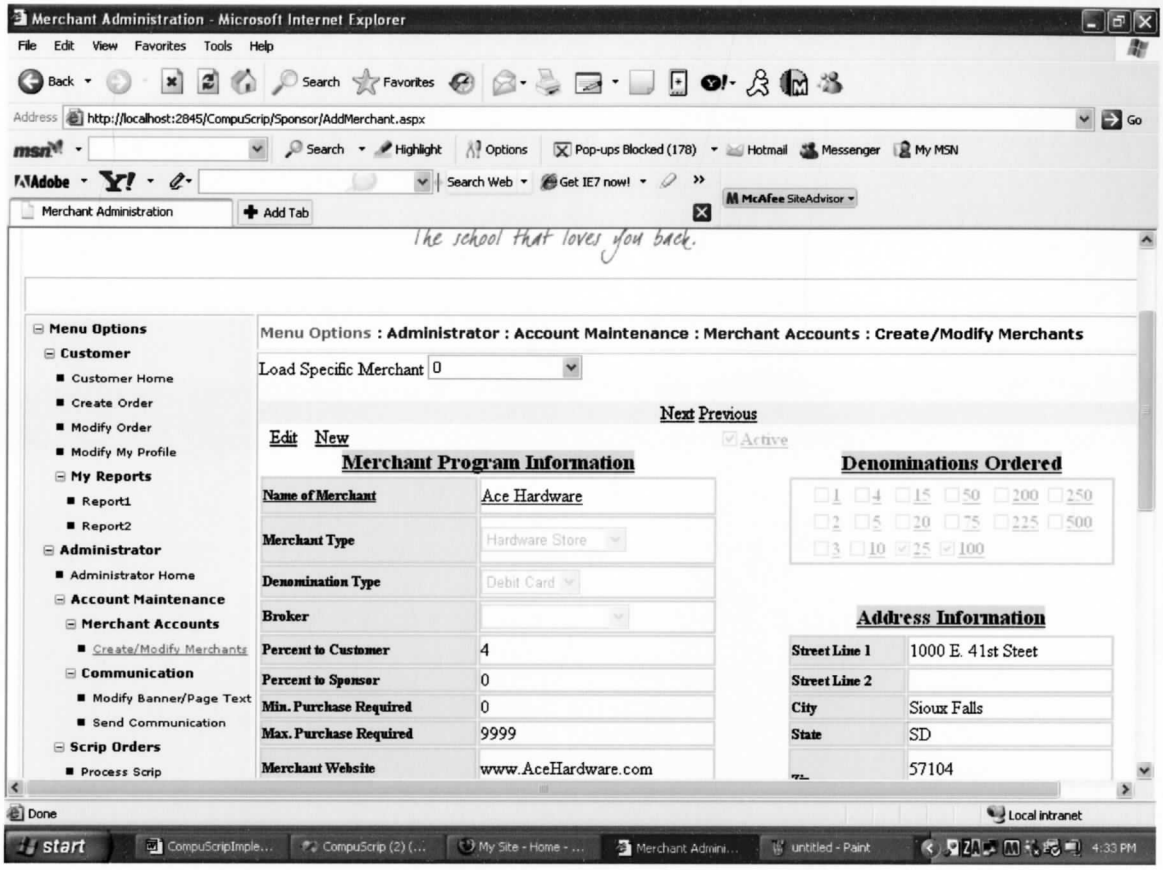


Figure 6. Create/Modify Merchants

To create a new merchant account, the Administrator will select the "New" hyperlink in the upper left portion of the screen. This will refresh the page and open the fields for data entry as seen in Figure 7 and Figure 8.

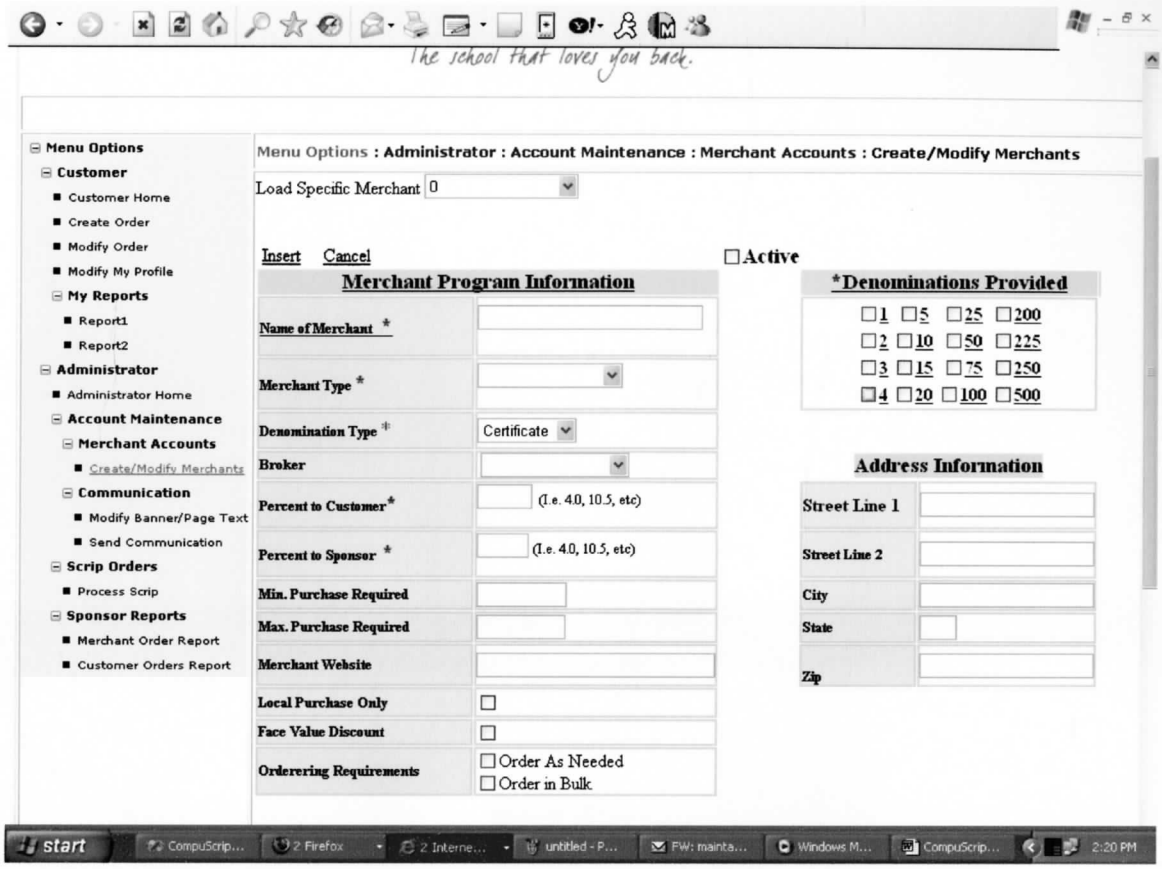


Figure 7. Create a Merchant Screen (Top Section of Screen)

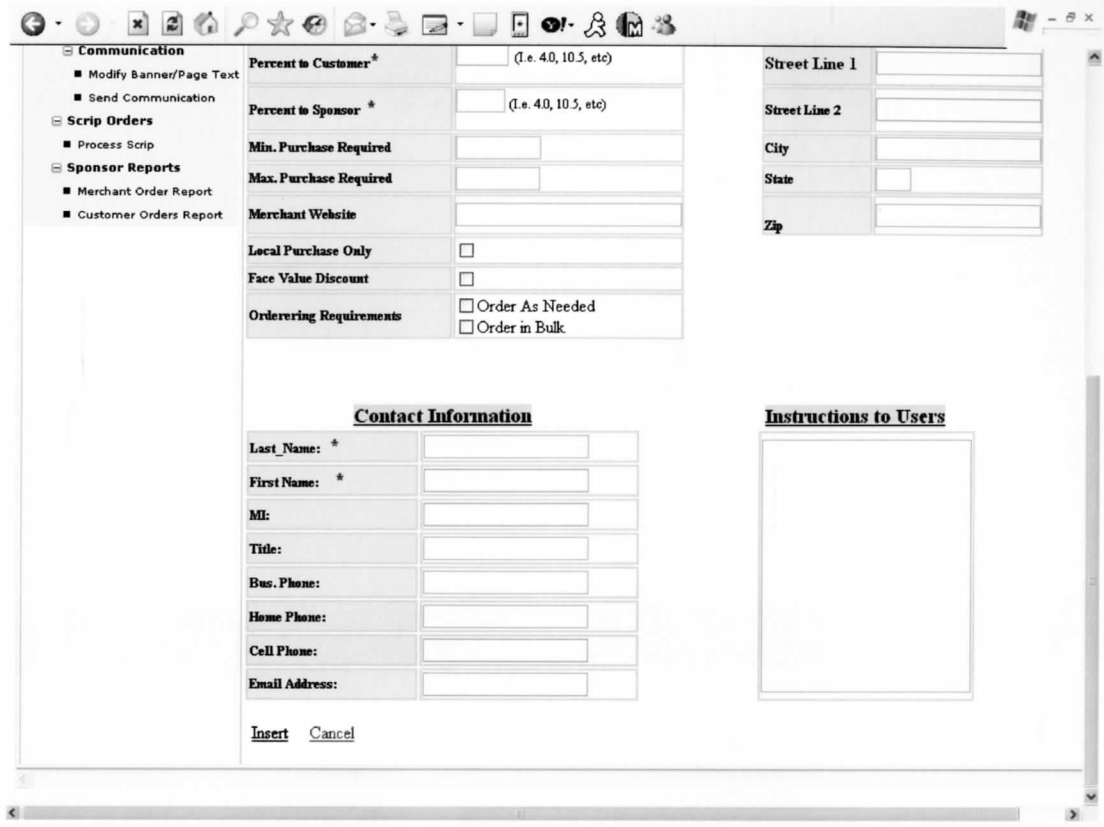


Figure 8. Create a Merchant Screen (Bottom Section of Screen)

Merchant Program Information Section

The Name of Merchant field is self-explanatory. The maximum length of this free form text field is 75 characters and requires a value to be entered.

The Merchant Type is a drop down list that contains various one-word descriptions of the type of business being added. Values in the drop down would be consistent with the types of categories listed in the yellow pages of a phone book (Automotive, Restaurant, Grocery, etc.).

The Denomination Type drop down list allows the Administrator to indicate whether the denominations provided by the merchant are in the form of a paper gift certificate or a plastic debit card.

The Broker drop down list currently has two available values from which to choose. The user may leave the value as blank or choose "Great Lakes Scrip Center", also known as GLSC. Some larger merchants use a third party brokerage organization like GLSC to manage their gift card distribution process. Schools/organizations that wish to purchase gift cards must purchase them from this third party vendor. The vendor then mails the cards for that business directly to the requestor. The primary vendor used by many larger organizations is Great Lakes Scrip based out of Michigan.

> "For retailers, GLSC offers access to loyal, motivated shoppers across the United States. GLSC non-profit organizations are eager to spend their shopping dollars with retailers who participate in scrip programs. GLSC works with participating retailers to maximize their scrip marketing efforts, and coordinate them with the retailer's overall merchandising strategies." http://www.glscrip.com/aboutus/index.aspx

The Broker drop down list currently only offers valid values of blank/null or Great Lakes Scrip. If additional brokerage firms need to be added in the future, it will be a simple addition to a table within the database by the developer.

The Percent to Customer field is a required freeform text field where the Administrator can identify the percentage that the business has agreed to contribute to the customer's tuition account if a gift card is purchased from them.

The Percent to Sponsor field is also a required free-form text field that allows the Administrator to identify a percentage that the merchant has agreed to contribute to the school/organization sponsoring the TRIP/Scrip program. For example, if a business is

willing to provide a ten percent "kick back", the sponsoring organization may decide to deduct two percent of that amount to cover administrative costs and leave the remaining eight percent in a customer's tuition account. Currently SFC allows the full percentage to go to the customer's tuition account. With this scenario, SFC would simply enter a value of zero in this field.

The next two fields are Min Purchase Required and Max Purchase Required. Some merchants have a policy indicating they will only sell gift cards in batches when a minimum dollar amount is met and may not sell beyond a maximum amount in one order. These two free form text fields are optional, however, they allow the Administrator to identify if the merchant has a minimum/maximum purchase policy.

The Merchant Website is an optional free-form text box field in which the Administrator can enter the merchant's URL. If a URL is entered, the CompuSCRIP application will identify this on any page the name of the merchant is displayed and convert it to a hyperlink. This will provide the customers placing orders the ability to quickly navigate to the merchant's website from any screen where the business name is reflected. If a URL is not entered for the merchant, then anytime the merchant name is listed on a page, it will be in plain text and will not be linked to a website.

The Local Purchase Only is a checkbox field. If checked, it indicates that the gift cards provided by the merchant can only be used locally. If this field is not checked, then the customer can use the gift cards at any of the merchant's locations nationwide.

The Face Value Discount is a checkbox field. Some merchants sell gift cards to organizations at a discounted price. For example, if an organization wanted $100 worth of debit cards and the merchant provides a ten percent donation back to the organization,

they will simply sell the $100 worth of gift cards to the organization for $90 instead of $100. Some merchants handle discounting in a slightly different way. Instead of selling the gift cards for a discounted price, they sell the cards at the full face value and then give the organization the extra percentage through additional gift cards. For example, the merchant will sell a $100 gift card to the organization at the face value of $100. The ten percent donation will be fulfilled through an extra $10 gift card. This Face Value Discount field allows the administrator to identify which of these two typical approaches a particular merchant has agreed to take.

Denominations Provided Section

This section allows the Administrator to select typical gift card denomination amounts provided by merchants. A single checkbox list control is utilized to display the denomination amounts which are sourced from a table within the backend database. If additional amounts would be requested in the future, it would only entail adding an entry to the specific table where these standard denomination values are housed.

This Address Information section is of course utilized for address details specific to the merchant. It consists of the optional freeform text fields of Street Line1, Street Line 2, City, State, and Zip.

The Contact Information section of the screen consists of freeform text fields that allow the Administrator to record the information corresponding to a particular merchant's point of contact. These fields include: Last Name, First Name, MI, Business Phone, Home Phone, Cell Phone, and Email. The Last Name and First Name fields in this section are required while the remaining fields are optional.

Instructions to Users Section

The final section on the Merchant Administration screen is the Instructions to Users. This is a freeform scrolling text box allowing the administrator to notate any specific instructions to users of the merchant's gift cards. For instance, some merchants have multiple locations within the area and perhaps only allow gift cards to be used at certain locations. Perhaps the merchants also want to let the customers know that they can use their gift cards to pay account balances maintained by the merchant. For example, Menards may allow customers to use their gift cards to make payments on their Menards credit card. The Instructions to Users field can be utilized to reflect these types of scenarios. Whatever comments are entered into this field will be viewable to the customer when they place their gift card orders.

The final important data entry field on the screen is the checkbox labeled "Active". To make this merchant available for customers to place on-line orders against, the Administrator will place a check mark in this field. If the Administrator wishes to remove the merchant from the Order screen, then it will remain unchecked.

After entering in the appropriate merchant information on this screen, the administrator will select the hyperlink field labeled "Insert" which will save each of the data fields on the screen to the database. If the administrator selects the Cancel hyperlink, the screen will not save any values to the database and return the user to the View only version of the Merchant Administration form.

After the Administrator has entered the list of merchants that wish to participate in the TRIP program, she can simply select the Next or Previous hyperlink on the

Merchant Administration page to view each merchant's specific account page. Each merchant is displayed in alphabetical ascending order when selecting the hyperlink labeled "Next". If the Administrator wishes to view an account for a specific merchant, she can select the name of the merchant from the Load Specific Merchant drop-down list box at the top of the screen (shown in Figure 9). If the Administrator wishes to edit an existing merchant account, she can simply select the hyperlink labeled "Edit" to open up all of the data entry fields for editing.

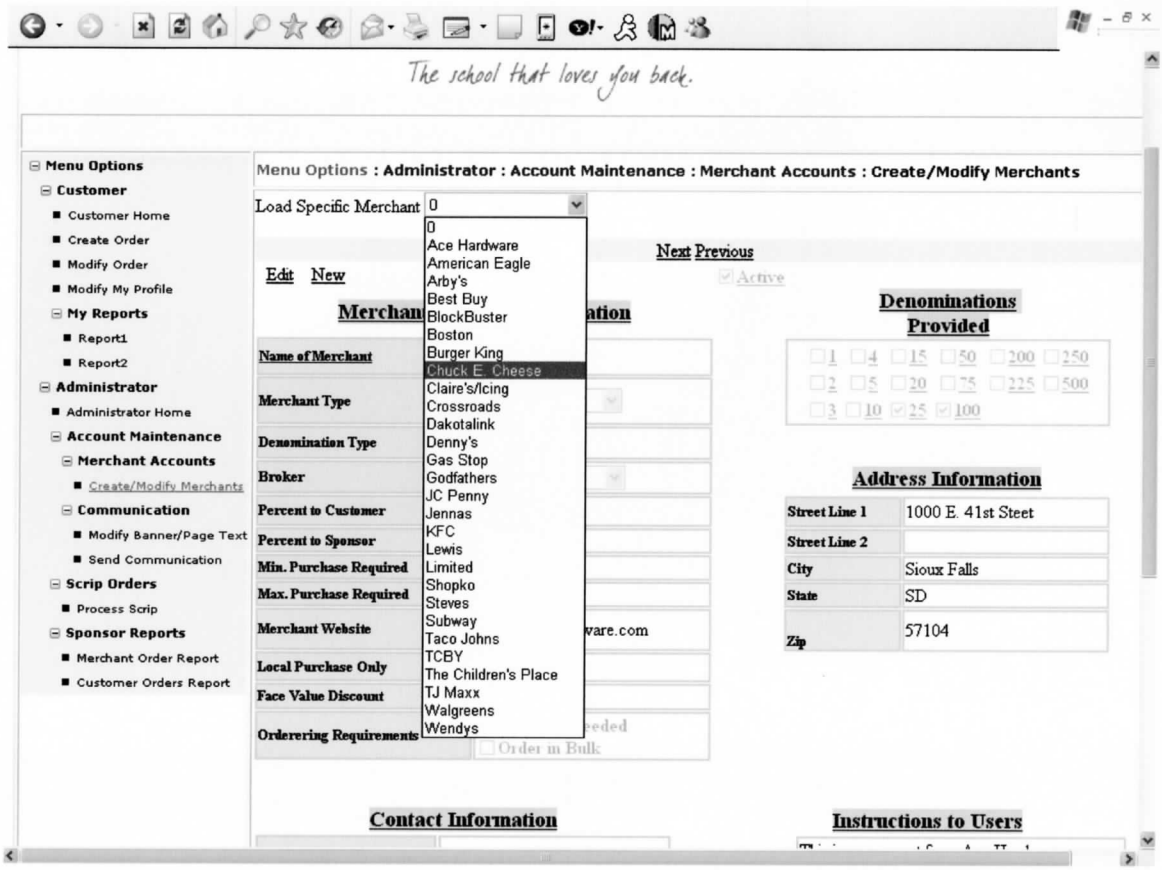


Figure 9. Load Specific Merchant List Box

Creating an Order

Now that customers have created their accounts within CompuSCRIP, and the Administrator has added a list of merchants to the system, the ordering process can begin. To create an order, the customer will sign into CompuSCRIP. After signing into CompuScrip, they will be automatically navigated to the Customer Home page based on their Customer Profile setting. The Customer Home page (shown in Figure 10) will display any messages from the TRIP administrator. It can include such information as office hours, order deadlines, upcoming TRIP events, etc. The administrator will also be able to alert customers to higher priority information by having a text message added to the scrolling text bar at the top of the Customer Home page. This scrolling text bar only appears on the Home page.

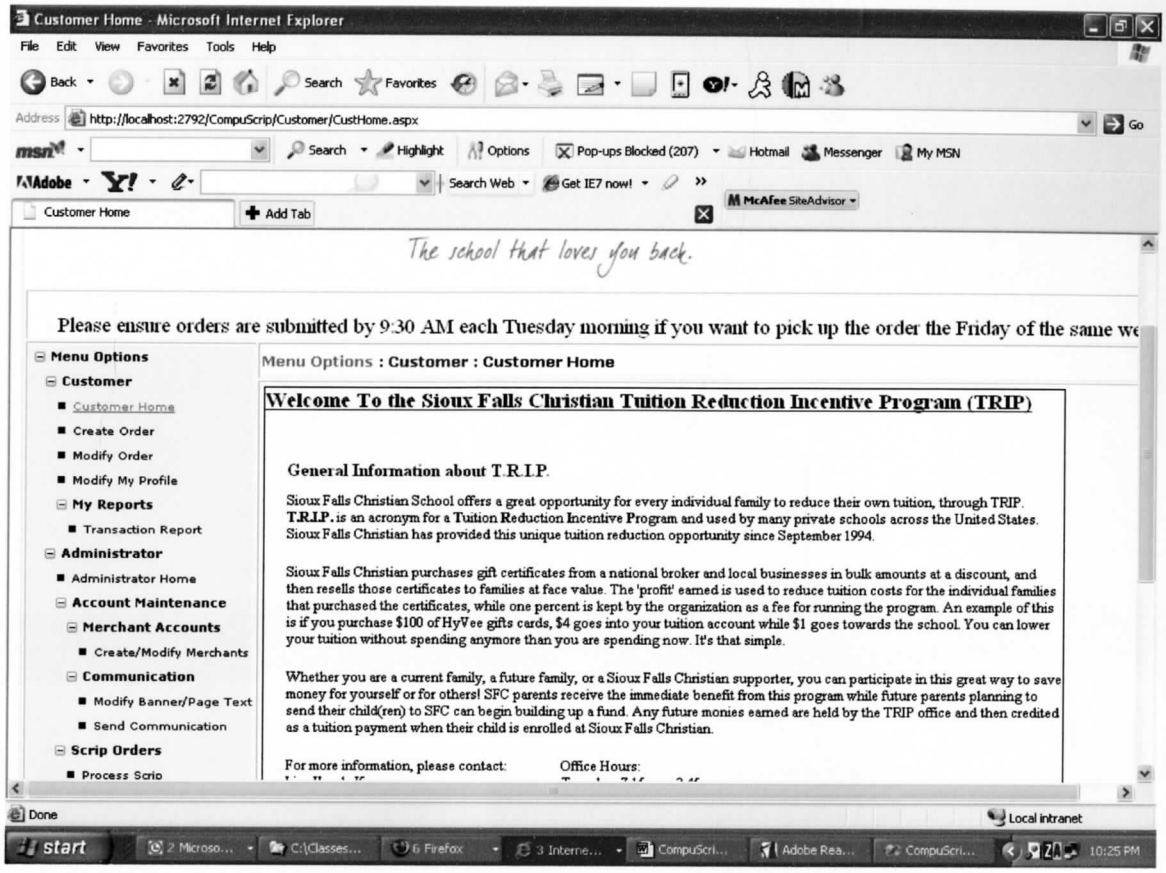


Figure 10. Customer Home Page

After signing in, the customer will select the Create Order hyperlink under the Customer folder in the navigation pane. This option will navigate the user to the Create Order screen seen in Figure 11. This screen shows a list of merchants that are participating in the TRIP program. Merchants are based on the Administrator creating an account for them and selecting the check box labeled "Active" on the Merchant Administration screen.

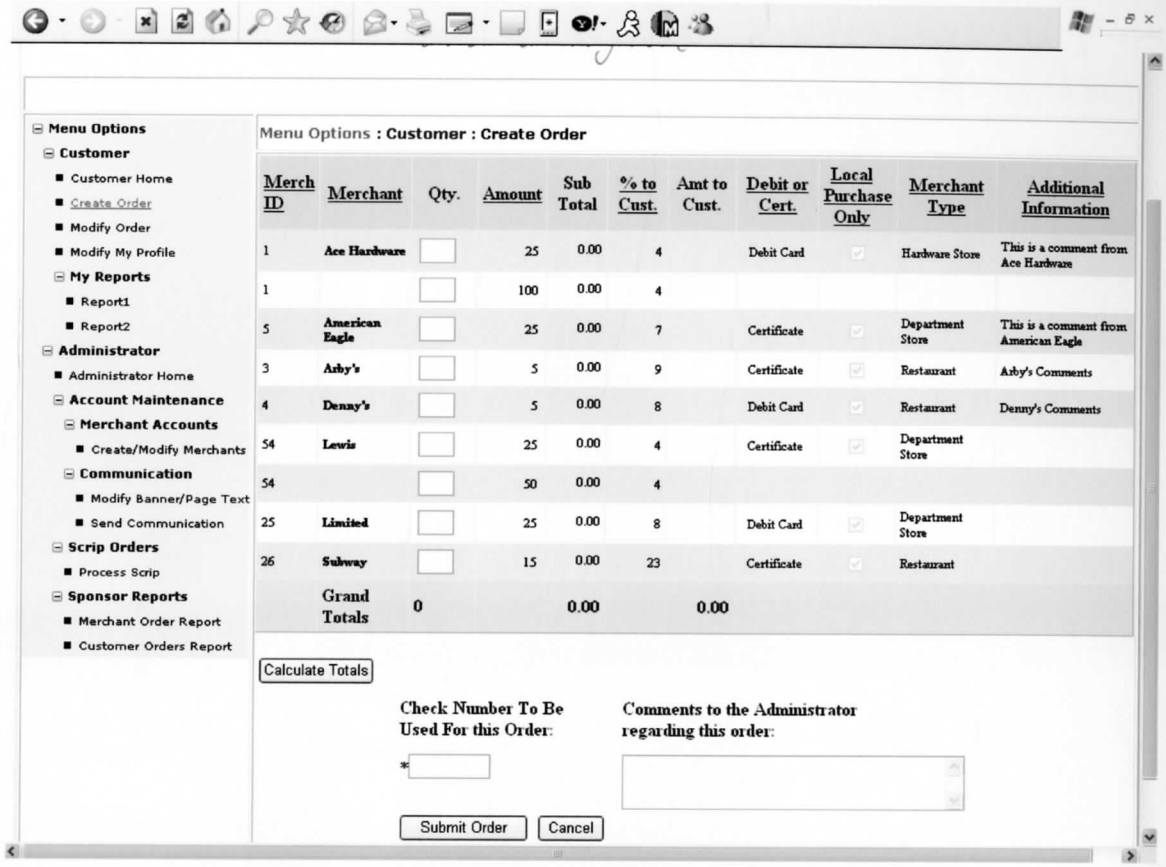


Figure 11. Create Order Screen

Description of Fields on Create Order Screen

The Merch ID field is a read-only field and is of no importance to the customer. The number displayed in the field is actually the primary key value for the business in the database and may be useful for future troubleshooting activities. It will give the developer of CompuSCRIP an understanding of what record in the database is causing potential issues.

The Merchant Name field is a read only field. When the screen is first loaded, the list is sorted by this field in ascending order. If there is a URL listed on the merchant's

account screen, this value would display in the form of a hyperlink. If the hyperlink is selected, a new browser window is opened and takes the user to the merchant's website.

The Qty. field is a text field that accepts only integer values. The customer will enter the number of specific denomination gift cards. Notice in Figure 11 that if the merchant offers multiple denominations, there is a separate row for those additional denomination amounts. On a side note, Figure11 shows only seven merchants listed on the screen. The smaller number of merchants is only for the purpose of this paper and being able to display the entire screen in one image. In reality, this screen would display as many merchants as the administrator has activated. It could contain a scrolling list of 100, 200, or more merchants.

The Sub Total field is a read-only and is calculated based on the value entered in the Qty field multiplied by the denomination amount.

The % to Cust. field is read only and displays the percentage the merchant will contribute back to the customer's tuition account after the order is processed. This percentage is derived from the value entered by the Administrator in the Percent to Customer field on the Create/Modify Merchant screen.

The Amt to Cust. is a calculated read-only field. The value that displays in this field is programmatically calculated using the Sub Total value multiplied by the % to Cust value. This is the dollar amount that will be deposited in the customer's tuition account after the order is processed.

The Debit or Cert. field is read-only and indicates whether the merchant provides the denomination amounts in the form of paper gift certificates or plastic debit cards.

The Local Purchase Only, Merchant Type, and Additional Information are all read-only fields that are directly sourced from the values entered by the Administrator on the Create/Modify Merchant screen.

Notice that some of the heading labels are underlined. This indicates that if the user clicks on that heading label, the screen will be refreshed, and the records will be sorted in ascending order by that particular field. If clicked a second time, it resorts the records on the screen in descending order.

After the user has entered in the quantity of each denomination they wish to order from any of the merchants, they can select the Calculate Totals button at the end of the merchant list. This button will calculate the Sub Total and the Amt. to Cust. values for each row where the user entered a number in the Quantity field. It will then add all values in each column and place the Grand Totals at the bottom of the page (shown in Figure 12). The Calculate Totals button does not save/submit the order. It simply calculates the screen totals so the user can determine the amount due for the order and view the amount they can expect to be deposited in their family's tuition account. In Figure 12 the user has entered a quantity for some of the merchants. They wish to order one $25 and two $100 debit cards from Ace Hardware, two Arby's gift certificates, and one $25 gift certificate from Lewis. Notice the Sub Total and Amt to Cust. columns have been calculated based on the user's input. The Grand Total row at the bottom of the list indicates the user can expect to receive a total of six certificates/debit cards totaling $260.00. Based on the percentages donated by each merchant for those orders, the user can expect to receive a credit to their child's tuition account of $10.90.

If the user is satisfied with the entries made, he or she will enter the check number he or she intends to use to pay for the order, as well as any comments associated with the order. The Aministrator will see the comments when she is processing the orders. Examples of comments could include situations where the customer wishes to indicate that they will be late in picking up the order, they will have someone else pick up the order, a suggestion on a merchant they would like to have added, etc. The comment field could also be used to submit a question to the Administrator about the order. In short, the field is primarily used as a simple communication method.

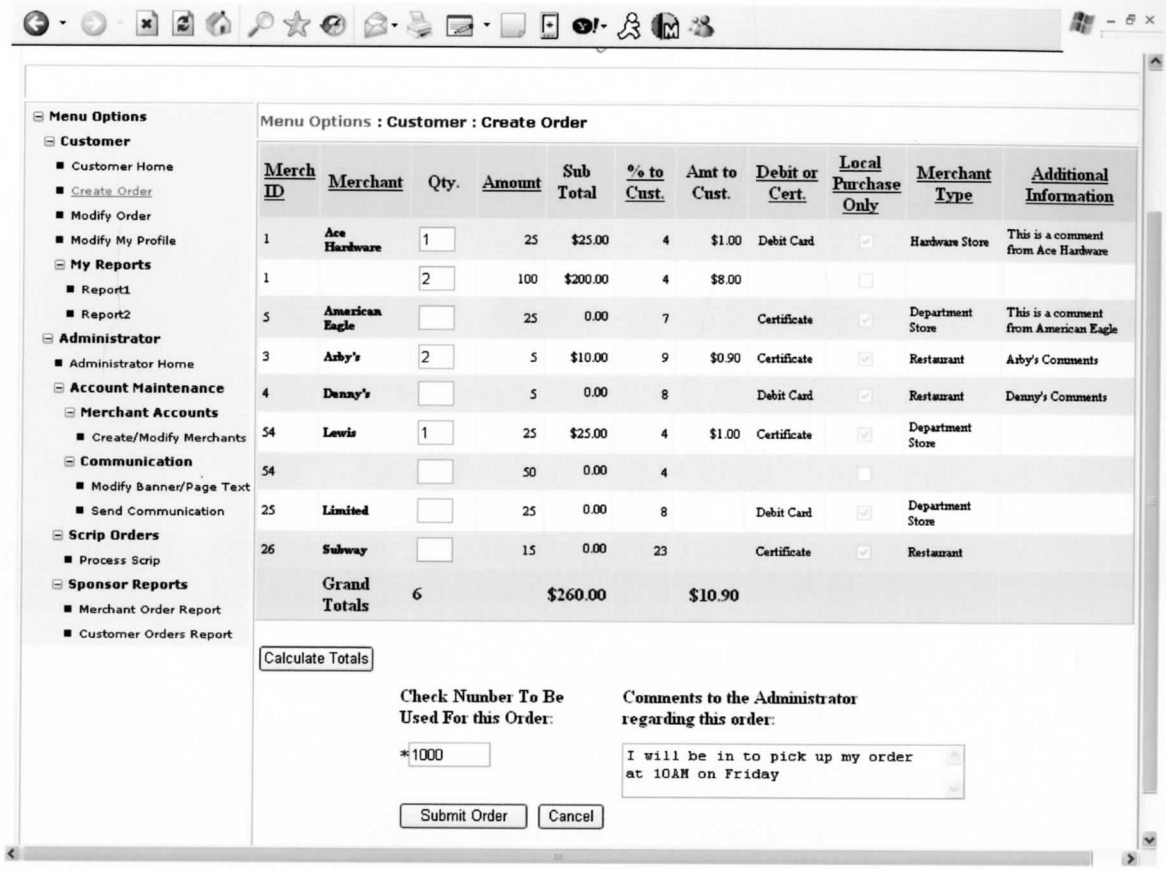


Figure 12. Create Order Screen (With Data Entry Values)

After the user has made their selections, entered the check number they intend to use, and typed any comments they might have, they can select the Submit Order button. Upon submitting the order, a message box will be displayed to the user with an Order Confirmation number as shown in Figure 13. Users are asked to take note of the confirmation number and, if paying by check, record the confirmation number in the memo field of their check. At this point the order will be saved to the database for processing by the Administrator.

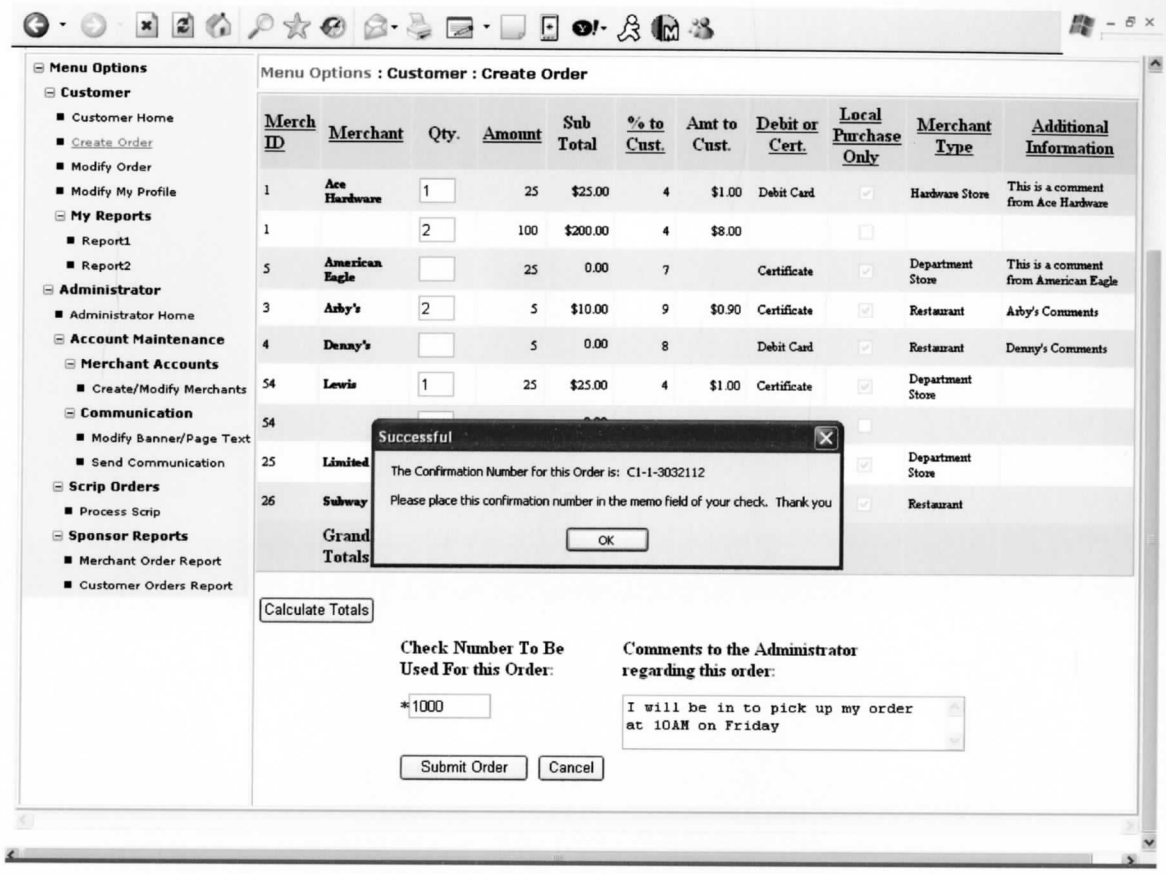


Figure 13. Create Order Confirmation Message

Processing Customer Orders (Administrator)

Customers who have placed on-line orders within CompuSCRIP will either mail their checks to the Administrator or physically take the check to the Administrator's office. The checks for that week's orders are to be given to the Administrator by 9:30 AM Tuesday morning.

After the Tuesday 9:30 AM deadline, the Administrator will begin to process the week's orders by signing into the CompuSCRIP application and navigating to the Administrator/ Scrip Orders/ Process Scrip page (shown in Figure 14). This screen will

display the list of orders submitted by customers for that week. In the example shown in Figure 14, there are only four customers that have submitted orders for the week. In reality, this screen would probably contain 50, 100, or more orders from customers.

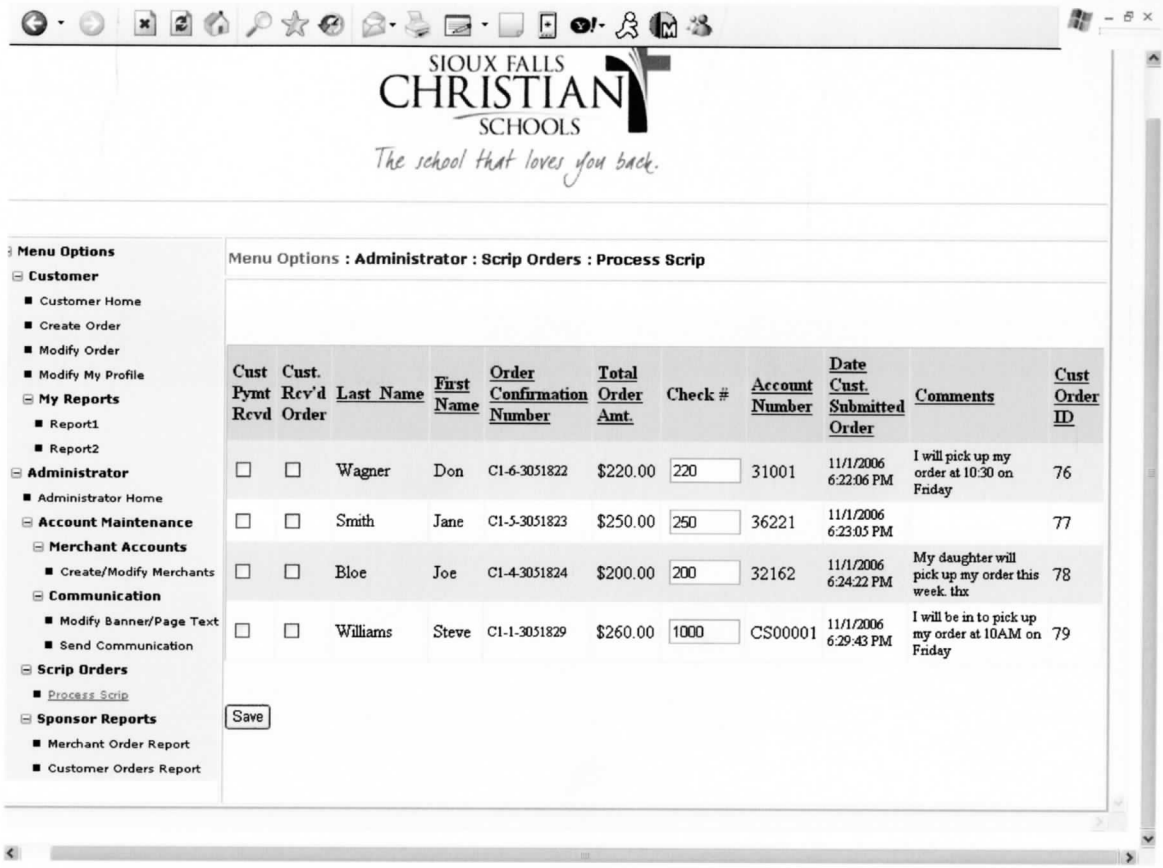


Figure 14. Process Scrip Screen

Process Scrip Fields

There are only three data entry fields on the Process Scrip screen. They are the Cust Pymt Rcvd field, Cust Rcv'd Order field and the Check # field. The remaining fields are read only fields and are self explanatory based on the column headings.

After navigating to this screen, the Administrator will take inventory of the checks received from the users for that week's orders. If the administrator has a physical check for an order, she will select the Cust Pymt Rcvd checkbox next to the corresponding order as shown in the example in Figure 15.

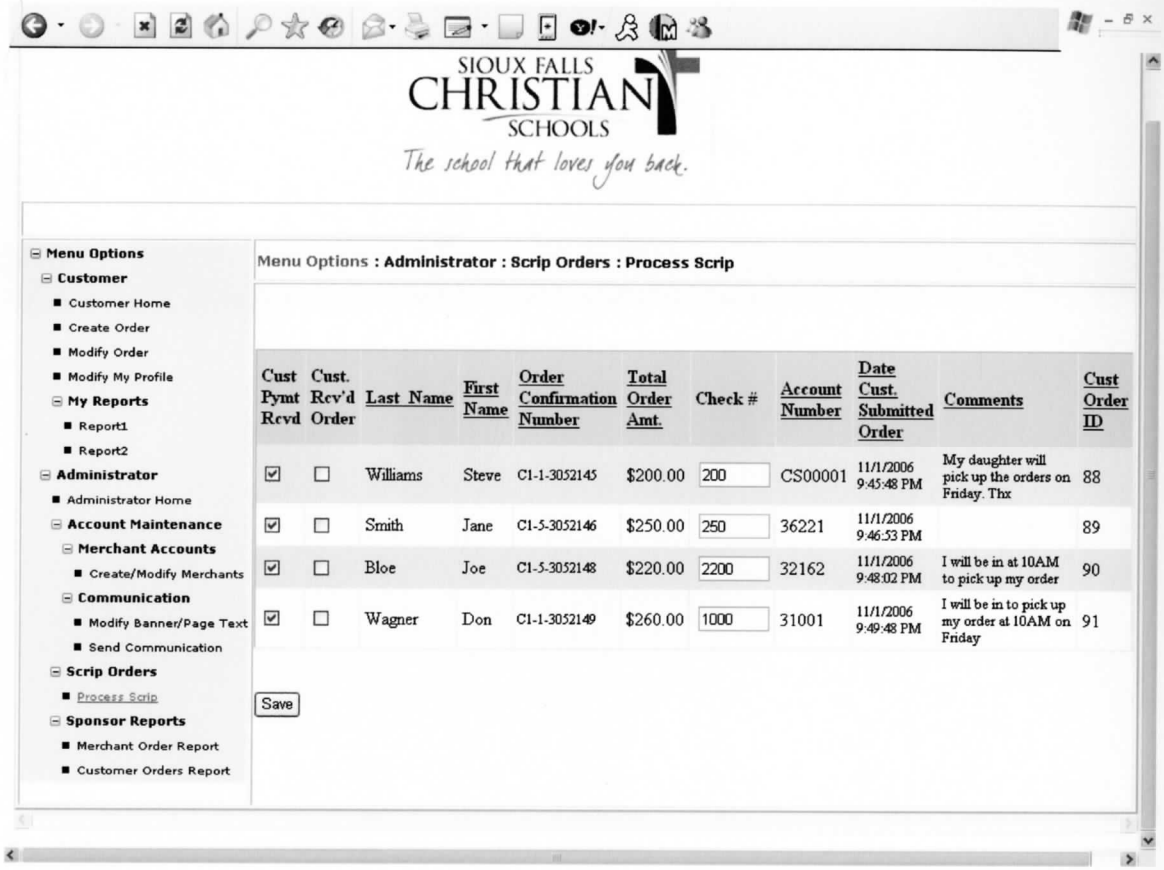


Figure 15. Process Scrip Screen (Payments Received)

If for whatever reason, the customer ends up using a different check number than the one keyed when placing the original order on-line, the Administrator can simply type over the check number value in the Check # field.

After the Administrator has taken inventory of the physical checks received, she is ready to run a report summarizing how many gift cards of each denomination need to be ordered from individual merchants.

Merchant Order Report

To identify the quantity and denominations ordered and paid for, the Administrator will navigate to Administrator/Sponsor Reports and select the Merchant Order Report hyperlink. When selecting this link, the report seen in Figure 16 will be displayed. It will summarize the list of orders the Administrator has marked on the Process Scrip screen as having been paid by the customer.

From the report shown in Figure 16, the Administrator will know to order two $25 dollar gift certificates and three $100 gift certificates from Ace Hardware. The total order for Ace comes to $350. The Administrator will use this report to either call or email the gift card requests to each merchant listed on the report. This report summarizes the quantity ordered, total dollar amount ordered of each denomination, and a final total for each merchant. At the end of the report, it will display a Grand Total for all the merchants.

11/2/2006 **Merchant Order Report As Of 11/2/2006**

		Quantity	Denomination	
Ace Hardware				
		2	25	50.00
		3	100	300.00
	Merchant Totals	5		350.00
American Eagle				
		4	25	100.00
	Merchant Totals	9		100.00
Arby's				
		2	5	10.00
	Merchant Totals	11		10.00
Denny's				
		12	5	60.00
	Merchant Totals	23		60.00
Lewis				
		4	25	100.00
		4	50	200.00
	Merchant Totals	31		300.00
Limited				
		2	25	50.00
	Merchant Totals	33		50.00
Subway				
		4	15	60.00
	Merchant Totals	37		60.00
			Grand Total	**930.00**

1

Figure 16. Merchant Order Report

Fulfilling Customer Orders

After running the Merchant Order Report, the Administrator will call or email the orders to each merchant listed on the report. This will typically happen on Tuesday afternoon. On Wednesday, the Administrator will drive to each merchant to pick up the gift cards. On Thursday, the Administrator will print a Customer Order Report in order to take inventory and sort the gift cards based on individual customer orders. The report is found in CompuSCRIP under Administrator/Sponsor Reports/Customer Orders Report. Selecting this link from the navigation menu will source information from the database to produce the report shown in Figure 17. From this report, the Administrator will determine how many specific gift cards each customer ordered. To begin the process, the Administrator will spread the entire physical inventory of gift cards on a large table; they will be sorted and grouped by denomination and by merchant. She will then take a few moments to cut apart each customer section in the Customer Orders report. After this is completed, she will take an individual customer order, select the appropriate gift cards from the corresponding merchants, and place the gift cards in an envelope. The section of the report for that customer will also be placed inside. The customer's name is listed on the outside of the envelope. The Administrator continues this process until all customer orders for that week have been placed in each customer's envelope along with their order receipt.

Customer Orders As Of 11/2/2006

	Last Name	First Name	Business Name	Quantity	Denomination	Sub Total
31001	Wagner	Don	Ace Hardware	1	25	$ 25.00
	Wagner	Don	Ace Hardware	2	100	$ 200.00
	Wagner	Don	Arby's	2	5	$ 10.00
	Wagner	Don	Lewis	1	25	$ 25.00
			Total For Customer	**6**		**$ 260.00**
32162	Last Name	First Name	Business Name	Quantity	Denomination	Sub Total
	Bice	Joe	American Eagle	2	25	$ 50.00
	Bice	Joe	Lewis	1	25	$ 25.00
	Bice	Joe	Lewis	2	50	$ 100.00
	Bice	Joe	Subway	3	15	$ 45.00
			Total For Customer	**14**		**$ 220.00**
33138	Last Name	First Name	Business Name	Quantity	Denomination	Sub Total
	Williams	Steve	American Eagle	2	25	$ 50.00
	Williams	Steve	Denny's	10	5	$ 50.00
	Williams	Steve	Lewis	2	25	$ 50.00
	Williams	Steve	Lewis	1	50	$ 50.00
			Total For Customer	**29**		**$ 200.00**
36221	Last Name	First Name	Business Name	Quantity	Denomination	Sub Total
	Smith	Jane	Ace Hardware	1	25	$ 25.00
	Smith	Jane	Ace Hardware	1	100	$ 100.00
	Smith	Jane	Denny's	2	5	$ 10.00
	Smith	Jane	Lewis	1	50	$ 50.00
	Smith	Jane	Limited	2	25	$ 50.00
	Smith	Jane	Subway	1	15	$ 15.00
			Total For Customer	**37**		**$ 250.00**
				Grand Total		**$ 930.00**

Page 1 of 1

Figure 17. Customer Order Report

Customer Pick-Up

On Friday morning, customers go to the Administrator's office to pick up their gift card order for that week. The Administrator is signed into CompuSCRIP and is on the Process Scrip screen. After handing the envelope filled with gift cards to the respective parent, the Administrator will select the check box in the second column of the screen labeled Cust Rcv'd Order (shown in figure 18). In this example, Williams, Smith, and Bloe have all picked up their gift card orders. The Administrator will select the Save button periodically to record to the database who has picked up their orders. If the Administrator has checked both check boxes (Cust Pymt Rcvd and Cust. Rcv'd Order), then the associated record will not display on the report the next time the screen is refreshed.

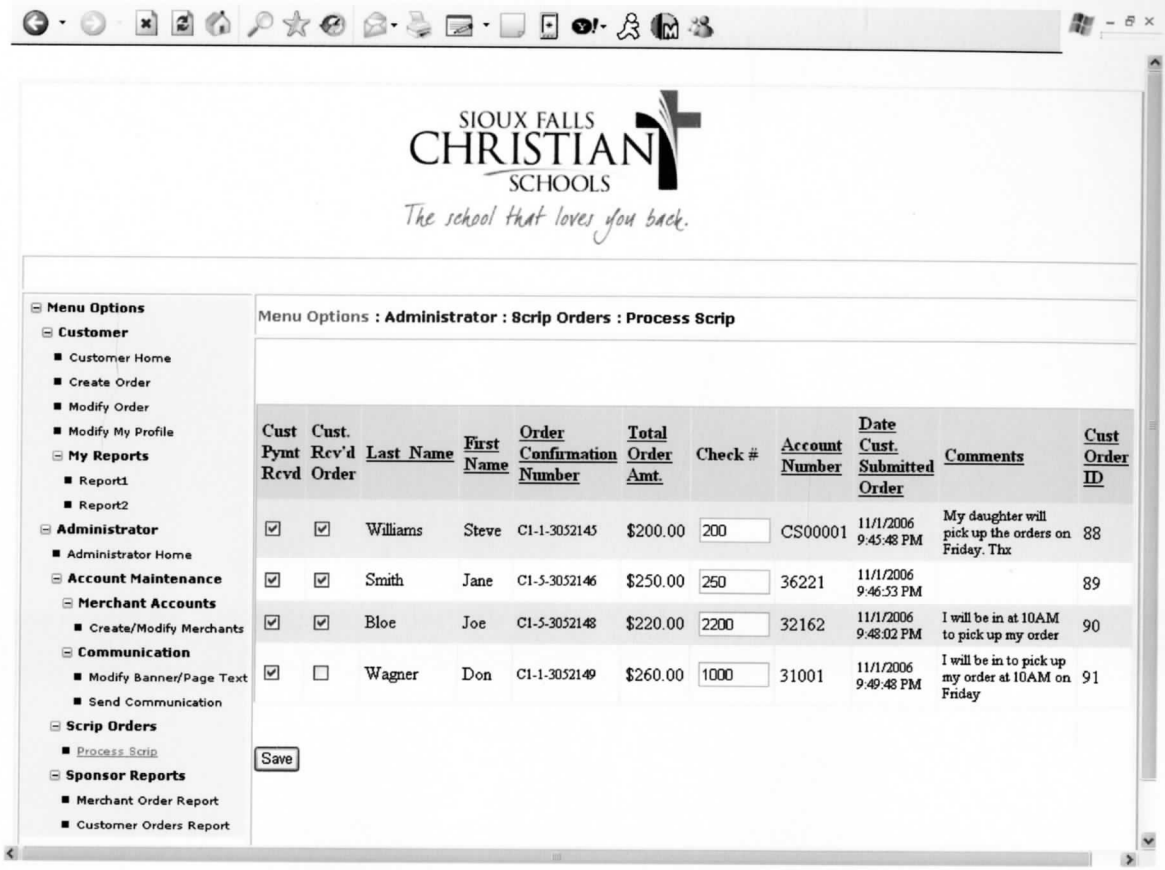


Figure 18. Process Scrip Screen (Customer Pick-up)

Database Design

The user interfaces, described in the sections above, were created in Asp.net 2.0 using SQL Server as the back-end database. A great deal of time and effort was spent creating a relational database that would prevent duplication of data. There are currently 30+ relational tables that were developed in support of the new CompSCRIP application. Not all of the tables are utilized with this first phase of the application, but creating them and including them in the initial design will provide for easier implementation of future functionality. For a high level view of the database see figure 19 below.

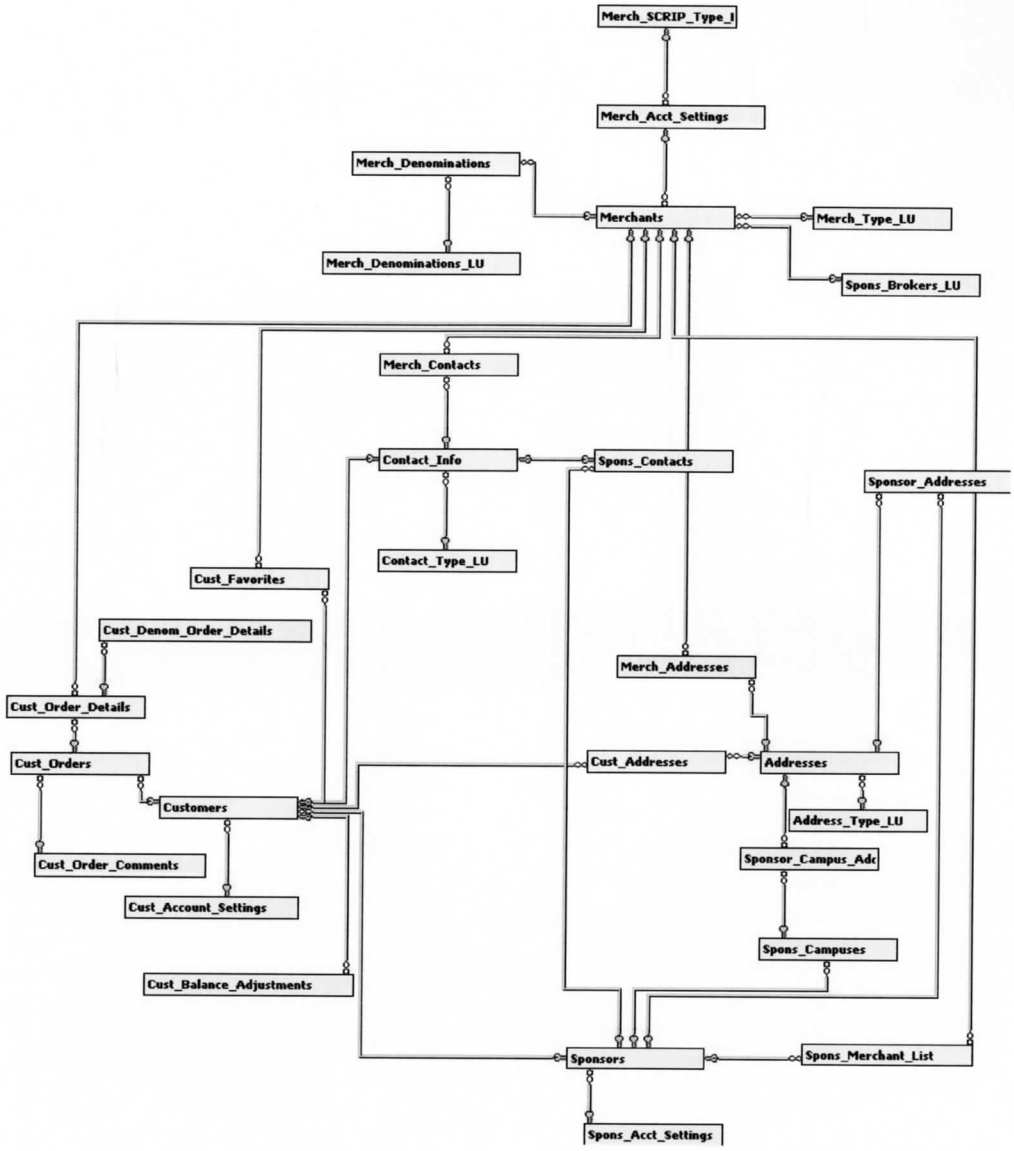


Figure 19. Database Diagram

In addition, most of the sourcing and updating of data for each of the pages is performed using SQL Server stored procedures that interact with the table structure shown above. (Appendix A contains further details regarding the table structures and stored procedure logic.)

CHAPTER 4

CASE STUDY

The initial phases of CompuSCRIP have been presented to Kristin Mulder, the SFC Information Technology Manager and Lisa Vande Kamp, the SFC TRIP Administrator. Some initial system testing has been completed with this initial version of CompuSCRIP. Mrs. Mulder and Mrs. Vande Kamp are enthused about opening the application to the entire TRIP customer base.

Although further coordination will need to take place, CompuSCRIP is on target for expanded testing in January and February of 2007. This expanded testing will include ten users, which will begin to use CompuSCRIP in the production environment. In March or April of 2007 the application will be opened to all participants in the TRIP program.

Although the first phase of this project will provide for a limited number of users to make orders on-line, there will need to be further development in order to make it feasible to bring a larger population onto the system. The enrollment process where users create their own user accounts and submit them to the TRIP coordinator for approval will need to be completed. Currently the developer of CompuSCRIP is required to make some manual entries in order to allow a new user to place their TRIP orders, which is not a viable long-term option.

Additionally, pages will need to be created where the TRIP administrator will be able to update contents of the Customer Home page. Currently the CompuSCRIP developer would need to hardcode the text directly into the aspx page.

Further development of CompuSCRIP will reduce and eventually eliminate time spent by the TRIP administrator keying orders from hardcopy order forms into separate software, like MANNAger.

CHAPTER 5

CONCLUSIONS

"While the impressive returns from scrip are very real, there's still no such thing as completely pain-free fundraising. Where scrip gets high grades for its earning potential and for the fact that parents don't have to spend an extra penny to help your school (they don't have to buy or pay more; they just have to buy differently), successful scrip volunteers do spend a lot of time on their programs" (Miller, Sullivan). Because fund raising can be challenging and SCRIP management is typically a time consuming job, CompuSCRIP has been created to alleviate some of this workload.

Initial indications show that CompuSCRIP will be a successful solution in automating a number of manual tasks by both the TRIP Administrator and the TRIP program participants.

Although the development of the application was a bigger effort and took longer than anticipated it was a valuable learning experience. The delays encountered in development of the application were primarily due to the lack of experience in the Asp.net and SQL Server environments. Future phases of the application should be easier and quicker to develop, based on the knowledge and experience gained through phase 1.

REFERENCES

United Scrip, Retrieved July 1, 2006 from http://www.unitedscrip.com/aboutscrip.html

Miller, Alicia; Sullivan, Tim; (Date Unknown), *The Fuss about Scrip*, PTO Today, Retrieved July 2, 2006 from http://www.ptotoday.com/0301scrip.html

Scrip.Net, Retrieved June 30, 2006 from http://scrip.net/faqs.htm#faq2

Great Lakes Scrip Center, Retrieved November 2, 2006, from http://www.glscrip.com/aboutus/index.aspx

Jovanovic, Nenad; Kruegel, Christopher; Kirda, Engin. (2006). *Precise Alias Analysis for Static Detection of Web Application Vulnerabilities,* Communications of the ACM

Bruno, Vince; Tam, Audrey; Thorn, James. (2005). *Characteristics of Web Applications That Affect Usability: A Review.* Communications of the ACM

APPENDICES

APPENDIX A: SYSTEM TECHNICAL DOCUMENTATION

Table 1. SQL: Create Tables Script

```
USE [compu32_CompuSCRIP]
GO
/****** Object:  Table [dbo].[Sponsors]    Script Date: 11/05/2006
15:26:49 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Sponsors](
      [Sponsor_ID] [int] IDENTITY(1,1) NOT NULL,
      [Spons_Acct_Settings_ID] [int] NULL DEFAULT ((0)),
      [Spons_Org_Name] [nvarchar](100) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL,
      [Spons_Acct_Num] [nvarchar](100) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL,
 CONSTRAINT [aaaaaSponsors_PK] PRIMARY KEY NONCLUSTERED
(
      [Sponsor_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Sponsors]  WITH CHECK ADD  CONSTRAINT
[FK_Sponsors_Spons_Acct_Settings] FOREIGN KEY([Spons_Acct_Settings_ID])
REFERENCES [dbo].[Spons_Acct_Settings] ([Spons_Acct_Settings_ID])
GO
ALTER TABLE [dbo].[Sponsors] CHECK CONSTRAINT
[FK_Sponsors_Spons_Acct_Settings]
/* End
Sponsor**********************************************************/


USE [compu32_CompuSCRIP]
GO
/****** Object:  Table [dbo].[Customers]    Script Date: 11/05/2006
15:27:59 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Customers](
      [Cust_ID] [int] IDENTITY(1,1) NOT NULL,
```

```
      [Contact_Info_ID] [int] NOT NULL CONSTRAINT
[DF__Temporary__Conta__1ADEEA9C]  DEFAULT ((0)),
      [Account_Settings_ID] [int] NOT NULL CONSTRAINT
[DF__Temporary__Accou__1BD30ED5]  DEFAULT ((0)),
      [Comp_AcctNum] [nvarchar](50) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL,
      [Sponsor_ID] [int] NOT NULL CONSTRAINT
[DF__Temporary__Spons__1CC7330E]  DEFAULT ((0)),
      [Cust_Enabled] [bit] NOT NULL CONSTRAINT
[DF__Temporary__Cust___1DBB5747]  DEFAULT ((0)),
      [Pending_Approval] [bit] NULL,
      [Acct_Num_w_Sponsor] [nvarchar](50) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL,
 CONSTRAINT [aaaaaCustomers_PK] PRIMARY KEY NONCLUSTERED
(
      [Cust_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Customers]  WITH CHECK ADD  CONSTRAINT
[fk_Customers_Contact_Info] FOREIGN KEY([Contact_Info_ID])
REFERENCES [dbo].[Contact_Info] ([Contact_Info_ID])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[Customers] CHECK CONSTRAINT
[fk_Customers_Contact_Info]
GO
ALTER TABLE [dbo].[Customers]  WITH CHECK ADD  CONSTRAINT
[fk_Customers_Cust_Acct_Settings] FOREIGN KEY([Account_Settings_ID])
REFERENCES [dbo].[Cust_Account_Settings] ([Account_Settings_ID])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[Customers] CHECK CONSTRAINT
[fk_Customers_Cust_Acct_Settings]
GO
ALTER TABLE [dbo].[Customers]  WITH CHECK ADD  CONSTRAINT
[FK_Customers_Sponsors] FOREIGN KEY([Sponsor_ID])
REFERENCES [dbo].[Sponsors] ([Sponsor_ID])
GO
ALTER TABLE [dbo].[Customers] CHECK CONSTRAINT [FK_Customers_Sponsors]
/*End Customer ************************************************/


/****** Object:  Table [dbo].[Address_Type_LU]    Script Date:
09/14/2006 20:14:48 ******/
USE [compu32_CompuSCRIP]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Address_Type_LU](
      [Address_Type_ID] [int] IDENTITY(1,1) NOT NULL,
```

```
      [Address_Type_Desc] [varchar](50) COLLATE
SQL_Latin1_General_CP1_CI_AS NOT NULL,
 CONSTRAINT [PK_Address_Type_LU] PRIMARY KEY CLUSTERED
(
      [Address_Type_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
SET ANSI_PADDING OFF

/********End Address_Type_LU  ****************************************/


USE [compu32_CompuSCRIP]
GO
/****** Object:  Table [dbo].[Addresses]    Script Date: 11/05/2006
15:29:33 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Addresses](
      [Address_ID] [int] IDENTITY(1,1) NOT NULL,
      [Street_Line_1] [nvarchar](50) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL,
      [Street_Line_2] [nvarchar](50) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL,
      [City] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
      [State] [nvarchar](2) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
      [Zip] [nvarchar](10) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
      [Address_Type] [int] NULL CONSTRAINT
[DF__Temporary__Addre__22951AFD]  DEFAULT ((0)),
 CONSTRAINT [aaaaaAddresses_PK] PRIMARY KEY NONCLUSTERED
(
      [Address_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Addresses]  WITH NOCHECK ADD  CONSTRAINT
[FK_Addresses_Address_Type_LU] FOREIGN KEY([Address_Type])
REFERENCES [dbo].[Address_Type_LU] ([Address_Type_ID])
GO
ALTER TABLE [dbo].[Addresses] CHECK CONSTRAINT
[FK_Addresses_Address_Type_LU]
/* End Addresses ************************************************/


USE [compu32_CompuSCRIP]
GO
/****** Object:  Table [dbo].[Cust_Addresses]    Script Date:
11/05/2006 15:30:29 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE TABLE [dbo].[Cust_Addresses](
      [Cust_Address_ID] [int] IDENTITY(1,1) NOT NULL,
      [Cust_ID] [int] NULL DEFAULT ((0)),
      [Address_ID] [int] NULL DEFAULT ((0)),
 CONSTRAINT [aaaaaCust_Addresses_PK] PRIMARY KEY NONCLUSTERED
(
      [Cust_Address_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Cust_Addresses]  WITH CHECK ADD  CONSTRAINT
[FK_Cust_Addresses_Addresses] FOREIGN KEY([Address_ID])
REFERENCES [dbo].[Addresses] ([Address_ID])
GO
ALTER TABLE [dbo].[Cust_Addresses] CHECK CONSTRAINT
[FK_Cust_Addresses_Addresses]
GO
ALTER TABLE [dbo].[Cust_Addresses]  WITH CHECK ADD  CONSTRAINT
[FK_Cust_Addresses_Customers] FOREIGN KEY([Cust_ID])
REFERENCES [dbo].[Customers] ([Cust_ID])
GO
ALTER TABLE [dbo].[Cust_Addresses] CHECK CONSTRAINT
[FK_Cust_Addresses_Customers]
/* End Cust Address ******************************************/


USE [compu32_CompuSCRIP]
GO
/****** Object:  Table [dbo].[Merch_Contacts]    Script Date:
11/05/2006 15:30:59 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Merch_Contacts](
      [Merchant_Contact_ID] [int] IDENTITY(1,1) NOT NULL,
      [Merchant_ID] [int] NULL DEFAULT ((0)),
      [Contact_Info_ID] [int] NULL DEFAULT ((0)),
 CONSTRAINT [aaaaaMerch_Contacts_PK] PRIMARY KEY NONCLUSTERED
(
      [Merchant_Contact_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Merch_Contacts]  WITH CHECK ADD  CONSTRAINT
[FK_Merch_Contacts_Contact_Info] FOREIGN KEY([Contact_Info_ID])
REFERENCES [dbo].[Contact_Info] ([Contact_Info_ID])
GO
ALTER TABLE [dbo].[Merch_Contacts] CHECK CONSTRAINT
[FK_Merch_Contacts_Contact_Info]
GO
ALTER TABLE [dbo].[Merch_Contacts]  WITH CHECK ADD  CONSTRAINT
[Merch_Contacts_FK00] FOREIGN KEY([Merchant_ID])
REFERENCES [dbo].[Merchants] ([Merch_ID])
GO
```

```
ALTER TABLE [dbo].[Merch_Contacts] CHECK CONSTRAINT
[Merch_Contacts_FK00]
/* Merchant_Contacts ********************************************/



/****** Object:  Table [dbo].[Merch_Denominations_LU]    Script Date:
11/05/2006 15:32:03 ******/

USE [compu32_CompuSCRIP]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Merch_Denominations_LU](
      [Denom_ID] [int] IDENTITY(1,1) NOT NULL,
      [Amount] [int] NULL DEFAULT ((0)),
 CONSTRAINT [aaaaaMerch_Denominations_LU_PK] PRIMARY KEY NONCLUSTERED
(
      [Denom_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

/* End Merch_Denomincations_LU ***************************/


USE [compu32_CompuSCRIP]
GO
/****** Object:  Table [dbo].[Spons_Contacts]    Script Date:
11/05/2006 15:32:03 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Spons_Contacts](
      [Sponsor_Contact_ID] [int] IDENTITY(1,1) NOT NULL,
      [Sponsor_ID] [int] NULL DEFAULT ((0)),
      [Contact_Info_ID] [int] NULL DEFAULT ((0)),
 CONSTRAINT [aaaaaSpons_Contacts_PK] PRIMARY KEY NONCLUSTERED
(
      [Sponsor_Contact_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Spons_Contacts]  WITH CHECK ADD  CONSTRAINT
[FK_Spons_Contacts_Sponsors] FOREIGN KEY([Sponsor_ID])
REFERENCES [dbo].[Sponsors] ([Sponsor_ID])
GO
ALTER TABLE [dbo].[Spons_Contacts] CHECK CONSTRAINT
[FK_Spons_Contacts_Sponsors]
GO
ALTER TABLE [dbo].[Spons_Contacts]  WITH NOCHECK ADD  CONSTRAINT
[Spons_Contacts_FK00] FOREIGN KEY([Sponsor_Contact_ID])
REFERENCES [dbo].[Contact_Info] ([Contact_Info_ID])
```

```
GO
ALTER TABLE [dbo].[Spons_Contacts] NOCHECK CONSTRAINT
[Spons_Contacts_FK00]
/* End Spons_Contacts ****************************************/


USE [compu32_CompuSCRIP]
GO
/****** Object:  Table [dbo].[Sponsor_Addresses]    Script Date:
11/05/2006 15:32:34 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Sponsor_Addresses](
      [Sponsor_Address_ID] [int] IDENTITY(1,1) NOT NULL,
      [Sponsor_ID] [int] NULL DEFAULT ((0)),
      [Address_ID] [int] NULL DEFAULT ((0)),
 CONSTRAINT [aaaaaSponsor_Addresses_PK] PRIMARY KEY NONCLUSTERED
(
      [Sponsor_Address_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Sponsor_Addresses]  WITH CHECK ADD  CONSTRAINT
[FK_Sponsor_Addresses] FOREIGN KEY([Address_ID])
REFERENCES [dbo].[Addresses] ([Address_ID])
GO
ALTER TABLE [dbo].[Sponsor_Addresses] CHECK CONSTRAINT
[FK_Sponsor_Addresses]
GO
ALTER TABLE [dbo].[Sponsor_Addresses]  WITH CHECK ADD  CONSTRAINT
[FK_Sponsor_Addresses_Sponsors] FOREIGN KEY([Sponsor_ID])
REFERENCES [dbo].[Sponsors] ([Sponsor_ID])
GO
ALTER TABLE [dbo].[Sponsor_Addresses] CHECK CONSTRAINT
[FK_Sponsor_Addresses_Sponsors]
/* End Sponsor_Addresses ****************************************/


USE [compu32_CompuSCRIP]
GO
/****** Object:  Table [dbo].[Spons_Campuses]    Script Date:
11/05/2006 15:33:08 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Spons_Campuses](
      [Campus_ID] [int] IDENTITY(1,1) NOT NULL,
      [Sponsor_ID] [int] NULL DEFAULT ((0)),
      [Campus_Name] [nvarchar](75) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
 CONSTRAINT [aaaaaSpons_Campuses_PK] PRIMARY KEY NONCLUSTERED
```

```
(
      [Campus_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Spons_Campuses]  WITH CHECK ADD  CONSTRAINT
[FK_Spons_Campuses_Sponsors] FOREIGN KEY([Sponsor_ID])
REFERENCES [dbo].[Sponsors] ([Sponsor_ID])
GO
ALTER TABLE [dbo].[Spons_Campuses] CHECK CONSTRAINT
[FK_Spons_Campuses_Sponsors]

/* End Spons_Campuses **********************************/


/****** Object:  Table [dbo].[Contact_Type_LU]    Script Date:
09/14/2006 21:55:30 ******/
USE [compu32_CompuSCRIP]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Contact_Type_LU](
      [Contact_Type_ID] [smallint] IDENTITY(1,1) NOT NULL,
      [Contact_Type_Desc] [nvarchar](50) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL,
 CONSTRAINT [PK_Contact_Type_LU] PRIMARY KEY CLUSTERED
(
      [Contact_Type_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]


/********End Contact_Type_LU**********************************/



USE [compu32_CompuSCRIP]
GO
/****** Object:  Table [dbo].[Contact_Info]    Script Date: 11/05/2006
15:34:29 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Contact_Info](
      [Contact_Info_ID] [int] IDENTITY(1,1) NOT NULL,
      [Last_Name] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS
NOT NULL,
      [First_Name] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS
NOT NULL,
      [MI] [nvarchar](1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
```

```
	[Home_Phone] [nvarchar](12) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
	[Business_Phone] [nvarchar](12) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL,
	[Cell_Phone] [nvarchar](12) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
	[Email_Address] [nvarchar](75) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL,
	[Contact_Type] [smallint] NOT NULL CONSTRAINT
[DF__Contact_I__Conta__5CA42306]  DEFAULT ((0)),
	[Title] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
	[UserID] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
 CONSTRAINT [aaaaaContact_Info_PK] PRIMARY KEY NONCLUSTERED
(
	[Contact_Info_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Contact_Info]  WITH NOCHECK ADD  CONSTRAINT
[FK_Contact_Info_Contact_Type_LU] FOREIGN KEY([Contact_Type])
REFERENCES [dbo].[Contact_Type_LU] ([Contact_Type_ID])
GO
ALTER TABLE [dbo].[Contact_Info] CHECK CONSTRAINT
[FK_Contact_Info_Contact_Type_LU]
/* End Contact Info  ******************************************/



USE [compu32_CompuSCRIP]
GO
/****** Object:  Table [dbo].[Merchants]    Script Date: 11/05/2006
15:34:57 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Merchants](
	[Merch_ID] [int] IDENTITY(1,1) NOT NULL,
	[Merch_Acct_Settings_ID] [int] NOT NULL CONSTRAINT
[DF__Temporary__Merch__789EE131]  DEFAULT ((0)),
	[Merch_Type_ID] [int] NOT NULL CONSTRAINT
[DF__Temporary__Merch__7993056A]  DEFAULT ((0)),
	[Business_Name] [nvarchar](75) COLLATE
SQL_Latin1_General_CP1_CI_AS NOT NULL,
	[Merch_Enabled] [bit] NOT NULL CONSTRAINT
[DF__Temporary__Merch__7B7B4DDC]  DEFAULT ((0)),
	[Sponsor_Broker_ID] [int] NULL,
 CONSTRAINT [aaaaaMerchants_PK] PRIMARY KEY NONCLUSTERED
(
	[Merch_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
```

```
ALTER TABLE [dbo].[Merchants]  WITH CHECK ADD  CONSTRAINT
[FK_Merchants_Merch_Acct_Settings] FOREIGN
KEY([Merch_Acct_Settings_ID])
REFERENCES [dbo].[Merch_Acct_Settings] ([Merch_Acct_Settings_ID])
GO
ALTER TABLE [dbo].[Merchants] CHECK CONSTRAINT
[FK_Merchants_Merch_Acct_Settings]
GO
ALTER TABLE [dbo].[Merchants]  WITH CHECK ADD  CONSTRAINT
[FK_Merchants_Spons_Brokers_LU] FOREIGN KEY([Sponsor_Broker_ID])
REFERENCES [dbo].[Spons_Brokers_LU] ([Spons_Brokers_ID])
GO
ALTER TABLE [dbo].[Merchants] CHECK CONSTRAINT
[FK_Merchants_Spons_Brokers_LU]
GO
ALTER TABLE [dbo].[Merchants]  WITH NOCHECK ADD  CONSTRAINT
[Merchants_FK01] FOREIGN KEY([Merch_Type_ID])
REFERENCES [dbo].[Merch_Type_LU] ([Merch_Type_ID])
NOT FOR REPLICATION
GO
ALTER TABLE [dbo].[Merchants] CHECK CONSTRAINT [Merchants_FK01]
/* End Merchants ************************************************/


/****** Object:  Table [dbo].[Cust_Favorites]    Script Date:
11/05/2006 15:35:22 ******/
USE [compu32_CompuSCRIP]
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Cust_Favorites](
      [Cust_Favorites_ID] [int] IDENTITY(1,1) NOT NULL,
      [Cust_ID] [int] NOT NULL CONSTRAINT
[DF__Temporary__Cust___35DCF99B]  DEFAULT ((0)),
      [Merch_ID] [int] NOT NULL CONSTRAINT
[DF__Temporary__Merch__36D11DD4]  DEFAULT ((0)),
 CONSTRAINT [aaaaaCust_Favorites_PK] PRIMARY KEY NONCLUSTERED
(
      [Cust_Favorites_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Cust_Favorites]  WITH CHECK ADD  CONSTRAINT
[FK_Cust_Favorites_Customers] FOREIGN KEY([Cust_ID])
REFERENCES [dbo].[Customers] ([Cust_ID])
GO
ALTER TABLE [dbo].[Cust_Favorites] CHECK CONSTRAINT
[FK_Cust_Favorites_Customers]
GO
ALTER TABLE [dbo].[Cust_Favorites]  WITH NOCHECK ADD  CONSTRAINT
[FK_Cust_Favorites_Merchants] FOREIGN KEY([Merch_ID])
REFERENCES [dbo].[Merchants] ([Merch_ID])
GO
```

```
ALTER TABLE [dbo].[Cust_Favorites] CHECK CONSTRAINT
[FK_Cust_Favorites_Merchants]
/** End Cust_Favorites  ****************************************/


/****** Object:  Table [dbo].[Spons_Brokers_LU]    Script Date:
11/05/2006 15:36:09 ******/
USE [compu32_CompuSCRIP]
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Spons_Brokers_LU](
      [Spons_Brokers_ID] [int] IDENTITY(1,1) NOT NULL,
      [Broker_Name] [nvarchar](75) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
 CONSTRAINT [aaaaaSpons_Brokers_LU_PK] PRIMARY KEY NONCLUSTERED
(
      [Spons_Brokers_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

/* End Spons_Brokers_LU  ****************************************/


/****** Object:  Table [dbo].[Sponsor_Campus_Addresses]    Script Date:
11/05/2006 15:36:39 ******/

USE [compu32_CompuSCRIP]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Sponsor_Campus_Addresses](
      [Campus_Address_ID] [int] IDENTITY(1,1) NOT NULL,
      [Campus_ID] [int] NULL DEFAULT ((0)),
      [Address_ID] [int] NULL DEFAULT ((0)),
 CONSTRAINT [aaaaaSponsor_Campus_Addresses_PK] PRIMARY KEY NONCLUSTERED
(
      [Campus_Address_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Sponsor_Campus_Addresses]  WITH CHECK ADD
CONSTRAINT [Sponsor_Campus_Addresses_Addresses] FOREIGN
KEY([Address_ID])
REFERENCES [dbo].[Addresses] ([Address_ID])
GO
ALTER TABLE [dbo].[Sponsor_Campus_Addresses] CHECK CONSTRAINT
[Sponsor_Campus_Addresses_Addresses]
GO
ALTER TABLE [dbo].[Sponsor_Campus_Addresses]  WITH CHECK ADD
CONSTRAINT [Sponsor_Campus_Addresses_FK00] FOREIGN KEY([Campus_ID])
```

```
REFERENCES [dbo].[Spons_Campuses] ([Campus_ID])
GO
ALTER TABLE [dbo].[Sponsor_Campus_Addresses] CHECK CONSTRAINT
[Sponsor_Campus_Addresses_FK00]
/* End Sponsor_Campus_Addresses
************************************/


/****** Object:  Table [dbo].[Cust_Balance_Adjustments]    Script Date:
11/05/2006 15:37:15 ******/
USE [compu32_CompuSCRIP]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Cust_Balance_Adjustments](
      [Bal_Adj_ID] [int] IDENTITY(1,1) NOT NULL,
      [Cust_ID] [int] NULL CONSTRAINT [DF__Temporary__Cust___4EA8A765]
DEFAULT ((0)),
      [Debit_Amt] [int] NULL CONSTRAINT
[DF__Temporary__Debit__4F9CCB9E]  DEFAULT ((0)),
      [Credit_Amt] [int] NULL CONSTRAINT
[DF__Temporary__Credi__5090EFD7]  DEFAULT ((0)),
      [Made_by] [int] NULL CONSTRAINT [DF__Temporary__Made___51851410]
DEFAULT ((0)),
      [Adj_Date] [datetime] NULL,
      [Comments] [nvarchar](255) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
 CONSTRAINT [aaaaaCust_Balance_Adjustments_PK] PRIMARY KEY NONCLUSTERED
(
      [Bal_Adj_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Cust_Balance_Adjustments]  WITH CHECK ADD
CONSTRAINT [FK_Cust_Balance_Adjustments_Customers] FOREIGN
KEY([Cust_ID])
REFERENCES [dbo].[Customers] ([Cust_ID])
GO
ALTER TABLE [dbo].[Cust_Balance_Adjustments] CHECK CONSTRAINT
[FK_Cust_Balance_Adjustments_Customers]
/* End Cust_Balance_Adjustments  ********************************/


/****** Object:  Table [dbo].[Spons_Acct_Settings]    Script Date:
11/05/2006 15:38:01 ******/
USE [compu32_CompuSCRIP]
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Spons_Acct_Settings](
      [Spons_Acct_Settings_ID] [int] IDENTITY(1,1) NOT NULL,
```

```
	[Spons_Logo_Pic] [image] NULL,
	[Spons_Logo_Text] [nvarchar](50) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL,
	[Spons_Scroll_Text] [nvarchar](max) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL,
	[Spons_Welcome_Page_Text] [nvarchar](max) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL,
	[Spons_Cust_Home_Page_Text] [nvarchar](max) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL,
 CONSTRAINT [aaaaaSpons_Acct_Settings_PK] PRIMARY KEY NONCLUSTERED
(
	[Spons_Acct_Settings_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

/* End Spons-Account_Settings   ************************************/


USE [compu32_CompuSCRIP]
GO
/****** Object:  Table [dbo].[Cust_Account_Settings]    Script Date:
11/05/2006 15:38:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Cust_Account_Settings](
	[Account_Settings_ID] [int] IDENTITY(1,1) NOT NULL,
	[Expiration_Date] [datetime] NULL,
	[Donate_To_Acct1] [int] NULL CONSTRAINT
[DF__Temporary__Donat__3B95D2F1]  DEFAULT ((0)),
	[Donate_To_Acct2] [int] NULL CONSTRAINT
[DF__Temporary__Donat__3C89F72A]  DEFAULT ((0)),
	[Donate_To_Acct3] [int] NULL CONSTRAINT
[DF__Temporary__Donat__3D7E1B63]  DEFAULT ((0)),
	[Donate_To_Acct4] [int] NULL CONSTRAINT
[DF__Temporary__Donat__3E723F9C]  DEFAULT ((0)),
	[Donate_To_Acct5] [int] NULL CONSTRAINT
[DF__Temporary__Donat__3F6663D5]  DEFAULT ((0)),
	[Anonymous1] [bit] NULL CONSTRAINT
[DF__Temporary__Anony__405A880E]  DEFAULT ((0)),
	[Anonymous2] [bit] NULL CONSTRAINT
[DF__Temporary__Anony__414EAC47]  DEFAULT ((0)),
	[Anonymous3] [bit] NULL CONSTRAINT
[DF__Temporary__Anony__4242D080]  DEFAULT ((0)),
	[Anonymous4] [bit] NULL CONSTRAINT
[DF__Temporary__Anony__4336F4B9]  DEFAULT ((0)),
	[Anonymous5] [bit] NULL CONSTRAINT
[DF__Temporary__Anony__442B18F2]  DEFAULT ((0)),
	[Name1] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
	[Name2] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
	[Name3] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
	[Name4] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
	[Name5] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
	[Account1_Visible] [bit] NULL CONSTRAINT
[DF__Temporary__Accou__451F3D2B]  DEFAULT ((0)),
```

```
      [Account2_Visible] [bit] NULL CONSTRAINT
[DF__Temporary__Accou__46136164]  DEFAULT ((0)),
      [Account3_Visible] [bit] NULL CONSTRAINT
[DF__Temporary__Accou__4707859D]  DEFAULT ((0)),
      [Account4_Visible] [bit] NULL CONSTRAINT
[DF__Temporary__Accou__47FBA9D6]  DEFAULT ((0)),
      [Account5_Visible] [bit] NULL CONSTRAINT
[DF__Temporary__Accou__48EFCE0F]  DEFAULT ((0)),
      [Future_Tuition_Payer] [bit] NULL CONSTRAINT
[DF__Temporary__Futur__49E3F248]  DEFAULT ((0)),
 CONSTRAINT [aaaaaCust_Account_Settings_PK] PRIMARY KEY NONCLUSTERED
(
      [Account_Settings_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
/* End Cust_Account_Settings  ***********************************/


USE [compu32_CompuSCRIP]
GO
/****** Object:  Table [dbo].[Cust_Orders]    Script Date: 11/05/2006
15:39:04 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Cust_Orders](
      [Cust_Order_ID] [int] IDENTITY(1,1) NOT NULL,
      [Cust_ID] [int] NULL CONSTRAINT [DF__Temporary__Cust___5649C92D]
DEFAULT ((0)),
      [Order_Confirmation_Num] [nvarchar](50) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL CONSTRAINT
[DF__Temporary__Order__573DED66]  DEFAULT ((0)),
      [Order_Submission_Date] [datetime] NULL,
      [Order_Processed_Date] [datetime] NULL,
      [Cust_Payment_Recieved] [bit] NULL CONSTRAINT
[DF__Temporary__Cust___5832119F]  DEFAULT ((0)),
      [Check_Num] [nvarchar](6) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
      [Cust_Order_Rcvd_Date] [datetime] NULL,
      [Credited_to_Tuition_Date] [datetime] NULL,
      [Auto_Submit_Date] [datetime] NULL,
      [Comment_ID] [int] NULL,
      [Payment_Rcvd_Date] [datetime] NULL,
      [Cust_Received] [bit] NULL,
      [TotalAmountOfOrder] [float] NULL,
 CONSTRAINT [aaaaaCust_Orders_PK] PRIMARY KEY NONCLUSTERED
(
      [Cust_Order_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Cust_Orders]  WITH CHECK ADD  CONSTRAINT
[FK_Cust_Orders_Cust_Order_Comments] FOREIGN KEY([Comment_ID])
REFERENCES [dbo].[Cust_Order_Comments] ([Comment_ID])
GO
```

```
ALTER TABLE [dbo].[Cust_Orders] CHECK CONSTRAINT
[FK_Cust_Orders_Cust_Order_Comments]
GO
ALTER TABLE [dbo].[Cust_Orders]  WITH CHECK ADD  CONSTRAINT
[FK_Cust_Orders_Customers] FOREIGN KEY([Cust_ID])
REFERENCES [dbo].[Customers] ([Cust_ID])
GO
ALTER TABLE [dbo].[Cust_Orders] CHECK CONSTRAINT
[FK_Cust_Orders_Customers]

/*****End Cust_Orders  ************************/


USE [compu32_CompuSCRIP]
GO
/****** Object:  Table [dbo].[Cust_Denom_Order_Details]    Script Date:
11/05/2006 15:39:43 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Cust_Denom_Order_Details](
      [Denom_Order_ID] [int] IDENTITY(1,1) NOT NULL,
      [Cust_Order_Detail_ID] [int] NULL DEFAULT ((0)),
      [Quantity] [int] NULL DEFAULT ((0)),
      [Denomination] [int] NULL DEFAULT ((0)),
 CONSTRAINT [aaaaaCust_Denom_Order_Details_PK] PRIMARY KEY NONCLUSTERED
(
      [Denom_Order_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Cust_Denom_Order_Details]  WITH CHECK ADD
CONSTRAINT [Cust_Denom_Order_Details_FK00] FOREIGN
KEY([Cust_Order_Detail_ID])
REFERENCES [dbo].[Cust_Order_Details] ([Cust_Order_Details_ID])
GO
ALTER TABLE [dbo].[Cust_Denom_Order_Details] CHECK CONSTRAINT
[Cust_Denom_Order_Details_FK00]
/* End Cust Denom Order Details   ***********************************/


USE [compu32_CompuSCRIP]
GO
/****** Object:  Table [dbo].[Cust_Order_Details]    Script Date:
11/05/2006 15:40:09 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Cust_Order_Details](
      [Cust_Order_Details_ID] [int] IDENTITY(1,1) NOT NULL,
      [Cust_Order_ID] [int] NULL CONSTRAINT
[DF__Temporary__Cust___668030F6]  DEFAULT ((0)),
```

```
      [Merchant_ID] [int] NULL CONSTRAINT
 [DF__Temporary__Merch__68687968]  DEFAULT ((0)),
      [Percent_to_Cust] [float] NULL CONSTRAINT
 [DF__Temporary__Perce__695C9DA1]  DEFAULT ((0)),
      [Percent_to_Sponsor] [float] NULL CONSTRAINT
 [DF__Temporary__Perce__6A50C1DA]  DEFAULT ((0)),
      [Total_Order_Amount] [float] NULL CONSTRAINT
 [DF__Temporary__Total__6B44E613]  DEFAULT ((0)),
  CONSTRAINT [aaaaaCust_Order_Details_PK] PRIMARY KEY NONCLUSTERED
 (
      [Cust_Order_Details_ID] ASC
 )WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
 ) ON [PRIMARY]

 GO
 ALTER TABLE [dbo].[Cust_Order_Details]  WITH CHECK ADD  CONSTRAINT
 [Cust_Order_Details_FK00] FOREIGN KEY([Cust_Order_ID])
 REFERENCES [dbo].[Cust_Orders] ([Cust_Order_ID])
 GO
 ALTER TABLE [dbo].[Cust_Order_Details] CHECK CONSTRAINT
 [Cust_Order_Details_FK00]
 GO
 ALTER TABLE [dbo].[Cust_Order_Details]  WITH CHECK ADD  CONSTRAINT
 [FK_Cust_Order_Details_Merchants] FOREIGN KEY([Merchant_ID])
 REFERENCES [dbo].[Merchants] ([Merch_ID])
 GO
 ALTER TABLE [dbo].[Cust_Order_Details] CHECK CONSTRAINT
 [FK_Cust_Order_Details_Merchants]
 /* End Cust Order Details  ***************************************/


 USE [compu32_CompuSCRIP]
 GO
 /****** Object:  Table [dbo].[Cust_Order_Comments]    Script Date:
 11/05/2006 15:40:34 ******/
 SET ANSI_NULLS ON
 GO
 SET QUOTED_IDENTIFIER ON
 GO
 CREATE TABLE [dbo].[Cust_Order_Comments](
      [Comment_ID] [int] IDENTITY(1,1) NOT NULL,
      [Comment_Text] [nvarchar](max) COLLATE
 SQL_Latin1_General_CP1_CI_AS NULL,
  CONSTRAINT [aaaaaCust_Order_Comments_PK] PRIMARY KEY NONCLUSTERED
 (
      [Comment_ID] ASC
 )WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
 ) ON [PRIMARY]

 /****End Cust_Order_Comments   *****************/


 USE [compu32_CompuSCRIP]
 GO
 /****** Object:  Table [dbo].[Merch_Addresses]    Script Date:
 11/05/2006 15:41:00 ******/
```

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Merch_Addresses](
      [Merch_Address_ID] [int] IDENTITY(1,1) NOT NULL,
      [Merch_ID] [int] NULL CONSTRAINT [DF__Temporary__Merch__1AF3F935]
DEFAULT ((0)),
      [Address_ID] [int] NULL CONSTRAINT
[DF__Temporary__Addre__1CDC41A7]  DEFAULT ((0)),
 CONSTRAINT [aaaaaMerch_Addresses_PK] PRIMARY KEY NONCLUSTERED
(
      [Merch_Address_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Merch_Addresses]  WITH CHECK ADD  CONSTRAINT
[FK_Merch_Addresses_Addresses] FOREIGN KEY([Address_ID])
REFERENCES [dbo].[Addresses] ([Address_ID])
GO
ALTER TABLE [dbo].[Merch_Addresses] CHECK CONSTRAINT
[FK_Merch_Addresses_Addresses]
GO
ALTER TABLE [dbo].[Merch_Addresses]  WITH CHECK ADD  CONSTRAINT
[FK_Merch_Addresses_Merchants] FOREIGN KEY([Merch_ID])
REFERENCES [dbo].[Merchants] ([Merch_ID])
GO
ALTER TABLE [dbo].[Merch_Addresses] CHECK CONSTRAINT
[FK_Merch_Addresses_Merchants]
/* End Merchant Addresses  ************************************/



USE [compu32_CompuSCRIP]
GO
/****** Object:  Table [dbo].[Merch_Denominations]    Script Date:
11/05/2006 15:41:24 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Merch_Denominations](
      [Merchant_Denom_ID] [int] IDENTITY(1,1) NOT NULL,
      [Merch_ID] [int] NULL CONSTRAINT [DF__Temporary__Merch__0E8E2250]
DEFAULT ((0)),
      [Denom_ID] [int] NULL CONSTRAINT [DF__Temporary__Denom__10766AC2]
DEFAULT ((0)),
 CONSTRAINT [aaaaaMerch_Denominations_PK] PRIMARY KEY NONCLUSTERED
(
      [Merchant_Denom_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Merch_Denominations]  WITH CHECK ADD  CONSTRAINT
[FK_Merch_Denominations_Merch_Denominations_LU] FOREIGN KEY([Denom_ID])
```

```
REFERENCES [dbo].[Merch_Denominations_LU] ([Denom_ID])
GO
ALTER TABLE [dbo].[Merch_Denominations] CHECK CONSTRAINT
[FK_Merch_Denominations_Merch_Denominations_LU]
GO
ALTER TABLE [dbo].[Merch_Denominations]  WITH CHECK ADD  CONSTRAINT
[Merch_Denominations_FK00] FOREIGN KEY([Merch_ID])
REFERENCES [dbo].[Merchants] ([Merch_ID])
GO
ALTER TABLE [dbo].[Merch_Denominations] CHECK CONSTRAINT
[Merch_Denominations_FK00]
/* End Merch_Denominations  ***********************************/


USE [compu32_CompuSCRIP]
GO
/****** Object:  Table [dbo].[Merch_SCRIP_Type_LU]    Script Date:
11/05/2006 15:41:53 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Merch_SCRIP_Type_LU](
      [SCRIP_Type_ID] [int] IDENTITY(1,1) NOT NULL,
      [SCRIP_Type] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS
NOT NULL,
 CONSTRAINT [PK_Merch_SCRIP_Type_LU] PRIMARY KEY CLUSTERED
(
      [SCRIP_Type_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]


/* End Merch_SCRIP_Type_LU  ***************************************/


USE [compu32_CompuSCRIP]
GO
/****** Object:  Table [dbo].[Merch_Acct_Settings]    Script Date:
11/05/2006 15:42:20 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Merch_Acct_Settings](
      [Merch_Acct_Settings_ID] [int] IDENTITY(1,1) NOT NULL,
      [SCRIP_Type_ID] [int] NULL CONSTRAINT
[DF__Temporary__SCRIP__01342732]  DEFAULT ((0)),
      [Merch_URL] [nvarchar](255) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
      [Merch_Percent_to_Cust] [float] NULL CONSTRAINT
[DF__Temporary__Merch__02284B6B]  DEFAULT ((0)),
      [Merch_Percent_to_Sponsor] [float] NULL CONSTRAINT
[DF__Temporary__Merch__031C6FA4]  DEFAULT ((0)),
      [Order_As_Needed] [bit] NULL CONSTRAINT
[DF__Temporary__Order__041093DD]  DEFAULT ((0)),
```

```
      [Order_In_Bulk] [bit] NULL CONSTRAINT
[DF__Temporary__Order__0504B816]  DEFAULT ((0)),
      [Local_Purchases_Only] [bit] NULL CONSTRAINT
[DF__Temporary__Local__05F8DC4F]  DEFAULT ((0)),
      [Broker_ID] [int] NULL CONSTRAINT
[DF__Temporary__Broke__06ED0088]  DEFAULT ((0)),
      [Face_Value_Discount] [bit] NULL CONSTRAINT
[DF__Temporary__Face___07E124C1]  DEFAULT ((0)),
      [Merch_Comments] [ntext] COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
      [Min_Purch_Required] [int] NULL CONSTRAINT
[DF__Temporary__Min_P__08D548FA]  DEFAULT ((0)),
      [Max_Purch_Required] [int] NULL CONSTRAINT
[DF__Temporary__Max_P__09C96D33]  DEFAULT ((0)),
 CONSTRAINT [aaaaaMerch_Acct_Settings_PK] PRIMARY KEY NONCLUSTERED
(
      [Merch_Acct_Settings_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
ALTER TABLE [dbo].[Merch_Acct_Settings]  WITH CHECK ADD  CONSTRAINT
[FK_Merch_Acct_Settings_Merch_SCRIP_Type_LU] FOREIGN
KEY([SCRIP_Type_ID])
REFERENCES [dbo].[Merch_SCRIP_Type_LU] ([SCRIP_Type_ID])
GO
ALTER TABLE [dbo].[Merch_Acct_Settings] CHECK CONSTRAINT
[FK_Merch_Acct_Settings_Merch_SCRIP_Type_LU]

/* End Merch_Acct_Settings
******************************************/


USE [compu32_CompuSCRIP]
GO
/****** Object:  Table [dbo].[Merch_Type_LU]    Script Date: 11/05/2006
15:42:46 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Merch_Type_LU](
      [Merch_Type_ID] [int] IDENTITY(1,1) NOT NULL,
      [Merch_Type] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
 CONSTRAINT [aaaaaMerch_Type_LU_PK] PRIMARY KEY NONCLUSTERED
(
      [Merch_Type_ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

/* End Merch_Type_LU
************************************************/


USE [compu32_CompuSCRIP]
```

```
GO
/****** Object:  Table [dbo].[Spons_Merchant_List]    Script Date:
11/05/2006 15:43:24 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Spons_Merchant_List](
      [Spons_Merch_ID] [int] IDENTITY(1,1) NOT NULL,
      [Sponsor_ID] [int] NOT NULL,
      [Merchant_ID] [int] NOT NULL
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Spons_Merchant_List]  WITH CHECK ADD  CONSTRAINT
[FK_Spons_Merchant_List_Merchants] FOREIGN KEY([Merchant_ID])
REFERENCES [dbo].[Merchants] ([Merch_ID])
GO
ALTER TABLE [dbo].[Spons_Merchant_List] CHECK CONSTRAINT
[FK_Spons_Merchant_List_Merchants]
GO
ALTER TABLE [dbo].[Spons_Merchant_List]  WITH CHECK ADD  CONSTRAINT
[FK_Spons_Merchant_List_Sponsors] FOREIGN KEY([Sponsor_ID])
REFERENCES [dbo].[Sponsors] ([Sponsor_ID])
GO
ALTER TABLE [dbo].[Spons_Merchant_List] CHECK CONSTRAINT
[FK_Spons_Merchant_List_Sponsors]
/****** End Sponsor Merchant List  *****************/
```

Table 2. Stored Prodedure: Add_Address

```
ALTER PROCEDURE dbo.Add_Address
      (
            @Merch_ID int,
            @Sponsor_ID int,
            @Cust_ID int,
            @Campus_ID int,
            @Street_Line_1 varchar(50),
            @Street_Line_2 varchar(50),
            @City varchar(50),
            @State varchar(2),
            @Zip varchar(10),
            @Address_Type int,
            @NewAddress_ID int = NULL OUTPUT

      )

AS
      SET NOCOUNT ON


      INSERT INTO Addresses(Street_Line_1, Street_Line_2, City, State,
ZIP, Address_Type) VALUES (@Street_Line_1, @Street_Line_2, @City,
@State, @ZIP, @Address_Type)
```

```
      SELECT @NewAddress_ID = @@IDENTITY


      IF @Merch_ID IS NOT NULL
                  INSERT INTO Merch_Addresses(Merch_ID, Address_ID)
VALUES (@Merch_ID, @NewAddress_ID)
      ELSE
            IF @Sponsor_ID IS NOT NULL
                  INSERT INTO Sponsor_Addresses(Sponsor_ID, Address_ID)
VALUES (@Sponsor_ID, @NewAddress_ID)
            ELSE
                  IF @Cust_ID IS NOT NULL
                        INSERT INTO Cust_Addresses(Cust_ID, Address_ID)
VALUES (@Cust_ID, @NewAddress_ID)
                  ELSE
                        IF @Campus_ID IS NOT NULL
                              INSERT INTO
Sponsor_Campus_Addresses(Campus_ID, Address_ID) VALUES (@Campus_ID,
@NewAddress_ID)
```

Table 3. Stored Prodedure: Add_Customer

```
ALTER PROCEDURE Add_Customer
/* Input Parameters for Merch_Acct_Settings */
(
      @Sponsor_ID int,
      @Cust_ID int,
      @Last_Name nvarchar(50),
      @First_Name nvarchar(50),
      @MI nvarchar(1),
      @Home_Phone nvarchar(12),
      @Business_Phone nvarchar(12),
      @Cell_Phone nvarchar(12),
      @Email_Address nvarchar(75),
      @Contact_Type smallint,
      @Street_Line_1 varchar(50),
      @Street_Line_2 varchar(50),
      @City varchar(50),
      @State varchar(2),
      @Zip varchar(10),
      @Address_Type int,
      @Name_To_Donate_To nvarchar(50),
      @Future_Tuition_Payer bit,
      @Comp_AcctNum nvarchar(50),
      @Cust_Enabled bit,
      @NewContact_Info_ID int = NULL OUTPUT,
      @NewCust_Account_Settings int = NULL OUTPUT,
      @NewCust_ID int = NULL OUTPUT

      )
```

```
AS
      SET NOCOUNT ON

DECLARE @NewMerch_ID int
DECLARE @Campus_ID int
DECLARE @Title nvarchar(75)

Set @NewMerch_ID = NULL
Set @Campus_ID = NULL
Set @Title = NULL

INSERT INTO Contact_Info (Last_Name,
                                    First_Name,
                                    MI,
                                    Home_Phone,
                                    Business_Phone,
                                    Cell_Phone,
                                    Email_Address,
                                    Contact_Type,
                                    Title)
                    VALUES      (@Last_Name,
                                    @First_Name,
                                    @MI,
                                    @Home_Phone,
                                    @Business_Phone,
                                    @Cell_Phone,
                                    @Email_Address,
                                    @Contact_Type,
                                    @Title)
      SELECT @NewContact_Info_ID = @@IDENTITY


                        INSERT INTO Cust_Account_Settings(Name1,

Future_Tuition_Payer)
                                                        VALUES
      (@Name_To_Donate_To,

@Future_Tuition_Payer)
                        SELECT @NewCust_Account_Settings = @@IDENTITY

                        INSERT INTO Customers(Contact_Info_ID,
                                    Account_Settings_ID,
                                    Sponsor_ID,
                                    Cust_Enabled)
                        VALUES      (@NewContact_Info_ID,
                                    @NewCust_Account_Settings,
                                    @Comp_AcctNum,
                                    @Cust_Enabled)
                        SELECT @NewCust_ID = @@IDENTITY


      IF @Street_Line_1 IS NOT NULL
            BEGIN
      Exec Add_Address
```

```
            @NewMerch_ID,
            @Sponsor_ID,
            @Cust_ID,
            @Campus_ID,
            @Street_Line_1,
            @Street_Line_2,
            @City,
            @State,
            @Zip,
            @Address_Type

      END
RETURN
```

Table 4. Stored Prodedure: AddContactInfo_Merch_Spons

```
ALTER PROCEDURE AddContactInfo_Merch_Spons
/* Input Parameters for Merch_Acct_Settings */
(
      @Merch_ID int,
      @Sponsor_ID int,
      @Cust_ID int,
      @Last_Name nvarchar(50),
      @First_Name nvarchar(50),
      @MI nvarchar(1),
      @Home_Phone nvarchar(12),
      @Business_Phone nvarchar(12),
      @Cell_Phone nvarchar(12),
      @Email_Address nvarchar(75),
      @Contact_Type smallint,
      @Title nvarchar(75),
      @NewContact_Info_ID int = NULL OUTPUT
      )

AS
      SET NOCOUNT ON

set @Contact_Type = 1

INSERT INTO Contact_Info (Last_Name, First_Name, MI, Home_Phone,
Business_Phone, Cell_Phone, Email_Address, Contact_Type, Title)
VALUES
(@Last_Name,@First_Name,@MI,@Home_Phone,@Business_Phone,@Cell_Phone,@Em
ail_Address,@Contact_Type,@Title)
      SELECT @NewContact_Info_ID = @@IDENTITY



      IF @Merch_ID IS NOT NULL
            BEGIN
                  IF @NewContact_Info_ID IS NOT NULL
```

```
                        INSERT INTO Merch_Contacts(Merchant_ID,
Contact_Info_ID) VALUES (@Merch_ID, @NewContact_Info_ID)
            /*
                  IF @S_NewContact_Info_ID IS NOT NULL
                        INSERT INTO Merch_Contacts VALUES (@Merch_ID,
@S_NewContact_Info_ID)
            */
            END

      ELSE

            IF @Sponsor_ID IS NOT NULL
                  BEGIN
                        IF @NewContact_Info_ID IS NOT NULL
                              INSERT INTO Spons_Contacts (Sponsor_ID,
Contact_Info_ID) VALUES (@Sponsor_ID, @NewContact_Info_ID)


                  END

RETURN
```

Table 5. Stored Prodedure: AddMerchant

```
ALTER PROCEDURE dbo.AddMerchant
            (
      /* Input Parameters for the Merch_Acct_Settings table */

      @SCRIP_Type_ID int,
      @Sponsor_ID int = 1,
      @Merch_URL varchar(255),
      @Merch_Percent_to_Cust FLOAT,
      @Merch_Percent_to_Sponsor FLOAT,
      @Order_As_Needed bit,
      @Order_In_Bulk bit,
      @Local_Purchases_Only bit,
      @Face_Value_Discount bit,
      @Merch_Comments ntext,
      @Min_Purch_Required int,
      @Max_Purch_Required int,


      /* Input Parameters for the Merchants table */
      /*@Merch_Acct_Settings_ID int,*/
      @Merch_Type_ID int,
      @Business_Name nvarchar(75),
      @Merch_Enabled bit,
      @Sponsor_Broker_ID int,
      @NewMerch_Acct_Settings_ID  int = NULL OUTPUT,
      @NewMerch_ID int = NULL OUTPUT,
      @NewSpons_Merch_ID int = NULL OUTPUT,

      /* Insert new Address record (Physical Address) */
```

```
        @Street_Line_1 varchar(50),
        @Street_Line_2 varchar(50),
        @City varchar(50),
        @State varchar(2),
        @Zip varchar(10),
        @Address_Type int,


        @Last_Name nvarchar(50),
        @First_Name nvarchar(50),
        @MI nvarchar(1),
        @Home_Phone nvarchar(12),
        @Business_Phone nvarchar(12),
        @Cell_Phone nvarchar(12),
        @Email_Address nvarchar(75),
        @Contact_Type smallint,
        @Title nvarchar(75)

)


AS
        DECLARE @Cust_ID int
        DECLARE @Campus_ID int
        DECLARE @Sponsor_ID_For_Address int

        SELECT @Cust_ID = NULL
        SELECT @Campus_ID = NULL
        SELECT @Sponsor_ID_For_Address = NULL
        SELECT @Sponsor_ID = 1

         SET NOCOUNT ON

INSERT INTO Merch_Acct_Settings (SCRIP_Type_ID, Merch_URL,
Merch_Percent_to_Cust,Merch_Percent_to_Sponsor,Order_As_Needed,
Order_In_Bulk, Local_Purchases_Only, Face_Value_Discount,
Merch_Comments, Min_Purch_Required, Max_Purch_Required)
 VALUES (@SCRIP_Type_ID, @Merch_URL,
@Merch_Percent_to_Cust,@Merch_Percent_to_Sponsor,@Order_As_Needed,
@Order_In_Bulk, @Local_Purchases_Only, @Face_Value_Discount,
@Merch_Comments, @Min_Purch_Required, @Max_Purch_Required)
SELECT @NewMerch_Acct_Settings_ID = @@IDENTITY



INSERT INTO Merchants (Merch_Acct_Settings_ID, Merch_Type_ID,
Business_Name, Merch_Enabled, Sponsor_Broker_ID)
VALUES (@NewMerch_Acct_Settings_ID,@Merch_Type_ID, @Business_Name,
@Merch_Enabled, @Sponsor_Broker_ID )
SELECT @NewMerch_ID = @@IDENTITY


IF @Last_Name IS NOT NULL
      BEGIN
            Exec AddContactInfo_Merch_Spons

                  @NewMerch_ID,
```

```
                @Sponsor_ID_For_Address,
                @Cust_ID,
                @Last_Name,
                @First_Name,
                @MI,
                @Home_Phone,
                @Business_Phone,
                @Cell_Phone,
                @Email_Address,
                @Contact_Type,
                @Title

        END


        IF @Street_Line_1 IS NOT NULL
              BEGIN
        Exec Add_Address

              @NewMerch_ID,
              @Sponsor_ID_For_Address,
              @Cust_ID,
              @Campus_ID,
              @Street_Line_1,
              @Street_Line_2,
              @City,
              @State,
              @Zip,
              @Address_Type

        END


        INSERT INTO Spons_Merchant_List
                              (Sponsor_ID, Merchant_ID)
        VALUES      (@Sponsor_ID, @NewMerch_ID)
```

Table 6. Stored Prodedure: AddNewOrder_Cust_Denom_Order_Details

```
ALTER PROCEDURE dbo.AddNewOrder_Cust_Denom_Order_Details
      (
            @NewCust_Order_Details_ID int,
        @Quantity int,
        @DenomAmount int
      )

AS
      /* SET NOCOUNT ON */


      INSERT INTO Cust_Denom_Order_Details(Cust_Order_Detail_ID,
                                        Quantity, Denomination)
      VALUES (@NewCust_Order_Details_ID, @Quantity, @DenomAmount)
```

```
RETURN
```

Table 7. Stored Prodedure: AddNewOrder_Cust_Order_Table

```
ALTER PROCEDURE dbo.AddNewOrder_Cust_Order_Table
      (
            @Cust_ID int,
          @Order_Confirmation_Num nvarchar(50),
            @CheckNum nvarchar(6),
            @Comments ntext,
            @GrandTotalForOrder float,
            @NewComment_ID int = NULL OUTPUT


      )

AS
      /* SET NOCOUNT ON */
DECLARE @Cust_Recvd_Order bit
declare @now datetime

SET @Cust_Recvd_Order = 'False'
select @now = getdate()

INSERT INTO Cust_Order_Comments (Comment_Text) VALUES (@Comments)
SELECT @NewComment_ID  = @@IDENTITY


INSERT INTO Cust_Orders
                        (Cust_ID, Order_Confirmation_Num,
Order_Submission_Date, Check_Num, Cust_Received, TotalAmountOfOrder,
Comment_ID)
VALUES      (@Cust_ID, @Order_Confirmation_Num, @now, @CheckNum,
@Cust_Recvd_Order, @GrandTotalForOrder, @NewComment_ID)


SELECT      Cust_Order_ID
FROM          Cust_Orders
WHERE Cust_ID = @Cust_ID and Order_Confirmation_Num =
@Order_Confirmation_Num


RETURN
```

Table 8. Stored Prodedure: DeleteMerchDenominations

```
ALTER PROCEDURE dbo.DeleteMerchDenominations
      (
            @Merch_ID int
```

```
      )

AS
      /* SET NOCOUNT ON */

DELETE FROM Merch_Denominations WHERE  Merch_ID = @Merch_ID


      RETURN
```

Table 9. Stored Prodedure: FindNewlyAddedMerchant

```
ALTER PROCEDURE dbo.FindNewlyAddedMerchant
      (
            @Sponsor_ID int,
            @Business_Name nvarchar(75)

       )

AS
       SET NOCOUNT ON


      SELECT     Merchants.Merch_ID
      FROM         Merchants INNER JOIN
                           Spons_Merchant_List ON Merchants.Merch_ID =
Spons_Merchant_List.Merchant_ID
      WHERE     (Spons_Merchant_List.Sponsor_ID = @Sponsor_ID) AND
(Merchants.Business_Name = @Business_Name)
      ORDER BY Merchants.Business_Name

RETURN
```

Table 10. Stored Prodedure: GetScripOrderList

```
ALTER PROCEDURE [dbo].[GetScripOrderList]

      (
      @Sponsor_ID int,
      @Business_Name varchar(75) = NULL OUTPUT


      )

AS
      /* SET NOCOUNT ON  */
/* ==============================================*/
SELECT     Merchants.Merch_ID, Merchants.Business_Name,
Merch_SCRIP_Type_LU.SCRIP_Type, Merch_Acct_Settings.Merch_URL,
```

```
                        Merch_Acct_Settings.Merch_Percent_to_Cust,
Merch_Acct_Settings.Merch_Percent_to_Sponsor,
Merch_Acct_Settings.Local_Purchases_Only,
                        Merch_Acct_Settings.Merch_Comments,
Merch_Type_LU.Merch_Type, Merch_Acct_Settings.Min_Purch_Required,
                        Merch_Acct_Settings.Max_Purch_Required
FROM          Merchants INNER JOIN
                        Merch_Acct_Settings ON
Merchants.Merch_Acct_Settings_ID =
Merch_Acct_Settings.Merch_Acct_Settings_ID INNER JOIN
                        Spons_Merchant_List ON Merchants.Merch_ID =
Spons_Merchant_List.Merchant_ID AND
                        Merchants.Merch_ID =
Spons_Merchant_List.Merchant_ID INNER JOIN
                        Merch_SCRIP_Type_LU ON
Merch_Acct_Settings.SCRIP_Type_ID = Merch_SCRIP_Type_LU.SCRIP_Type_ID
INNER JOIN
                        Merch_Type_LU ON Merchants.Merch_Type_ID =
Merch_Type_LU.Merch_Type_ID
WHERE      (Spons_Merchant_List.Sponsor_ID = @Sponsor_ID)


      RETURN (1)
```

Table 11. Stored Prodedure: Select_Merch_Acct_Settings_ID

```
ALTER PROCEDURE dbo.Select_Merch_Acct_Settings_ID
      (
        @Merch_ID int
      )

AS
      /* SET NOCOUNT ON */
SELECT Merch_Acct_Settings_ID FROM Merchants WHERE (Merch_ID =
@Merch_ID)

RETURN
```

Table 12. Stored Prodedure: SelectBrokers_All

```
ALTER PROCEDURE dbo.SelectBrokers_All

AS
      /* SET NOCOUNT ON */

      SELECT [Spons_Brokers_ID], [Broker_Name] FROM [Spons_Brokers_LU]

      RETURN
```

Table 13. Stored Prodedure: SelectContact

```
ALTER PROCEDURE dbo.SelectContact
      (
        @Merch_ID int,
        @Contact_Type int
      )

AS
      /* SET NOCOUNT ON */
SELECT     Contact_Info.Last_Name, Contact_Info.First_Name,
Contact_Info.MI, Contact_Info.Home_Phone, Contact_Info.Business_Phone,
                      Contact_Info.Cell_Phone,
Contact_Info.Email_Address, Contact_Info.Title
FROM         Merch_Contacts INNER JOIN
                      Contact_Info ON Merch_Contacts.Contact_Info_ID =
Contact_Info.Contact_Info_ID
WHERE     (Merch_Contacts.Merchant_ID = @Merch_ID) AND
(Contact_Info.Contact_Type = @Contact_Type)

RETURN
```

Table 14. Stored Prodedure: SelectCustomers

```
ALTER PROCEDURE dbo.SelectCustomers

AS
      /* SET NOCOUNT ON */

      SELECT     Customers.Account_Settings_ID, Customers.Comp_AcctNum,
Customers.Sponsor_ID, Customers.Cust_Enabled,
Customers.Pending_Approval, Cust_Account_Settings.Name1,
Cust_Account_Settings.Future_Tuition_Payer, Contact_Info.Last_Name,
Contact_Info.First_Name, Contact_Info.MI, Contact_Info.Home_Phone,
Contact_Info.Business_Phone, Contact_Info.Cell_Phone,
Contact_Info.Email_Address, Contact_Info.Contact_Type,
Contact_Info.Title, Addresses.Street_Line_1, Addresses.Street_Line_2,
Addresses.City, Addresses.State, Addresses.Zip,
Addresses.Address_Type
      FROM Customers INNER JOIN
                        Cust_Addresses ON Customers.Cust_ID =
Cust_Addresses.Cust_ID INNER JOIN
                        Cust_Account_Settings ON
Customers.Account_Settings_ID =
Cust_Account_Settings.Account_Settings_ID INNER JOIN
                        Addresses ON Cust_Addresses.Address_ID =
Addresses.Address_ID INNER JOIN
                        Contact_Info ON Customers.Contact_Info_ID =
Contact_Info.Contact_Info_ID
      RETURN
```

Table 15. Stored Prodedure: SelectCustomerTransactionHist

```
ALTER PROCEDURE dbo.SelectCustomerTransactionHist


      (
      @AccountNum int,
      @Sponsor_ID int
      )
AS
      /* SET NOCOUNT ON */

SELECT     Merchants.Business_Name,
SUM(Cust_Denom_Order_Details.Quantity) AS Expr1,
Cust_Denom_Order_Details.Denomination AS Expr2,
                      Contact_Info.Last_Name, Contact_Info.First_Name,
Customers.Acct_Num_w_Sponsor, Cust_Order_Details.Percent_to_Cust,
                      Cust_Orders.Order_Submission_Date,
Cust_Orders.Payment_Rcvd_Date, Customers.Sponsor_ID
FROM         Customers INNER JOIN
                      Contact_Info ON Customers.Contact_Info_ID =
Contact_Info.Contact_Info_ID INNER JOIN
                      Cust_Orders ON Customers.Cust_ID =
Cust_Orders.Cust_ID INNER JOIN
                      Cust_Order_Details ON Cust_Orders.Cust_Order_ID =
Cust_Order_Details.Cust_Order_ID INNER JOIN
                      Merchants ON Cust_Order_Details.Merchant_ID =
Merchants.Merch_ID INNER JOIN
                      Cust_Denom_Order_Details ON
Cust_Order_Details.Cust_Order_Details_ID =
Cust_Denom_Order_Details.Cust_Order_Detail_ID
WHERE     (Cust_Orders.Cust_Payment_Recieved = 1)
GROUP BY Customers.Acct_Num_w_Sponsor, Merchants.Business_Name,
Cust_Denom_Order_Details.Denomination, Contact_Info.Last_Name,
                      Contact_Info.First_Name,
Customers.Acct_Num_w_Sponsor, Cust_Order_Details.Percent_to_Cust,
Cust_Orders.Order_Submission_Date,
                      Cust_Orders.Payment_Rcvd_Date,
Customers.Sponsor_ID, Cust_Orders.Cust_Received
HAVING      (Customers.Acct_Num_w_Sponsor = @AccountNum) AND
(Customers.Sponsor_ID = @Sponsor_ID)
ORDER BY Cust_Orders.Cust_Received, Merchants.Business_Name
      RETURN
```

Table 16. Stored Prodedure: SelectMerchantDenominations

```
ALTER PROCEDURE dbo.SelectMerchantDenominations
 (
      @Merchant_ID int,
      @Sponsor_ID int
 )
AS
      /* SET NOCOUNT ON */
SELECT     Merch_Denominations_LU.Amount,
Spons_Merchant_List.Sponsor_ID, Spons_Merchant_List.Merchant_ID
```

```
FROM          Merch_Denominations INNER JOIN
                      Merch_Denominations_LU ON
Merch_Denominations.Denom_ID = Merch_Denominations_LU.Denom_ID INNER
JOIN
                      Merchants ON Merch_Denominations.Merch_ID =
Merchants.Merch_ID INNER JOIN
                      Spons_Merchant_List ON Merchants.Merch_ID =
Spons_Merchant_List.Merchant_ID
WHERE      (Spons_Merchant_List.Sponsor_ID = @Sponsor_ID) AND
(Spons_Merchant_List.Merchant_ID = @Merchant_ID)
      RETURN
```

Table 17. Stored Prodedure: SelectMerchantDenominations_AddActive

```
ALTER PROCEDURE dbo.SelectMerchantDenominations_AllActive

AS
      /* SET NOCOUNT ON */
      SELECT      Merch_Denominations_LU.Amount,
Spons_Merchant_List.Sponsor_ID, Spons_Merchant_List.Merchant_ID
      FROM          Merch_Denominations INNER JOIN
                            Merch_Denominations_LU ON
Merch_Denominations.Denom_ID = Merch_Denominations_LU.Denom_ID INNER
JOIN
                            Merchants ON Merch_Denominations.Merch_ID =
Merchants.Merch_ID INNER JOIN
                            Spons_Merchant_List ON Merchants.Merch_ID =
Spons_Merchant_List.Merchant_ID
      WHERE      (Merchants.Merch_Enabled = 1) AND
(Spons_Merchant_List.Sponsor_ID = 1)
```

Table 18. Stored Prodedure: SelectMerchantListForSponsor

```
ALTER PROCEDURE dbo.SelectMerchantListForSponsor
      (
            @Sponsor_ID int = 1,
            @Contact_Type int = 1,
            @Merch_ID int
       )

AS
       SET NOCOUNT ON

IF @Merch_ID > 0
BEGIN
      SELECT      Merchants.Business_Name, Merchants.Merch_Enabled,
Merch_Acct_Settings.Merch_URL,
Merch_Acct_Settings.Merch_Percent_to_Cust,
                        Merch_Acct_Settings.Merch_Percent_to_Sponsor,
Merch_Acct_Settings.Order_As_Needed, Merch_Acct_Settings.Order_In_Bulk,
```

```
                        Merch_Acct_Settings.Local_Purchases_Only,
Merch_Acct_Settings.Face_Value_Discount,
Merch_Acct_Settings.Merch_Comments,
                        Merch_Acct_Settings.Min_Purch_Required,
Merch_Acct_Settings.Max_Purch_Required, Contact_Info.Last_Name,
Contact_Info.First_Name,
                        Contact_Info.MI, Contact_Info.Home_Phone,
Contact_Info.Business_Phone, Contact_Info.Cell_Phone,
Contact_Info.Email_Address, Contact_Info.Title,
                        Addresses.Street_Line_1, Addresses.Street_Line_2,
Addresses.City, Addresses.State, Addresses.Zip,
Merchants.Sponsor_Broker_ID,
                        Merch_Acct_Settings.SCRIP_Type_ID,
Merchants.Merch_ID, Spons_Merchant_List.Spons_Merch_ID,
Merchants.Merch_Acct_Settings_ID,
                        Merch_Type_LU.Merch_Type,
Merch_SCRIP_Type_LU.SCRIP_Type, Merchants.Merch_Type_ID,
Spons_Brokers_LU.Broker_Name
      FROM          Addresses INNER JOIN
                        Merch_Addresses ON Addresses.Address_ID =
Merch_Addresses.Address_ID RIGHT OUTER JOIN
                        Merch_Type_LU INNER JOIN
                        Merchants INNER JOIN
                        Spons_Brokers_LU ON Merchants.Sponsor_Broker_ID =
Spons_Brokers_LU.Spons_Brokers_ID ON
                        Merch_Type_LU.Merch_Type_ID =
Merchants.Merch_Type_ID INNER JOIN
                        Spons_Merchant_List ON Merchants.Merch_ID =
Spons_Merchant_List.Merchant_ID INNER JOIN
                        Contact_Info INNER JOIN
                        Merch_Contacts ON Contact_Info.Contact_Info_ID =
Merch_Contacts.Contact_Info_ID ON
                        Merchants.Merch_ID = Merch_Contacts.Merchant_ID
INNER JOIN
                        Merch_SCRIP_Type_LU INNER JOIN
                        Merch_Acct_Settings ON
Merch_SCRIP_Type_LU.SCRIP_Type_ID = Merch_Acct_Settings.SCRIP_Type_ID
ON
                        Merchants.Merch_Acct_Settings_ID =
Merch_Acct_Settings.Merch_Acct_Settings_ID ON Merch_Addresses.Merch_ID
= Merchants.Merch_ID
      WHERE     (Spons_Merchant_List.Sponsor_ID = @Sponsor_ID) AND
(Contact_Info.Contact_Type = @Contact_Type) AND (Merchants.Merch_ID =
@Merch_ID)
      ORDER BY Merchants.Business_Name
      END
ELSE
      BEGIN


      SELECT Merchants.Business_Name, Merchants.Merch_Enabled,
Merch_Acct_Settings.Merch_URL,
Merch_Acct_Settings.Merch_Percent_to_Cust,
Merch_Acct_Settings.Merch_Percent_to_Sponsor,
Merch_Acct_Settings.Order_As_Needed, Merch_Acct_Settings.Order_In_Bulk,
Merch_Acct_Settings.Local_Purchases_Only,
Merch_Acct_Settings.Face_Value_Discount,
```

```
Merch_Acct_Settings.Merch_Comments,
Merch_Acct_Settings.Min_Purch_Required,
Merch_Acct_Settings.Max_Purch_Required, Contact_Info.Last_Name,
Contact_Info.First_Name, Contact_Info.MI, Contact_Info.Home_Phone,
Contact_Info.Business_Phone, Contact_Info.Cell_Phone,
Contact_Info.Email_Address, Contact_Info.Title,
Addresses.Street_Line_1, Addresses.Street_Line_2, Addresses.City,
Addresses.State, Addresses.Zip, Merchants.Sponsor_Broker_ID,
Merch_Acct_Settings.SCRIP_Type_ID, Merchants.Merch_ID,
Spons_Merchant_List.Spons_Merch_ID, Merchants.Merch_Acct_Settings_ID,
Merch_Type_LU.Merch_Type, Merch_SCRIP_Type_LU.SCRIP_Type,
Merchants.Merch_Type_ID, Spons_Brokers_LU.Broker_Name
      FROM Addresses INNER JOIN Merch_Addresses ON Addresses.Address_ID
= Merch_Addresses.Address_ID RIGHT OUTER JOIN Merch_Type_LU INNER JOIN
Merchants INNER JOIN Spons_Brokers_LU ON Merchants.Sponsor_Broker_ID =
Spons_Brokers_LU.Spons_Brokers_ID ON Merch_Type_LU.Merch_Type_ID =
Merchants.Merch_Type_ID INNER JOIN Spons_Merchant_List ON
Merchants.Merch_ID = Spons_Merchant_List.Merchant_ID INNER JOIN
Contact_Info INNER JOIN Merch_Contacts ON Contact_Info.Contact_Info_ID
= Merch_Contacts.Contact_Info_ID ON Merchants.Merch_ID =
Merch_Contacts.Merchant_ID INNER JOIN Merch_SCRIP_Type_LU INNER JOIN
Merch_Acct_Settings ON Merch_SCRIP_Type_LU.SCRIP_Type_ID =
Merch_Acct_Settings.SCRIP_Type_ID ON Merchants.Merch_Acct_Settings_ID =
Merch_Acct_Settings.Merch_Acct_Settings_ID ON Merch_Addresses.Merch_ID
= Merchants.Merch_ID WHERE (Spons_Merchant_List.Sponsor_ID =
@Sponsor_ID) AND (Contact_Info.Contact_Type = @Contact_Type) ORDER BY
Merchants.Business_Name

      END

RETURN
```

Table 19. Stored Prodedure: SelectMerchType_All

```
ALTER PROCEDURE dbo.SelectMerchType_All

AS
      /* SET NOCOUNT ON */
SELECT     Merch_Type_ID, Merch_Type
FROM         Merch_Type_LU
ORDER BY Merch_Type

      RETURN
```

Table 20. Stored Prodedure: SelectMerchTypeForMerchant

```
ALTER PROCEDURE dbo.SelectMerchTypeForMerchant
      (
            @Merch_ID int
      )
AS
      /* SET NOCOUNT ON */
```

```
SELECT      Merch_Type_ID
FROM          Merchants
WHERE      (Merch_ID = @Merch_ID)

      RETURN
```

Table 21. Stored Prodedure: SelectOrdersToProcess

```
ALTER PROCEDURE dbo.SelectOrdersToProcess
   /*
      (
      @parameter1 int = 5,
      @parameter2 datatype OUTPUT
      )
      */
AS
      /* SET NOCOUNT ON */

SELECT      Merchants.Business_Name,
SUM(Cust_Denom_Order_Details.Quantity) AS Expr1,
Cust_Denom_Order_Details.Denomination AS Expr2
FROM          Customers INNER JOIN
                      Contact_Info ON Customers.Contact_Info_ID =
Contact_Info.Contact_Info_ID INNER JOIN
                      Cust_Orders ON Customers.Cust_ID =
Cust_Orders.Cust_ID INNER JOIN
                      Cust_Order_Details ON Cust_Orders.Cust_Order_ID =
Cust_Order_Details.Cust_Order_ID INNER JOIN
                      Merchants ON Cust_Order_Details.Merchant_ID =
Merchants.Merch_ID LEFT OUTER JOIN
                      Cust_Denom_Order_Details ON
Cust_Order_Details.Cust_Order_Details_ID =
Cust_Denom_Order_Details.Cust_Order_Detail_ID
WHERE      (Cust_Orders.Cust_Payment_Recieved = 1) AND
(Cust_Orders.Cust_Received = 0)
GROUP BY Merchants.Business_Name, Cust_Denom_Order_Details.Denomination
ORDER BY Merchants.Business_Name
      RETURN
```

Table 22. Stored Prodedure: SelectPendingCustomerOrders

```
ALTER PROCEDURE dbo.SelectPendingCustomerOrders
   /*
      (
      @parameter1 int = 5,
      @parameter2 datatype OUTPUT
      )
      */
AS
      /* SET NOCOUNT ON */
```

```
SELECT     Merchants.Business_Name,
SUM(Cust_Denom_Order_Details.Quantity) AS Expr1,
Cust_Denom_Order_Details.Denomination AS Expr2,
                      Contact_Info.Last_Name, Contact_Info.First_Name,
Customers.Acct_Num_w_Sponsor
FROM         Customers INNER JOIN
                      Contact_Info ON Customers.Contact_Info_ID =
Contact_Info.Contact_Info_ID INNER JOIN
                      Cust_Orders ON Customers.Cust_ID =
Cust_Orders.Cust_ID INNER JOIN
                      Cust_Order_Details ON Cust_Orders.Cust_Order_ID =
Cust_Order_Details.Cust_Order_ID INNER JOIN
                      Merchants ON Cust_Order_Details.Merchant_ID =
Merchants.Merch_ID INNER JOIN
                      Cust_Denom_Order_Details ON
Cust_Order_Details.Cust_Order_Details_ID =
Cust_Denom_Order_Details.Cust_Order_Detail_ID
WHERE     (Cust_Orders.Cust_Payment_Recieved = 1) AND
(Cust_Orders.Cust_Received = 0)
GROUP BY Customers.Acct_Num_w_Sponsor, Merchants.Business_Name,
Cust_Denom_Order_Details.Denomination, Contact_Info.Last_Name,
Contact_Info.First_Name,
                      Customers.Acct_Num_w_Sponsor
ORDER BY Contact_Info.Last_Name
      RETURN
```

Table 23. Stored Prodedure: SelectPendingOrders

```
ALTER PROCEDURE dbo.SelectPendingOrders
      /*
      (
      @parameter1 int = 5,
      @parameter2 datatype OUTPUT
      )
      */
AS
      /* SET NOCOUNT ON */

SELECT     Customers.Account_Settings_ID, Customers.Comp_AcctNum,
Customers.Sponsor_ID, Cust_Orders.Order_Confirmation_Num,
Contact_Info.Last_Name,
                      Contact_Info.First_Name,
Cust_Orders.Order_Submission_Date, Cust_Orders.Cust_Payment_Recieved,
Cust_Orders.Check_Num,
                      Customers.Acct_Num_w_Sponsor,
Customers.Contact_Info_ID, Cust_Orders.Order_Processed_Date,
Cust_Orders.Cust_Received,
                      Cust_Orders.Cust_Order_Rcvd_Date,
Cust_Orders.Payment_Rcvd_Date, Cust_Orders.Cust_Order_ID,
Cust_Orders.TotalAmountOfOrder,
                      Cust_Order_Comments.Comment_Text
FROM         Customers INNER JOIN
                      Contact_Info ON Customers.Contact_Info_ID =
Contact_Info.Contact_Info_ID INNER JOIN
```

```
                        Cust_Orders ON Customers.Cust_ID =
Cust_Orders.Cust_ID INNER JOIN
                        Cust_Order_Comments ON Cust_Orders.Comment_ID =
Cust_Order_Comments.Comment_ID
WHERE      (Cust_Orders.Order_Processed_Date IS NULL)
      RETURN
```

Table 24. Stored Prodedure: SelectScripOrderList

```
ALTER PROCEDURE [dbo].[SelectScripOrderList]

      (
      @Sponsor_ID int
      )

AS
      /* SET NOCOUNT ON  */
/* ================================================*/
SELECT     Merchants.Merch_ID, Merchants.Business_Name,
Merch_SCRIP_Type_LU.SCRIP_Type, Merch_Acct_Settings.Merch_URL,
                        Merch_Acct_Settings.Merch_Percent_to_Cust,
Merch_Acct_Settings.Merch_Percent_to_Sponsor,
Merch_Acct_Settings.Local_Purchases_Only,
                        Merch_Acct_Settings.Merch_Comments,
Merch_Type_LU.Merch_Type, Merch_Acct_Settings.Min_Purch_Required,
                        Merch_Acct_Settings.Max_Purch_Required,
Merch_Denominations.Merchant_Denom_ID, Merch_Denominations_LU.Amount,
                        Merch_Denominations.Denom_ID
FROM          Merchants INNER JOIN
                        Merch_Acct_Settings ON
Merchants.Merch_Acct_Settings_ID =
Merch_Acct_Settings.Merch_Acct_Settings_ID INNER JOIN
                        Spons_Merchant_List ON Merchants.Merch_ID =
Spons_Merchant_List.Merchant_ID AND
                        Merchants.Merch_ID =
Spons_Merchant_List.Merchant_ID INNER JOIN
                        Merch_SCRIP_Type_LU ON
Merch_Acct_Settings.SCRIP_Type_ID = Merch_SCRIP_Type_LU.SCRIP_Type_ID
INNER JOIN
                        Merch_Type_LU ON Merchants.Merch_Type_ID =
Merch_Type_LU.Merch_Type_ID INNER JOIN
                        Merch_Denominations ON Merchants.Merch_ID =
Merch_Denominations.Merch_ID INNER JOIN
                        Merch_Denominations_LU ON
Merch_Denominations.Denom_ID = Merch_Denominations_LU.Denom_ID
WHERE      (Spons_Merchant_List.Sponsor_ID = @Sponsor_ID) AND
(Merchants.Merch_Enabled = 1)
ORDER BY Merchants.Business_Name


      RETURN
```

Table 25. Stored Prodedure: SelectScripTypes_All

```
ALTER PROCEDURE dbo.SelectScripTypes_All

AS
      /* SET NOCOUNT ON */

      SELECT SCRIP_Type_ID, SCRIP_Type FROM Merch_SCRIP_Type_LU

      RETURN
```

Table 26. Stored Prodedure: SelectSponsorID_Info

```
ALTER PROCEDURE dbo.SelectSponsorID_Info

      (
      @UserName nvarchar(50)
      )

AS
      /* SET NOCOUNT ON */

SELECT     Customers.Sponsor_ID, Contact_Info.Email_Address,
Contact_Info.First_Name, Customers.Cust_ID, Customers.Cust_Enabled,
                      Customers.Acct_Num_w_Sponsor
FROM         Customers INNER JOIN
                      Contact_Info ON Customers.Contact_Info_ID =
Contact_Info.Contact_Info_ID
WHERE     (Contact_Info.UserID = @UserName)


      RETURN
```

Table 27. Stored Prodedure: UpdateAddress

```
ALTER PROCEDURE dbo.UpdateAddress
      (
/* Insert new record in the Address Table */
      @Address_ID int,
      @Street_Line_1 varchar(50),
      @Street_Line_2 varchar(50) = NULL,
      @City varchar(50),
      @State varchar(2),
      @Zip varchar(10),
      @Address_Type int

      )

AS
      /* SET NOCOUNT ON */

      UPDATE Addresses
```

```
SET

Street_Line_1 = @Street_Line_1,
Street_Line_2 = @Street_Line_2,
City =      @City,
State = @State,
Zip =@Zip,
Address_Type = @Address_Type

WHERE Address_ID = @Address_ID

RETURN
```

Table 28. Stored Prodedure: UpdateContactInfo

```
ALTER PROCEDURE dbo.UpdateContactInfo
      /* Input Parameters for Merch_Acct_Settings */

      (

      @Contact_Info_ID int,
      @Last_Name nvarchar(50),
      @First_Name nvarchar(50),
      @MI nvarchar(1),
      @Home_Phone nvarchar(12),
      @Business_Phone nvarchar(12),
      @Cell_Phone nvarchar(12),
      @Email_Address nvarchar(75),
      @Contact_Type smallint
      )

AS

/*    SET NOCOUNT ON  */

      IF @Contact_Info_ID IS NOT NULL
            BEGIN
                  UPDATE dbo.Contact_Info

                  SET

                  Last_Name = @Last_Name,
                  First_Name = @First_Name,
                  MI = @MI,
                  Home_Phone = @Home_Phone,
                  Business_Phone = @Business_Phone,
                  Cell_Phone = @Cell_Phone,
                  Email_Address = @Email_Address,
                  Contact_Type = @Contact_Type

                  WHERE Contact_Info_ID = @Contact_Info_ID

                  RETURN
            END
```

Table 29. Stored Prodedure: UpdateDenominations

```
ALTER PROCEDURE dbo.UpdateDenominations
      (
            @Merch_ID int,
            @Denomination_Amount_to_Add int
      )

AS
      /* SET NOCOUNT ON */


IF @Denomination_Amount_to_Add IS NOT NULL
      INSERT INTO Merch_Denominations(Merch_ID, Denom_ID) VALUES
(@Merch_ID, @Denomination_Amount_to_Add)


      RETURN
```

Table 30. Stored Prodedure: UpdateMerchant

```
ALTER PROCEDURE dbo.UpdateMerchant
      (

      /* Input Parameters for Merch_Acct_Settings */

      @SCRIP_Type_ID int,
      @Sponsor_ID int,
      @Merch_URL varchar(255),
      @Merch_Percent_to_Cust FLOAT,
      @Merch_Percent_to_Sponsor FLOAT,
      @Order_As_Needed bit,
      @Order_In_Bulk bit,
      @Local_Purchases_Only bit,
      @Face_Value_Discount bit,
      @Merch_Comments ntext,
      @Min_Purch_Required int,
      @Max_Purch_Required int,


      /* Input Parameters for the Merchants table */
      @Business_Name nvarchar(75),
      @Merch_Enabled bit,
      @Sponsor_Broker_ID int,
      @Merch_ID int,

      /* Insert new Address record (Physical Address) */
      @Address_ID int,
      @Street_Line_1 varchar(50),
      @Street_Line_2 varchar(50),
      @City varchar(50),
      @State varchar(2),
      @Zip varchar(10),
      @Address_Type int,
```

```
        @Last_Name nvarchar(50),
        @First_Name nvarchar(50),
        @MI nvarchar(1),
        @Home_Phone nvarchar(12),
        @Business_Phone nvarchar(12),
        @Cell_Phone nvarchar(12),
        @Email_Address nvarchar(75),
        @Contact_Type smallint,
        @Title nvarchar(75),
        @Merch_Type_ID int

        )

AS
        /* SET NOCOUNT ON */

DECLARE @Merch_Acct_Settings_ID nvarchar(100)
/*DECLARE @Merch_Type_ID nvarchar(100)*/
DECLARE @Contact_Info_ID nvarchar(100)
DECLARE @SQLString nvarchar(500)
DECLARE @ParmDefinition nvarchar(500)
DECLARE @Primary_Contact int

Set @Primary_Contact = 1

IF @Sponsor_Broker_ID IS NULL
      BEGIN
            Set @Sponsor_Broker_ID = 1
      END

IF @SCRIP_Type_ID IS NULL
      BEGIN
            Set @SCRIP_Type_ID = 1
      END


/*Retrieve the Acct_Settings_ID required to make updates to the
Merch_Acct_Settings table*/
SET @SQLString = N'SELECT @Merch_Acct_Settings_ID_OUT =
Merch_Acct_Settings_ID FROM Merchants WHERE Merch_ID = @Merch_ID'
SET @ParmDefinition = N'@Merch_ID int, @Merch_Acct_Settings_ID_OUT int
OUTPUT'
EXEC sp_executesql @SQLString, @ParmDefinition, @Merch_ID,
@Merch_Acct_Settings_ID_OUT=@Merch_Acct_Settings_ID OUTPUT
SELECT @Merch_Acct_Settings_ID


SET @SQLString = N'SELECT @Address_ID_OUT = Address_ID FROM
Merch_Addresses WHERE Merch_ID = @Merch_ID'
SET @ParmDefinition = N'@Merch_ID int, @Address_ID_OUT int OUTPUT'
EXEC sp_executesql @SQLString, @ParmDefinition, @Merch_ID,
@Address_ID_OUT=@Address_ID OUTPUT
SELECT @Address_ID

SET @SQLString = N'SELECT @Contact_Info_ID_OUT = Contact_Info_ID FROM
Merch_Contacts WHERE Merchant_ID = @Merch_ID'
SET @ParmDefinition = N'@Merch_ID int, @Contact_Info_ID_OUT int OUTPUT'
```

```
EXEC sp_executesql @SQLString, @ParmDefinition, @Merch_ID,
@Contact_Info_ID_OUT=@Contact_Info_ID OUTPUT
SELECT @Contact_Info_ID


UPDATE    Merch_Acct_Settings
SET      SCRIP_Type_ID = @SCRIP_Type_ID,
            Merch_URL = @Merch_URL,
            Merch_Percent_to_Cust = @Merch_Percent_to_Cust,
         Merch_Percent_to_Sponsor = @Merch_Percent_to_Sponsor,
         Order_As_Needed = @Order_As_Needed,
         Order_In_Bulk = @Order_In_Bulk,
         Local_Purchases_Only = @Local_Purchases_Only,
         Face_Value_Discount = @Face_Value_Discount,
         Merch_Comments = @Merch_Comments,
         Min_Purch_Required = @Min_Purch_Required,
         Max_Purch_Required = @Max_Purch_Required
WHERE      (Merch_Acct_Settings_ID = @Merch_Acct_Settings_ID)


UPDATE     Merchants
SET        Merch_Acct_Settings_ID = Merch_Acct_Settings_ID,
               Merch_Type_ID = @Merch_Type_ID,
               Business_Name = @Business_Name,
           Merch_Enabled = @Merch_Enabled,
           Sponsor_Broker_ID = @Sponsor_Broker_ID
WHERE      (Merch_ID = @Merch_ID)


      Exec UpdateAddress
      @Address_ID,
      @Street_Line_1,
      @Street_Line_2,
      @City,
      @State,
      @Zip,
      @Address_Type


Exec UpdateContactInfo

      @Contact_Info_ID,
      @Last_Name,
      @First_Name,
      @MI,
      @Home_Phone,
      @Business_Phone,
      @Cell_Phone,
      @Email_Address,
      @Primary_Contact

      RETURN
```

Table 31. Stored Prodedure: UpdateOrder_Pay_Rcvd

```
ALTER PROCEDURE dbo.UpdateOrder_Pay_Rcvd
      (

      /* Input Parameters for Merch_Acct_Settings */

      @Payment_Rcvd bit,
      @Cust_Rcvd_Order bit,
      @Cust_Order_ID int,
      @Check_Number nvarchar(6),
      @Payment_Rcvd_Date as datetime,
      @Cust_Rcvd_Order_Date as datetime
      )

AS
       SET NOCOUNT ON


UPDATE    Cust_Orders
SET     Cust_Payment_Recieved = @Payment_Rcvd,
            Payment_Rcvd_Date = @Payment_Rcvd_Date,
            Cust_Received = @Cust_Rcvd_Order,
            Cust_Order_Rcvd_Date = @Cust_Rcvd_Order_Date,
            Check_Num = @Check_Number
WHERE     (Cust_Order_ID = @Cust_Order_ID)


      RETURN
```

Table 32. Code: Web.config

```
<?xml version="1.0"?>
<!--
    Note: As an alternative to hand editing this file you can use the
    web admin tool to configure settings for your application. Use
    the Website->Asp.Net Configuration option in Visual Studio.
    A full list of settings and comments can be found in
    machine.config.comments usually located in
    \Windows\Microsoft.Net\Framework\v2.x\Config
-->
<configuration
xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
      <!--xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">-
->
      <appSettings>
            <!--<add key="CrystalImageCleaner-AutoStart" value="true"
/>
     <add key="CrystalImageCleaner-Sleep" value="60000" />
     <add key="CrystalImageCleaner-Age" value="120000" />-->
      </appSettings>
```

```
  <connectionStrings>
    <clear />
    <!--<add name="CompuSCRIP" connectionString="Data
Source=CD023507902-1\SQLEXPRESS;Initial Catalog=compu32_CompuSCRIP;User
ID=Steve; Password=93kx250;" />-->
    <add name="CompuSCRIP" connectionString="Data
Source=209.200.235.3;database=Compu32_CompuSCRIP;User ID=Steve;
Password=93kx250"/>
  </connectionStrings>
      <system.web>

    <authentication mode="Forms"/>
          <roleManager enabled="true">
                <providers>
                      <clear/>
                      <add name="AspNetSqlRoleProvider"
connectionStringName="CompuSCRIP" applicationName="/"
type="System.Web.Security.SqlRoleProvider, System.Web, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"/>
                </providers>
          </roleManager>
          <profile>
                <providers>
                      <clear/>
                      <add name="AspNetSqlProfileProvider"
connectionStringName="CompuSCRIP" applicationName="/"
type="System.Web.Profile.SqlProfileProvider, System.Web,&#xA;
Version=2.0.0.0, Culture=neutral,&#xA;
PublicKeyToken=b03f5f7f11d50a3a"/>
                </providers>
                <properties>
                      <add name="_LastName"/>
                      <add name="_FirstName"/>
                      <add name="_Email"/>
                      <add name="_UserID"/>
                      <add name="_SponsorID"/>
                      <add name="_CustID"/>
                      <add name="_Enabled"/>
                      <add name="_AcctNumWithSponsor"/>
                </properties>
          </profile>
          <membership>
                <providers>
                      <clear/>
                      <add name="AspNetSqlMembershipProvider"
type="System.Web.Security.SqlMembershipProvider, System.Web,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"
connectionStringName="CompuSCRIP" enablePasswordRetrieval="false"
enablePasswordReset="true" requiresQuestionAndAnswer="true"
applicationName="/" requiresUniqueEmail="true" passwordFormat="Hashed"
maxInvalidPasswordAttempts="5" minRequiredPasswordLength="7"
minRequiredNonalphanumericCharacters="0" passwordAttemptWindow="10"
passwordStrengthRegularExpression=""/>
                </providers>
          </membership>
          <!--
          Set compilation debug="true" to insert debugging
```

```
            symbols into the compiled page. Because this
            affects performance, set this value to true only
            during development.

            Visual Basic options:
            Set strict="true" to disallow all data type conversions
            where data loss can occur.
            Set explicit="true" to force declaration of all variables.
        -->
            <compilation debug="true" strict="false" explicit="true">
    <assemblies>
     <add assembly="System.Design, Version=2.0.0.0, Culture=neutral,
 PublicKeyToken=B03F5F7F11D50A3A" />
     <add assembly="System.Windows.Forms, Version=2.0.0.0,
 Culture=neutral, PublicKeyToken=B77A5C561934E089" />
     <add assembly="System.Configuration.Install, Version=2.0.0.0,
 Culture=neutral, PublicKeyToken=B03F5F7F11D50A3A" />
    </assemblies>
    <buildProviders>
     <add extension=".rdlc" type="Microsoft.Reporting.RdlBuildProvider,
 Microsoft.ReportViewer.Common, Version=8.0.0.0, Culture=neutral,
 PublicKeyToken=b03f5f7f11d50a3a" />
    </buildProviders>
   </compilation>
            <pages>
                  <namespaces>
                        <clear/>
                        <add namespace="System"/>
                        <add namespace="System.Collections"/>
                        <add
 namespace="System.Collections.Specialized"/>
                        <add namespace="System.Configuration"/>
                        <add namespace="System.Text"/>
                        <add
 namespace="System.Text.RegularExpressions"/>
                        <add namespace="System.Web"/>
                        <add namespace="System.Web.Caching"/>
                        <add namespace="System.Web.SessionState"/>
                        <add namespace="System.Web.Security"/>
                        <add namespace="System.Web.Profile"/>
                        <add namespace="System.Web.UI"/>
                        <add namespace="System.Web.UI.WebControls"/>
                        <add
 namespace="System.Web.UI.WebControls.WebParts"/>
                        <add namespace="System.Web.UI.HtmlControls"/>
                  </namespaces>
            </pages>
            <!--
      </system.web>
</configuration>
```

Table 33. Code: ScripOrder.aspx.vb

```
Imports System.Data.SqlClient
```

```
Imports System.Drawing
Partial Class Customer_Scrip_Order
    Inherits System.Web.UI.Page


    Const m_intCurrentMerchID As Integer = 0
    Const m_intBusNameCol As Integer = 1
    Const m_intNumToOrderCol As Integer = 2
    Const m_intAmountCol As Integer = 3
    Const m_intSubTotalCol As Integer = 4
    Const m_intPercentToCustCol As Integer = 5
    Const m_intPercentToSponsor As Integer = 6
    Const m_intAmountToCust As Integer = 7
    Const m_intDebitCertCol As Integer = 8
    Const m_intLocalPurchCol As Integer = 9

    Const m_intBusTypeCol As Integer = 10
    Const m_intCommentsCol As Integer = 11

    Dim m_MerchantsFound As Boolean = True
    Dim m_strConfirmationNum As String



    Protected Sub GridView1_PreRender(ByVal sender As Object, ByVal e
As System.EventArgs) Handles GridView1.PreRender

''*************************************************************
        ''Author:  Steve Williams
        ''Date:    10/15/06
        ''Description:  This proc will read through the gridview
control row by row
        ''              prior to it being displayed to the user.
Because the same row
        ''              of data is repeated for each business based on
that denomination
        ''              it will erase all duplicate row info, except
for the denomination
        ''              anmount for that row.
        ''Pre:  Gridview shows duplicate rows for each business based
on denomination
        ''      I.e.   Ace Hardware  25
        ''             Ace Hardware  50
        ''             Ace Hardware 100
        ''      Changes to:
        ''             Ace Hardware 25
        ''                          50
        ''                         100
        ''Post:  All duplicate rows of a business are erased except for
the denomination
        ''       amount which is left.

''*************************************************************
        Dim intRowCount As Integer
        Dim intRowNum As Integer
        Dim strCurBusName As String
        Dim strLastBusName As String
```

```
        Dim intNumColumns As Integer
        Dim intColNum As Integer
        Dim strCurBackColor As System.Drawing.Color
        Dim strLastBackColor As System.Drawing.Color



        'Count the number of rows in the Gridview
        intRowCount = GridView1.Rows.Count - 1
        intNumColumns = GridView1.Columns.Count - 1

        strLastBusName = Space(0)

        If intRowCount > 0 Then
            m_MerchantsFound = True 'There are merchants to display
            'For each row in the grid
            For intRowNum = 0 To intRowCount
                'Read in the next Business name from the grid row
                strCurBusName =
GridView1.Rows.Item(intRowNum).Cells(m_intBusNameCol).Text
                strCurBackColor =
GridView1.Rows.Item(intRowNum).BackColor
                'If the current business name read in is the same as
the last then
                'don't display it because it is a dup
                If strCurBusName = strLastBusName Then
                    GridView1.Rows.Item(intRowNum).BackColor =
strLastBackColor

GridView1.Rows.Item(intRowNum).Cells(m_intBusNameCol).Text = Space(0)

                    'Erase the remaining cells in the the row
                    For intColNum = m_intDebitCertCol To intNumColumns

GridView1.Rows.Item(intRowNum).Cells(intColNum).Text = Space(0)

                    Next

                End If
                'Assign the bus name in the row just read to the last
Bus name var
                strLastBusName = strCurBusName
                strLastBackColor = strCurBackColor

            Next
            GridView1.FooterRow.Cells(m_intBusNameCol).Text = "Grand
Totals"
        Else
            m_MerchantsFound = False


            Response.Write("<script>alert('There are no merchants
listed for this organization.  Please contact the Administrator for
your organization.');</script>")
            Me.Server.Transfer("CustHome.aspx")

        End If
```

```
    End Sub

    Protected Function TotalTheOrders() As Double

        Dim intRowCount As Integer
        Dim intRowNum As Integer
        Dim intNumToOrder As Integer
        Dim objNumOrderTextBox As TextBox
        Dim objDenomAmount As Label
        Dim dblOrderAmt As Double
        Dim dblGrandTotal As Double
        Dim dblSubTotalForMerchant As Double
        Dim strSubTotalForMerchant As String
        Dim dblSubTotalForCust As Double
        Dim strSubTotalForCust As String
        Dim dblPercentToCust As Double
        Dim strGrandTotal As String
        Dim strGrandTotToCust As String
        Dim dblGrandTotToCust As Double
        Dim intGrandTotToOrder As Integer
        Dim strGrandTotToOrder As String

        objNumOrderTextBox = New TextBox
        objDenomAmount = New Label

        intRowCount = GridView1.Rows.Count - 1
        dblOrderAmt = (0.0)
        dblGrandTotToCust = 0
        strGrandTotal = ""
        strGrandTotToCust = ""


        lblConfirmationHeading.Text = ""
        lblConfirmationNum.Text = ""
        lblConfirmationNum.BackColor = Color.White


        'For each row in the grid
        For intRowNum = 0 To intRowCount
            objNumOrderTextBox =
GridView1.Rows.Item(intRowNum).FindControl("txtNumberToOrder")


            If objNumOrderTextBox.Text <> "" Then

                'Read order info from screen in prep for calculations
                intNumToOrder =
Convert.ToInt16(objNumOrderTextBox.Text)
                intGrandTotToOrder = intGrandTotToOrder + intNumToOrder

                dblOrderAmt =
GridView1.Rows.Item(intRowNum).Cells(m_intAmountCol).Text
                dblPercentToCust =
GridView1.Rows.Item(intRowNum).Cells(m_intPercentToCustCol).Text
```

```
                'Calculate the sub total amount of each denomination
for merchant
                dblSubTotalForMerchant = intNumToOrder * dblOrderAmt
                'Calc the amount owed to customer for each customer
                dblSubTotalForCust = (dblSubTotalForMerchant *
dblPercentToCust) * 0.01


                dblGrandTotal = dblGrandTotal + dblSubTotalForMerchant
                strSubTotalForMerchant =
dblSubTotalForMerchant.ToString("$#,##0.00")
                strSubTotalForCust =
dblSubTotalForCust.ToString("$#,##0.00")
                strGrandTotal = strGrandTotal + strSubTotalForMerchant
                dblGrandTotToCust = dblGrandTotToCust +
dblSubTotalForCust


                'Populate sub totals for each row in the grid

GridView1.Rows.Item(intRowNum).Cells(m_intSubTotalCol).Text =
strSubTotalForMerchant

GridView1.Rows.Item(intRowNum).Cells(m_intAmountToCust).Text =
strSubTotalForCust
            End If
        Next

        strGrandTotal = dblGrandTotal.ToString("$#,##0.00")
        strGrandTotToCust = dblGrandTotToCust.ToString("$#,##0.00")
        strGrandTotToOrder = intGrandTotToOrder.ToString

        GridView1.FooterRow.Cells(m_intNumToOrderCol).Text =
strGrandTotToOrder
        GridView1.FooterRow.Cells(m_intSubTotalCol).Text =
strGrandTotal
        GridView1.FooterRow.Cells(m_intAmountToCust).Text =
strGrandTotToCust

        TotalTheOrders = dblGrandTotal

    End Function


    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnTotalOrder.Click
        Call TotalTheOrders()
    End Sub


    Protected Sub btnSubmit_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnSubmit.Click
        Dim dblGrandTotalForOrder As Double
        Dim strConfirmationNum As String
        Dim dteDate As Date = DateTime.Now
        Dim strCheckOrEFTNum As String
        Dim intNew_Cust_Order_ID As Int64
```

```
        Dim strComments As String
        Dim bolFormCleared As Boolean = False

        btnSubmit.Enabled = False
        btnSubmit.Text = "Processing"

        dblGrandTotalForOrder = TotalTheOrders()
        strConfirmationNum = GenerateConfirmationNum()
        strCheckOrEFTNum = txtCheckNum.Text
        strComments = Server.HtmlEncode(txtComments.Text)


        intNew_Cust_Order_ID =
AddNewOrderToCustOrderTable(strConfirmationNum, strCheckOrEFTNum,
strComments, dblGrandTotalForOrder)
        If dblGrandTotalForOrder <> 0.0 Then
            Call SaveAndSubmitOrder(dblGrandTotalForOrder,
intNew_Cust_Order_ID)



            Response.Write("<script>alert('Successful Completion:
Please write the confirmation number, shown at the bottom of the
screen, in the memo field of your check.');</script>")

            lblConfirmationHeading.Text = "Confirmation #"
            lblConfirmationNum.Text = strConfirmationNum
            lblConfirmationNum.BackColor = Color.GreenYellow


            btnSubmit.Enabled = True
            btnSubmit.Text = "Submit Order"



        Else
            Response.Write("<script>alert('Your order must be greater
than $0.00.  Please make selections and re-submit');</script>")

        End If

    End Sub


    Protected Sub SaveAndSubmitOrder(ByVal dblGrandTotalForOrder, ByVal
intNew_Cust_Order_ID)

        Dim intRowCount As Integer
        Dim intRowNum As Integer
        Dim objNumOrderTextBox As TextBox
        Dim intQuantity As Integer
        Dim dblDenomAmount As Double
        Dim intMerchID As Int64
        Dim dblPercentToCust As Double
        Dim dblPercentToSponsor As Double
        Dim intLastMerchID As Integer
        Dim intNewCust_Order_Details_ID As Int64
```

```
        objNumOrderTextBox = New TextBox

        'Count the number of rows in the Gridview
        intRowCount = GridView1.Rows.Count - 1

        objNumOrderTextBox =
GridView1.Rows.Item(intRowNum).FindControl("txtNumberToOrder")

        'For each row in the grid
        For intRowNum = 0 To intRowCount

            objNumOrderTextBox =
GridView1.Rows.Item(intRowNum).FindControl("txtNumberToOrder")

            'If user entered a value in the Number to Order text box
then
            If objNumOrderTextBox.Text <> "" Then

                'Read in the next Business name from the grid row
                intMerchID =
Convert.ToInt64(GridView1.Rows.Item(intRowNum).Cells(m_intCurrentMerchI
D).Text)
                intQuantity = Convert.ToInt16(objNumOrderTextBox.Text)
                dblDenomAmount =
Convert.ToDouble(GridView1.Rows.Item(intRowNum).Cells(m_intAmountCol).T
ext)
                dblPercentToCust =
Convert.ToDouble(GridView1.Rows.Item(intRowNum).Cells(m_intPercentToCus
tCol).Text)
                If
GridView1.Rows.Item(intRowNum).Cells(m_intPercentToSponsor).Text <> ""
Then
                    dblPercentToSponsor =
Convert.ToDouble(GridView1.Rows.Item(intRowNum).Cells(m_intPercentToSpo
nsor).Text)
                Else
                    dblPercentToSponsor = 0.0
                End If



                intNewCust_Order_Details_ID =
AddNew_Cust_Order_Details(intNew_Cust_Order_ID, _

intMerchID, _

dblPercentToCust, _

dblPercentToSponsor, _

dblGrandTotalForOrder)

                Call
AddNew_Cust_Denom_Order_Details(intNewCust_Order_Details_ID, _
                                                    intQuantity, _
```

```
                                                        dblDenomAmount)
                intLastMerchID = intMerchID

            End If

        Next

    End Sub

    Protected Function AddNew_Cust_Order_Details(ByVal
intNew_Cust_Order_ID, _
                                                 ByVal intMerchID, _
                                                 ByVal
dblPercentToCust, _
                                                 ByVal
dblPercentToSponsor, _
                                                 ByVal
dblGrandTotalForOrder) As Int64

        Dim conn As Data.SqlClient.SqlConnection = Nothing
        Dim intNewCustOrderDetailsID As Int64


        Try
            'Retrieve value in 1st column of result set
            Dim intNewCustOrderDetailsIDColNum As Integer = 0

            conn = New SqlConnection

            conn.ConnectionString =
ConfigurationManager.ConnectionStrings("CompuSCRIP").ConnectionString

            conn.Open()

            Dim cmd As New SqlCommand("EXECUTE
AddNewOrder_Cust_Order_Details " & _
                                      intNew_Cust_Order_ID & "," & _
                                      intMerchID & "," & _
                                      dblPercentToCust & "," & _
                                      dblPercentToSponsor & "," & _
                                      dblGrandTotalForOrder, conn)

            Dim rdr As SqlDataReader = cmd.ExecuteReader()

            If rdr.HasRows = True Then
                While rdr.Read
                    intNewCustOrderDetailsID =
rdr.GetInt32(intNewCustOrderDetailsIDColNum)
                End While

            End If
            rdr.Close()
            rdr = Nothing
            cmd.Dispose()

            Return intNewCustOrderDetailsID
```

```
        Finally
            If Not conn Is Nothing Then
                If conn.State <> Data.ConnectionState.Closed Then
                    conn.Close()
                    conn.Dispose()
                End If
            End If
        End Try
    End Function


    Protected Sub AddNew_Cust_Denom_Order_Details(ByVal
intNewCust_Order_Details_ID, _
                                                  ByVal
intQuantity, _
                                                  ByVal
dblDenomAmount)

        Dim conn As Data.SqlClient.SqlConnection = Nothing

        Try
            'Retrieve value in 1st column of result set
            Dim intNewDenomDetailsIDColNum As Integer = 0

            conn = New SqlConnection

            conn.ConnectionString =
ConfigurationManager.ConnectionStrings("CompuSCRIP").ConnectionString

            conn.Open()

            Dim cmd As New SqlCommand("EXECUTE
AddNewOrder_Cust_Denom_Order_Details " & _
                                      intNewCust_Order_Details_ID &
"," & _
                                      intQuantity & "," & _
                                      dblDenomAmount, conn)

            Dim rdr As SqlDataReader = cmd.ExecuteReader()

        Finally
            If Not conn Is Nothing Then
                If conn.State <> Data.ConnectionState.Closed Then
                    conn.Close()
                    conn.Dispose()
                End If
            End If
        End Try
    End Sub



    Protected Function GenerateConfirmationNum() As String
        Dim strConfirmationNum As String
        Dim strDayOfYear As String
        Dim strHour As String
```

```
        Dim strMinute As String

        strDayOfYear = DateTime.Now.DayOfYear.ToString
        strHour = DateTime.Now.Hour.ToString
        strMinute = DateTime.Now.Minute.ToString

        strConfirmationNum = "C" & Profile._SponsorID & "-" &
 Profile._CustID & "-" & strDayOfYear & strHour & strMinute
        GenerateConfirmationNum = strConfirmationNum


    End Function


    Protected Function AddNewOrderToCustOrderTable(ByVal
strConfirmationNum As String, _
                                                   ByVal
strCheckOrEFTNum As String, _
                                                   ByVal strComments
As String, _
                                                   ByVal
dblGrandTotalForOrder As Double) As Int32
        Dim conn As Data.SqlClient.SqlConnection = Nothing
        Dim intNewCustOrderIDNum As Int32
        Dim intCustID As Int32


        Try
            'Retrieve value in 1st column of result set
            Dim intNewCustOrderIDColNum As Integer = 0

            If dblGrandTotalForOrder > 0.0 Then
                conn = New SqlConnection
                conn.ConnectionString =
ConfigurationManager.ConnectionStrings("CompuSCRIP").ConnectionString
                conn.Open()

                intCustID = Convert.ToInt32(Profile._CustID)

                strComments = Replace(strComments, "'", "''")
                Dim cmd As New SqlCommand("EXECUTE
AddNewOrder_Cust_Order_Table " & intCustID & ",'" & strConfirmationNum
& "','" & strCheckOrEFTNum & "', '" & strComments & "'," &
dblGrandTotalForOrder, conn)

                Dim rdr As SqlDataReader = cmd.ExecuteReader()


                If rdr.HasRows = True Then
                    rdr.Read()
                    intNewCustOrderIDNum =
rdr.GetInt32(intNewCustOrderIDColNum)
                End If

                Return intNewCustOrderIDNum
            End If
```

```
        Finally

            If Not conn Is Nothing Then
                If conn.State <> Data.ConnectionState.Closed Then
                    conn.Close()
                    conn.Dispose()
                End If
            End If
        End Try

    End Function


    Protected Sub Reset_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Reset.Click
        Call ClearTheForm()
    End Sub

    Protected Function ClearTheForm() As Boolean
        Me.Server.Transfer("Scrip_Order.aspx")
    End Function


End Class
```

Table 34. Code: AddMerchant.aspx.vb

```
Imports System.Data.SqlClient


Partial Class Sponsor_AddMerchant
    Inherits System.Web.UI.Page
    Dim strCurrentMerchID As String
    Dim strButtonSelected As String



    Public Function GetCurrentDenominations() As StringCollection
        '*************************************************************
        'Author:  Steve Williams
        'Date:    9/30/06
        'Description:          '
        'Pre:
        'Post:
        '*************************************************************
        Dim conn As Data.SqlClient.SqlConnection = Nothing


        Try
            Dim ret As New StringCollection()
            Dim intNumRecords As Integer = 0
            'Retrieve value in 1st column of result set
```

```
            Dim intDenomIDColumn As Integer = 0
            conn = New SqlConnection

            conn.ConnectionString =
ConfigurationManager.ConnectionStrings("CompuSCRIP").ConnectionString

            conn.Open()

            Dim cmd As New SqlCommand("EXECUTE
SelectMerchantDenominations " & strCurrentMerchID & "," &
Profile._SponsorID, conn)
            Dim rdr As SqlDataReader = cmd.ExecuteReader()


            If rdr.HasRows = True Then
                Do While rdr.Read
                    ret.Add(rdr.GetInt32(intDenomIDColumn).ToString())
                Loop

            End If

            Return ret


        Finally
            If Not conn Is Nothing Then
                If conn.State <> Data.ConnectionState.Closed Then
                    conn.Close()
                    conn.Dispose()
                End If
            End If
        End Try

    End Function

    Public Sub UpdateDenominations(ByVal sender As Object, ByVal e As
System.EventArgs)
        Call UpdateCurrentDenominations()
    End Sub

    Public Sub UpdateCurrentDenominations()
        Dim conn As Data.SqlClient.SqlConnection = Nothing
        Dim cmdresults As Integer
        Dim strDenominationsFromScreen As StringCollection
        Dim sqlDeleteDenominations As String =
"DeleteMerchDenominations"
        Dim sqlUpdateDenominations As String = "UpdateDenominations"


        '   Dim conn As New SqlConnection(strConn)

        Dim intIndex1 As Integer

        conn = New SqlConnection
```

```
        conn.ConnectionString =
ConfigurationManager.ConnectionStrings("CompuSCRIP").ConnectionString
        Dim Cmd As New SqlCommand(sqlUpdateDenominations, conn)
        Dim cmdDelete As New SqlCommand(sqlDeleteDenominations, conn)

        Try
            '   Dim cmd As SqlCommand

            Call SetCurrentMerchantID()

            strDenominationsFromScreen = New StringCollection

            strDenominationsFromScreen = GetDenomSelections()


            If strDenominationsFromScreen.Count > 0 Then
                conn.Open()

                Cmd.CommandType = Data.CommandType.StoredProcedure

                cmdDelete.CommandType =
Data.CommandType.StoredProcedure
                cmdDelete.Parameters.Add(New SqlParameter("@Merch_ID",
strCurrentMerchID))

                'Delete existing denominations for merchant in prep for
inserting
                'the updated list of denominations from the denom
checkbox list
                cmdresults = cmdDelete.ExecuteNonQuery()

                'For each denomination found in the string collection
                For intIndex1 = 0 To strDenominationsFromScreen.Count -
1
                    Cmd.Parameters.Add(New SqlParameter("@Merch_ID",
strCurrentMerchID))
                    Cmd.Parameters.Add(New
SqlParameter("@Denomination_Amount_to_Add", _

strDenominationsFromScreen.Item(intIndex1)))


                    cmdresults = Cmd.ExecuteNonQuery()
                    Cmd.Parameters.Clear()

                Next

            End If

        Finally
            If Not conn Is Nothing Then
                If conn.State <> Data.ConnectionState.Closed Then
                    conn.Close()
                    conn.Dispose()
                End If
            End If
```

```
        End Try

    End Sub

    Private Function GetDenomSelections() As StringCollection
        '*****************************************************************
        'Author:  Steve Williams
        'Date:    9/30/06
        'Description:
        'Pre:
        'Post:
        '*****************************************************************

        Dim intIndex2 As Integer 'index to loop through checkbox list
in formview
        Dim strDenominations As StringCollection
        Dim ChkBoxList As CheckBoxList  'instance of checkboxlist
        strDenominations = New StringCollection


        'Find the checkbox list in the Formview control on the page
        ChkBoxList = New CheckBoxList
        ChkBoxList =
FormView4.Row.FindControl("chkBoxLst_GetAllDenoms")

        'Find the corresponding denomination checkbox on the screen
        For intIndex2 = 0 To ChkBoxList.Items.Count - 1
            'If the denomination value is listed for the merchant in
the DB
            'then put a check next to the denom amount on the screen

            If ChkBoxList.Items.Item(intIndex2).Selected = True Then

strDenominations.Add(ChkBoxList.Items.Item(intIndex2).Value.ToString())
            End If

        Next
        Return (strDenominations)

    End Function


    Private Sub MakeSelections(ByVal strDenominations As
StringCollection)
        '*****************************************************************
        'Author:  Steve Williams
        'Date:    9/30/06
        'Description:
        '*****************************************************************
        Dim intIndex1 As Integer 'index to loop through the string
collection
        Dim intIndex2 As Integer 'index to loop through checkbox list
in formview
        Dim ChkBoxList As CheckBoxList  'instance of checkboxlist

        'Find the checkbox list in the Formview control on the page
```

```
        ChkBoxList =
FormView4.Row.FindControl("chkBoxLst_GetAllDenoms")

        'Initialize all checkboxes to unchecked
        For intIndex2 = 0 To ChkBoxList.Items.Count - 1
            ChkBoxList.Items(intIndex2).Selected = False
        Next

        'For each denomination found in the string collection
        For intIndex1 = 0 To strDenominations.Count - 1

            'Find the corresponding denomination checkbox on the screen
            For intIndex2 = 0 To ChkBoxList.Items.Count - 1
                'If the denomination value is listed for the merchant
in the DB
                'then put a check next to the denom amount on the
screen

                If ChkBoxList.Items(intIndex2).Text =
strDenominations.Item(intIndex1) Then
                    ChkBoxList.Items(intIndex2).Selected = True
                    Exit For
                End If
            Next
        Next

    End Sub
    Protected Sub PreSelections(ByVal sender As Object, ByVal e As
System.EventArgs)
        '*****************************************************************
        'Author:  Steve Williams
        'Date:    9/30/06
        'Description:
        'Pre:
        'Post:
        '*****************************************************************
        Dim strDenominations As StringCollection
        'Retrieves denomination amounts for the merchant from the
        'Merch_Denominations table and places the list of amounts
        'in a string collection.
        Call SetCurrentMerchantID()
        strDenominations = GetCurrentDenominations()
        If strDenominations.Count > 0 Then
            'Call the procedure that selects the corresponding
denomination
            'checkboxes that are reflected in the Merch_Denominations
table
            Call MakeSelections(strDenominations)
        End If



    End Sub

    Protected Sub SetCurrentMerchantID()
        Dim lblMerch_ID As Label
        lblMerch_ID = New Label
```

```
        lblMerch_ID = FormView4.Row.FindControl("lblMerch_ID")
        strCurrentMerchID = lblMerch_ID.Text.ToString

    End Sub


    Public Sub SelectMerchTypeForSpons(ByVal sender As Object, ByVal e
As System.EventArgs)
        '*************************************************************
        'Author:  Steve Williams
        'Date:    9/30/06
        'Description:
        'Pre:
        'Post:
        '*************************************************************
        Dim conn As Data.SqlClient.SqlConnection = Nothing
        Dim lstMerchType As DropDownList  'instance of checkboxlist
        Dim strMerchType As String
        Dim intMerchTypeIDCol As Integer = 0
        Dim intMerchTypeDescCol As Integer = 1
        Dim intMerchTypeIDValue As Integer
        Dim strSelectedMerchType As String

        Try
            Dim ret As New StringCollection()



            lstMerchType =
FormView4.Row.FindControl("DropDownList_MerchTypes")
            conn = New SqlConnection
            conn.ConnectionString =
ConfigurationManager.ConnectionStrings("CompuSCRIP").ConnectionString

            conn.Open()


            Dim cmd As New SqlCommand("EXECUTE SelectMerchType_All",
conn)
            Dim rdr As SqlDataReader = cmd.ExecuteReader()

            strSelectedMerchType = lstMerchType.SelectedItem.Text


            lstMerchType.Items.Clear()

            If rdr.HasRows = True Then

                Do While rdr.Read

                    intMerchTypeIDValue =
rdr.GetInt32(intMerchTypeIDCol)
                    strMerchType = rdr.GetString(intMerchTypeDescCol)
                    'intMerchTypeIDValue = rdr.GetInt32(1)
```

```
                    'Find the checkbox list in the Formview control on
the page
                    lstMerchType =
FormView4.Row.FindControl("DropDownList_MerchTypes")
                    'lstMerchType.Items.Add(strMerchType)
                    lstMerchType.Items.Add(New ListItem(strMerchType,
intMerchTypeIDValue))



                Loop
                rdr.Close()

                lstMerchType.ClearSelection()

lstMerchType.Items.FindByText(strSelectedMerchType).Selected = True

            End If



        Finally
            If Not conn Is Nothing Then
                If conn.State <> Data.ConnectionState.Closed Then
                    conn.Close()
                    conn.Dispose()
                End If
            End If
        End Try

    End Sub

    Public Sub SelectBrokerForSpons(ByVal sender As Object, ByVal e As
System.EventArgs)
        '*************************************************************
        'Author:  Steve Williams
        'Date:    9/30/06
        'Description:
        'Pre:
        'Post:
        '*************************************************************
        Dim conn As Data.SqlClient.SqlConnection = Nothing
        Dim lstBroker As DropDownList  'instance of checkboxlist
        Dim strBrokerType As String
        Dim intBrokerTypeIDCol As Integer = 0
        Dim intBrokerTypeDescCol As Integer = 1
        Dim intBrokerTypeIDValue As Integer
        Dim strSelectedBrokerType As String
        Dim intSelectedBrokertype As Integer

        Try
            Dim ret As New StringCollection()



            lstBroker =
FormView4.Row.FindControl("DropDownList_Broker")
```

```
            conn = New SqlConnection

            conn.ConnectionString =
ConfigurationManager.ConnectionStrings("CompuSCRIP").ConnectionString
            conn.Open()


            Dim cmd As New SqlCommand("EXECUTE SelectBrokers_All",
conn)
            Dim rdr As SqlDataReader = cmd.ExecuteReader()

            strSelectedBrokerType = lstBroker.SelectedItem.Text
            intSelectedBrokertype = lstBroker.SelectedItem.Value

            lstBroker.Items.Clear()

            If rdr.HasRows = True Then

                Do While rdr.Read

                    intBrokerTypeIDValue =
rdr.GetInt32(intBrokerTypeIDCol)
                    strBrokerType = rdr.GetString(intBrokerTypeDescCol)
                    'intMerchTypeIDValue = rdr.GetInt32(1)


                    'Find the checkbox list in the Formview control on
the page
                    lstBroker =
FormView4.Row.FindControl("DropDownList_Broker")

                    lstBroker.Items.Add(New ListItem(strBrokerType,
intBrokerTypeIDValue))


                Loop
                rdr.Close()

                lstBroker.ClearSelection()

lstBroker.Items.FindByText(strSelectedBrokerType).Selected = True

            End If



        Finally
            If Not conn Is Nothing Then
                If conn.State <> Data.ConnectionState.Closed Then
                    conn.Close()
                    conn.Dispose()
                End If
            End If
        End Try

    End Sub
```

```
    Public Sub SelectScripType(ByVal sender As Object, ByVal e As
System.EventArgs)
        '*****************************************************************
        'Author:  Steve Williams
        'Date:    9/30/06
        'Description:
        'Pre:
        'Post:
        '*****************************************************************
        Dim conn As Data.SqlClient.SqlConnection = Nothing
        Dim lstScrip As DropDownList  'instance of checkboxlist
        Dim strScripType As String
        Dim intScripTypeIDCol As Integer = 0
        Dim intScripTypeDescCol As Integer = 1
        Dim intScripTypeIDValue As Integer
        Dim strSelectedScripType As String

        Try
            Dim ret As New StringCollection()



            lstScrip = FormView4.Row.FindControl("DropDownList_Scrip")
            conn = New SqlConnection

            conn.ConnectionString =
ConfigurationManager.ConnectionStrings("CompuSCRIP").ConnectionString

            conn.Open()


            Dim cmd As New SqlCommand("EXECUTE SelectScripTypes_All",
conn)
            Dim rdr As SqlDataReader = cmd.ExecuteReader()

            strSelectedScripType = lstScrip.SelectedItem.Text


            lstScrip.Items.Clear()

            If rdr.HasRows = True Then

                Do While rdr.Read

                    intScripTypeIDValue =
rdr.GetInt32(intScripTypeIDCol)
                    strScripType = rdr.GetString(intScripTypeDescCol)


                    'Find the checkbox list in the Formview control on
the page
                    lstScrip =
FormView4.Row.FindControl("DropDownList_Scrip")
                    'lstScrip.Items.Add(strScripType)
                    lstScrip.Items.Add(New ListItem(strScripType,
intScripTypeIDValue))
```

```
                Loop
                rdr.Close()

                lstScrip.ClearSelection()

lstScrip.Items.FindByText(strSelectedScripType).Selected = True

            End If



        Finally
            If Not conn Is Nothing Then
                If conn.State <> Data.ConnectionState.Closed Then
                    conn.Close()
                    conn.Dispose()
                End If
            End If
        End Try

    End Sub


    Protected Sub FormView4_ModeChanging(ByVal sender As Object, ByVal
e As System.Web.UI.WebControls.FormViewModeEventArgs) Handles
FormView4.ModeChanging
        Dim strEnteringMode As String
        strEnteringMode = e.NewMode().ToString

        If strButtonSelected = "Insert" And strEnteringMode =
"ReadOnly" Then
            Call FindMerchID()
            Call UpdateCurrentDenominations()
            strButtonSelected = Space(0)
            lstMerchantName.Items.Clear()
            lstMerchantName.DataBind()

        End If

    End Sub

    Protected Sub InsertCancelButton_Click(ByVal sender As Object,
ByVal e As System.EventArgs)

    End Sub

    Protected Sub InsertButton_Click(ByVal sender As Object, ByVal e As
System.EventArgs)
        strButtonSelected = "Insert"
    End Sub



    Public Sub FindMerchID()
        Dim conn As Data.SqlClient.SqlConnection = Nothing
```

```
        Dim cmdresults As Integer
        Dim intNewMerchIDColNum As Integer = 0
        Dim intNewMerchID As Int64
        Dim objBusNameTextBox As TextBox
        Dim strBusNameJustAdded As String
        Dim sqlFindMerchID As String = "FindNewlyAddedMerchant"
        Dim lblMerch_ID As Label
        lblMerch_ID = New Label



        Try
            conn = New SqlConnection

            conn.ConnectionString =
ConfigurationManager.ConnectionStrings("CompuSCRIP").ConnectionString

            'Dim conn As New SqlConnection(strConn)
            Dim cmd As New SqlCommand(sqlFindMerchID, conn)

            objBusNameTextBox =
FormView4.FindControl("txtBusiness_Name")
            strBusNameJustAdded = objBusNameTextBox.Text


            conn.Open()

            cmd.CommandType = Data.CommandType.StoredProcedure

            cmd.CommandType = Data.CommandType.StoredProcedure

            cmd.Parameters.Add("@Sponsor_ID", Data.SqlDbType.Int).Value
= Profile._SponsorID
            cmd.Parameters.Add("@Business_Name",
Data.SqlDbType.NVarChar).Value = strBusNameJustAdded

            cmdresults = cmd.ExecuteNonQuery()

            Dim rdr As SqlDataReader = cmd.ExecuteReader()

            If rdr.HasRows = True Then
                rdr.Read()
                intNewMerchID = rdr.GetInt32(intNewMerchIDColNum)
            End If

            cmd.Parameters.Clear()



            lblMerch_ID = FormView4.Row.FindControl("lblMerch_ID")
            lblMerch_ID.Text = intNewMerchID

            Call SetCurrentMerchantID()
```

```
        Finally
            If Not conn Is Nothing Then
                If conn.State <> Data.ConnectionState.Closed Then
                    conn.Close()
                    conn.Dispose()
                End If
            End If
        End Try

    End Sub


End Class
```

Table 35. Code: ProcessScrip.aspx.vb

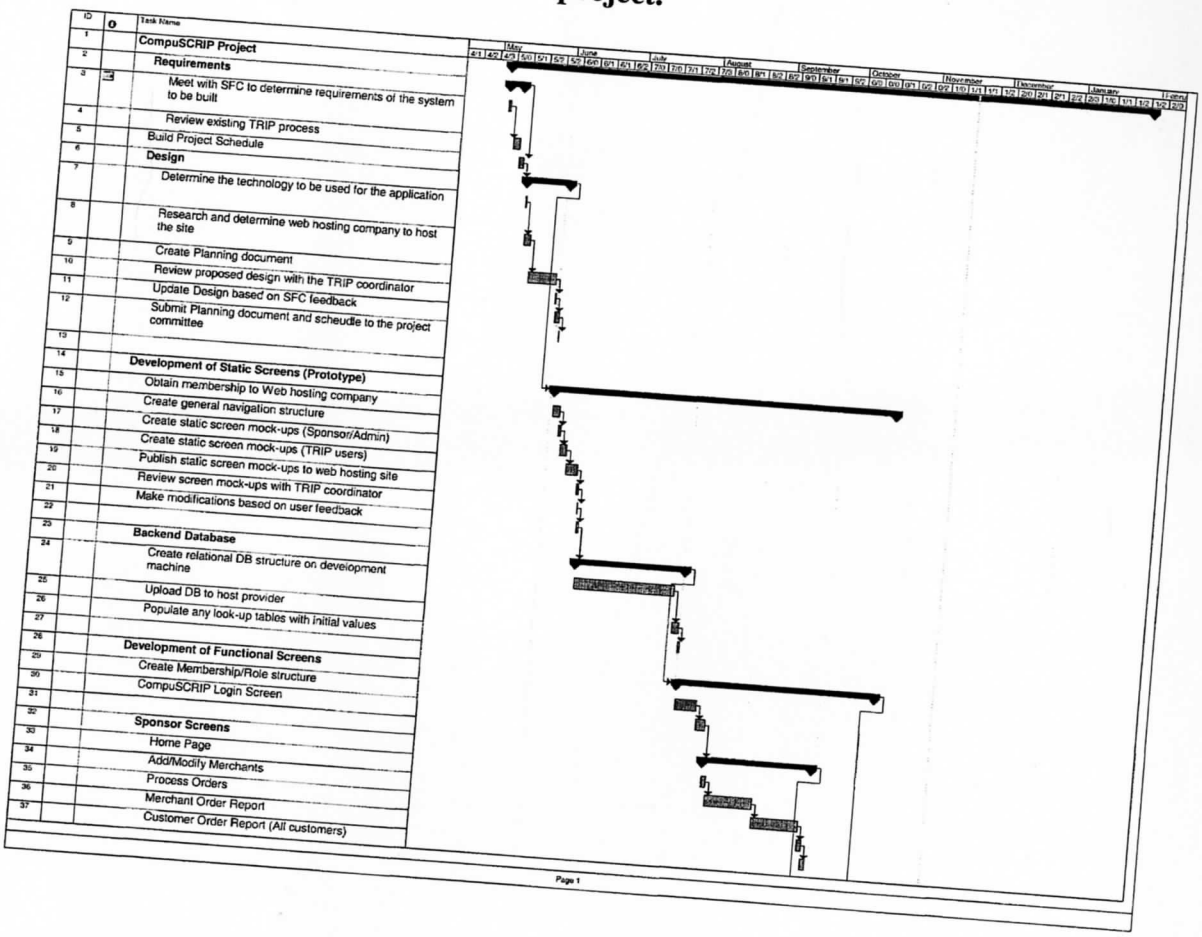
```
Imports System.Data.SqlClient
Imports System.Data.SqlTypes

Partial Class Sponsor_ProcessScrip
    Inherits System.Web.UI.Page

    Const m_intAmountCol As Integer = 6
    Const m_intCustOrderIDCol As Integer = 14


    Protected Sub GridView1_PreRender(ByVal sender As Object, ByVal e
As System.EventArgs) Handles GridView1.PreRender

        Dim intRowCount As Integer
        Dim intRowNum As Integer
        Dim objNumOrderTextBox As TextBox
        Dim objDenomAmount As Label
        Dim dblOrderAmt As Double
        Dim strGrandTotal As String
        Dim strGrandTotToCust As String
        Dim dblGrandTotToCust As Double
        Dim strOrderAmt As String

        objNumOrderTextBox = New TextBox
        objDenomAmount = New Label

        intRowCount = GridView1.Rows.Count - 1
        dblOrderAmt = (0.0)
        dblGrandTotToCust = 0
        strGrandTotal = ""
        strGrandTotToCust = ""

        'For each row in the grid
        For intRowNum = 0 To intRowCount
```

```
            dblOrderAmt =
GridView1.Rows.Item(intRowNum).Cells(m_intAmountCol).Text

            strOrderAmt = dblOrderAmt.ToString("$#,##0.00")
            GridView1.Rows.Item(intRowNum).Cells(m_intAmountCol).Text =
strOrderAmt

            'Populate sub totals for each row in the grid
        Next

    End Sub

    Protected Sub cmdSave_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles cmdSave.Click
        Dim conn As Data.SqlClient.SqlConnection = Nothing
        Dim intNewCustOrderID As Int64
        Dim objPaymentRcvd As CheckBox
        Dim objCustPickedUpOrder As CheckBox
        Dim objCheckNumber As TextBox
        Dim bolPaymentRcvd As Boolean
        Dim bolCustPickedUpOrder As Boolean
        Dim intRowNum As Integer
        Dim intRowCount As Integer
        Dim strCheckNum As String


        Try
            'Retrieve value in 1st column of result set
            Dim intNewCustOrderDetailsIDColNum As Integer = 0
            conn = New SqlConnection

            conn.ConnectionString =
ConfigurationManager.ConnectionStrings("CompuSCRIP").ConnectionString


            conn.Open()

            intRowCount = GridView1.Rows.Count - 1
            Dim cmd As SqlCommand = New
SqlCommand("UpdateOrder_Pay_Rcvd ", conn)
            cmd.CommandType = Data.CommandType.StoredProcedure

            For intRowNum = 0 To intRowCount
                objPaymentRcvd =
GridView1.Rows.Item(intRowNum).FindControl("chkPaymentRcvd")
                objCustPickedUpOrder =
GridView1.Rows.Item(intRowNum).FindControl("chkCustOrderRcvd")
                objCheckNumber =
GridView1.Rows.Item(intRowNum).FindControl("txtCheckNum")

                bolPaymentRcvd = objPaymentRcvd.Checked
                bolCustPickedUpOrder = objCustPickedUpOrder.Checked
                strCheckNum = objCheckNumber.Text
                intNewCustOrderID =
Convert.ToInt32(GridView1.Rows.Item(intRowNum).Cells(m_intCustOrderIDCo
l).Text)
```

```
                If bolPaymentRcvd = True Then
                    cmd.Parameters.Add("@Payment_Rcvd_Date",
Data.SqlDbType.DateTime).Value = DateTime.Parse(Now.ToString)
                Else
                    cmd.Parameters.Add("@Payment_Rcvd_Date",
Data.SqlDbType.DateTime).Value = SqlDateTime.Null
                End If

                cmd.Parameters.Add("@Cust_Rcvd_Order_Date",
Data.SqlDbType.DateTime).Value = SqlDateTime.Null


                cmd.Parameters.Add("@Payment_Rcvd",
Data.SqlDbType.Bit).Value = bolPaymentRcvd
                cmd.Parameters.Add("@Cust_Rcvd_Order",
Data.SqlDbType.Bit).Value = bolCustPickedUpOrder
                cmd.Parameters.Add("@Cust_Order_ID",
Data.SqlDbType.Int).Value = intNewCustOrderID
                cmd.Parameters.Add("@Check_Number",
Data.SqlDbType.NVarChar).Value = strCheckNum


                cmd.ExecuteNonQuery()

                cmd.Parameters.Clear()
            Next


        Finally
            If Not conn Is Nothing Then
                If conn.State <> Data.ConnectionState.Closed Then
                    conn.Close()
                    conn.Dispose()
                End If
            End If
        End Try
    End Sub


    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load

    End Sub
End Class
```

APPENDIX B: GHANT CHART

Ghantt Chart for the CompuSCRIP project.



ID	Task Name
1	**CompuSCRIP Project**
2	**Requirements**
3	Meet with SFC to determine requirements of the system to be built
4	Review existing TRIP process
5	Build Project Schedule
6	**Design**
7	Determine the technology to be used for the application
8	Research and determine web hosting company to host the site
9	Create Planning document
10	Review proposed design with the TRIP coordinator
11	Update Design based on SFC feedback
12	Submit Planning document and scheudle to the project committee
13	
14	**Development of Static Screens (Prototype)**
15	Obtain membership to Web hosting company
16	Create general navigation structure
17	Create static screen mock-ups (Sponsor/Admin)
18	Create static screen mock-ups (TRIP users)
19	Publish static screen mock-ups to web hosting site
20	Review screen mock-ups with TRIP coordinator
21	Make modifications based on user feedback
22	
23	**Backend Database**
24	Create relational DB structure on development machine
25	Upload DB to host provider
26	Populate any look-up tables with initial values
27	
28	**Development of Functional Screens**
29	Create Membership/Role structure
30	CompuSCRIP Login Screen
31	
32	**Sponsor Screens**
33	Home Page
34	Add/Modify Merchants
35	Process Orders
36	Merchant Order Report
37	Customer Order Report (All customers)

APPENDIX B: GHANT CHART

Ghantt Chart for the CompuSCRIP project.