Database to Assist Auditors with Client Data Testing and Analysis for KPMG Ireland
KPMG Ireland
Create Database to Assist Auditors with Client Data Testing and Analysis

24th March 2014

Prepared By: Dasha Gorovenco  
Supervisor: Aideen Keaney
I declare that the work described in this dissertation has been carried out in full compliance with the ethical research requirements of the School of Computer Science and Statistics.

Signed: ___________________

Dasha Gorovenco

24/03/2014
ABSTRACT

The purpose of this project was to design and develop a database with appropriate data testing for Auditors within KPMG Ireland. The aim of this project was to develop a proof of concept to demonstrate that the existing Excel testing tool would operate better using a different programming language. It was essential for the testing tool to be user-friendly and provide comprehensive user output.

The deliverable consisted of a back-end database in MySQL used for storing and testing client data and an additional sample front-end user interface to allow for a user-friendly data testing process. Five languages were combined to achieve the desired outcome, SQL, HTML, PHP, JavaScript and CSS.
This project was undertaken on behalf of KPMG Ireland. Within KPMG Ireland the main contact throughout the project was the Partner of Information Risk Management Department, Mr. Selwyn Hearns, hereafter referred to as “the client”.

The client wanted a proof of concept for the existing testing tool. The existing tool built in Excel using VBA Macros had some limitations. The client was interested in testing the suitability of the SQL programming language to ascertain whether this will allow the user to store more client data and efficiently execute data testing whilst providing user friendly output. The resulting system meets the client’s terms of reference and it also provides additional user functionality for manipulation of the database in the form of a user interface.

The main challenges associated with this project were development-related, identifying the best tools and languages to apply, coding SQL queries and providing the client with a user friendly sample interface. In total five languages were used to develop the testing tool; SQL, PHP, CSS, HTML and JavaScript.

I would like to thank Mr. Selwyn Hearns for his invaluable expertise and feedback throughout the completion of this project. I would also like to thank Mr. Jaume Mora Pedros for his continuous help and support. Finally, I would like to thank my project supervisor, Dr. Aideen Keaney, for her advice, support and guidance throughout the project which greatly contributed to the completion of this project.
KPMG Ireland

Database to Assist Auditors with Client Data Testing and Analysis

March 2014

TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>NO.</th>
<th>SECTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTRODUCTION AND SUMMARY</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Client Background</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Project Background</td>
<td>1</td>
</tr>
<tr>
<td>1.3</td>
<td>Terms of Reference</td>
<td>2</td>
</tr>
<tr>
<td>1.4</td>
<td>Chapter Summaries</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>SYSTEM OVERVIEW</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>System Objectives</td>
<td>3</td>
</tr>
<tr>
<td>2.2</td>
<td>Data Familiarisation and Software Research</td>
<td>4</td>
</tr>
<tr>
<td>2.3</td>
<td>Technical Environment</td>
<td>4</td>
</tr>
<tr>
<td>2.4</td>
<td>System Overview Diagram</td>
<td>5</td>
</tr>
<tr>
<td>2.5</td>
<td>Interface Walkthrough</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>DESCRIPTION OF WORK DONE</td>
<td>9</td>
</tr>
<tr>
<td>3.1</td>
<td>Requirements</td>
<td>9</td>
</tr>
<tr>
<td>3.2</td>
<td>Design</td>
<td>14</td>
</tr>
<tr>
<td>3.3</td>
<td>Development Tools</td>
<td>15</td>
</tr>
<tr>
<td>3.4</td>
<td>Development</td>
<td>16</td>
</tr>
<tr>
<td>3.5</td>
<td>Implementation and Testing</td>
<td>20</td>
</tr>
<tr>
<td>3.6</td>
<td>Challenges Encountered</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>CONCLUSIONS AND RECOMMENDATIONS</td>
<td>23</td>
</tr>
<tr>
<td>4.1</td>
<td>Conclusions</td>
<td>23</td>
</tr>
<tr>
<td>4.2</td>
<td>Recommendation</td>
<td>24</td>
</tr>
</tbody>
</table>
## APPENDICES

<table>
<thead>
<tr>
<th>NO.</th>
<th>SECTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>Original Project Outline</td>
<td>A.1</td>
</tr>
<tr>
<td>B.</td>
<td>Interim Project Report</td>
<td>B.1</td>
</tr>
<tr>
<td>C.</td>
<td>User Manual</td>
<td>C.1</td>
</tr>
<tr>
<td>D.</td>
<td>Technical Manual</td>
<td>D.1</td>
</tr>
<tr>
<td>E.</td>
<td>Design Documentation</td>
<td>E.1</td>
</tr>
<tr>
<td>F.</td>
<td>Screenshots</td>
<td>F.1</td>
</tr>
<tr>
<td>G.</td>
<td>Source Code</td>
<td>G.1</td>
</tr>
<tr>
<td>H.</td>
<td>Test Documentation</td>
<td>H.1</td>
</tr>
</tbody>
</table>

GLOSSARY TERMS USED

REFERENCES
1. INTRODUCTION AND SUMMARY

This chapter describes the client, outlines the project background and agreed terms of reference, along with providing summaries of the remaining chapters.

1.1. Client Background

KPMG is a leading provider of professional services in Ireland. With 79 partners and 1,900 people working in offices in Dublin, Galway, Cork and Belfast; KPMG provide a range of Audit, Tax and Advisory services to clients in all sectors of Irish and worldwide businesses.

The client for this project was the Information Risk Management (IRM) team within the Audit sector of KPMG. The IRM team are responsible for advising on managing risks from the use of information technology. They provide their clients with detailed feedback and recommendations from their analysis.

1.2. Project Background

At present, the Audit practice within KPMG manually analyses and tests client data, which can include millions of records at a time. This is time consuming and susceptible to human error. Currently, simple data tests such as identifying negative values or duplicate values, consumes a lot of valuable time.

The IRM team have developed a testing tool for client data in Excel using VBA Macros. This tool contains a switchboard which allows Auditors to input desired client data and choose various tests to be performed. The results of these tests are then printed in a tabular format on a separate Excel sheet. This allows the Auditors to quickly and efficiently analyse the data. The current Excel tool contains the following limitations:

- At present, Excel only handles 1,048,576 records at a time. This means that if a KPMG client has provided more than 1,048,576 records, all tests have to be executed multiple times, making the current testing tool inefficient.
- Structures within VBA for Excel also contain limitations for the number of transactions that can be processed.
- The existing Excel testing tool takes a long time to execute a large number of transactions, which becomes a significant factor for bigger KPMG clients. Currently it takes one hour to process 100,000 invoices. This in turn limits the amount of KPMG clients to which the tool can be applied.
- The current tool assumes that data files do not need to be merged; this means that the Excel input files must be manually pre-processed where this is required.
- Certain tasks which would be standard and efficient for databases are difficult to perform in spreadsheets – particularly sequence checking, looking for gaps and duplicates, and data consistency. A high level of errors in the data causes performance to dip significantly.
• The Excel tool is not built to allow ad-hoc questions or queries; this means that such requests must be performed manually within Excel.

The proof of concept was carried out in order to identify if it would be possible to test more than 1,048,576 records of client data at a time and eliminate many of the limitations that exist in the current testing tool. The new testing tool had to be able to take in and store relevant data, input it into SQL and return the necessary output into an Excel file for the user to analyse. This process should not require the user to possess a thorough understanding of programming languages in order to utilise all the functionalities of the tool.

1.3. Terms of Reference

• Examine the format of data currently collected for Audit
• Examine reports that are currently generated
• Design a database to match data collected and to generate selected output reports
• Print output to Excel or Word file in a tabular format
• Produce comprehensive user documentation for the client

The application exceeds the original terms of reference by including:

• A sample web interface that provides
  o An option to select CSV files to be imported into the database
  o A choice of testing to be executed
  o Automatic printing of output to Excel

1.4. Chapter Summaries

Chapter 2 provides an overview of the system. It describes in detail the technical background of the system, how the system operates and also includes a user-walkthrough of the web interface.

Chapter 3 contains a description of the work done in order to create the application. It also describes all the development and testing phases, and the problems encountered.

Chapter 4 contains conclusions drawn from the project, as well as recommendations for further development of the system.
2. **SYSTEM OVERVIEW**

This section describes the system objectives, the technical environment, a system overview diagram and a detailed walk-through of the sample user interface.

2.1. **System Objectives**

The purpose of this system, as outlined in the terms of reference in Section 1.3, is to identify if the current Excel testing tool used by KPMG Auditors can function better using the SQL language and a MySQL database. Furthermore, the proof of concept is to identify if the new system can store millions of records and provide the desired output. The new system will store large amounts of client data in a database, pass it through SQL queries and return the results to the user in the form of an Excel file.

The data provided by the client was reviewed in detail. It was essential to understand the output desired by the client, and the links between the input data and the output data. Also, detailed research was carried out into software to use for the development of the system. This segment of the project took approximately three weeks. Data familiarisation and software research are further discussed in Section 3.1.

After the research and familiarisation with the data, the queries were coded using SQL and thoroughly tested. This was done in order to identify if the transition to a MySQL database and SQL queries is possible. The design and development of the queries are further discussed in Section 3.2 and Section 3.4 respectively. Details on Testing can be found in Appendix H.

During the development phase it became clear that the production of a sample user web interface was essential in order to demonstrate the full capabilities of the application to the client. The web user interface facilitates data input by non-technical users which will minimise any chance of data input error. The interface provides a unified view for data collection, whilst the back end of the interface also improves the efficiency of analyses carried out - larger scale analyses can be performed much quicker in SQL.

This system is loaded onto a server, meaning that there is no need for software installation on end-user PCs.

The user interface introduces an aspect of “control” to the new application and without it the system becomes ineffective and difficult to use for non-technical users. Without the sample user interface the client would have been presented with unpleasant lines of SQL code and an inefficient manner to extract the output. In order to update the database and run SQL queries, the user would have had to manually access both; PHPMyAdmin to update the database and SQL Workbench 5.0 to execute queries. The user interface unifies all back-end processes and enables users from a non-technical background to avail of all the functionalities of the tool.
2.2. **Data Familiarisation and Software Research**

After the terms of reference were clearly defined and agreed on, the client provided sample data for the system. Familiarisation with the data was important and included an in-depth understanding of the input provided by KPMG clients and the output desired by the client. This in-depth understanding of the data also consisted of identifying the links between the input data and the output data i.e. specifically how the output was achieved. A detailed explanation of how this was attained is described in Section 3.1.

Thorough research was conducted into identifying the best software to use for the development of the system. This included downloading various software tools and testing them for suitability. Section 3.1 provides further details on the research carried out and explains why the WAMP Stack was chosen for the development of the system.

2.3. **Technical Environment**

A wide selection of languages and software tools were used in order to develop the new system. The programming languages included SQL, PHP, HTML, CSS and JavaScript. The vast majority of the proof of concept was coded in SQL, but an in-depth knowledge of the rest of the languages was essential. The purpose of having an in-depth knowledge of all the programming languages used was to develop the sample user web interface and edit the code. For example, including a package within the PHP code or adjusting the SQL code to match the PHP requirements.

![WAMP Stack Breakdown and Additionally Installed Software](image)

Figure 2.3.1 illustrates the server components and additionally installed software necessary for the development of the system. The WAMP Stack was installed which provided a complete PHP, MySQL and Apache 2.2 development environment for Windows. PHP is a simple programming language used to develop dynamic web pages. The MySQL database
used with PHPMyAdmin provides an interface to easily create databases and interact with them. The application runs on an open-source Apache 2.2 web server which is compatible with the highly popular Windows operating system.

SQL queries were developed in SQL Workbench 5.0 which connected to the MySQL database stored on the server. The entire web interface was created in Microsoft Expression Web 4 using mainly PHP, as well as HTML, CSS and JavaScript. The WAMP Stack used in the development of this system is free and open-source. It is available to download from http://bitnami-wampstack.en.softonic.com/.

In the user interface PHP controls the system. The PHP language is compatible with SQL functions, such as running queries to import CSV files into tables using a specific format. When the new files are stored in the database, PHP can then execute SQL queries for the tests and automatically print the results to an Excel file.

2.4. System Overview Diagram

Figure 2.4.1 represents the overview of the SQL queries prior to the development of the user interface.

The blue box represents the software used in the development of the SQL queries. The SQL code, which is shows in the green box, is written in SQL Workbench 5.0 which is connected to the database (the grey cylinder). When the code is executed the request is sent to the database. The database sends back the requested data and the output of the data is printed in a separate window in SQL Workbench 5.0. The output can be exported to an Excel file, represented by the orange box, by manually clicking the “Export to Excel” button available within the SQL Workbench 5.0 interface.
Figure 2.4.2 represents a map of the back-end processes of the sample user web interface.

![Diagram](image)

Figure 2.4.2.: Back-end process of the User Interface

The blue boxes represent the HTML pages. These are the pages that are physically visible to the user and contain various options for the user to choose from. If the user chooses to update/create a new database, they will be presented with the option of uploading CSV files. When the files have been chosen, the “Upload CSV Files” PHP code is called (represented as the green box). The PHP code first of all stores the CSV files onto the server and then uploads them into specified database tables. If the user desires to view the existing database tables, a PHP file will be called depending on the name of the table that was requested to view. The PHP code within this file will send a request to the database and then provide the user with the results of that request. When the user chooses to complete Audit testing, they will be required to select the tests that they wish to complete on the “Audit Testing” HTML page. After clicking the “Run” button at the bottom of that page, the “Execute Audit Testing” PHP file is called. The code within this file sends the request to the database. It then creates an Excel file (represented as the orange box) and prints the requested output from the database to that file. The PHP file also creates a new Excel sheet for each of the tests completed.
2.5. **Interface Walkthrough**

This section provides a detailed walkthrough of the sample user web interface. Further details are available in the User Manual in Appendix C. All screenshots of the interface can be found in Appendix F.

**Home**

This is the main page of the web user interface. This is the page where an Auditor will begin the “Client Data Testing” process. The page contains the company’s logo and the name of the tool. It provides the user with three options; to update the database, to view existing database tables or to use the existing database files to complete testing. When the user chooses to update the database, he/she will be redirected to the “Upload CSV File” page. When the user chooses to view an existing table, he/she will be redirected to the “View Tables” page. When the user chooses to use the existing tables within the database, he/she will be redirected to the “Audit Testing” page.

**Upload CSV File**

This page contains ten names and buttons for each of the database tables. When clicking the button for a certain table, the user has the option to select a CSV file to be uploaded to that table. This allows the user to use the same database but update the contents within the existing database tables. When the CSV file is uploaded, it is first of all saved onto the server and then uploaded into the database table. The page then allows the user to either view existing tables or move onto data testing. When the user chooses to view existing tables, he/she will be redirected to the “View Tables” page. If the user chooses to move onto testing, he/she will be redirected to the “Audit Testing” page. Figure 2.5.1 illustrates the process of uploading a CSV file.
Figure 1.5.1.: CSV File Upload

View Tables

This page contains ten buttons and names for each of the database tables. When clicking the button for a certain table, the user is provided with a quick overview of the table on the next page. This allows the user to make sure that the file input was correct. On this page the user has the option to go back to the “Upload CSV File” page, in case some data went in incorrectly. The user also has the option to move onto the “Audit Testing” page.

Audit Testing

This page contains all the data analysis tests. The user is asked to input dates and select tests they wish to run. There are validations present on this page. The user will be notified if there was no input for dates, or if the input was incorrect. The user will also be notified if no tests were selected. When the user runs the tests, the output is automatically transferred to an Excel file.

Excel File

The Excel file provides the user with the comprehensive output. Each of the result tables are printed on separate sheets for clarity and neatness. The user is also notified how long it took to carry out each test, how much computer memory was used in the process and the computer path for where the Excel file was saved to.

The user has the option to return to the “Home” page at any time, by clicking the Home link at the bottom of each page. These steps are described in more detail in the User Manual in Appendix C.
3. DESCRIPTION OF WORK DONE

This section provides a detailed description of the work carried out during the development of the application. The application was developed using an incremental model meaning that some stages of the model were repeated as changes were made to the system [1]. The incremental model is represented in Figure 3.0.1 and consists of several phases; requirements, design, development, testing and implementation.

![Incremental Model](image)

**Figure 3.0.1.: Incremental Model**

3.1. Requirements

The most important task of the project was to clearly define the requirements. This task had to be carried out before the design or the development phases began. The requirements were established during client meetings.

A primary set of requirements were set out during the first meeting with Mr. Selwyn Hearns and Mr. Jaume Mora Pedros. The potential ideas and recommendations were listed, and the feasibility of each of these were further discussed with the project supervisor, Dr. Aideen Keaney. Detailed analysis of the data provided was then conducted and research was carried out into the best possible software and programming languages to develop the application. Another meeting was arranged with the client to discuss in detail the feasibility of the current requirements and the possibility of incorporating additional requirements in the form of a user interface. It was agreed that the most important aspect of the project was to determine if the transition of the existing testing tool to a different programming language is possible. A fully functioning sample user interface was also identified as a priority.
Analysis and Research

- Client Data

The client provided both; the input data – for database tables, and the output data – what the desired output should look like. The input data consisted of ten Excel sheets – each representing a database table. The sheets were for a sample 2012 Audit; Cash 2011, Cash 2012, Credit Notes 2011, Credit Notes 2012, Despatch Notes 2011, Despatch Notes 2012, Sales 2011, Sales 2012, Debtors 2011 and Debtors 2012. These sheets were specific to 2012; however the new testing tool had to be able to function for any yearend. Each sheet had a common column which made the connection between tables possible. All sheets contained varying numbers of rows of data; a sample Excel data input sheet is shown in Figure 3.1.1.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invoice No</td>
<td>Cash Amount</td>
<td>Date Received</td>
<td>Customer ID</td>
<td>Months</td>
<td>Year</td>
<td>Invoice No</td>
</tr>
<tr>
<td>2</td>
<td>20000</td>
<td>299</td>
<td>01/03/2012</td>
<td>10220</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>5</td>
<td>20001</td>
<td>279</td>
<td>01/03/2012</td>
<td>10491</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>4</td>
<td>20002</td>
<td>300</td>
<td>31/03/2012</td>
<td>10704</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>5</td>
<td>20003</td>
<td>199</td>
<td>01/03/2012</td>
<td>10430</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>6</td>
<td>20004</td>
<td>129</td>
<td>31/03/2012</td>
<td>10041</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>7</td>
<td>20005</td>
<td>229</td>
<td>01/03/2012</td>
<td>10777</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>8</td>
<td>20006</td>
<td>129</td>
<td>01/03/2012</td>
<td>10053</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>9</td>
<td>20007</td>
<td>279</td>
<td>01/03/2012</td>
<td>10413</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>10</td>
<td>20008</td>
<td>299</td>
<td>01/03/2012</td>
<td>10884</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>11</td>
<td>20009</td>
<td>279</td>
<td>01/03/2012</td>
<td>10200</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>12</td>
<td>20010</td>
<td>288</td>
<td>31/03/2012</td>
<td>10439</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>13</td>
<td>20011</td>
<td>206</td>
<td>01/04/2012</td>
<td>10439</td>
<td>16</td>
<td>2012</td>
</tr>
<tr>
<td>14</td>
<td>20012</td>
<td>179</td>
<td>01/04/2012</td>
<td>10919</td>
<td>16</td>
<td>2012</td>
</tr>
<tr>
<td>15</td>
<td>20013</td>
<td>279</td>
<td>02/03/2012</td>
<td>10680</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>16</td>
<td>20014</td>
<td>609</td>
<td>02/03/2012</td>
<td>10511</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>17</td>
<td>20015</td>
<td>48</td>
<td>02/03/2012</td>
<td>10363</td>
<td>15</td>
<td>2012</td>
</tr>
</tbody>
</table>

Figure 3.1.1.: Sample of Cash 2012 input sheet.

The client also provided samples of the desired user output. In total, the client provided 16 output tables that required duplication using SQL queries. A sample output table is shown in Figure 3.1.2.
Figure 3.1.2.: Sample output table representing the current year and prior year of top 20 customers.

<table>
<thead>
<tr>
<th>Customer name</th>
<th>Customer ID</th>
<th>Current year amount</th>
<th>% Total</th>
<th>Prior year amount</th>
<th>% Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abbott Laboratories</td>
<td>10022</td>
<td>17,548</td>
<td>0.50%</td>
<td>18,092</td>
<td>0.50%</td>
</tr>
<tr>
<td>Advance Home Control Ltd T/A AHCC Ltd</td>
<td>10067</td>
<td>25,222</td>
<td>0.75%</td>
<td>14,074</td>
<td>0.45%</td>
</tr>
<tr>
<td>AGFIA Ltd</td>
<td>10074</td>
<td>16,546</td>
<td>0.54%</td>
<td>7,054</td>
<td>0.23%</td>
</tr>
<tr>
<td>Ardfern Pharmacy Ltd</td>
<td>10229</td>
<td>14,554</td>
<td>0.48%</td>
<td>8,987</td>
<td>0.29%</td>
</tr>
<tr>
<td>Atreus Credit Insurance NV</td>
<td>10280</td>
<td>33,854</td>
<td>1.08%</td>
<td>6,170</td>
<td>0.20%</td>
</tr>
<tr>
<td>Avia Walker</td>
<td>10292</td>
<td>15,520</td>
<td>0.43%</td>
<td>6,897</td>
<td>0.22%</td>
</tr>
<tr>
<td>Balbriggan Credit Union Limited</td>
<td>10334</td>
<td>14,529</td>
<td>0.47%</td>
<td>9,770</td>
<td>0.33%</td>
</tr>
<tr>
<td>Balbriggan Estates Property Limited</td>
<td>10355</td>
<td>27,421</td>
<td>0.87%</td>
<td>11,263</td>
<td>0.36%</td>
</tr>
<tr>
<td>Ballyhoch Limited T/A Hamilton Long</td>
<td>10366</td>
<td>16,289</td>
<td>0.52%</td>
<td>10,508</td>
<td>0.34%</td>
</tr>
<tr>
<td>Barrow Street Pharmacy Ltd</td>
<td>10413</td>
<td>16,282</td>
<td>0.52%</td>
<td>9,077</td>
<td>0.29%</td>
</tr>
<tr>
<td>Baggs pharmacy trading as Pure pharma</td>
<td>10448</td>
<td>16,614</td>
<td>0.53%</td>
<td>5,448</td>
<td>0.17%</td>
</tr>
<tr>
<td>Berehaven Credit Union Limited</td>
<td>10464</td>
<td>14,077</td>
<td>0.45%</td>
<td>7,671</td>
<td>0.26%</td>
</tr>
<tr>
<td>Brandeaux Administrators Limited</td>
<td>10560</td>
<td>12,969</td>
<td>0.44%</td>
<td>5,384</td>
<td>0.17%</td>
</tr>
<tr>
<td>Broadstone C.I.E. Employees Credit Union</td>
<td>10516</td>
<td>15,469</td>
<td>0.49%</td>
<td>7,774</td>
<td>0.25%</td>
</tr>
<tr>
<td>Central Applications Office</td>
<td>10783</td>
<td>13,363</td>
<td>0.42%</td>
<td>11,293</td>
<td>0.36%</td>
</tr>
<tr>
<td>Certus</td>
<td>10795</td>
<td>16,464</td>
<td>0.52%</td>
<td>6,448</td>
<td>0.21%</td>
</tr>
<tr>
<td>CLI Insurance Services Ltd</td>
<td>10891</td>
<td>14,593</td>
<td>0.46%</td>
<td>5,472</td>
<td>0.17%</td>
</tr>
<tr>
<td>Clinical Laboratory Services</td>
<td>10888</td>
<td>25,559</td>
<td>0.74%</td>
<td>5,355</td>
<td>0.17%</td>
</tr>
<tr>
<td>Clonority Town Council</td>
<td>10906</td>
<td>16,302</td>
<td>0.51%</td>
<td>6,487</td>
<td>0.22%</td>
</tr>
<tr>
<td>Totals top 20 debtors</td>
<td>339,064</td>
<td>10.78%</td>
<td></td>
<td>163,223</td>
<td>5.21%</td>
</tr>
</tbody>
</table>

The next important step was to identify the links between the input and the output data provided in order to create matching SQL queries. It was vital to understand the data generated and how exactly it was created. Some output tables were straightforward; they included columns available in input tables. This meant that the associated queries required a simple “call” of the desired column to generate the output. However, Figure 3.1.3 represents a complicated and rather large output. This output contained 22 columns and only two of them were columns available in the input tables. The other 20 columns were manually generated inside the query code. This output table required a combination of three input tables as well as complicated mathematical calculations and extensive data manipulation to achieve the desired physical appearance of the output table. The mathematical calculations included; quarterly totals of invoices and credit notes, percentage of quarterly invoices and credit notes, the total of each separately and the percentage difference between the current year and the next year of invoices and credit notes. In the input data provided by the client the yearend was December 2012, however it was important to make the code flexible by allowing the yearend to be any time of the year. This would ensure that the queries would operate even if the yearend was different. It was important to have the layout of the output identical to that provided by the client. The output table in Excel was a pivot table and this meant that the output had to be manually replicated. This included a lot of new code being added and manipulated within the query. Figure 3.1.3 shows that the names of the client companies had to be in an alphabetical order. Placing the client names in an alphabetical order required confirmation that the rest of the output in each column corresponded to the right client name.
Figure 3.1.3.: Detailed analysis by customer is a sample of a complicated output table

See Appendix G for SQL query code related to the diagram above.

- **Software Research**

Detailed research was conducted into identifying the best possible software to develop the application. The initial software requirements consisted of an interface for database creation and manipulation, a server to store the database and software for coding SQL queries. At a later stage, research was conducted into software to use for the development of the sample user interface.

The ideal server for developing and implementing the application would have required a minimal amount of time for familiarisation and allow access from any computer and location. If the system was developed on the client’s server that would mean that the project could only be worked on in the client’s offices. If the system was developed on the testing database server in the School of Computer Science and Statistics, TCD then the system could only be worked on and demonstrated in College - this is inefficient as the client would have to regularly be present in College for the demonstration of the application. The best solution was to install the required software as a “localhost” on a laptop. This means that the laptop can be brought to the client’s offices or College and the system fully demonstrated. This also allowed for the system to be worked on and modified from any location.

Various servers were looked at in detail, these included: SQL Server Express, Oracle Server, Apache Server and others. Descriptive documentations accompanying these servers were read carefully [2], [3]. The vast majority of servers reviewed required an in-depth understanding of its back-end procedures and an individual installation of all software required. Some servers even required a financial investment. These servers appeared complex and familiarisation with a server of this type would take up a lot of valuable time. The Apache server appeared more appealing than others as it included both; PHP web interface and PHPMyAdmin – which can be installed additionally onto a server. Further research into the Apache server located the WAMP Stack which includes the Apache server as well as PHP and MySQL. The WAMP Stack which operates on the Windows platform provided all the required software and programming languages to develop the system. The package also provides easy installation. The WAMP Stack is fully described in Section 2.3.

PHPMyAdmin was used to develop the MySQL database for the application. This tool was chosen as it has been previously used in the course. It is compatible with the Apache server
and consists of a simple interface that allows for easy modifications and manipulations of the database.

In order to develop and test the SQL queries, additional software was required that easily connected to the database and consisted of a user-friendly interface. This software was used to develop and test the queries, and primarily to identify if the transition to the SQL language was feasible. The SQL software tools examined were; Oracle SQL Developer and SQL Workbench 5.0. Through research, these two proved to be the most popular providing a lot of online support. Oracle SQL Developer was downloaded from http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html?ssSourceSiteId=otnpt and installed. The interface of this tool was complicated and proved difficult to use. Initial problems with Oracle SQL Developer occurred during the connection to the database. Lengthy online research was conducted into identifying code to modify the connection details. Oracle SQL Developer also included built-in SQL code functions, such as creating a pivot table. Although this is convenient, it provides a limited approach of coding SQL queries and doesn’t harness the full capabilities of the language. Review of the documentation attached to Oracle SQL Developer revealed that some aspects of the SQL code implemented in this software are not compatible with any other SQL servers and will require extensive code modifications [4]. Further research showed that MySQL Workbench 5.0 can connect to a MySQL database on the Apache server and provides a unified visual tool for database developers. The tool was downloaded from http://www.mysql.com/products/workbench/ and installed. As SQL Workbench 5.0 is a component of the MySQL family, the Workbench automatically connected to the database. The interface is simple and easy to use. It allowed for immediate code writing and testing without tedious adaptations to the tool. The SQL Workbench 5.0 also provided visual tools for creating, executing and optimising SQL queries. The tool provides a visual demonstration of the database stored within MySQL. SQL Workbench 5.0 contains the functionality to easily, with a simple click of a button, export the output to Excel to visualise how it will look in Excel. The SQL Workbench 5.0 operated perfectly in providing the desired outcome.

At a later stage research was conducted into software to use to develop the sample user interface. Since the WAMP Stack package included PHP, Windows Expression Web 4 was used for the development of the user interface. This software was chosen as it has been previously used in the course. Microsoft Expression Web 4 can be downloaded from http://www.microsoft.com/en-us/download/details.aspx?id=36179. It provides a straightforward interface that allows for frequent testing and visualisation of the code on the server. The software is also compatible with all the required languages; PHP, SQL, HTML, CSS and JavaScript.
3.2. Design

After the requirements were clearly established, the design phase could begin. During this phase, the appearance and the structure of the testing tool were designed.

Database

PHPMyAdmin was chosen to develop the MySQL database as it is easily accessible and provides a user-friendly interface for back-end database manipulation. The data provided by the client consisted of ten “input” Excel sheets. Each sheet represented a database table. During the initial meeting with the client, it was established that the input data will always be provided in the same format. Thus, ten static database tables were created and each column was assigned a name and a variable type. A detailed description of the database tables and their relationships can be found in Appendix E.

SQL Queries

The design and structure development of the SQL queries were conducted in SQL Workbench 5.0. This software was ideal as it easily connected to the database and allowed constant testing of the code. It also provided proof that the switch to SQL programming language was feasible by producing the required output; this is further discussed in Section 3.4. SQL Workbench 5.0 is free software that was additionally installed to the WAMP Stack, described in Section 2.3.

User Interface

PHP was the best programming language to use in the development of the sample user interface as it has the ability to interpret the SQL language and easily connect to the MySQL database.

The appearance of the interface was to be clear and concise, so that a non-technical user could easily operate the application. The interface structure contains a front-end “Home” page that allows smooth connections between other pages. All the main pages also contain a Home hyperlink at the bottom which allows the user to return to the “Home” page at any time. The fully functioning back-end of the application was more important than the graphical appearance. Therefore, it was decided to leave the graphical design of the application until all back-end processes were perfectly operating.
3.3. Development Tools

The entire development process of the application required the use of various programming languages and software applications. These are described in more detail in Figure 3.3.1 and Figure 3.3.2.

<table>
<thead>
<tr>
<th>Development Tools</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Workbench 5.0</td>
<td>Used to develop and test SQL queries</td>
</tr>
<tr>
<td>Windows Expression Web 4</td>
<td>Used to develop the sample user interface</td>
</tr>
<tr>
<td>PHPMyAdmin</td>
<td>Used to set up the database</td>
</tr>
</tbody>
</table>

Table 3.3.1.: Tools used to develop the application

<table>
<thead>
<tr>
<th>Development Languages</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL</td>
<td>Used to manipulate database data</td>
</tr>
<tr>
<td>PHP</td>
<td>Used to connect to the database, query the database and output result</td>
</tr>
<tr>
<td>HTML</td>
<td>Used to develop web interface pages</td>
</tr>
<tr>
<td>CSS</td>
<td>Used to develop graphical appearance of HTML pages</td>
</tr>
<tr>
<td>JavaScript</td>
<td>Used to develop necessary user validations</td>
</tr>
</tbody>
</table>

Table 3.3.2.: Programming languages used to develop the application
3.4. Development

The development of the application commenced at the beginning of January 2014 when the design phase of the application was complete. This phase consumed the majority of time as it required an in-depth understanding of the input data and SQL, and a re-familiarisation with the remaining programming languages [5], [6]. Each section of the application was developed separately. It was then thoroughly tested and implemented if successful. Further details on testing are available in Appendix H.

Populating the Database Tables

As it has been previously stated, the MySQL database was designed and developed using PHPMyAdmin. Figure 3.4.1 shows one of the tables. There are two methods available for populating a database table:

- Manually – by writing SQL query in SQL Workbench 5.0. This will include manual coding of each row in the Excel sheet.
- Automatically – using functions available within PHPMyAdmin’s user-friendly interface.

Considering that the smallest Excel sheet provided by the client contained 385 rows of data, the manual population of a table would result in a loss of valuable development time. Therefore the automatic approach was used. All the Excel sheets that were imported into the MySQL database were converted to a CSV format as required by PHPMyAdmin. This process is described in further detail in the User Manual in Appendix C.

<table>
<thead>
<tr>
<th>Sales 2011 Database Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column Heading</td>
</tr>
<tr>
<td>invoiceNo</td>
</tr>
<tr>
<td>invoiceDate</td>
</tr>
<tr>
<td>dueDate</td>
</tr>
<tr>
<td>customerID</td>
</tr>
<tr>
<td>customerName</td>
</tr>
<tr>
<td>dateTrans</td>
</tr>
<tr>
<td>postDate</td>
</tr>
<tr>
<td>description</td>
</tr>
<tr>
<td>grossSale</td>
</tr>
<tr>
<td>vat</td>
</tr>
<tr>
<td>netSale</td>
</tr>
<tr>
<td>quantity</td>
</tr>
<tr>
<td>branch</td>
</tr>
<tr>
<td>despatchNote</td>
</tr>
<tr>
<td>despatchDate</td>
</tr>
<tr>
<td>month</td>
</tr>
</tbody>
</table>

Figure 3.4.1.: Sales 2011 Database Table
Development of SQL Queries

The development of the SQL queries consumed approximately 4 weeks of the development time devoted to the system. In total there were 16 queries created, some were more complicated than others and required more time. This step of the development process was crucial as it determined the further success of the project. If the SQL queries were unable to fulfill the desired output, the transition of the current system to a new programming language would have been considered unsuccessful.

The straightforward queries easily produced the desired output. However, the more problematic queries required major modifications and manipulations of the SQL code. For example, they required the creation of aliases and manual coding of pivot tables. They also required testing after each line of newly written code. To match the output desired by the client, extra columns that were not present in any of the tables, had to be created within the query code. This virtually created a new table without it being stored in MySQL database. These code manipulations required intensive practice and familiarisation with the SQL language. Figure 3.4.2 shows a sample of SQL code that requires aliases. Aliases are created to make column names more readable since the code is complex and columns overlap. In the graph below aliases are shown as an addition to a column name, for example d1.customerID or a1.customerID. The full database structure is shown in Appendix E and the source code can be found in Appendix G.

```sql
SELECT sales2012.customerName, sales2012.customerID, sales2012.gSale sale2012,
    ROUND((sales2012.gSale/(SELECT round(sum(GrossSale)) FROM sales2012)*100),2) perc2012,
    sales2011.gSale sale2011,
    ROUND(((sales2011.gSale/(SELECT round(sum(GrossSale)) FROM sales2011))*100),2) perc2011 FROM
    (SELECT al.customerName, a1.customerID, a1.gSale FROM (SELECT customerName, customerID,
        ROUND(sum(grossSale)) gSale FROM sales2012 GROUP BY 1,2 ORDER BY 2 DESC LIMIT 20 ) a1 ORDER BY 1)
        sales2012,
    (SELECT b1.customerName, b1.customerID, ROUND(sum(b1.grossSale)) gSale FROM sales2011 b1
    WHERE b1.customerID IN (SELECT a2.customerID FROM (SELECT customerID, ROUND(sum(grossSale)) gSale FROM sales2012
        GROUP BY 1 ORDER BY 2 DESC LIMIT 20 ) a2) GROUP BY 1,2 ORDER BY 1 ) sales2011
WHERE sales2011.customerID=sales2012.customerID;
```

Figure 3.4.2.: SQL code for Compare Top 20 Customers query.

SQL Workbench 5.0 provides functionality for the user to export their output with a click of a button to an Excel file. However, this was deemed unsuitable as the Auditor would be exposed to the back-end coding of the queries. This is one of the reasons why the sample user interface was created, to unify all back-end processes and connections – SQL queries, MySQL database and Apache server.
Development of Sample Web Interface

Using Microsoft Expression Web 4, the interface was constructed with PHP to extract data from the database and to produce comprehensive output in the form of Excel files. Using the interface template produced during the design phase, simple HTML pages were created. Figure 3.4.3 shows the links between HTML and PHP pages.

A connection to the MySQL database had to be established before any code could be written in PHP. This was achieved using an additional PHP file - “detail.php”, which contained connection details to the database. This connection was tested by requesting PHP to run a simple SQL query. The use of a separate file to store database connection details, allowed the connection to be more secure, as the database connection details are hidden from unwanted users.

During the interface development process, some SQL queries required code modification as some of the code used was not compatible with the PHP version installed. For example, some of the queries required the introduction of aliases, which are temporary names given to columns to eliminate confusion when columns overlap, which were not required in SQL Workbench 5.0. SQL code such as “Having By” was also not compatible. Finding the cause of the problems and the solutions to the problems, created a minor time set-back in the development of the interface.

As it has been stated previously, one of the main reasons for implementing a sample user interface was to provide the user with automatic output to Excel. In order to generate an Excel file a PHPExcel plug-in was required by PHP. This plug-in was successfully downloaded from http://phpexcel.codeplex.com/releases/view/107442 and installed onto the server. The plug-in was tested by running a simple line of PHP code that was provided with the downloaded plug-in files. The HTML page for client data testing contains checkboxes for each of the tests, allowing the user to select the tests that they wish to complete. Full screenshots of the system are available in Appendix F. These checkboxes are connected to PHP and SQL code. By clicking the “Run” button at the bottom of the HTML page, the user calls the PHP code. The SQL code within PHP is responsible for fetching the data from the database and producing the requested result. The PHP code creates a connection with the database to request data and produce the output in an Excel file. Once the process of creating an Excel file is complete, the user is redirected to the next page which details the amount of time it took each test to complete, the amount of memory used in order to create the Excel file, the name of the Excel file and the exact path of where the Excel file was saved to. The functionality of the PHPExcel plug-in was tested by selecting some and then all checkboxes and the output was compared to the desired output supplied by the client.

An additional HTML page was created to allow the user to update database tables without requiring knowledge of the back-end process. Ten buttons were created for each of the tables which allow the user to browse their computer for the desired CSV file, the file then gets uploaded onto the server and from the server into the chosen database table. The user
is then provided with the option to view the tables. This PHP code was tested by updating database tables and viewing their content on the next page.

Using JavaScript, validations were put into place. The user is notified if text boxes are left blank or if the input is incorrect. A warning message box also appears if no checkboxes were checked for data testing.

The last step of developing the user interface related to the interface design. The graphics consist of the company’s logo and a gradient background, as well as different fonts and styles for the text. This was achieved by creating a separate CSS file and incorporating it into each HTML file. The gradient background was developed by a free online CSS background generator available at http://www.colorzilla.com/gradient-editor/. Having a separate CSS file allows the graphics to appear identical on each HTML page. Changes made to the CSS code will also automatically take place within HTML pages, i.e. if a font colour was changed within CSS code, this change will also appear on all HTML pages that implement the CSS file. Figure 3.4.4 demonstrates the design of the user interface.

Figure 2.4.3.: Links between HTML and PHP pages in the user interface.
3.5. Implementation and Testing

Implementation

Due to the nature of the clients work the application will only be required to be deployed onto one computer. This application will be used for each KPMG client once a year. This means that the client may use the same database and update the contents of each database table with the relevant KPMG client data. A backup of the database should be performed before the database is updated with new data.

To implement the system, the client should install the WAMP Stack. This is an open-source software package that does not require any financial investment and provides the client with all recommended software – Apache server, PHPMyAdmin and PHP. The root folder for the server should be identified and all files that contain code copied into it. The root folder is usually available on the C drive. This folder will contain all PHP, HTML, CSS and JavaScript files. All the files that are present in the root folder may be viewed and executed on any browser installed on the computer used. The installation of additionally required software should be performed. This includes PHPMyAdmin, SQL Workbench 5.0 and Microsoft Expression Web 4, further described in section 2.3. The User Manual, along with the Technical Manual for future modifications can be found at Appendix C and Appendix D. The files required for this application are also included on the accompanying CD.
Testing

As it has been stated previously, the project followed the incremental model shown in Section 3. This means that the code was tested continuously throughout the development process before being implemented. The client provided the input for the database as well as the desired output. When developing the SQL queries, the output achieved from running each query was compared to the output provided by the client. The query was marked as correct when both outputs matched perfectly.

The PHP code was tested by repeatedly choosing various combinations of tests to complete and comparing the Excel file output to the output provided by the client. Each step that required extraction or manipulation of data was tested thoroughly.

The interface was successfully tested on various browsers which included Google Chrome, Mozilla Firefox, Internet Explorer and Opera. This was carried out to ensure software compatibility.

This methodology of testing is believed to be the most effective as each line of code was tested before implementation [7]. Any errors that occurred during the coding process were quickly and easily identified and dealt with. This allowed for the project to run rather smoothly with no major set-backs.

Appendix H provides further details on testing.

3.6. Challenges Encountered

During the development process of the application some issues were encountered. This section will describe the issues in detail and solutions adopted.

Server

Initial issues occurred with the selection of the server to use for the system. Using the client’s server was inefficient as it would require the whole development of the system to take place in the client’s offices. Using the School of Computer Science and Statistics, TCD server was also inefficient as the system could only be developed on College grounds. This would in turn provide difficulty with demonstrating the system’s functionalities to the client. The solution was found by implementing the “localhost” Apache 2.2 server. Section 3.1 further explains why the Apache server was chosen.
Excel Output

As defined earlier the SQL Workbench 5.0 provided an inefficient manner to extract output to Excel file – as the Auditor is exposed to the back-end processes. Therefore, it was decided to create a simple sample user interface to unify all back-end process and provide the user with an automatic way to extract the output to Excel. This process is described further in Section 3.4.

PHP Plug-Ins

Issues were encountered when installing PHP plug-ins. These issues consisted of the plug-in not installing properly onto the server, the plug-in being incompatible with the version of PHP used and code complexities. These issues occurred with the PHPExcel plug in which is used to develop automatic output of generated results to Excel. These problems were dealt with by coding different variations and testing them until the desired output was achieved. Some PHPExcel versions were not compatible with Microsoft Expression Web 4; therefore research was conducted into establishing suitable versions which were then used.
4. CONCLUSION AND RECOMMENDATIONS

4.1. Conclusions

The finished application meets the client’s requirements as outlined in Section 1.3. The old system developed in Excel using VBA Macros was slow, taking up hours of valuable time. Excel was inefficient – as it only stored a maximum of 1,048,576 records of client data at a time. Excel also has some limitations (discussed in Section 1.2). The new system completes all the required functions within ten minutes and it also has the ability to store millions of records. The application also provides uniformity to the data collection, storage and manipulation processes.

Below is the outline of how each requirement as specified in the Terms of Reference was met, as well as the additional requirements:

- Detailed analysis was carried out into the Excel testing tool, the sample data provided and the output generated by the Excel testing tool. (Section 3.1)
- Detailed research was carried out into different servers and development software, to identify the best possible path to take in the development of the new system. (Section 3.1)
- A database was designed and developed to match the current data format. (Section 3.2 and Section 3.4)
- Queries were generated in SQL to ensure that desired output was achieved. (Section 3.4)
- A sample user interface was designed and developed using PHP. (Section 3.2 and Section 3.4)
- SQL queries were implemented within PHP and the desired output achieved. (Section 3.4)
- The output was automatically exported into an Excel file through the user interface. (Section 3.4)
- The user interface allows the user to update database tables with CSV files without being exposed to the back-end procedures. (Section 3.4)
- Audit tests outlined by the client were successfully generated. The user has the option to select Audit tests to complete. (Section 3.4)
- The interface is user-friendly and easy to navigate. (Section 2.5)
- A User Manual was prepared along with a detailed Technical Manual to allow for further manipulations and modifications. These can be found at Appendix C and Appendix D respectively.

This project has been challenging technically and academically. Bringing a range of different software and languages together in one application is a complex process and created some issues. The end product included the use of five different programming languages; SQL, PHP, CSS, HTML and JavaScript. This means that it was necessary to use and expand a range of current skills, as well as developing many new ones. Building such a
complicated system was rewarding as it involved learning how to deal with issues that occurred.

The client may further extend the system if needed. The Technical Manual in Appendix D describes in detail how to deploy the system.

4.2. Recommendations

It is recommended that the client install and implement the WAMP Stack, as described in Section 2.3. The WAMP Stack operates on the Windows operating system, which is used by the client. The WAMP Stack also does not require any financial investments and provides the client with the required software. The Technical Manual in Appendix D provides a step by step guide on how to download the WAMP Stack and setup its components.

It is also recommended that the client install the PHPMyAdmin open source software. This software will allow the client to design and easily develop the database, along with being able to manipulate the database contents, for example deleting a table. In addition to that, the client should install SQL Workbench 5.0 for the development of SQL queries. This software will allow to code and test the queries before implementing them in the PHP code. Microsoft Web Expression 4 should also be installed for the user interface. This software will allow for the manipulation of the PHP code that is responsible for the functionality of the user web interface.

Once the server has been setup, the server root folder called “htdocs” should be identified and all the code documents copied into it; an electronic copy of these is provided on the CD attached. The client is recommended to follow the Test Document available in Appendix H to test if a connection has been established. It is further recommended that the client thoroughly test the application before deploying it. The application should be deployed onto an Apache server only; using a different server will result in substantial code modifications.

Since the client will be implementing the same database to run Audit testing for each KPMG client, the database will be constantly updated with new data. Therefore, it is recommended that the client to maintain a backup of the database for each KPMG client to avoid potential data loss. The backup of a database is carried out in PHPMyAdmin and is a simple process. This is further explained in the Technical Manual in Appendix D.
# APPENDICES

<table>
<thead>
<tr>
<th>NO.</th>
<th>SECTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>Original Project Outline</td>
<td>A.1</td>
</tr>
<tr>
<td>B.</td>
<td>Interim Project Report</td>
<td>B.1</td>
</tr>
<tr>
<td>C.</td>
<td>User Manual</td>
<td>C.1</td>
</tr>
<tr>
<td>D.</td>
<td>Technical Manual</td>
<td>D.1</td>
</tr>
<tr>
<td>E.</td>
<td>Design Documentation</td>
<td>E.1</td>
</tr>
<tr>
<td>F.</td>
<td>Screenshots</td>
<td>F.1</td>
</tr>
<tr>
<td>G.</td>
<td>Source Code</td>
<td>G.1</td>
</tr>
<tr>
<td>H.</td>
<td>Test Documentation</td>
<td>H.1</td>
</tr>
</tbody>
</table>

**GLOSSARY TERMS USED**

**REFERENCES**
A. ORIGINAL PROJECT OUTLINE

Management Science and Information System Studies

Client: KPMG Ireland
Project: Create Database to Assist Auditors with Client Data Testing and Analysis
Address: 1 Stokes Place, St Stephen's Green, Dublin 2
Client Contact: Selwyn Hearns, (01) 410 1958, selwyn.hearns@kpmg.ie
Dept. Contact: IRM Audit

Client Background

KPMG is a leading provider of professional services in Ireland, working with clients in all sectors of Irish business, providing a range of Audit, Tax and Advisory services. The firm has 78 partners and 1,800 people in offices in Dublin, Cork, Galway and Belfast.

Project Requirements

At present, the Audit practice within KPMG manually analyses and tests obtained client data, which can include millions of records at a time. This is time consuming and susceptible to human error. Currently, simple data tests such as identifying negative values or duplicate values, consumes a lot of valuable time.

The Information Risk Management (IRM) team have developed a testing tool in Excel using VBA Macros. The existing testing tool contains many limitations, these include; taking too much time to process a request and inability to test more than a million records at a time. As a result, the IRM team wish to develop a proof of concept database system to evaluate whether the testing and analysis of client data can be run more efficiently. The new system must store the obtained client data in a database; complete testing selected by the user and output the results to an Excel file. It is essential to make the new system user-friendly.

What is involved for the student?

- Review the data collected for Audit
- In-depth understanding of the output reports currently generated
- Create a database and generate selected output reports
- Provide output in a tabular format in an Excel or Word file
- Produce comprehensive user documentation
B. INTERIM PROJECT REPORT

Management Science and Information System Studies

Project: Create Database to Assist Auditors with Client Data Testing and Analysis

Client: KPMG Ireland

Student: Dasha Gorovenco

Supervisor: Aideen Keaney

Review of Background and Work to Date

KPMG Ireland provide Tax, Audit and Advisory services to various clients worldwide. On a daily basis, an Auditor will go through a lot of client data manually. This data is submitted by the client to the Audit team every time a new audit takes place.

KPMG Ireland are developing and evaluating means by which data can be extracted, and audit tests automated, in an electronic fashion thereby increasing the efficiency of the audit process. As a result the Information Risk Management Audit team wish to develop a proof of concept database system to evaluate whether the testing and analysis of client data can be run more efficiently. It is essential to make the product user friendly. The IRM Audit team have developed a data testing and analysis tool using VBA Macros in Excel. IRM Audit wish to find out if it is possible to test and analyse client data on a database platform that can manage more than two million records at a time.

To date, several meetings with the client and project supervisor have been held in order to establish the work involved. Research has been conducted into the software required and data has been provided.

Terms of Reference

- Examine the format of data currently collected for Audit
- Examine reports that are currently generated
- Design a database to match data collected and to generate selected output reports
- Print output to Excel or Word file in a tabular format
- Produce comprehensive user documentation for the client

Further Work

- Carry out basic analysis to become familiar with the manipulation of the dataset using SQL
- Examine the data structure and reporting products of the current excel tool
- Develop the database
- Develop queries to match output reports
- Produce comprehensive user documentation
C. USER MANUAL

The User Manual is attached separately to this report. It provides the user with a guide on how to operate various functions within the user interface. A copy of the manual is also provided on the attached CD.
D. TECHNICAL MANUAL

The Technical Manual is additionally attached to this report. It is a guide written for the client on how to deploy the application. An electronic copy of the manual is also available on the CD attached.
E. DESIGN DOCUMENTATION

This appendix describes the process undertaken while designing the system. It describes the methodology used, the design of database tables and SQL queries, as well as the purpose of each PHP, HTML and CSS page in the interface.

E.1. Incremental Model

The developed system follows the incremental model which involves designing, implementing and testing the product until it’s finished. The model involves both development and maintenance, and consists of five stages; requirements analysis, design, development, implementation and testing.

![Incremental Model Diagram]

Figure E.1.1: Incremental Model

E.2. Database Tables

An entity relationship diagram shows the links between the database tables. As described in Section 3.2, the client stated that the data provided for Audit testing will always appear in the same format; therefore the database consists of ten tables. Figure E.2.1 below describes the links between tables in the new system.

The tables can be populated automatically and manually as described in the User Manual in Appendix C.
Figure E.2.1: Entity Relationship Diagram

Key

- Blue column names represent the primary keys. These are unique identifiers of that particular table.
- The black arrows show the links between tables.
- Red column names represent another way of linking table together. Eight out of ten tables have the Customer ID column. This means that a query can be created using tables that only have the Customer ID in common.
- For this application the links between tables were coded.
E.3. SQL Queries

The SQL queries were designed to fully match the output provided by the client. In total 16 queries were created. These are outlined below:

- Duplicates in Sales – displays all duplicate invoice IDs found.
- Duplicates in Despatch Notes - displays all duplicate despatch note IDs found.
- Duplicates in Credit Notes – displays all duplicate credit note IDs found.
- Duplicates in Debtors – displays all duplicate debtor IDs found.
- Negative Values in Sale – displays all negative values in Sales tables.
- Negative Values in Credit Notes – displays all negative values in Credit Note tables.
- Negative Values in Debtors – displays all negative values in Debtor tables.
- Compare Top 20 Debtors – compares the current year and prior year of top 20 debtors.
- Summary Table of Sales by Branch – provides a summary of sales per each month per branch.
- Detailed Analysis by Customer - provides the analysis of invoices and credit notes per each customer, as well as percentage breakdown of each.
- Posted and Issues Date Cut-Off Testing

<table>
<thead>
<tr>
<th>Sales Invoices</th>
<th>Issued Date Before Year End, Posted After</th>
<th>Issued Date After Year End, Posted Before</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credit Notes</td>
<td>Issued Date Before Year End, Posted After</td>
<td>Issued Date After Year End, Posted Before</td>
</tr>
<tr>
<td>Despatch Notes</td>
<td>Issued Date Before Year End, Posted After</td>
<td>Issued Date After Year End, Posted Before</td>
</tr>
</tbody>
</table>

E.4. Interface Page Descriptions

This section describes what each page within the interface is responsible for. Figure 3.4.3 in Section 3.4 on page 19 shows the links between these pages.

HTML

- home.html – this is the home page of the interface. Provides the user with three options; update the database, view current database tables or complete Audit testing on current database.
- uploadFilesBackup.html – this page allows the user to update database tables by selecting CSV files.
- viewTables.html – provides the user with two options; to view current database tables or to complete Audit tests.
- selectViewTables.html – allows the user to select tables to overview.
- runTestPage.html – allows the user to select Audit tests to complete and print the output to Excel.
PHP

- clearTables.php – clears the content of current database tables and uploads new data.
- detail.php – includes connection details for the database.
- showTableCash2011.php – shows the contents of Cash2011 database table.
- showTableCash2012.php – shows the contents of Cash2012 database table.
- showTableCredit2011.php – shows the contents of Cn2011 database table.
- showTableCredit2012.php – shows the contents of Cn2012 database table.
- showTableDebtors2011.php – shows the contents of Dr2011 database table.
- showTableDebtors2012.php – shows the contents of Dr2012 database table.
- showTableDespatch2011.php – shows the contents of Des2011 database table.
- showTableDespatch2012.php – shows the contents of Des2012 database table.
- showTableSales2011.php – shows the contents of Sales2011 database table.
- showTableSales2012.php – shows the contents of Sales2012 database table.
- testconnect.php – tests if connection has been established with the database.

CSS

- background.css – contains the graphics of the interface
This section contains screenshots of the user interface.

Figure F.2: Home page

Figure F.3: Update Database page
Figure F.4: Browse file to upload

Figure F.5: View Current Database Tables page
Figure F.6: Audit testing page

Figure F.7: Date Validations on Audit testing page
Figure F.8: Test Validations on Audit testing page

Figure F.9: Chose Action after Uploading Files page
There are 4574 records.

<table>
<thead>
<tr>
<th></th>
<th>Date</th>
<th>Amount</th>
<th>Code</th>
<th>Code 2</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>328</td>
<td>2011-03-33</td>
<td>10569</td>
<td>10001</td>
<td>15</td>
<td>2011</td>
</tr>
<tr>
<td>45</td>
<td>2011-03-33</td>
<td>10006</td>
<td>10002</td>
<td>15</td>
<td>2011</td>
</tr>
<tr>
<td>159</td>
<td>2011-03-33</td>
<td>10883</td>
<td>10003</td>
<td>15</td>
<td>2011</td>
</tr>
<tr>
<td>443</td>
<td>2011-03-33</td>
<td>10923</td>
<td>10004</td>
<td>15</td>
<td>2011</td>
</tr>
<tr>
<td>2.938</td>
<td>2011-03-33</td>
<td>10698</td>
<td>10005</td>
<td>15</td>
<td>2011</td>
</tr>
<tr>
<td>383</td>
<td>2011-03-33</td>
<td>10335</td>
<td>10006</td>
<td>15</td>
<td>2011</td>
</tr>
<tr>
<td>229</td>
<td>2011-03-33</td>
<td>10235</td>
<td>10007</td>
<td>15</td>
<td>2011</td>
</tr>
<tr>
<td>500</td>
<td>2011-03-33</td>
<td>10019</td>
<td>10008</td>
<td>15</td>
<td>2011</td>
</tr>
<tr>
<td>549</td>
<td>2011-03-33</td>
<td>10390</td>
<td>10009</td>
<td>15</td>
<td>2011</td>
</tr>
<tr>
<td>339</td>
<td>2011-03-33</td>
<td>10508</td>
<td>10010</td>
<td>15</td>
<td>2011</td>
</tr>
<tr>
<td>179</td>
<td>2011-03-33</td>
<td>10134</td>
<td>10011</td>
<td>15</td>
<td>2011</td>
</tr>
<tr>
<td>353</td>
<td>2011-03-33</td>
<td>10329</td>
<td>10012</td>
<td>15</td>
<td>2011</td>
</tr>
<tr>
<td>159</td>
<td>2011-03-34</td>
<td>10201</td>
<td>10013</td>
<td>15</td>
<td>2011</td>
</tr>
<tr>
<td>300</td>
<td>2011-04-03</td>
<td>10089</td>
<td>10014</td>
<td>15</td>
<td>2011</td>
</tr>
<tr>
<td>199</td>
<td>2011-03-34</td>
<td>10594</td>
<td>10015</td>
<td>15</td>
<td>2011</td>
</tr>
<tr>
<td>579</td>
<td>2011-03-34</td>
<td>10320</td>
<td>10016</td>
<td>15</td>
<td>2011</td>
</tr>
<tr>
<td>279</td>
<td>2011-03-34</td>
<td>10175</td>
<td>10017</td>
<td>15</td>
<td>2011</td>
</tr>
</tbody>
</table>

Figure F.10: View Table Cash 2011 page

There are 6122 records.

<table>
<thead>
<tr>
<th></th>
<th>Date</th>
<th>Amount</th>
<th>Code</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>299</td>
<td>2012-03-01</td>
<td>10220</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>279</td>
<td>2012-03-01</td>
<td>10451</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>300</td>
<td>2012-03-31</td>
<td>10704</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>199</td>
<td>2012-03-01</td>
<td>10430</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>129</td>
<td>2012-03-31</td>
<td>10841</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>229</td>
<td>2012-03-01</td>
<td>10777</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>129</td>
<td>2012-03-01</td>
<td>10553</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>3.429</td>
<td>2012-03-01</td>
<td>10413</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>279</td>
<td>2012-03-01</td>
<td>10654</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>279</td>
<td>2012-03-01</td>
<td>10300</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>283</td>
<td>2012-03-31</td>
<td>10439</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>259</td>
<td>2012-04-01</td>
<td>10439</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>179</td>
<td>2012-04-01</td>
<td>10912</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>279</td>
<td>2012-03-02</td>
<td>10600</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>869</td>
<td>2012-03-02</td>
<td>10511</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>48</td>
<td>2012-03-02</td>
<td>10353</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>279</td>
<td>2012-03-02</td>
<td>10732</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>303</td>
<td>2012-03-02</td>
<td>10516</td>
<td>15</td>
<td>2012</td>
</tr>
<tr>
<td>779</td>
<td>2012-03-02</td>
<td>10411</td>
<td>15</td>
<td>2012</td>
</tr>
</tbody>
</table>

Figure F.11: View Table Cash 2012 page
There are 439 records.

<table>
<thead>
<tr>
<th>Code</th>
<th>Date</th>
<th>Amount</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN00032</td>
<td>2011-01-15</td>
<td>10005</td>
<td>Adrienne Kennedy</td>
</tr>
<tr>
<td>CN00033</td>
<td>2011-01-15</td>
<td>10848</td>
<td>Co Tipperary (NR) VEC</td>
</tr>
<tr>
<td>CN00034</td>
<td>2011-01-24</td>
<td>10938</td>
<td>Co Kildare VEC</td>
</tr>
<tr>
<td>CN00035</td>
<td>2011-01-25</td>
<td>10075</td>
<td>Aghada Pharmacy Ltd</td>
</tr>
<tr>
<td>CN00036</td>
<td>2011-01-21</td>
<td>10210</td>
<td>Applefield Ltd</td>
</tr>
<tr>
<td>CN00037</td>
<td>2011-01-14</td>
<td>10576</td>
<td>Callaghans the Chemist Ltd T/A Irishtown Pharmacy</td>
</tr>
<tr>
<td>CN00038</td>
<td>2011-01-25</td>
<td>10879</td>
<td>Clare County Council</td>
</tr>
<tr>
<td>CN00039</td>
<td>2011-01-14</td>
<td>10082</td>
<td>Adrian Copeland t/a Physio Solutions</td>
</tr>
<tr>
<td>CN00040</td>
<td>2011-01-25</td>
<td>10901</td>
<td>Clinical Trial Endpoint (CTEP)</td>
</tr>
<tr>
<td>CN00041</td>
<td>2011-01-15</td>
<td>10508</td>
<td>Brian McDonald</td>
</tr>
<tr>
<td>CN00042</td>
<td>2011-01-27</td>
<td>10762</td>
<td>Castletroy SCP Ltd</td>
</tr>
</tbody>
</table>

Figure F.12: View Table Credit 2011

There are 435 records.

<table>
<thead>
<tr>
<th>Code</th>
<th>Date</th>
<th>Amount</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN00032</td>
<td>2012-01-10</td>
<td>2012-01-12</td>
<td>10005</td>
</tr>
<tr>
<td>CN00033</td>
<td>2012-01-11</td>
<td>2012-01-13</td>
<td>10049</td>
</tr>
<tr>
<td>CN00034</td>
<td>2012-01-19</td>
<td>2012-01-21</td>
<td>10939</td>
</tr>
<tr>
<td>CN00035</td>
<td>2012-01-20</td>
<td>2012-01-22</td>
<td>10075</td>
</tr>
<tr>
<td>CN00036</td>
<td>2012-01-16</td>
<td>2012-01-21</td>
<td>10210</td>
</tr>
<tr>
<td>CN00037</td>
<td>2012-01-08</td>
<td>2012-01-11</td>
<td>10876</td>
</tr>
<tr>
<td>CN00038</td>
<td>2012-01-23</td>
<td>2012-01-25</td>
<td>10876</td>
</tr>
<tr>
<td>CN00039</td>
<td>2012-01-09</td>
<td>2012-01-11</td>
<td>10062</td>
</tr>
<tr>
<td>CN00040</td>
<td>2012-01-19</td>
<td>2012-01-20</td>
<td>10901</td>
</tr>
<tr>
<td>CN00041</td>
<td>2012-01-13</td>
<td>2012-01-15</td>
<td>10508</td>
</tr>
<tr>
<td>CN00042</td>
<td>2012-01-22</td>
<td>2012-01-24</td>
<td>10762</td>
</tr>
</tbody>
</table>

Figure F.13: View Table Credit 2012 page
There are 399 records.

<table>
<thead>
<tr>
<th>#</th>
<th>Name of Company</th>
<th>Amount</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>OxygenS Media Ireland Limited</td>
<td>115.93</td>
<td>1000</td>
</tr>
<tr>
<td>1001</td>
<td>0800 Reverse Pty Ltd</td>
<td>451.66</td>
<td>1000</td>
</tr>
<tr>
<td>1002</td>
<td>20/20 Imaging Limited</td>
<td>404.58</td>
<td>1000</td>
</tr>
<tr>
<td>1003</td>
<td>328dental</td>
<td>967.5</td>
<td>1000</td>
</tr>
<tr>
<td>1004</td>
<td>3V Transaction Services Ltd</td>
<td>9054.89</td>
<td>5000</td>
</tr>
<tr>
<td>1005</td>
<td>4E Fulfillment.com Ltd t/a GiftsDirect.com, Slopes.com, TheFishStore.com, ReaderOffers</td>
<td>397.75</td>
<td>1000</td>
</tr>
<tr>
<td>1006</td>
<td>A and A Pharmacy Limited-trading as A and A Pharmacy</td>
<td>410.93</td>
<td>1000</td>
</tr>
<tr>
<td>1007</td>
<td>A C Boles Ltd</td>
<td>1248.33</td>
<td>5000</td>
</tr>
<tr>
<td>1008</td>
<td>A E Digan Ltd (Pharmacy) t/a McCartans Pharmacy</td>
<td>551.57</td>
<td>1000</td>
</tr>
<tr>
<td>1009</td>
<td>A J Hallahan Limited</td>
<td>1355.39</td>
<td>5000</td>
</tr>
<tr>
<td>1010</td>
<td>A J Quirke (Pharmacist) Ltd</td>
<td>8016.83</td>
<td>5000</td>
</tr>
<tr>
<td>1011</td>
<td>A.C.M Ltd t/a Glasscare</td>
<td>820.92</td>
<td>1000</td>
</tr>
<tr>
<td>1012</td>
<td>Abbey Healthcare Limited</td>
<td>433.33</td>
<td>1000</td>
</tr>
</tbody>
</table>

Figure F.14: View Table Debtors 2011 page

There are 395 records.

<table>
<thead>
<tr>
<th>#</th>
<th>Name of Company</th>
<th>Amount</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1004</td>
<td>3V Transaction Services Ltd</td>
<td>104.88</td>
<td>200</td>
</tr>
<tr>
<td>1005</td>
<td>4E Fulfillment.com Ltd t/a GiftsDirect.com, Slopes.com, TheFishStore.com, ReaderOffers</td>
<td>243.09</td>
<td>1000</td>
</tr>
<tr>
<td>1006</td>
<td>A and A Pharmacy Limited-trading as A and A Pharmacy</td>
<td>405.69</td>
<td>1000</td>
</tr>
<tr>
<td>1007</td>
<td>A C Boles Ltd</td>
<td>1217.85</td>
<td>5000</td>
</tr>
<tr>
<td>1008</td>
<td>A J Quirke (Pharmacist) Ltd</td>
<td>454.47</td>
<td>1000</td>
</tr>
<tr>
<td>1009</td>
<td>A.C.M Ltd t/a Glasscare</td>
<td>1313.82</td>
<td>5000</td>
</tr>
<tr>
<td>1010</td>
<td>Abbey Healthcare Limited</td>
<td>881.3</td>
<td>1000</td>
</tr>
<tr>
<td>1011</td>
<td>Abbey Leigh Limited</td>
<td>3325.2</td>
<td>5000</td>
</tr>
<tr>
<td>1012</td>
<td>Abbey Pharmacy</td>
<td>726.82</td>
<td>1000</td>
</tr>
<tr>
<td>1013</td>
<td>Abbey Street Pharmacy Ltd T/A Abbey Street Pharmacy</td>
<td>243.82</td>
<td>1000</td>
</tr>
<tr>
<td>1014</td>
<td>Abbeyvix Credit Union Ltd</td>
<td>908.33</td>
<td>1000</td>
</tr>
<tr>
<td>1015</td>
<td>Ability West</td>
<td>1688.2</td>
<td>5000</td>
</tr>
<tr>
<td>1016</td>
<td>ABN AMRC Administration Services (Ireland) Limited</td>
<td>251.06</td>
<td>1000</td>
</tr>
<tr>
<td>1017</td>
<td>Abian Limited</td>
<td>400.61</td>
<td>1000</td>
</tr>
<tr>
<td>1018</td>
<td>Acadis International Insurance Limited</td>
<td>840.56</td>
<td>1000</td>
</tr>
</tbody>
</table>

Figure F.15: View Table Debtors 2012 page
There are 5816 records.

| DES8001 | 2011-01-31 | 2011-12-29 | 10686 | Bencoopharm Ltd t/a Kieran's Pharmacy | 85 | 273.33 | 1 |
| DES8001 | 2011-12-27 | 2011-12-28 | 10908 | Clonaslee Pharmacy Ltd | 86 | 37.38 | 0 |
| DES8001 | 2010-12-28 | 2010-12-31 | 10637 | Canada Life International Assurance Limited | 25 | 105.93 | 0 |
| DES8001 | 2011-12-31 | 2011-01-03 | 10923 | Close Brothers Limited | 28 | 396.17 | 0 |
| DES80025 | 2010-12-30 | 2010-12-27 | 10896 | Capita IB Solutions | 82 | 2448.33 | 0 |
| DES80025 | 2010-12-28 | 2010-12-24 | 10357 | Balbriggan Estate Property Limited | 14 | 327.5 | 0 |
| DES80025 | 2010-12-28 | 2010-12-28 | 10235 | Avelibe Communications Limited | 5 | 100.93 | 0 |
| DES80025 | 2010-12-30 | 2010-12-27 | 10019 | Abbey Street Pharmacy Ltd t/A Abbey Street Pharmacy | 2 | 416.88 | 0 |
| DES80025 | 2010-12-28 | 2010-12-27 | 10398 | Bank of Ireland Mortgage Bank, t/a Bank of Ireland Mortgages | 3 | 290.83 | 0 |
| DES80025 | 2010-12-31 | 2011-12-29 | 10506 | Blackrock Clinic Ltd | 66 | 202.5 | 0 |

Figure F.16: View Table Despatch 2011 page

There are 6115 records.

| DES0015 | 2011-12-22 | 2011-12-22 | 10220 | Arbor Property & Casualty Ireland Limited | 54 | 243.09 | 0 | 0 | 0 | 0 |
| DES0016 | 2011-12-22 | 2011-12-19 | 10491 | BIOGEN IDEC HEMOPHILIA INC t/a BIOGEN IDEC HEMOPHILIA INC | 81 | 226.83 | 0 | 1 | 1 | 0 |
| DES0017 | 2011-12-22 | 2011-12-20 | 10704 | Capway Ltd. | 10 | 250.33 | 0 | 1 | 1 | 0 |
| DES0018 | 2011-12-23 | 2011-12-24 | 10430 | Beacon Care Fertility | 11 | 101.79 | 0 | 1 | 1 | 0 |
| DES0019 | 2011-12-23 | 2011-12-20 | 10841 | Christy O Rourke | 22 | 104.88 | 0 | 1 | 1 | 0 |
| DES0020 | 2011-12-23 | 2011-12-28 | 10777 | Cavanator t/a Lator's pharmacy | 41 | 186.18 | 0 | 1 | 1 | 0 |
| DES0021 | 2011-12-23 | 2011-12-24 | 10053 | C H Chemists (Tralee) Ltd | 12 | 104.39 | 0 | 1 | 1 | 0 |
| DES0022 | 2011-12-24 | 2011-12-24 | 10803 | Bewleys | 1 | 66.1 | 0 | 1 | 1 | 0 |

Figure F.17: View Table Despatch 2012 page
There are 5820 records.

<table>
<thead>
<tr>
<th>Customer ID</th>
<th>Date</th>
<th>Product</th>
<th>Product Code</th>
<th>Product Type</th>
<th>Product Name</th>
<th>Year</th>
<th>Location</th>
<th>Category</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>10001</td>
<td>2011-01-01</td>
<td>2011-02-01</td>
<td>10568</td>
<td>Product 0058</td>
<td>328</td>
<td>64.67</td>
<td>273.33</td>
<td>55</td>
<td>Galway</td>
</tr>
<tr>
<td>10002</td>
<td>2011-01-01</td>
<td>2011-02-01</td>
<td>10909</td>
<td>Product 0012</td>
<td>44.5</td>
<td>7.42</td>
<td>37.08</td>
<td>05</td>
<td>Dublin</td>
</tr>
<tr>
<td>10003</td>
<td>2011-01-01</td>
<td>2011-02-01</td>
<td>10883</td>
<td>Product 0074</td>
<td>199</td>
<td>33.17</td>
<td>165.92</td>
<td>66</td>
<td>Dublin</td>
</tr>
<tr>
<td>10004</td>
<td>2011-01-01</td>
<td>2011-02-01</td>
<td>10923</td>
<td>Product 0052</td>
<td>493</td>
<td>73.83</td>
<td>258.17</td>
<td>28</td>
<td>Dublin</td>
</tr>
<tr>
<td>10005</td>
<td>2011-01-01</td>
<td>2011-02-01</td>
<td>10894</td>
<td>Product 0005</td>
<td>2030</td>
<td>459.67</td>
<td>4449.33</td>
<td>82</td>
<td>Dublin</td>
</tr>
<tr>
<td>10006</td>
<td>2011-01-01</td>
<td>2011-02-01</td>
<td>10333</td>
<td>Product 0030</td>
<td>189</td>
<td>86.6</td>
<td>277.6</td>
<td>14</td>
<td>Dublin</td>
</tr>
<tr>
<td>10007</td>
<td>2011-01-01</td>
<td>2011-02-01</td>
<td>10230</td>
<td>Product 0014</td>
<td>229</td>
<td>38.17</td>
<td>190.83</td>
<td>5</td>
<td>Dublin</td>
</tr>
<tr>
<td>10008</td>
<td>2011-01-01</td>
<td>2011-02-01</td>
<td>10019</td>
<td>Product 0003</td>
<td>289.6</td>
<td>83.32</td>
<td>418.88</td>
<td>2</td>
<td>Cork</td>
</tr>
</tbody>
</table>

Figure F.18: View Table Sales 2011 page

There are 6122 records.

<table>
<thead>
<tr>
<th>Customer ID</th>
<th>Date</th>
<th>Product</th>
<th>Product Code</th>
<th>Product Type</th>
<th>Product Name</th>
<th>Year</th>
<th>Location</th>
<th>Category</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>20000</td>
<td>2012-01-01</td>
<td>2012-02-01</td>
<td>10220</td>
<td>Product 0023</td>
<td>259</td>
<td>55.91</td>
<td>245.05</td>
<td>8</td>
<td>Cork</td>
</tr>
<tr>
<td>20001</td>
<td>2012-01-01</td>
<td>2012-02-01</td>
<td>10481</td>
<td>Product 0024</td>
<td>279</td>
<td>52.17</td>
<td>220.63</td>
<td>4</td>
<td>Dublin</td>
</tr>
<tr>
<td>20002</td>
<td>2012-01-01</td>
<td>2012-02-01</td>
<td>10704</td>
<td>Product 0002</td>
<td>289.3</td>
<td>49.57</td>
<td>250.33</td>
<td>1</td>
<td>Dublin</td>
</tr>
<tr>
<td>20003</td>
<td>2012-01-01</td>
<td>2012-02-01</td>
<td>10450</td>
<td>Product 0020</td>
<td>129</td>
<td>24.12</td>
<td>104.88</td>
<td>11</td>
<td>Dublin</td>
</tr>
<tr>
<td>20004</td>
<td>2012-01-01</td>
<td>2012-02-01</td>
<td>10481</td>
<td>Product 0002</td>
<td>169</td>
<td>37.21</td>
<td>161.79</td>
<td>54</td>
<td>Dublin</td>
</tr>
<tr>
<td>20005</td>
<td>2012-01-01</td>
<td>2012-02-01</td>
<td>10681</td>
<td>Product 0020</td>
<td>229</td>
<td>24.12</td>
<td>104.88</td>
<td>11</td>
<td>Dublin</td>
</tr>
<tr>
<td>20006</td>
<td>2012-01-01</td>
<td>2012-02-01</td>
<td>10704</td>
<td>Product 0002</td>
<td>289.3</td>
<td>49.57</td>
<td>250.33</td>
<td>1</td>
<td>Dublin</td>
</tr>
<tr>
<td>20007</td>
<td>2012-01-01</td>
<td>2012-02-01</td>
<td>10450</td>
<td>Product 0020</td>
<td>129</td>
<td>24.12</td>
<td>104.88</td>
<td>11</td>
<td>Dublin</td>
</tr>
<tr>
<td>20008</td>
<td>2012-01-01</td>
<td>2012-02-01</td>
<td>10481</td>
<td>Product 0002</td>
<td>169</td>
<td>37.21</td>
<td>161.79</td>
<td>54</td>
<td>Dublin</td>
</tr>
</tbody>
</table>

Figure F.19: View Table Sales 2012 page
Create new PHPExcel object
Testing for Duplicates in Sales (Year end)
Testing for Duplicates in Credit Notes (Year end)
Testing for Duplicates in Despatch Notes (Year end)
Testing for Duplicates in Debtor (Year end)
Testing for Negatives in Sales (Year end)
Testing for Negatives in Credit Notes (Year end)
Testing for Negatives in Debtor (Year end)
Testing for Top 20 Debtor (Year end)
Testing for Summary of Sales by Branch (Year end)
Detailed Analysis of Customers
Cut-Off Testing for Invoices
Cut-Off Testing for Credit Notes
Cut-Off Testing for Despatch Notes
Cut-Off Testing for Despatch Notes
Write to Excel2007 format
File written to run Testing.xlsx
Time to write Workbook was 4.1821 seconds
Current memory usage: 27.25 MB
Peak memory usage: 28.75 MB
Done writing files
Files have been created in C:\Apache2.2\htdocs

Figure F.20: Confirmation and details of Audit tests completed

Figure F.21: Excel output of result
G. SOURCE CODE

This appendix contains the source code used in the application. A digital copy of the source code can be found on the CD provided.

SQL Queries

- **Duplicate of Invoice ID in Sales2012**

```sql
SELECT invoiceNo, invoiceDate, dueDate, customerID, customerName, dateTrans, postDate, description, grossSale, vat, netSale, quantity, branch, despatchNote
FROM sales2012
WHERE invoiceNo IN (SELECT invoiceNo FROM sales2012 GROUP BY invoiceNo HAVING COUNT(invoiceNo) > 1);
```

- **Duplicate of Despatch Notes ID in Des2012**

```sql
SELECT despatchNote, despatchDate, postDate, customerID, customerName, quantity
FROM des2012
WHERE despatchNote IN (SELECT despatchNote FROM des2012 GROUP BY despatchNote HAVING COUNT(despatchNote) > 1);
```

- **Duplicate of Credit Notes ID in Cn2012**

```sql
SELECT `creditNoteNumber`, `creditNoteDate`, `postDate`, `customerID`, `customerName`, `creditNoteAmount`
FROM cn2012
WHERE creditNoteNumber IN (SELECT creditNoteNumber FROM cn2012 GROUP BY creditNoteNumber HAVING COUNT(creditNoteNumber) > 1);
```

- **Duplicate of Debtors ID from Dr2012**

```sql
SELECT debtorID, debtorName, currentBal, creditLimit
FROM dr2012
WHERE debtorID IN (SELECT debtorID FROM dr2012);
```
GROUP BY debtorID
HAVING COUNT(debtorID)>1
);

- **Negative Net Sales Values is Sales2012**

```
SELECT invoiceNo, invoiceDate, dueDate, customerID, customerName, dateTrans, postDate, description, grossSale, vat, netSale, quantity, branch, despatchNote
FROM sales2012
WHERE netSale<0;
```

- **Negative Credit Note Amount Values in Cn2012**

```
SELECT creditNoteNumber, creditNoteDate, postDate, customerID, customerName, creditNoteAmount
FROM cn2012
WHERE creditNoteAmount<0;
```

- **Negative Current Balance Values in Dr2012**

```
SELECT debtorID, debtorName, currentBal, CreditLimit
FROM dr2012
WHERE currentBal<0;
```

- **Posted and Issued Cut-Off Testing for Sales Invoices from Sales2012 – issued date before year end date, posted after year end date**

```
SELECT invoiceNo, invoiceDate, dueDate, customerID, customerName, dateTrans, postDate, description, grossSale, vat, netSale, quantity, branch, despatchNote
FROM sales2012
WHERE invoiceDate<20130101 AND postDate>20121231
ORDER BY InvoiceDate;
```

- **Posted and Issued Cut-Off Testing for Sales Invoices from Sales2012 – issued date after year end date, posted before year end date**

```
SELECT invoiceNo, invoiceDate, dueDate, customerID, customerName, dateTrans, postDate, description, grossSale, vat, netSale, quantity, branch, despatchNote
FROM sales2012
WHERE invoiceDate>20121231 AND postDate<20130101
ORDER BY InvoiceDate;
```

- **Posted and Issued Cut-Off Testing for Credit Notes from Cn2012 – issued date before year end date, posted after year end date**

```
SELECT creditNoteNumber, creditNoteDate, postDate, customerID, customerName, creditNoteAmount
FROM cn2012
WHERE creditNoteDate<20130101 AND postDate>20121231
ORDER BY creditNoteDate;
```
• Posted and Issued Cut-Off Testing for Credit Notes from Cn2012 – issued date after year end date, posted before year end date

```sql
SELECT creditNoteNumber, creditNoteDate, postDate, customerID, customerName, creditNoteAmount
FROM cn2012
WHERE creditNoteDate>20121231 AND postDate<20130101
ORDER BY creditNoteDate;
```

• Posted and Issued Cut-Off Testing for Despatch Notes from Cn2012 – issued date before year end date, posted after year end date

```sql
SELECT despatchNote, despatchDate, postDate, customerID, customerName, quantity
FROM des2012
WHERE despatchDate<20130101 AND postDate>20121231
ORDER BY despatchDate;
```

• Posted and Issued Cut-Off Testing for Despatch Notes from Cn2012 – issued date after year end date, posted before year end date

```sql
SELECT despatchNote, despatchDate, postDate, customerID, customerName, quantity
FROM des2012
WHERE despatchDate>20121231 AND postDate<20130101
ORDER BY despatchDate;
```

• Compare Top 20 Debtors

```sql
(select a1.customerName,a1.customerID, a1.GSale from (select customerName,customerID,Round(sum(grossSale)) GSale from sales2012 group by 1,2 order by 3 desc limit 20 ) a1 order by 1) sales2012,

(select b1.customerName, b1.customerID, round(sum(b1.grossSale)) Gsl from sales2011 b1 where b1.customerId in ( select a2.customerId from (select customerName,customerID,Round(sum(grossSale)) GSale from sales2012 group by 1 order by 2 desc limit 20 ) a2) group by 1,2 order by 1 ) sales2011

where sales2011.customerID=sales2012.customerID;
```

• Summary Table for Sales by Branch

```sql
```
from

Select d1.customername,d1.customerId, d1.INV_Q1_2012, d1.INV_Q2_2012, d1.INV_Q3_2012,d1.INV_Q4_2012,d1.INV_Q1_2013,d1.INV_ToTal_2012,ifnull(d2.CN_Q1_2012,0) CNT_Q1_2012 , ifnull(d2.CN_Q2_2012,0) CNT_Q2_2012, ifnull(d2.CN_Q3_2012,0) CNT_Q3_2012, ifnull(d2.CN_Q4_2012,0) CNT_Q4_2012,ifnull(d2.CN_Q1_2013,0) CNT_Q1_2013,ifnull(d2.CN_ToTal_2012,0) CNT_ToTal_2012,concat(ifnull(round(((ifnull(d2.CN_Q1_2012,0)/d1.INV_Q1_2012)* 100),0),'%') CrByInvQ1_2012,concat(ifnull(round(((ifnull(d2.CN_Q2_2012,0)/d1.INV_Q2_2012)*1 00),0),'%') CrByInvQ2_2012,concat(ifnull(round(((ifnull(d2.CN_Q3_2012,0)/d1.INV_Q3_2012)*1 00),0),'%') CrByInvQ3_2012,concat(ifnull(round(((ifnull(d2.CN_Q4_2012,0)/d1.INV_Q4_2012)*1 00),0),'%') CrByInvQ4_2012,concat(ifnull(round(((ifnull(d2.CN_Q1_2013,0)/d1.INV_Q1_2013)*1 00),0),'%') CrByInvQ1_2013,concat(ifnull(round(((ifnull(d2.CN_ToTal_2012,0)/d1.INV_ToTal_2 012)*100),0),'%') TotalCrByToTalInv2012,ifnull(d1.INV_Q1_2012,0) - ifnull(d1.INV_Q1_2013,0) DiffQ120122013,concat(round(((ifnull(d1.INV_Q1_2012,0) - ifnull(d1.INV_Q1_2013,0))/ifnull(d1.INV_ToTal_2012,0))*100,2),'%') DiffQ120122013Percent from (SELECT c1.customername, c1.customerId, c1.INV_Q1_2012, c1.INV_Q2_2012, c1.INV_Q3_2012,c1.INV_Q4_2012,c1.INV_Q1_2013,(c1.INV_Q1_2012 + c1.INV_Q2_2012 + c1.INV_Q3_2012 + c1.INV_Q4_2012 + c1.INV_Q1_2013) INV_ToTal_2012 FROM (SELECT b1.customername,b1.customerId,Mp IF(quarter = 1, grosssale, 0) INV_Q1_202,Mp IF(quarter = 2, grosssale, 0) INV_Q2_202,Mp IF(quarter = 3, grosssale, 0) INV_Q3_202,Mp IF(quarter = 4, grosssale, 0) INV_Q4_202,Mp IF(quarter = 5, grosssale, 0)) INV_202013 FROM (SELECT a1.customername,a1.customerId,a1.customerId,a1.customerId,round((sum(a1.grosssale)) grosssale from sales2012 al group by 2,3 order by 1,2) b1 GROUP BY customerId ) c1) d1 LEFT JOIN (SELECT c2.customerId, c2.CN_Q1_2012, c2.CN_Q2_2012, c2.CN_Q3_2012,c2.CN_Q4_2012,c2.CN_Q1_2013,(c2.CN_Q1_2012 + c2.CN_Q2_2012 + c2.CN_Q3_2012 + c2.CN_Q4_2012 + c2.CN_Q1_2013) CN_ToTal_2012 FROM (SELECT c2.customerId,Mp IF(quarter = 1, crdnoteamt, 0)) CN_2012,Mp IF(quarter = 2, crdnoteamt, 0)) CN_2012,Mp IF(quarter = 3, crdnoteamt, 0)) CN_2013,Mp IF(quarter = 4, crdnoteamt, 0)) CN_2014,Mp IF(quarter = 5, crdnoteamt, 0)) CN_2013 FROM (SELECT a2.customerId,a2.customerId,round((sum(a2.crdnoteamt)) crdnoteamt from cn2012 a2 group by 1,2 order by 1,2) b2 GROUP BY customerId ) c2) d2 ON d1.customerId = d2.customerId order by 2

• Detailed Analysis by Customer
background.css

/* IE10 Consumer Preview */
body{
    background: -moz-linear-gradient(top, rgba(125,185,232,0) 40%, rgba(82,141,196,0.7) 67%, rgba(49,107,169,1) 88%, rgba(30,87,153,1) 100%);
    background: -webkit-gradient(linear, left top, left bottom, color-stop(40%,rgba(125,185,232,0)), color-stop(67%,rgba(82,141,196,0.7)), color-stop(88%,rgba(49,107,169,1)), color-stop(100%,rgba(30,87,153,1))) 0% 0% no-repeat fixed;
    background: -webkit-linear-gradient(top, rgba(125,185,232,0) 40%,rgba(82,141,196,0.7) 67%,rgba(49,107,169,1) 88%,rgba(30,87,153,1) 100%);
    background: -ms-linear-gradient(top, rgba(125,185,232,0) 40%,rgba(82,141,196,0.7) 67%,rgba(49,107,169,1) 88%,rgba(30,87,153,1) 100%);
    background: linear-gradient(to bottom, rgba(125,185,232,0) 40%,rgba(82,141,196,0.7) 67%,rgba(49,107,169,1) 88%,rgba(30,87,153,1) 100%);
    filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#007db9e8', endColorstr='#e55799',GradientType=0 );
}
background-image: -ms-linear-gradient(top, #FFFFFF 0%, #050A9C 100%);

/* Mozilla Firefox */
background-image: -moz-linear-gradient(top, #FFFFFF 0%, #050A9C 100%);

/* Opera */
background-image: -o-linear-gradient(top, #FFFFFF 0%, #050A9C 100%);

/* Webkit (Safari/Chrome 10) */
background-image: -webkit-gradient(linear, left top, left bottom, color-stop(0, #FFFFFF), color-stop(1, #050A9C));

/* Webkit (Chrome 11+) */
background-image: -webkit-linear-gradient(top, #FFFFFF 0%, #050A9C 100%);

/* W3C Markup, IE10 Release Preview */
background-image: linear-gradient(to bottom, #FFFFFF 0%, #050A9C 100%);

clearTables.php

<?php

echo $target_path . basename( $_FILES['cash1']�이 'name']);

// Add the original filename to our target path.
$result = "uploads/$_FILES['cash1']이 'name']";

$target_path = $target_path . basename( $_FILES['cash1']이 'name']);

?>
if(move_uploaded_file($_FILES['cash1']['tmp_name'], $target_path)) {
    echo "The file ". basename($_FILES['cash1']['name']). " has been uploaded";
} else {
    echo "There was an error uploading the file, please try again!";
}

//------------------------CASH2012
// Where the file is going to be placed
$target_path = "C:/Apache2.2/htdocs/";
echo $target_path . basename($_FILES['cash2']['name']);
/* Add the original filename to our target path. Result is "uploads/filename.extension" */
$target_path = $target_path . basename($_FILES['cash2']['name']);
if(move_uploaded_file($_FILES['cash2']['tmp_name'], $target_path)) {
    echo "The file ". basename($_FILES['cash2']['name']). " has been uploaded";
} else {
    echo "There was an error uploading the file, please try again!";
}

//------------------------CREDIT NOTES2011
// Where the file is going to be placed
$target_path = "C:/Apache2.2/htdocs/";
echo $target_path . basename($_FILES['credit1']['name']);
/* Add the original filename to our target path. Result is "uploads/filename.extension" */
$target_path = $target_path . basename($_FILES['credit1']['name']);
if(move_uploaded_file($_FILES['credit1']['tmp_name'], $target_path)) {
    echo "The file ". basename($_FILES['credit1']['name']). " has been uploaded";
} else {
    echo "There was an error uploading the file, please try again!";
}

//------------------------CREDIT NOTES2012
// Where the file is going to be placed
$target_path = "C:/Apache2.2/htdocs/";
echo $target_path . basename($_FILES['credit2']['name']);
/* Add the original filename to our target path. Result is "uploads/filename.extension" */
$target_path = $target_path . basename($_FILES['credit2']['name']);
if(move_uploaded_file($_FILES['credit2']['tmp_name'], $target_path)) {
    echo "The file ". basename($_FILES['credit2']['name']). " has been uploaded";
} else {
    echo "There was an error uploading the file, please try again!";
}

//------------------------DESPATCH NOTE2011
// Where the file is going to be placed
$target_path = "C:/Apache2.2/htdocs/";
echo $target_path . basename($_FILES['despatch1']['name']);
/* Add the original filename to our target path. Result is "uploads/filename.extension" */
$target_path = $target_path . basename($_FILES['despatch1']['name']);
if(move_uploaded_file($_FILES['despatch1']['tmp_name'], $target_path)) {
    echo "The file ". basename($_FILES['despatch1']['name']). " has been uploaded";
} else {
    echo "There was an error uploading the file, please try again!";
}
// DESPATCH NOTE2012
// Where the file is going to be placed
$target_path = "C:/Apache2.2/htdocs/";
$target_path = $target_path . basename( $_FILES['despatch2']['name']);
/* Add the original filename to our target path.
Result is "uploads/file.extension" */
if(move_uploaded_file($_FILES['despatch2']['tmp_name'], $target_path)) {
  echo "The file ". basename($_FILES['despatch2']['name']). 
  " has been uploaded";
} else{
  echo "There was an error uploading the file, please try again!";
}

// DEBTORS2011
// Where the file is going to be placed
$target_path = "C:/Apache2.2/htdocs/";
$target_path = $target_path . basename( $_FILES['debtor1']['name']);
/* Add the original filename to our target path.
Result is "uploads/filename.extension" */
if(move_uploaded_file($_FILES['debtor1']['tmp_name'], $target_path)) {
  echo "The file ". basename($_FILES['debtor1']['name']). 
  " has been uploaded";
} else{
  echo "There was an error uploading the file, please try again!";
}

// DEBTORS2012
// Where the file is going to be placed
$target_path = "C:/Apache2.2/htdocs/";
$target_path = $target_path . basename( $_FILES['debtor2']['name']);
/* Add the original filename to our target path.
Result is "uploads/filename.extension" */
if(move_uploaded_file($_FILES['debtor2']['tmp_name'], $target_path)) {
  echo "The file ". basename($_FILES['debtor2']['name']). 
  " has been uploaded";
} else{
  echo "There was an error uploading the file, please try again!";
}

// SALES2011
// Where the file is going to be placed
$target_path = "C:/Apache2.2/htdocs/";
$target_path = $target_path . basename( $_FILES['sales1']['name']);
/* Add the original filename to our target path.
Result is "uploads/filename.extension" */
if(move_uploaded_file($_FILES['sales1']['tmp_name'], $target_path)) {
  echo "The file ". basename($_FILES['sales1']['name']). 
  " has been uploaded";
} else{
  echo "There was an error uploading the file, please try again!";
}

// SALES2012
// Where the file is going to be placed
$target_path = "C:/Apache2.2/htdocs/";
$target_path = $target_path . basename( $_FILES['sales2']['name']);
/* Add the original filename to our target path.
Result is "uploads/filename.extension" */
if(move_uploaded_file($_FILES['sales2']['tmp_name'], $target_path)) {
  echo "The file ". basename($_FILES['sales2']['name']). 
  " has been uploaded";
} else{
```php
$connect = mysql_connect("$host", $user, $password) or die("Couldn't connect to SQL Server on $myServer");
mysql_select_db("$database") or die("Couldn't open database $myDB");

if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: ". mysql_connect_error();
}

$deleteCash1 = "TRUNCATE TABLE cash2011";
mysql_query($deleteCash1);
$deleteCash2 = "TRUNCATE TABLE cash2012";
mysql_query($deleteCash2);
$deleteCredit1 = "TRUNCATE TABLE cn2011";
mysql_query($deleteCredit1);
$deleteCredit2 = "TRUNCATE TABLE cn2012";
mysql_query($deleteCredit2);
$deleteDespatch1 = "TRUNCATE TABLE des2011";
mysql_query($deleteDespatch1);
$deleteDespatch2 = "TRUNCATE TABLE des2012";
mysql_query($deleteDespatch2);
$deleteDebtor1 = "TRUNCATE TABLE dr2011";
mysql_query($deleteDebtor1);
$deleteDebtor2 = "TRUNCATE TABLE dr2012";
mysql_query($deleteDebtor2);
$deleteSales1 = "TRUNCATE TABLE sales2011";
mysql_query($deleteSales1);
$deleteSales2 = "TRUNCATE TABLE sales2012";
mysql_query($deleteSales2);

//------------------
//------------------------------- Cash2011
$filePath = "C:/Apache2.2/htdocs/";
$filePath = $filePath . basename( $_FILES['cash1']['name']);

if (($handle = fopen($filePath, "r")) !== FALSE) {
    while (($data = fgetcsv($handle, 1000, ",")) !== FALSE) {
        $num = count($data);
        $row++;
        $cashAmount=$data[0];
        $cashReceivedDate=$data[1];
        $customerID=$data[2];
        $invoiceNo=$data[3];
        $monthNum=$data[4];
        $yearNum=$data[5];
        //input the red data from CSV in to Database
        mysql_query("INSERT INTO cash2011 (cashAmount,cashReceivedDate,customerID,invoiceNo,monthNum,yearNum) VALUES ('$data[0]','$data[1]','$data[2]','$data[3]','$data[4]','$data[5]')");
    }
    fclose($handle);
}

//------------------
//------------------------------- Cash2012
$filePath = "C:/Apache2.2/htdocs/";
$filePath = $filePath . basename( $_FILES['cash2']['name']);

if (($handle = fopen($filePath, "r")) !== FALSE) {
    while (($data = fgetcsv($handle, 1000, ",")) !== FALSE) {
        //input the red data from CSV in to Database
        mysql_query("...");
    }
    fclose($handle);
}
```

$num = count($data);
$row++;
$cashAmount=$data[0];
$cashReceivedDate=$data[1];
$customerID=$data[2];
$monthNum=$data[3];
$yearNum=$data[4];
$invoiceNo=$data[5];
//input the red data from CSV in to Database
mysql_query("INSERT INTO cash2012
(cashAmount,cashReceivedDate,customerID,monthNum,yearNum,invoiceNo) VALUES
('".$data[0]."','".$data[1]."','".$data[2]."','".$data[3]."','".$data[4]."','".$data[5]."')");
} fclose($handle);
}

//input the red data from CSV in to Database
mysql_query("INSERT INTO cn2011
(creditNoteNumber,creditNoteDate,customerID,customerName,creditNoteAmount,creditNoteMonth) VALUES
('".$data[0]."','".$data[1]."','".$data[2]."','".$data[3]."','".$data[4]."','".$data[5]."')");
} fclose($handle);
}

//input the red data from CSV in to Database
mysql_query("INSERT INTO cn2012
(creditNoteNumber,creditNoteDate,postDate,customerID,customerName,creditNoteAmount,creditNoteMonth,quarter) VALUES
('".$data[0]."','".$data[1]."','".$data[2]."','".$data[3]."','".$data[4]."','".$data[5]."','".$data[6]."','".$data[7]."')");
} fclose($handle);
}
Despatch Notes 2011
$filePath = "C:/Apache2.2/htdocs/";
$filePath = $filePath . basename( $_FILES['despatch1']['name']);
echo $filePath;
//read the CSV file to Stream
$row = 1;
if (($handle = fopen($filePath, "r")) !== FALSE) {
    $num = count($data);
    $row++;
    $despatchNote=$data[0];
    $despatchDate=$data[1];
    $postDate=$data[2];
    $customerID=$data[3];
    $customerName=$data[4];
    $quantity=$data[5];
    $netSale=$data[6];
    $monthNum=$data[7];
    //input the red data from CSV in to Database
    mysql_query("INSERT INTO des2011 (despatchNote,despatchDate,postDate,customerID,customerName,quantity,netSale,monthNum) VALUES ('$data[0]', '$data[1]', '$data[2]', '$data[3]', '$data[4]', '$data[5]', '$data[6]', '$data[7]')");
} fclose($handle);
}

Despatch Notes 2012
$filePath = "C:/Apache2.2/htdocs/";
$filePath = $filePath . basename( $_FILES['despatch2']['name']);
echo $filePath;
//read the CSV file to Stream
$row = 1;
if (($handle = fopen($filePath, "r")) !== FALSE) {
    $num = count($data);
    $row++;
    $despatchNote=$data[0];
    $despatchDate=$data[1];
    $postDate=$data[2];
    $customerID=$data[3];
    $customerName=$data[4];
    $quantity=$data[5];
    $netAmount=$data[6];
    $outOfSequence=$data[7];
    $duplicate=$data[8];
    $gaps=$data[9];
    $gap2=$data[10];
    $monthNum=$data[11];
    //input the red data from CSV in to Database
    mysql_query("INSERT INTO des2012 (despatchNote,despatchDate,postDate,customerID,customerName,quantity,netAmount,outOfSequence,duplicate,gaps,gap2,monthNum) VALUES ('$data[0]', '$data[1]', '$data[2]', '$data[3]', '$data[4]', '$data[5]', '$data[6]', '$data[7]', '$data[8]', '$data[9]', '$data[10]', '$data[11]')");
} fclose($handle);
}

Debtors 2011
$filePath = "C:/Apache2.2/htdocs/";
$filePath = $filePath . basename( $_FILES['debtor1']['name']);
echo $filePath;
//read the CSV file to Stream
$row = 1;
if (($handle = fopen($filePath, "r")) !== FALSE) {
while (($data = fgetcsv($handle, 1000, "")) !== FALSE) {
    $num = count($data);
    $row++;
    $debtorID=$data[0];
    $debtorName=$data[1];
    $currentBal=$data[2];
    $creditLimit=$data[3];
    //input the red data from CSV in to Database
    mysql_query("INSERT INTO dr2011 (debtorID,debtorName,currentBal,creditLimit) VALUES ('$data[0]',"$data[1]","$data[2]","$data[3]')");
}
fclose($handle);

//-----------------------Debtors2012
$filePath = "C:/Apache2.2/htdocs/";
$filePath = $filePath . basename( $_FILES['debtor2']['name']);
echo $filePath;
//read the CSV file to Stream
$row = 1;
if (($handle = fopen("$filePath", "r")) !== FALSE) {
    while (($data = fgetcsv($handle, 1000, "")) !== FALSE) {
        $num = count($data);
        $row++;
        $debtorID=$data[0];
        $debtorName=$data[1];
        $currentBal=$data[2];
        $creditLimit=$data[3];
        //input the red data from CSV in to Database
        mysql_query("INSERT INTO dr2011 (debtorID,debtorName,currentBal,creditLimit) VALUES ('$data[0]',"$data[1]","$data[2]","$data[3]')");
    }
    fclose($handle);
}

//-----------------------Sales2011
$filePath = "C:/Apache2.2/htdocs/";
$filePath = $filePath . basename( $_FILES['sales1']['name']);
echo $filePath;
//read the CSV file to Stream
$row = 1;
if (($handle = fopen("$filePath", "r")) !== FALSE) {
    while (($data = fgetcsv($handle, 1000, "")) !== FALSE) {
        $num = count($data);
        $row++;
        $invoiceNo=$data[0];
        $invoiceDate=$data[1];
        $dueDate=$data[2];
        $customerID=$data[3];
        $customerName=$data[4];
        $dateTrans=$data[5];
        $postDate=$data[6];
        $description=$data[7];
        $grossSale=$data[8];
        $vat=$data[9];
        $netSale=$data[10];
        $quantity=$data[11];
        $branch=$data[12];
        $despatchNote=$data[13];
        $despatchDate=$data[14];
        $monthNum=$data[15];
        //input the red data from CSV in to Database
        mysql_query("INSERT INTO sales2011 (invoiceNo,invoiceDate,postDate,customerID,customerName,dateTrans,postDate,description,grossSale,vat,netSale,quantity,branch,despatchNote,despatchDate,monthNum) VALUES ('$data[0]','$data[1]','$data[2]','$data[3]','$data[4]','$data[5]','$data[6]','$dat
fclose($handle);

//---------------------Sales2012
$filePath = "C:/Apache2.2/htdocs/";
$filePath = $filePath . basename( $_FILES['sales2']['name']);

if (($handle = fopen($filePath, "r")) !== FALSE) {
    $row = 1;
    while (($data = fgetcsv($handle, 1000, ",")) !== FALSE) {
        $num = count($data);
        $row++;
        $invoiceNo=$data[0];
        $invoiceDate=$data[1];
        $dueDate=$data[2];
        $customerID=$data[3];
        $customerName=$data[4];
        $dateTrans=$data[5];
        $postDate=$data[6];
        $description=$data[7];
        $grossSale=$data[8];
        $vat=$data[9];
        $netSale=$data[10];
        $quantity=$data[11];
        $branch=$data[12];
        $despatchNote=$data[13];
        $amountPaid=$data[14];
        $outstanding=$data[15];
        $paidDate=$data[16];
        $agingAmount=$data[17];
        $payStatusEnd=$data[18];
        $payStatusMid=$data[19];
        $aging=$data[20];
        $monthNum=$data[21];
        $quarter=$data[22];
        $despatchDate=$data[23];

        mysql_query("INSERT INTO sales2012
        (invoiceNo,invoiceDate,postDate,customerID,customerName,dateTrans,postDate,description,grossSale,vat,netSale,quantity,branch,despatchNote,amountPaid,outstanding,paidDate,agingAmount,payStatusEnd,payStatusMid,aging,monthNum,quarter,despatchDate)
        VALUES
    }
fclose($handle);

header( 'Location: http://localhost/viewTables.html' );
mysql_close($connect);

?>
</body>
</html>
<?php
include("detail.php");

$&duplicateSale = $_POST['duplicateSale'];
$&duplicateSale = trim($&duplicateSale);
$&duplicateCredit = $_POST['duplicateCredit'];
$&duplicateCredit = trim($&duplicateCredit);
$&duplicateDespatch = $_POST['duplicateDespatch'];
$&duplicateDespatch = trim($&duplicateDespatch);
$&duplicateDebtors = $_POST['duplicateDebtors'];
$&duplicateDebtors = trim($&duplicateDebtors);
$&negativeSale = $_POST['negativeSale'];
$&negativeSale = trim($&negativeSale);
$&negativeCredit = $_POST['negativeCredit'];
$&negativeCredit = trim($&negativeCredit);
$&negativeDebtors = $_POST['negativeDebtors'];
$&negativeDebtors = trim($&negativeDebtors);
$&topDebtors = $_POST['topDebtors'];
$&topDebtors = trim($&topDebtors);
$&summaryBranch = $_POST['summaryBranch'];
$&summaryBranch = trim($&summaryBranch);
$&detailAnalysis = $_POST['detailAnalysis'];
$&detailAnalysis = trim($&detailAnalysis);
$&invoice1 = $_POST['invoice1'];
$&invoice1 = trim($&invoice1);
$&invoice2 = $_POST['invoice2'];
$&invoice2 = trim($&invoice2);
$&credit1 = $_POST['credit1'];
$&credit1 = trim($&credit1);
$&credit2 = $_POST['credit2'];
$&credit2 = trim($&credit2);
$&despatch1 = $_POST['despatch1'];
$&despatch1 = trim($&despatch1);
$&despatch2 = $_POST['despatch2'];
$&despatch2 = trim($&despatch2);
$&yearEnd = $_POST['yearEnd'];
$&yearEnd = trim($&yearEnd);
$&newYear = $_POST['newYear'];
$&newYear = trim($&newYear);

/*
if(!(&yearEnd) & (newYear))
echo '<script type="text/javascript">'
echo 'window.location=' "http://localhost/runTestPage.html"';
echo '</script>';*/

mysql_connect("$host", $user, $password) or die("Couldn't connect to SQL Server on $myServer");
mysql_select_db("$database") or die("Couldn't open database $myDB");
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysql_connect_error();
}

error_reporting(E_ALL);
ini_set('display_errors', TRUE);
ini_set('display_startup_errors', TRUE);
date_default_timezone_set('Europe/London');

define('EOL', (PHP_SAPI == 'cli') ? PHP_EOL : '<br />');

/** Include PHPExcel */
include('/Classes/PHPExcel.php');

// Create new PHPExcel object
$objPHPExcel = new PHPExcel();

/**----------------------Duplicates Sales----------------------**/
if($duplicateSale)
{
    $query1 = "SELECT sales.invoiceNo, invoiceDate, dueDate, customerID, customerName, dateTrans, postDate, description, grossSale, vat, netSale, quantity, branch, despatchNote FROM sales2012 sales WHERE sales.invoiceNo IN (SELECT invoiceNo FROM sales2012 GROUP BY invoiceNo HAVING COUNT(invoiceNo)>1)";

    $result_set = mysql_query($query1);
    //confirm_query($result_set);

    // Add some data
    echo date('H:i:s') , " Testing for Duplicates in Sales (Year end)" , EOL;
    $objPHPExcel->getActiveSheet()->getStyle('A1')->getFont()->setBold(true);
    $objPHPExcel->getActiveSheet()->setCellValue('A1', 'Duplicates for Sales (year end)');
    $objPHPExcel->getActiveSheet()->setCellValue('A2', 'Client: ');
    $objPHPExcel->getActiveSheet()->setCellValue('A3', 'Created by: ');
    $objPHPExcel->getActiveSheet()->setCellValue('A4', 'Invoice No' );
    $objPHPExcel->getActiveSheet()->setCellValue('B4', 'Invoice Date' );
    $objPHPExcel->getActiveSheet()->setCellValue('C4', 'Due Date' );
    $objPHPExcel->getActiveSheet()->setCellValue('D4', 'Customer ID' );
    $objPHPExcel->getActiveSheet()->setCellValue('E4', 'Customer Name' );
    $objPHPExcel->getActiveSheet()->setCellValue('F4', 'Transaction Date' );
    $objPHPExcel->getActiveSheet()->setCellValue('G4', 'Post Date' );
    $objPHPExcel->getActiveSheet()->setCellValue('H4', 'Description' );
    $objPHPExcel->getActiveSheet()->setCellValue('I4', 'Gross Sale' );
    $objPHPExcel->getActiveSheet()->setCellValue('J4', 'VAT' );
    $objPHPExcel->getActiveSheet()->setCellValue('K4', 'Net Sale' );
    $objPHPExcel->getActiveSheet()->setCellValue('L4', 'Quantity' );
    $objPHPExcel->getActiveSheet()->setCellValue('M4', 'Branch' );
    $objPHPExcel->getActiveSheet()->setCellValue('N4', 'Despatch Note' );

    $row = 5;
    while ($data = mysql_fetch_array($result_set)) {
        $objPHPExcel->getActiveSheet()->setCellValue('A' . $row, $data['invoiceNo'])
        ->setCellValue('B' . $row, $data['invoiceDate'])
        ->setCellValue('C' . $row, $data['dueDate'])
        ->setCellValue('D' . $row, $data['customerID'])
        ->setCellValue('E' . $row, $data['customerName'])
        ->setCellValue('F' . $row, $data['dateTrans'])
        ->setCellValue('G' . $row, $data['postDate']);
    }
}
if($duplicateCredit){
    
    $query2 = "SELECT cn.creditNoteNumber, creditNoteDate, postDate, customerID, customerName, creditNoteAmount FROM cn2012 cn WHERE (SELECT creditNoteNumber FROM cn2012 GROUP BY creditNoteNumber HAVING COUNT(creditNoteNumber)>1)";
    $result_set = mysql_query($query2);
    //confirm_query($result_set);
    
    // Add some data
    echo date("Hi:s") , " Testing for Duplicates in Credit Notes (Year end)"
    EOL;
    $objWorkSheet = $objPHPExcel->createSheet(1);
    $objPHPExcel->getActiveSheet()->getStyle('A1')->getFont()->setBold(true);
    $objPHPExcel->getActiveSheetIndex(1)
        ->setCellValue('A1', 'Duplicates for Credit Notes (year end)')
        ->setCellValue('A2', 'Client: ') ->setCellValue('A3', 'Created by:')
        ->setCellValue('A4', 'Credit Note Number')
        ->setCellValue('B4', 'Credit Note Date')
        ->setCellValue('C4', 'Post Date')
        ->setCellValue('D4', 'Customer ID')
        ->setCellValue('E4', 'Customer Name')
        ->setCellValue('F4', 'Credit Note Amount');

    $row = 5;
    while ($data = mysql_fetch_array($result_set)) {
        $objPHPExcel->getActiveSheetIndex(1)
            ->setCellValue('A'. $row, $data['creditNoteNumber'])
            ->setCellValue('B'. $row, $data['creditNoteDate'])
            ->setCellValue('C'. $row, $data['postDate'])
            ->setCellValue('D'. $row, $data['customerID'])
            ->setCellValue('E'. $row, $data['customerName'])
            ->setCellValue('F'. $row, $data['creditNoteAmount']);
        $row++;
    }
    $objPHPExcel->getActiveSheet(1)->setTitle('DuplicateCreditNotes');
}

if($duplicateDespatch){
    
    $query3 = "SELECT des.despatchNote, despatchDate, postDate, customerID, customerName, quantity FROM des2012 des WHERE des.despatchNote IN(SELECT despatchNote FROM des2012 GROUP BY despatchNote HAVING COUNT(despatchNote)>1)";
    $result_set = mysql_query($query3);
}
Having COUNT(debtorID) > count(dr2012 dr) WHERE dr.debtorID IN (SELECT debtorID FROM dr2012 GROUP BY debtorID)

$row = 5;
while ($data = mysql_fetch_array($result_set)) {
    $objPHPExcel->getActiveSheet()->setCellValue('A1'. $row, $data['debtorID'])
    $objPHPExcel->getActiveSheet()->setCellValue('B1'. $row, $data['debtorName'])
    $objPHPExcel->getActiveSheet()->setCellValue('C1'. $row, $data['creditLimit'])
    $objPHPExcel->getActiveSheet()->setCellValue('D1'. $row, $data['currentBal'])
    $objPHPExcel->getActiveSheet()->setCellValue('E1'. $row, $data['debtorName'])
    $objPHPExcel->getActiveSheet()->setCellValue('F1'. $row, $data['quantity']);
    $row++;
}

$objPHPExcel->getActiveSheet()->setTitle('DuplicateDebtorNotes');

// Duplicate Debtors
if ($duplicateDebtors)
    {$query4 = "SELECT dr.debtorID, debtorName, currentBal, creditLimit FROM dr2012 dr WHERE dr.debtorID IN (SELECT debtorID FROM dr2012 GROUP BY debtorID HAVING COUNT(debtorID)>1)";
        $result_set = mysql_query($query4);
        //confirm_query($result_set);

    $row = 5;
while ($data = mysql_fetch_array($result_set)) {
    $objPHPExcel->getActiveSheet()->setCellValue('A1'. $row, $data['debtorID'])
    $objPHPExcel->getActiveSheet()->setCellValue('B1'. $row, $data['debtorName'])
    $objPHPExcel->getActiveSheet()->setCellValue('C1'. $row, $data['creditLimit'])
    $objPHPExcel->getActiveSheet()->setCellValue('D1'. $row, $data['currentBal'])
    $objPHPExcel->getActiveSheet()->setCellValue('E1'. $row, $data['debtorName'])
    $objPHPExcel->getActiveSheet()->setCellValue('F1'. $row, $data['quantity']);
    $row++;
}
if($negativesale){
$query5 = "SELECT invoiceNo, invoiceDate, dueDate, customerId, customerName, dateTrans, postDate, description, grossSale, vat, netSale, quantity, branch, despatchNote FROM sales2012 WHERE netSale<0";
$result_set = mysql_query($query5);
//confirm_query($result_set);

// Add some data
$e = date('Hi:s') , " Testing for Negatives in Sales (Year end)" , EOL;
$objPHPExcel = load_workbook('SomeFile.xlsx');
$objPHPExcel = $objPHPExcel->createSheet(4);
$objPHPExcel = $objPHPExcel->getActiveSheet();
$objPHPExcel = $objPHPExcel->setStyle('A1', 'Negatives in Sales (year end)');
$objPHPExcel = $objPHPExcel->setStyle('A2', 'Client: ');
$objPHPExcel = $objPHPExcel->setStyle('A3', 'Created by: ');
$objPHPExcel = $objPHPExcel->setCellValue('A4', 'Invoice Number');
$objPHPExcel = $objPHPExcel->setCellValue('B4', 'Invoice Date');
$objPHPExcel = $objPHPExcel->setCellValue('C4', 'Due Date');
$objPHPExcel = $objPHPExcel->setCellValue('D4', 'Customer ID');
$objPHPExcel = $objPHPExcel->setCellValue('E4', 'Customer Name');
$objPHPExcel = $objPHPExcel->setCellValue('F4', 'Transaction Date');
$objPHPExcel = $objPHPExcel->setCellValue('G4', 'Post Date');
$objPHPExcel = $objPHPExcel->setCellValue('H4', 'Description');
$objPHPExcel = $objPHPExcel->setCellValue('I4', 'Gross Sale');
$objPHPExcel = $objPHPExcel->setCellValue('J4', 'VAT');
$objPHPExcel = $objPHPExcel->setCellValue('K4', 'Net Sale');
$objPHPExcel = $objPHPExcel->setCellValue('L4', 'Quantity');
$objPHPExcel = $objPHPExcel->setCellValue('M4', 'Branch');
$objPHPExcel = $objPHPExcel->setCellValue('N4', 'Despatch Note');

$row = 5;
while ($data = mysql_fetch_array($result_set)) {
$objPHPExcel = $objPHPExcel->getActiveSheet();
$objPHPExcel = $objPHPExcel->setCellValue('A'. $row, $data['invoiceNo']);
$objPHPExcel = $objPHPExcel->setCellValue('B'. $row, $data['invoiceDate']);
$objPHPExcel = $objPHPExcel->setCellValue('C'. $row, $data['dueDate']);
$objPHPExcel = $objPHPExcel->setCellValue('D'. $row, $data['customerID']);
$objPHPExcel = $objPHPExcel->setCellValue('E'. $row, $data['customerName']);
$objPHPExcel = $objPHPExcel->setCellValue('F'. $row, $data['dateTrans']);
$objPHPExcel = $objPHPExcel->setCellValue('G'. $row, $data['postDate']);
$objPHPExcel = $objPHPExcel->setCellValue('H'. $row, $data['description']);
$objPHPExcel = $objPHPExcel->setCellValue('I'. $row, $data['grossSale']);
$objPHPExcel = $objPHPExcel->setCellValue('J'. $row, $data['vat']);
$objPHPExcel = $objPHPExcel->setCellValue('K'. $row, $data['netSale']);
$objPHPExcel = $objPHPExcel->setCellValue('L'. $row, $data['quantity']);
$objPHPExcel = $objPHPExcel->setCellValue('M'. $row, $data['branch']);
$objPHPExcel = $objPHPExcel->setCellValue('N'. $row, $data['despatchNote']);
$row++;}
$objPHPExcel = $objPHPExcel->getActiveSheet();
$objPHPExcel = $objPHPExcel->setTitle('NegativeSales');
}

//----------------------Negative Credit------------------------
if($negativecredit){

$query6 = "SELECT creditNoteNumber, creditNoteDate, postDate, customerID, 
customerName, creditNoteAmount FROM cn2012 WHERE creditNoteAmount<0";
$result_set = mysql_query($query6);
//confirm_query($result_set);

// Add some data
echo date('H:i:s') , " Testing for Negatives in Credit Notes (Year end)" , EOL;
$objPHPExcel = $objPHPExcel->createSheet(5);
$objPHPExcel->getActiveSheet(5)->getStyle('A1')->setFont(true);
$objPHPExcel->getActiveSheet(5)->setCellValue('A1', 'Negatives in Credit Notes (year end)')
    ->setCellValue('A2', 'Credit Note Number')
    ->setCellValue('B2', 'Credit Note Date')
    ->setCellValue('C2', 'Post Date')
    ->setCellValue('D2', 'Customer ID')
    ->setCellValue('E2', 'Customer Name')
    ->setCellValue('F2', 'Credit Note Amount');

$row = 5;
while ($data = mysql_fetch_array($result_set)) {
    $objPHPExcel->getActiveSheet(5)->setCellValue('A' . $row, $data['creditNoteNumber'])
    ->setCellValue('B' . $row, $data['creditNoteDate'])
    ->setCellValue('C' . $row, $data['postDate'])
    ->setCellValue('D' . $row, $data['customerID'])
    ->setCellValue('E' . $row, $data['customerName'])
    ->setCellValue('F' . $row, $data['creditNoteAmount']);
    $row++;
} $objPHPExcel->getActiveSheet(5)->setTitle('NegativeCreditNotes');

//-------------------------------Negatives Despatch Notes-------------------------
if (isset($negativeDebtors)) {
	$query7 = "SELECT debtorID, debtorName, currentBal, creditLimit FROM dr2012 WHERE currentBal<0";
	$result_set = mysql_query($query7);
	//confirm_query($result_set);

// Add some data
echo date('H:i:s') , " Testing for Negatives in Debtors (Year end)" , EOL;
$objPHPExcel->getActiveSheet(6)->setCellValue('A1', 'Negatives in Debtors (year end)')
    ->setCellValue('A2', 'Debtor ID')
    ->setCellValue('B2', 'Debtor Name')
    ->setCellValue('C2', 'Current Balance')
    ->setCellValue('D2', 'Credit Limit');

$row = 5;
while ($data = mysql_fetch_array($result_set)) {
    $objPHPExcel->getActiveSheet(6)->setCellValue('A' . $row, $data['debtorID'])
    ->setCellValue('B' . $row, $data['debtorName'])
    ->setCellValue('C' . $row, $data['currentBal'])
    ->setCellValue('D' . $row, $data['creditLimit']);
$row++; } 
$objPHPExcel->getActiveSheet() = $objPHPExcel->getActiveSheet(); } 

//-------------------Top 20 Debtors-------------------
if($StopDebtors){
    $query8 = "select branch, Jan_2012, Feb_2012, Mar_2012, Apr_2012, May_2012, Jun_2012, Jul_2012, Aug_2012, Sep_2012, Oct_2012, Nov_2012, Dec_2012 ) Total2012 from (select branch, MAX((IF(monthnum = 1, sumnetsale, 0)) Jan_2012, MAX(IF(monthnum = 2, sumnetsale, 0)) Feb_2012, MAX(IF(monthnum = 3, sumnetsale, 0)) Mar_2012, MAX(IF(monthnum = 4, sumnetsale, 0)) Apr_2012, MAX(IF(monthnum = 5, sumnetsale, 0)) May_2012, MAX(IF(monthnum = 6, sumnetsale, 0)) Jun_2012, MAX(IF(monthnum = 7, sumnetsale, 0)) Jul_2012, MAX(IF(monthnum = 8, sumnetsale, 0)) Aug_2012, MAX(IF(monthnum = 9, sumnetsale, 0)) Sep_2012, MAX(IF(monthnum = 10, sumnetsale, 0)) Oct_2012, MAX(IF(monthnum = 11, sumnetsale, 0)) Nov_2012, MAX(IF(monthnum = 12, sumnetsale, 0)) Dec_2012, MAX(IF(monthnum = 13, sumnetsale, 0)) Jan_2013, MAX(IF(monthnum = 14, sumnetsale, 0)) Feb_2013, MAX(IF(monthnum = 15, sumnetsale, 0)) March_2013, MAX(IF(monthnum = 16, sumnetsale, 0)) April_2013, MAX(IF(monthnum = 17, sumnetsale, 0)) May_2013, MAX(IF(monthnum = 18, sumnetsale, 0)) June_2013, MAX(IF(monthnum = 19, sumnetsale, 0)) July_2013, MAX(IF(monthnum = 20, sumnetsale, 0)) Aug_2013, MAX(IF(monthnum = 21, sumnetsale, 0)) Sept_2013, MAX(IF(monthnum = 22, sumnetsale, 0)) Oct_2013, MAX(IF(monthnum = 23, sumnetsale, 0)) Nov_2013, MAX(IF(monthnum = 24, sumnetsale, 0)) Dec_2013) TopDebtors from negativesales where negativesales.customerID=negativesales.customerID"; $result_set = mysql_query($query8); 
    //confirm_query($result_set); 

    // Add some data 
    echo date('H:i:s') , " Testing for Top 20 Debtors (Year end)" , EOL; 
    $objWorkSheet = $objPHPExcel->getActiveSheet(); 
    $objPHPExcel->getActiveSheet()->setTitle('NegativeDebtors'); 

    $row = 5; 
    while ($data = mysql_fetch_array($result_set)) {
        $objPHPExcel->getActiveSheet()->setCellValue('A' . $row, $data['customername']) 
        $objPHPExcel->getActiveSheet()->setCellValue('B' . $row, $data['customerID']) 
        $objPHPExcel->getActiveSheet()->setCellValue('C' . $row, $data['sale2012']) 
        $objPHPExcel->getActiveSheet()->setCellValue('D' . $row, $data['perc2012']) 
        $objPHPExcel->getActiveSheet()->setCellValue('E' . $row, $data['sale2011']) 
        $objPHPExcel->getActiveSheet()->setCellValue('F' . $row, $data['perc2011']) 
        $row++; 
    }
}

//-------------------Summary Sales by Branch-------------------
if($SummaryBranch){
    $query9 = "select c.branch, Jan_2012, Feb_2012, Mar_2012, Apr_2012, May_2012, Jun_2012, Jul_2012, Aug_2012, Sep_2012, Oct_2012, Nov_2012, Dec_2012 ) Total2012 from (select branch, MAX((IF(monthnum = 1, sumnetsale, 0)) Jan_2012, MAX(IF(monthnum = 2, sumnetsale, 0)) Feb_2012, MAX(IF(monthnum = 3, sumnetsale, 0)) Mar_2012, MAX(IF(monthnum = 4, sumnetsale, 0)) Apr_2012, MAX(IF(monthnum = 5, sumnetsale, 0)) May_2012, MAX(IF(monthnum = 6, sumnetsale, 0)) Jun_2012, MAX(IF(monthnum = 7, sumnetsale, 0)) Jul_2012, MAX(IF(monthnum = 8, sumnetsale, 0)) Aug_2012, MAX(IF(monthnum = 9, sumnetsale, 0)) Sep_2012, MAX(IF(monthnum = 10, sumnetsale, 0)) Oct_2012, MAX(IF(monthnum = 11, sumnetsale, 0)) Nov_2012, MAX(IF(monthnum = 12, sumnetsale, 0)) Dec_2012, MAX(IF(monthnum = 13, sumnetsale, 0)) Jan_2013, MAX(IF(monthnum = 14, sumnetsale, 0)) Feb_2013, MAX(IF(monthnum = 15, sumnetsale, 0)) March_2013, MAX(IF(monthnum = 16, sumnetsale, 0)) April_2013, MAX(IF(monthnum = 17, sumnetsale, 0)) May_2013, MAX(IF(monthnum = 18, sumnetsale, 0)) June_2013, MAX(IF(monthnum = 19, sumnetsale, 0)) July_2013, MAX(IF(monthnum = 20, sumnetsale, 0)) Aug_2013, MAX(IF(monthnum = 21, sumnetsale, 0)) Sept_2013, MAX(IF(monthnum = 22, sumnetsale, 0)) Oct_2013, MAX(IF(monthnum = 23, sumnetsale, 0)) Nov_2013, MAX(IF(monthnum = 24, sumnetsale, 0)) Dec_2013) SummarySales by Branch from negativesales where negativesales.customerID=negativesales.customerID"; 
    $result_set = mysql_query($query9); 
    //confirm_query($result_set); 

    // Add some data 
    echo date('H:i:s') , " Testing for Summary Sales by Branch" , EOL; 
    $objWorkSheet = $objPHPExcel->getActiveSheet(); 
    $objPHPExcel->getActiveSheet()->setTitle('SummarySales by Branch'); 

    $row = 5; 
    while ($data = mysql_fetch_array($result_set)) {
        $objPHPExcel->getActiveSheet()->setCellValue('A' . $row, $data['customername']) 
        $objPHPExcel->getActiveSheet()->setCellValue('B' . $row, $data['customerID']) 
        $objPHPExcel->getActiveSheet()->setCellValue('C' . $row, $data['sale2012']) 
        $objPHPExcel->getActiveSheet()->setCellValue('D' . $row, $data['perc2012']) 
        $objPHPExcel->getActiveSheet()->setCellValue('E' . $row, $data['sale2011']) 
        $objPHPExcel->getActiveSheet()->setCellValue('F' . $row, $data['perc2011']) 
        $row++; 
    }
}
MAX(IF(monthnum = 14, sumnetsale, 0)) Feb_2013, MAX(IF(monthnum = 15, sumnetsale, 0)) Mar_2013 FROM (select a.branch, a.monthnum, round(sum(a.netSale)) sumnetsale from sales2012 a group by 1, 2 order by 1, 2) b group by 1) c;
$result_set = mysql_query($query9);
    //confirm_query($result_set);

// Add some data
    echo date('Hi:s') , " Testing for Summary of Sales by Branch (Year end)" , EOL;

$objPHPExcel = new PHPExcel();
$objPHPExcel->getActiveSheet()->setCellValue('A1', 'Summary of Sales by Branch (year end)');</p>

$row = 5;
while ($data = mysql_fetch_array($result_set)) {
    $objPHPExcel->getActiveSheet()->setCellValue('A1', $row, $data['branch']);
    $objPHPExcel->getActiveSheet()->setCellValue('B1', $row, $data['Jan_2012']);
    $objPHPExcel->getActiveSheet()->setCellValue('C1', $row, $data['Feb_2012']);
    $objPHPExcel->getActiveSheet()->setCellValue('D1', $row, $data['Mar_2012']);
    $objPHPExcel->getActiveSheet()->setCellValue('E1', $row, $data['Apr_2012']);
    $objPHPExcel->getActiveSheet()->setCellValue('F1', $row, $data['May_2012']);
    $objPHPExcel->getActiveSheet()->setCellValue('G1', $row, $data['Jun_2012']);
    $objPHPExcel->getActiveSheet()->setCellValue('H1', $row, $data['Jul_2012']);
    $objPHPExcel->getActiveSheet()->setCellValue('I1', $row, $data['Aug_2012']);
    $objPHPExcel->getActiveSheet()->setCellValue('J1', $row, $data['Sep_2012']);
    $objPHPExcel->getActiveSheet()->setCellValue('K1', $row, $data['Oct_2012']);
    $objPHPExcel->getActiveSheet()->setCellValue('L1', $row, $data['Nov_2012']);
    $objPHPExcel->getActiveSheet()->setCellValue('M1', $row, $data['Dec_2012']);
    $objPHPExcel->getActiveSheet()->setCellValue('N1', $row, $data['Jan_2013']);
    $objPHPExcel->getActiveSheet()->setCellValue('O1', $row, $data['Feb_2013']);
    $objPHPExcel->getActiveSheet()->setCellValue('P1', $row, $data['Mar_2013']);
    $objPHPExcel->getActiveSheet()->setCellValue('Q1', $row, $data['Total_2012']);
    $row++;
}
$objPHPExcel->getActiveSheet()->setTitle('SummarySaleBranch');

//-------------------------------Detailed Analysis--------------------------------------
if(!$detailAnalysis){
$query10 = "Select d1.customername,d1.customerId, d1.INV_Q1_2012, d1.INV_Q2_2012, d1.INV_Q3_2012, d1.INV_Q4_2012,d1.INV_Q1_2013,d1.INV_Total_2012,ifnull(d2.CN_Q1_2012,0) CNT_Q1_2012 , ifnull(d2.CN_Q2_2012,0) CNT_Q2_2012, ifnull(d2.CN_Q3_2012,0)

Page | G.20
### Detailed Analysis of Customers

<table>
<thead>
<tr>
<th>Customer Name</th>
<th>Customer ID</th>
<th>Invoice Q1 (Current Year)</th>
<th>Invoice Q2 (Current Year)</th>
<th>Invoice Q3 (Current Year)</th>
<th>Invoice Q4 (Current Year)</th>
<th>Total Invoice (Current Year)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>1</td>
<td>1000</td>
<td>2000</td>
<td>3000</td>
<td>4000</td>
<td>10000</td>
</tr>
<tr>
<td>A2</td>
<td>2</td>
<td>2000</td>
<td>3000</td>
<td>4000</td>
<td>5000</td>
<td>14000</td>
</tr>
<tr>
<td>A3</td>
<td>3</td>
<td>3000</td>
<td>4000</td>
<td>5000</td>
<td>6000</td>
<td>18000</td>
</tr>
<tr>
<td>A4</td>
<td>4</td>
<td>4000</td>
<td>5000</td>
<td>6000</td>
<td>7000</td>
<td>22000</td>
</tr>
</tbody>
</table>

### Diff Q1 % (Current Year/Next Year)

<table>
<thead>
<tr>
<th>Customer Name</th>
<th>% Diff Q1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>0%</td>
</tr>
<tr>
<td>A2</td>
<td>5%</td>
</tr>
<tr>
<td>A3</td>
<td>10%</td>
</tr>
<tr>
<td>A4</td>
<td>15%</td>
</tr>
</tbody>
</table>

### Total Cr/Inv (Current Year)

<table>
<thead>
<tr>
<th>Customer Name</th>
<th>Total Cr/Inv</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>10000</td>
</tr>
<tr>
<td>A2</td>
<td>14000</td>
</tr>
<tr>
<td>A3</td>
<td>18000</td>
</tr>
<tr>
<td>A4</td>
<td>22000</td>
</tr>
</tbody>
</table>

### Cr/Inv Q1 (Current Year)

<table>
<thead>
<tr>
<th>Customer Name</th>
<th>Cr/Inv Q1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>10000</td>
</tr>
<tr>
<td>A2</td>
<td>14000</td>
</tr>
<tr>
<td>A3</td>
<td>18000</td>
</tr>
<tr>
<td>A4</td>
<td>22000</td>
</tr>
</tbody>
</table>

### Cr/Inv Q4 (Current Year)

<table>
<thead>
<tr>
<th>Customer Name</th>
<th>Cr/Inv Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>4000</td>
</tr>
<tr>
<td>A2</td>
<td>5000</td>
</tr>
<tr>
<td>A3</td>
<td>6000</td>
</tr>
<tr>
<td>A4</td>
<td>7000</td>
</tr>
</tbody>
</table>

### Credit Note Q1 (Current Year)

<table>
<thead>
<tr>
<th>Customer Name</th>
<th>Credit Note Q1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>10000</td>
</tr>
<tr>
<td>A2</td>
<td>14000</td>
</tr>
<tr>
<td>A3</td>
<td>18000</td>
</tr>
<tr>
<td>A4</td>
<td>22000</td>
</tr>
</tbody>
</table>

### Credit Note Q4 (Current Year)

<table>
<thead>
<tr>
<th>Customer Name</th>
<th>Credit Note Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>4000</td>
</tr>
<tr>
<td>A2</td>
<td>5000</td>
</tr>
<tr>
<td>A3</td>
<td>6000</td>
</tr>
<tr>
<td>A4</td>
<td>7000</td>
</tr>
</tbody>
</table>

### Credit Note Q3 (Current Year)

<table>
<thead>
<tr>
<th>Customer Name</th>
<th>Credit Note Q3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>3000</td>
</tr>
<tr>
<td>A2</td>
<td>4000</td>
</tr>
<tr>
<td>A3</td>
<td>5000</td>
</tr>
<tr>
<td>A4</td>
<td>6000</td>
</tr>
</tbody>
</table>

### Credit Note Q2 (Current Year)

<table>
<thead>
<tr>
<th>Customer Name</th>
<th>Credit Note Q2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>2000</td>
</tr>
<tr>
<td>A2</td>
<td>3000</td>
</tr>
<tr>
<td>A3</td>
<td>4000</td>
</tr>
<tr>
<td>A4</td>
<td>5000</td>
</tr>
</tbody>
</table>

### Credit Note Q1 (Next Year)

<table>
<thead>
<tr>
<th>Customer Name</th>
<th>Credit Note Q1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>3000</td>
</tr>
<tr>
<td>A2</td>
<td>4000</td>
</tr>
<tr>
<td>A3</td>
<td>5000</td>
</tr>
<tr>
<td>A4</td>
<td>6000</td>
</tr>
</tbody>
</table>

### Total Credit Note (Current Year)

<table>
<thead>
<tr>
<th>Customer Name</th>
<th>Total Credit Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>20000</td>
</tr>
<tr>
<td>A2</td>
<td>24000</td>
</tr>
<tr>
<td>A3</td>
<td>28000</td>
</tr>
<tr>
<td>A4</td>
<td>32000</td>
</tr>
</tbody>
</table>

### Diff Q1 % (Current Year/Next Year)

<table>
<thead>
<tr>
<th>Customer Name</th>
<th>Diff Q1 %</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>0%</td>
</tr>
<tr>
<td>A2</td>
<td>5%</td>
</tr>
<tr>
<td>A3</td>
<td>10%</td>
</tr>
<tr>
<td>A4</td>
<td>15%</td>
</tr>
</tbody>
</table>
while ($data = mysql_fetch_array($result_set)) {
    $objPHPExcel->setActiveSheetIndex(9)
    ->setCellValue('A' . $row, $data['customerName'])
    ->setCellValue('B' . $row, $data['customerID'])
    ->setCellValue('C' . $row, $data['INV_Q1_2012'])
    ->setCellValue('D' . $row, $data['INV_Q2_2012'])
    ->setCellValue('E' . $row, $data['INV_Q3_2012'])
    ->setCellValue('F' . $row, $data['INV_Q4_2012'])
    ->setCellValue('G' . $row, $data['INV_Total_2012'])
    ->setCellValue('H' . $row, $data['CNT_Q1_2012'])
    ->setCellValue('I' . $row, $data['CNT_Q2_2012'])
    ->setCellValue('J' . $row, $data['CNT_Q3_2012'])
    ->setCellValue('K' . $row, $data['CNT_Q4_2012'])
    ->setCellValue('L' . $row, $data['CNT_Total_2012'])
    ->setCellValue('M' . $row, $data['C��TotalInv2012'])
    ->setCellValue('N' . $row, $data['TotalCrByTotalInv2012'])
    ->setCellValue('O' . $row, $data['CrByInvQ12012'])
    ->setCellValue('P' . $row, $data['CrByInvQ22012'])
    ->setCellValue('Q' . $row, $data['CrByInvQ32012'])
    ->setCellValue('R' . $row, $data['CrByInvQ42012'])
    ->setCellValue('S' . $row, $data['CrByInvQ12013'])
    ->setCellValue('T' . $row, $data['TotalCrByTotalInv2012'])
    ->setCellValue('U' . $row, $data['DiffQ120122013'])
    ->setCellValue('V' . $row, $data['DiffQ120122013Percent'])
    $row++;
}
$objPHPExcel->getActiveSheet(9)->setTitle('DetailedAnalysis');

//--------------------------------Cut-Off Testing Invoice 1-------------------------------------

if($invoice1){
    $query11 = "SELECT invoiceNo, invoiceDate, dueDate, customerID, customerName, dateTrans, postDate, description, grossSale, vat, netSale, quantity, branch, despatchNote FROM sales2012 WHERE invoiceDate< $newYear AND postDate> $yearEnd ORDER BY InvoiceDate";
    $result_set = mysql_query($query11);

echo date('Hi:i:s') , " Cut-Off Testing for Invoices " , EOL;
    $objWorkSheet = $objPHPExcel->createSheet(10);
    $objPHPExcel->getActiveSheet(10)->getStyle('A1')->getFont()->setBold(true);
    $objPHPExcel->getActiveSheet(10)->setCellValue('A1', 'Cut-Off Testing for Invoices')
    ->setCellValue('A2', 'Issued date before year end, posted after')
    ->setCellValue('A3', 'Client: ')
    ->setCellValue('A4', 'Created by: ')
    ->setCellValue('A5', 'Invoice Number')
    ->setCellValue('B5', 'Invoice Date')
    ->setCellValue('C5', 'Due Date')
    ->setCellValue('D5', 'Customer ID')
    ->setCellValue('E5', 'Customer Name')
    ->setCellValue('F5', 'Transaction Date')
    ->setCellValue('G5', 'Post Date')
    ->setCellValue('H5', 'Description')
    ->setCellValue('I5', 'Gross Sale')
    ->setCellValue('J5', 'VAT')
    ->setCellValue('K5', 'Net Sale')
    ->setCellValue('L5', 'Quantity')
    ->setCellValue('M5', 'Branch')
    ->setCellValue('N5', 'Despatch Note');
    $row = 6;
    while ($data = mysql_fetch_array($result_set)) {
        $objPHPExcel->getActiveSheet(10)
$yearEnd = date('Y');
$query12 = "SELECT invoiceNo, invoiceDate, dueDate, customerID, customerName, dateTrans, postDate, description, grossSale, vat, netSale, quantity, branch, despatchNote FROM sales2012 WHERE invoiceDate > $yearEnd AND postDate < $newYear
ORDER BY InvoiceDate";
$result_set = mysql_query($query12);

echo date("Hi:s"), " Cut-Off Testing for Invoices", EOL;
$objPHPExcel->createSheet(11);
$objPHPExcel->setActiveSheetIndex(11);
$objPHPExcel->getStyle('A1')->setFont()->setBold(true);
$objPHPExcel->setActiveStyleSheet(11);
$objPHPExcel->setCellValue('A1', 'Cut-Off Testing for Invoices');
$objPHPExcel->setCellValue('A2', 'Issued date after year end, posted before');
$objPHPExcel->setCellValue('A3', 'Client: ');
$objPHPExcel->setCellValue('A4', 'Created by: ');
$objPHPExcel->setCellValue('A5', 'Invoice Number');
$objPHPExcel->setCellValue('B5', 'Invoice Date');
$objPHPExcel->setCellValue('C5', 'Due Date');
$objPHPExcel->setCellValue('D5', 'Customer ID');
$objPHPExcel->setCellValue('E5', 'Customer Name');
$objPHPExcel->setCellValue('F5', 'Transaction Date');
$objPHPExcel->setCellValue('G5', 'Post Date');
$objPHPExcel->setCellValue('H5', 'Description');
$objPHPExcel->setCellValue('I5', 'Gross Sale');
$objPHPExcel->setCellValue('J5', 'VAT');
$objPHPExcel->setCellValue('K5', 'Net Sale');
$objPHPExcel->setCellValue('L5', 'Quantity');
$objPHPExcel->setCellValue('M5', 'Branch');
$objPHPExcel->setCellValue('N5', 'Despatch Note');

$row = 6;
while ($data = mysql_fetch_array($result_set)) {
    $objPHPExcel->setActiveSheetIndex(11);
    $objPHPExcel->setCellValue('A'. $row, $data['invoiceNo']);
    $objPHPExcel->setCellValue('B'. $row, $data['invoiceDate']);
    $objPHPExcel->setCellValue('C'. $row, $data['dueDate']);
    $objPHPExcel->setCellValue('D'. $row, $data['customerID']);
    $objPHPExcel->setCellValue('E'. $row, $data['customerName']);
    $objPHPExcel->setCellValue('F'. $row, $data['dateTrans']);
    $objPHPExcel->setCellValue('G'. $row, $data['postDate']);
    $objPHPExcel->setCellValue('H'. $row, $data['description']);
    $objPHPExcel->setCellValue('I'. $row, $data['grossSale']);
    $objPHPExcel->setCellValue('J'. $row, $data['vat']);
    $objPHPExcel->setCellValue('K'. $row, $data['netSale']);
    $row++;
}
if($credit1){
    $query13 = "SELECT creditNoteNumber, creditNoteDate, postDate, customerID,
customerName, creditNoteAmount FROM cn2012 WHERE creditNoteDate>$query13
    $newYear AND
    postDate> $yearEnd ORDER BY creditNoteDate";
    $result_set = mysql_query($query13);
    $objPHPExcel->createSheet(12);
    $objPHPExcel->setTitle('CutOffTestCreditNote1');
    $objPHPExcel->getActiveSheet(12)->setTitle('CutOff Test Credit Note 1');
    $objPHPExcel->getActiveSheet(12)->setCellValue('A1', 'Cut Off Testing for Credit Notes');
    $objPHPExcel->getActiveSheet(12)->setCellValue('A2', 'Issued date after year end, posted before');
    $objPHPExcel->getActiveSheet(12)->setCellValue('A3', 'Client: ');  // EOL
    $objPHPExcel->getActiveSheet(12)->setCellValue('A4', 'Created by:');  // EOL
    $objPHPExcel->getActiveSheet(12)->setCellValue('A5', 'Credit Note Number');
    $objPHPExcel->getActiveSheet(12)->setCellValue('B5', 'Credit Note Date');
    $objPHPExcel->getActiveSheet(12)->setCellValue('C5', 'Post Date');
    $objPHPExcel->getActiveSheet(12)->setCellValue('D5', 'Customer ID');
    $objPHPExcel->getActiveSheet(12)->setCellValue('E5', 'Customer Name');
    $objPHPExcel->getActiveSheet(12)->setCellValue('F5', 'Credit Note Amount');
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->getFont()->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->getFont()->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setBold(true);
    $objPHPExcel->getActiveSheet(12)->getStyle('A3:F5')->setB
if

while ($query15 = "SELECT despatchNote, despatchDate, postDate, customerID, customerName, quantity FROM des2012 WHERE despatchDate < $yearEnd AND postDate > $yearEnd ORDER BY despatchDate");

$stdClass = mysql_query($query15);

$stdClass = mysql_fetch_array($result_set); {
    $objPHPExcel = getWorksheet($objPHPExcel);
    $objPHPExcel = setActiveSheetIndex($objPHPExcel, 13);
    $objPHPExcel = setCellValue($objPHPExcel, 'A', $row, $data[creditNoteNumber]);
    $objPHPExcel = setCellValue($objPHPExcel, 'B', $row, $data[creditNoteDate]);
    $objPHPExcel = setCellValue($objPHPExcel, 'C', $row, $data[postDate]);
    $objPHPExcel = setCellValue($objPHPExcel, 'D', $row, $data[customerID]);
    $objPHPExcel = setCellValue($objPHPExcel, 'E', $row, $data[customerName]);
    $objPHPExcel = setCellValue($objPHPExcel, 'F', $row, $data[creditNoteAmount]);
    $row++;
}

$objPHPExcel = setActiveSheetIndex($objPHPExcel, 13)->setTitle('CutOffTestCr2');

if($despatch2){

if($query16 = "SELECT despatchNote, despatchDate, postDate, customerID, customerName, quantity FROM des2012 WHERE despatchDate > $yearEnd AND postDate < $yearEnd ORDER BY despatchDate");

$stdClass = mysql_query($query16);

$stdClass = mysql_fetch_array($result_set); {
    $objPHPExcel = getWorksheet($objPHPExcel);
    $objPHPExcel = setActiveSheetIndex($objPHPExcel, 14);
    $objPHPExcel = setCellValue($objPHPExcel, 'A', $row, $data[despatchNote]);
    $objPHPExcel = setCellValue($objPHPExcel, 'B', $row, $data[despatchNote]);
    $objPHPExcel = setCellValue($objPHPExcel, 'C', $row, $data[postDate]);
    $objPHPExcel = setCellValue($objPHPExcel, 'D', $row, $data[customerID]);
    $objPHPExcel = setCellValue($objPHPExcel, 'E', $row, $data[customerName]);
    $objPHPExcel = setCellValue($objPHPExcel, 'F', $row, $data[quantity]);
    $row++;
}

$objPHPExcel = setActiveSheetIndex($objPHPExcel, 14)->setTitle('CutOffTestDesp1');

//-------------------------------Cut Off Testing Despatch 2-------------------------------

if($despatch2){

if($query16 = "SELECT despatchNote, despatchDate, postDate, customerID, customerName, quantity FROM des2012 WHERE despatchDate > $yearEnd AND postDate < $yearEnd ORDER BY despatchDate");

$stdClass = mysql_query($query16);

$stdClass = mysql_fetch_array($result_set); {
    $objPHPExcel = getWorksheet($objPHPExcel);
    $objPHPExcel = setActiveSheetIndex($objPHPExcel, 14);
    $objPHPExcel = setCellValue($objPHPExcel, 'A', $row, $data[despatchNote]);
    $objPHPExcel = setCellValue($objPHPExcel, 'B', $row, $data[despatchNote]);
    $objPHPExcel = setCellValue($objPHPExcel, 'C', $row, $data[postDate]);
    $objPHPExcel = setCellValue($objPHPExcel, 'D', $row, $data[customerID]);
    $objPHPExcel = setCellValue($objPHPExcel, 'E', $row, $data[customerName]);
    $objPHPExcel = setCellValue($objPHPExcel, 'F', $row, $data[quantity]);
    $row++;
}

$objPHPExcel = setActiveSheetIndex($objPHPExcel, 14)->setTitle('CutOffTestDesp1');

//-------------------------------Cut Off Testing Despatch 2-------------------------------
$objPHPExcel->getActiveSheet()->getStyle('A1')->setFont()->setBold(true);
$objPHPExcel->getActiveSheet()->getStyle('A1')->getFont()->setBold(true);
$objPHPExcel->getActiveSheet()->getStyle('A1')->getFont()->setBold(true);
$objPHPExcel->getActiveSheet()->getStyle('A1')->getFont()->setBold(true);
$objPHPExcel->getActiveSheet()->getStyle('A1')->getFont()->setBold(true);

$objPHPExcel->getActiveSheet()->setCellValue('A3', 'Client: ');
$objPHPExcel->getActiveSheet()->setCellValue('A4', 'Created by: ');
$objPHPExcel->getActiveSheet()->setCellValue('A5', 'Despatch Note ');
$objPHPExcel->getActiveSheet()->setCellValue('A6', 'Despatch Note Date ');
$objPHPExcel->getActiveSheet()->setCellValue('A7', 'Post Date ');
$objPHPExcel->getActiveSheet()->setCellValue('A8', 'Customer ID ');
$objPHPExcel->getActiveSheet()->setCellValue('A9', 'Customer Name ');
$objPHPExcel->getActiveSheet()->setCellValue('A10', 'Quantity ');

$row = 6;
while ($data = mysql_fetch_array($result_set)) {
    $objPHPExcel->getActiveSheet()->setCellValue('A'. $row, $data['despatchNote']);
    $objPHPExcel->getActiveSheet()->setCellValue('B'. $row, $data['despatchDate']);
    $objPHPExcel->getActiveSheet()->setCellValue('C'. $row, $data['postDate']);
    $objPHPExcel->getActiveSheet()->setCellValue('D'. $row, $data['customerID']);
    $objPHPExcel->getActiveSheet()->setCellValue('E'. $row, $data['customerName']);
    $objPHPExcel->getActiveSheet()->setCellValue('F'. $row, $data['quantity']);
    $row++;
}
$objPHPExcel->getActiveSheet()->setCellValue('F5', 'Quantity ');

$objPHPExcel->getActiveSheet()->setCellValue('A3', 'Files have been created in ');
$objPHPExcel->getActiveSheet()->setCellValue('A4', 'Call time ');
$objPHPExcel->getActiveSheet()->setCellValue('A5', 'Date written to ');
$objPHPExcel->getActiveSheet()->setCellValue('A6', 'Call time to write Workbook was ');
$objPHPExcel->getActiveSheet()->setCellValue('A7', 'current memory usage: ');
$objPHPExcel->getActiveSheet()->setCellValue('A8', 'Memory peak usage ');
$objPHPExcel->getActiveSheet()->setCellValue('A9', 'Files have been created in ');
$objPHPExcel->getActiveSheet()->setCellValue('A10', 'Getcwd()');
?>
**detail.php**

```php
<?php
$host = "127.0.0.1";
$database = "fyp";
$user = "root";
$password = "yee4Tech";
mysql_connect("$host", "$user", "$password");
mysql_select_db("$database");
?>
```

**testconnect.php**

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
<title>Untitled 1</title>
</head>
<body>

<?php
$user_name = "root";
$password = "yee4Tech";
$database = "fyp";
$server = "127.0.0.1";
mysql_connect($server, $user_name, $password);
print "Connection to the Server opened";
$db_found = mysql_select_db($database);
if ($db_found) {
    print "Database Found";
} else {
    print "Database NOT Found";
    print mysql_error();
}
?>
</body>
</html>
```

**showTableCash2011.php**

```html
<head>
<meta content="en-gb" http-equiv="Content-Language">
<style type="text/css">
.auto-style1 {
    text-align: center;
}
</style>
</head>
```
<?php

include("detail.php");

$connect = mysql_connect("$host", $user, $password) or die("Couldn't connect to SQL Server on $myServer");
mysql_select_db("$database") or die("Couldn't open database $myDB");

if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysql_connect_error();
}

$query1 = "SELECT * FROM cash2011";
$result = mysql_query($query1) or die(mysql_error());
$num_results = mysql_num_rows($result);

print "There are $num_results records.<P>";
print "<table width=400 border=1>

while ($get_info = mysql_fetch_row($result)){
    print "<tr>
    foreach ($get_info as $field)
    print "<td><font face=arial size=1/2>$field</font></td>
    print "</tr>

?>

<table style="width: 100%">
    <tr>
        <td class="auto-style1"><a href="http://home.html">Home</a></td>
        <td class="auto-style1"><a href="selectViewTables.html">Back</a></td>
    </tr>
</table>

showTableCash2012.php

<?php

include("detail.php");

$connect = mysql_connect("$host", $user, $password) or die("Couldn't connect to SQL Server on $myServer");
mysql_select_db("$database") or die("Couldn't open database $myDB");
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysql_connect_error();
}

$query1 = "SELECT * FROM cash2012";
$result = mysql_query($query1) or die(mysql_error());
$num_results = mysql_num_rows($result);
print "There are $num_results records.<P>";
print "<table width=400 border=1><tr>";
while ($get_info = mysql_fetch_row($result)) {
print "<tr>";
foreach ($get_info as $field)
print "<td><font face=arial size=1>$field</font></td>";
print "</tr>";
}
?>
<table style="width: 100%">
<tr>
	<td class="auto-style1"><a href="http://home.html">Home</a></td>
	<td class="auto-style1"><a href="selectViewTables.html">Back</a></td>
</tr>
</table>

showTableCn2011.php

<?php
include("detail.php");

$connect = mysql_connect("$host", $user, $password) or die("Couldn't connect to SQL Server on $myServer");
mysql_select_db("$database") or die("Couldn't open database $myDB");

if (mysqli_connect_errno())
{
	die("Failed to connect to MySQL: ". mysqli_connect_error());
}

$query1 = "SELECT * FROM cn2011";
$result = mysql_query($query1) or die(mysql_error());
$num_results = mysql_num_rows($result);

print "There are $num_results records.<P>";
print "<table width=400 border=1><tr>";
while ($get_info = mysql_fetch_row($result)) {
print "<tr>";
foreach ($get_info as $field)
print "<td><font face=arial size=1>$field</font></td>";
print "</tr>";
}
?>
<table style="width: 100%">
<tr>
	<td class="auto-style1"><a href="http://home.html">Home</a></td>
	<td class="auto-style1"><a href="selectViewTables.html">Back</a></td>
</tr>
</table>
showTableCn2012.php

<?php
include("detail.php");

$connect = mysql_connect("$host", $user, $password) or die("Couldn't connect to SQL Server on $myServer");
mysql_select_db("$database") or die("Couldn't open database $myDB");

if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: ". mysql_connect_error();
}

$query1 = "SELECT * FROM cn2012";
$result = mysql_query($query1) or die(mysql_error());
$num_results = mysql_num_rows($result);

print "There are $num_results records.";
print "<table width=400 border=1>";
while ($get_info = mysql_fetch_row($result)) {
    print "<tr>
    
    foreach ($get_info as $field)
    print "<td>$field</td>
    print "</tr>
    }

?>

<table style="width: 100%">
    <tr>
        <td class="auto-style1"><a href="http://home.html">Home</a></td>
        <td class="auto-style1"><a href="selectViewTables.html">Back</a></td>
    </tr>
</table>

showTableDr2011.php

<?php
include("detail.php");

$connect = mysql_connect("$host", $user, $password) or die("Couldn't connect to SQL Server on $myServer");
mysql_select_db("$database") or die("Couldn't open database $myDB");
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysql_connect_error();
}

$query1 = "SELECT * FROM dr2011";
$result = mysql_query($query1) or die(mysql_error());
$num_results = mysql_num_rows($result);

print "There are $num_results records.<P>";
print "<table width=400 border=1>
    <tr>
        <td class="auto-style1"><a href="http://home.html">Home</a></td>
        <td class="auto-style1"><a href="selectViewTables.html">Back</a></td>
    </tr>
</table>";

showTableDr2012.php

<?php
include("detail.php");
$connect = mysql_connect("$host", $user, $password) or die("Couldn't connect to SQL Server on $myServer");
mysql_select_db("$database") or die("Couldn't open database $myDB");

if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysql_connect_error();
}

$query1 = "SELECT * FROM dr2012";
$result = mysql_query($query1) or die(mysql_error());
$num_results = mysql_num_rows($result);

print "There are $num_results records.<P>";
print "<table width=400 border=1>
    <tr>
        <td class="auto-style1"><a href="http://home.html">Home</a></td>
        <td class="auto-style1"><a href="selectViewTables.html">Back</a></td>
    </tr>
</table>";
showTableDes2011.php

<?php
include("detail.php");

$connect = mysql_connect("$host", $user, $password) or die("Couldn't connect to SQL Server on $myServer");
mysql_select_db("$database") or die("Couldn't open database $myDB");

if (mysqli_connect_errno())
    {
        echo "Failed to connect to MySQL: " . mysql_connect_error();
    }

$query1 = "SELECT * FROM des2011";
$result = mysql_query($query1) or die(mysql_error());
$num_results = mysql_num_rows($result);

print "There are $num_results records."."\n";
print "<table width=400 border=1>"."\n";
while ($get_info = mysql_fetch_row($result)){
    print "<tr>"."\n";
    foreach ($get_info as $field)
        print "<td><font face=arial size=1>/$field</font></td>"."\n";
    print "</tr>"."\n";
}

?>
</table>

showTablesDes2012.php

<?php
include("detail.php");

$connect = mysql_connect("$host", $user, $password) or die("Couldn't connect to SQL Server on $myServer");
mysql_select_db("$database") or die("Couldn't open database $myDB");

if (mysqli_connect_errno())
    {
        echo "Failed to connect to MySQL: " . mysql_connect_error();
    }

$query1 = "SELECT * FROM des2012";
$result = mysql_query($query1) or die(mysql_error());
$num_results = mysql_num_rows($result);

print "There are $num_results records."."\n";
print "<table width=400 border=1>"."\n";
while ($get_info = mysql_fetch_row($result)){
    print "<tr>"."\n";
    foreach ($get_info as $field)
        print "<td><font face=arial size=1>/$field</font></td>"."\n";
    print "</tr>"."\n";
}

?>
</table>
<?php
include("detail.php");
$connect = mysql_connect("$host", $user, $password) or die("Couldn't connect to SQL Server on $myServer");
mysql_select_db("$database") or die("Couldn't open database $myDB");
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: ". mysql_connect_error();
}
$query1 = "SELECT * FROM des2012";
$result = mysql_query($query1) or die(mysql_error());
$num_results = mysql_num_rows($result);

print "There are $num_results records.";
print "<table width=400 border=1><tr>
while ($get_info = mysql_fetch_row($result)){
    print "<td>
    foreach ($get_info as $field) print "<td>$field</td>
    print "</tr>
}
?></table>

showTableSales2011.php

<?php
include("detail.php");
$connect = mysql_connect("$host", $user, $password) or die("Couldn't connect to SQL Server on $myServer");
mysql_select_db("$database") or die("Couldn't open database $myDB");
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: ". mysql_connect_error();
}
$query1 = "SELECT * FROM sales2011";
$result = mysql_query($query1) or die(mysql_error());
$num_results = mysql_num_rows($result);

print "There are $num_results records.<P>";
print "<table width=400 border=1><n>
while ($get_info = mysql_fetch_row($result)) {
print "<tr><n>
foreach ($get_info as $field) {
print "<td><font face=arial size=1/>$field</font></td>
}
print "</tr><n>
?

showTablesSales2012.php

<?php
include("detail.php");

$connect = mysql_connect("$host", $user, $password) or die("Couldn't connect to SQL Server on $myServer");
mysql_select_db("$database") or die("Couldn't open database $myDB");

if (mysqli_connect_errno()) {
    echo "Failed to connect to MySQL: " . mysql_connect_error();
}

$query1 = "SELECT * FROM sales2012";
$result = mysql_query($query1) or die(mysql_error());
$num_results = mysql_num_rows($result);

print "There are $num_results records.<P>";
print "<table width=400 border=1><n>
while ($get_info = mysql_fetch_row($result)) {
print "<tr><n>
foreach ($get_info as $field) {
print "<td><font face=arial size=1/>$field</font></td>
}
print "</tr><n>
?>
validations.js

<script type="text/javascript">
function validateForm() {
    var x=document.forms["myForm"]['yearEnd'].value;
    var y=document.forms["myForm"]['newYear'].value;

    if (x==null || x=="") {
        alert("Year End date must be entered.");
        return false;
    }
    if (y==null || y=="") {
        alert("New Year date must be entered.");
        return false;
    }

    var DS=document.getElementById("myCheck1").checked;
    var DC=document.getElementById("myCheck2").checked;
    var DDN=document.getElementById("myCheck3").checked;
    var DD=document.getElementById("myCheck4").checked;
    var NS=document.getElementById("myCheck5").checked;
    var NC=document.getElementById("myCheck6").checked;
    var ND=document.getElementById("myCheck7").checked;
    var TOP=document.getElementById("myCheck8").checked;
    var SB=document.getElementById("myCheck9").checked;
    var DA=document.getElementById("myCheck10").checked;

    var S1=document.getElementById("myCheck11").checked;
    var S2=document.getElementById("myCheck12").checked;
    var C1=document.getElementById("myCheck13").checked;
    var C2=document.getElementById("myCheck14").checked;
    var D1=document.getElementById("myCheck15").checked;
    var D2=document.getElementById("myCheck16").checked;

    if (DS==false && DC==false && DDN==false && DD==false && NS==false && NC==false && ND==false && TOP==false && SB==false && DA==false && S1==false && S2==false && C1==false && C2==false && D1==false && D2==false) {
        alert("At least one test must be chosen.");
        return false;
    }
}</script>
H. TEST DOCUMENTATION

This appendix outlines the testing carried out during the development of the application.

H.1. Software Testing

This application involved the merging of several software tools and programming languages, therefore it was important to test the connections between them to make sure that they operate properly.

Firstly, the connection between SQL Workbench 5.0 and MySQL database was set up. The connection was tested by requesting SQL Workbench 5.0 to complete a simply query and output the result data from the MySQL database.

Secondly, the connection between PHP and MySQL database was set up. The connection was tested by running the “testconnect.php” which identifies if the connection was successful. If the connection was unsuccessful, PHP explains the error that has occurred.

Lastly, the connection between PHP and SQL was set up. This connection was tested by including an SQL query within PHP code and executing the query.

H.2. Function Testing

This section explains how all functions in the system were tested to identify if they perform correctly.

**SQL Queries**

<table>
<thead>
<tr>
<th>Test</th>
<th>Expected Result</th>
<th>Testing Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Each SQL query</td>
<td>Output requested by the query</td>
<td>Execute each query. Checking that output achieved matched output provided.</td>
</tr>
</tbody>
</table>

**Interface**

<table>
<thead>
<tr>
<th>Test</th>
<th>Expected Result</th>
<th>Testing Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload CSV file</td>
<td>Correctly formatted CSV file populates the database table. If the format of CSV file is incorrect an error is printed.</td>
<td>Upload a CSV file into a database table and check that table. Upload a non-CSV file or unformatted CSV file and check if error printed.</td>
</tr>
<tr>
<td>View Tables</td>
<td>The entire table should print on the next page.</td>
<td>Click “View” for the table to print. Check that the correct table printed.</td>
</tr>
<tr>
<td>Date Validations</td>
<td>Message box should pop-up if date was not filled in.</td>
<td>Leave the date space black and click “Run Testing”. Check if the validations popped-up.</td>
</tr>
<tr>
<td>Checkbox Validations</td>
<td>At least one checkbox must be checked for Audit testing. If all checkboxes are left black, a pop-up message box appears.</td>
<td>Leave all checkboxes black and run Audit testing. Check if message box appeared.</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Complete Audit Testing</td>
<td>Output should print to Excel. Each query output should be printed to a separate Excel sheet. The output should match the output provided by the client.</td>
<td>Select some and all Audit tests and ran the PHP code. Check that the Excel file was created. Check that each test was printed separately. Check that the output was correct.</td>
</tr>
</tbody>
</table>
GLOSSARY OF TERMS USED

CSS
Cascading Style Sheets is a style sheet language used for describing the presentation semantics of a document written in a mark-up language.

HTML
Hypertext Mark-up Language is a standardised system for tagging text files to achieve font, colour, graphic and hyperlink effects on World Wide Web pages.

PHP
Hypertext Pre-processor is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

SQL
Structured Query Language is a specialised language for updating, deleting and requesting information from a database.

MySQL
MySQL is a database management system.

PHPMmysql
PHPMysql is a free and open source tool written in PHP intended to handle the administration of MySQL with the use of a web browser. It can perform tasks such as creating and deleting databases, modifying table fields and rows, and executing SQL statements.

JavaScript
JavaScript is an object-oriented computer programming language often used to create interactive effects in the form of validations within web pages.
REFERENCES


University of Dublin
Trinity College

Management Science and
Information Systems Studies

Final Year Project Report

Database to Assist Auditors with Client
Data Testing and Analysis
for
KPMG Ireland

User Manual
and
Technical Manual

Dasha Gorovenco

March 2014
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>NO.</th>
<th>SECTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2.</td>
<td>HOME PAGE</td>
<td>2</td>
</tr>
<tr>
<td>3.</td>
<td>UPDATE DATABASE</td>
<td>3</td>
</tr>
<tr>
<td>3.1</td>
<td>Convert Files to CSV Format</td>
<td>3</td>
</tr>
<tr>
<td>3.2</td>
<td>Automatically Import CSV Files</td>
<td>5</td>
</tr>
<tr>
<td>3.3</td>
<td>Manually Import CSV File</td>
<td>6</td>
</tr>
<tr>
<td>4.</td>
<td>VIEW DATABASE TABLES</td>
<td>8</td>
</tr>
<tr>
<td>5.</td>
<td>COMPLETE AUDIT TESTING</td>
<td>9</td>
</tr>
<tr>
<td>6.</td>
<td>OUTPUT TO EXCEL</td>
<td>10</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

The User Manual was created to provide the user with a step by step guide on how to use the system. It will cover basic interface usage as well as other features.
2. HOME PAGE

The Home page provides you with three options – update the database, view current database tables or complete Audit testing on the existing database (Work with Existing Database Files). The application was designed such that the Home page can be returned to at anytime by clicking the *Home* link at the bottom of each page.
3. UPDATE DATABASE

3.1 Convert Files to CSV Format

Only a CSV file may be imported into a database table. To make sure that the data is submitted correctly, the file should follow a specific format. Follow the steps below:

a. Open the Excel sheet you wish to import into a database table.

b. Click the button and click “Save As”.

c. Now save the file as CSV (Comma delimited).

d. Open the newly saved CSV file and delete the first row with the headings.

e. Databases are very specific with the format for dates. Highlight column(s) that contain dates. Right click on the highlighted column and click “Format Cells” in the list of options. In the pop-up window select “Date” from the list on the left hand side and chose the date in the following format: YYYY-MM-DD.
f. Repeat above steps for each Excel sheet.

The columns within CSV files must match the column names in the database tables. Below is the list of tables and their column headings. Make sure that the columns in your CSV files match the names listed in the tables below i.e. you might have to swap columns around in your CSV file before uploading it into the database table.
3.2 Automatically Import CSV Files

You can update the data in the database automatically. Click the “Update Database” option on the Home page and then click the “Choose File” button beside the table that you want to update.

Browse the computer for the required CSV files and when all the files have been selected click the “Upload Files” button at the bottom of the page. This will automatically update the contents of the database tables.
3.3 Manually Import CSV Files

The CSV files can also be imported manually. Log into PHPMyAdmin.

Chose the database you wish to work with on the left hand side of the screen and then chose the table to import new data into. Click the “Import” button.

Browse for the CSV file and set the parameters as shown below. Then click “Go” and the data is imported into the table.
Partial Import:

- Allow the interruption of an import in case the script detects it is close to the PHP timeout limit. (This might be a good way to import large files, however it can break transactions.)

- Skip this number of queries (for SQL) or lines (for other formats), starting from the first one: 0

Format:

- CSV using LOAD DATA

Format-Specific Options:

- Replace table data with file
- Columns separated with: [ ]
- Columns enclosed with: [ ]
- Columns escaped with: [ ]
- Lines terminated with: [auto]
- Column names: [ ]

- Do not abort on INSERT error
- Use LOCAL keyword
4. **VIEW CURRENT TABLES**

In the Home page there is an option to view the current data in the database tables. Also, after the CSV files are automatically uploaded into the database tables, you will be automatically redirected to this page as well. This page will allow you to either view the tables to make sure data input was correct, or move onto Audit data testing.

On the View Tables page, each table can be viewed individually by clicking the “View” button. Each table has its own individual “View” button.

If you noticed that a mistake occurred when the CSV files were uploaded, you can return and upload the files again.
5. COMPLETE AUDIT TESTING

When conducting the Audit tests, you will be presented with an option to choose which tests you wish to complete. You must also enter the yearend date and the next year date. These are mandatory and you will be notified if these are missing or the input format was incorrect. After selecting the tests you wish to complete, click the "Run" button at the bottom of the page.

The output will automatically print to an Excel file and you will be presented with the breakdown of the process. This breakdown will indicate how long it took for each test to complete and where the file has been saved to, as well as other useful information.
6. **OUTPUT TO EXCEL FILE**

The output is generated into an Excel file. Each test result is printed on a separate Excel sheet for neatness and clarity. You may manipulate the file if required.
## TECHNICAL MANUAL

## TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>NO.</th>
<th>SECTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2.</td>
<td>TECHNICAL ENVIRONMENT AND SOFTWARE REQUIREMENTS</td>
<td>2</td>
</tr>
<tr>
<td>3.</td>
<td>SYSTEM SETUP</td>
<td>3</td>
</tr>
<tr>
<td>3.1</td>
<td>Server</td>
<td>3</td>
</tr>
<tr>
<td>3.2</td>
<td>Database</td>
<td>3</td>
</tr>
<tr>
<td>3.3</td>
<td>PHP Interface</td>
<td>8</td>
</tr>
<tr>
<td>4.</td>
<td>TESTING</td>
<td>10</td>
</tr>
<tr>
<td>5.</td>
<td>MODIFICATIONS</td>
<td>12</td>
</tr>
<tr>
<td>6.</td>
<td>DATABASE BACKUP</td>
<td>13</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

The Technical Manual is a step by step guide on how to deploy the application. It will cover items such as the software required, the technical environment, how to setup various aspects of the system as well as testing, modifications and database backup.
2. TECHNICAL ENVIRONMENT

The application was fully developed using the WAMP Stack (Windows, Apache, MySQL, PHP). You are recommended to download the WAMP Stack from http://www.softpedia.com/get/Internet/Servers/Server-Tools/BitNami-WAMPStack.shtml and follow the simple installation instructions. Within the WAMP Stack Windows is the operating system, Apache is the server, MySQL is the database and PHP is the user interface.

Additionally to the WAMP Stack, you are recommended to download the Microsoft Web Expression 4 from http://www.microsoft.com/en-us/download/details.aspx?id=36179 for the user interface, and SQL Workbench 5.0 from http://dev.mysql.com/downloads/tools/workbench/ for the development and testing of SQL queries. This is where the queries should be built before being implemented within PHP. Also, you should download and install PHPMyAdmin from http://www.phpmyadmin.net/home_page/downloads.php for the creation and manipulation of the database.

The application includes five languages in total: SQL, PHP, CSS, HTML and JavaScript.
3. **SYSTEM SETUP**

This section will describe in detail how to setup and connect all software used for the application.

3.1. **Server**

To install the server you should download the WAMP Stack and follow the installation steps. The WAMP Stack automatically installs the server onto your system as well as MySQL and PHP. To test if your installation was successful, you should type in “localhost” into the search box in your browser. If the installation was completed correctly, you should be presented with the “Index” and the list of all files present in the “htdocs” folder.

3.2. **Database**

To set up the database you should log into PHPMyAdmin with the username and password setup during the installation of the WAMP Stack. Create a new database on the left hand side of the interface and then create the tables required. All table names must match the names of the tables used in this application; otherwise a vast amount of code modification in SQL and PHP will be required. The table names do not include the bracketed text. The column names must also be identical to the ones used in this application.

In the tables below the numbers represent the column order. The “name” column represents the names of each column in the database table. The “type” column in each table below represents the characteristic of each database table column. The parameters in your database should be the exact same.

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>cashAmount</td>
<td>varchar(11)</td>
</tr>
<tr>
<td>2</td>
<td>cashReceivedDate</td>
<td>date</td>
</tr>
<tr>
<td>3</td>
<td>customerID</td>
<td>int(11)</td>
</tr>
<tr>
<td>4</td>
<td>invoiceNo</td>
<td>int(11)</td>
</tr>
<tr>
<td>5</td>
<td>monthNum</td>
<td>int(11)</td>
</tr>
<tr>
<td>6</td>
<td>yearNum</td>
<td>int(11)</td>
</tr>
</tbody>
</table>
### Table cash2012 (Cash Current Year)

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>cashAmount</td>
<td>varchar(15)</td>
</tr>
<tr>
<td>2</td>
<td>cashReceivedDate</td>
<td>date</td>
</tr>
<tr>
<td>3</td>
<td>customerID</td>
<td>int(11)</td>
</tr>
<tr>
<td>4</td>
<td>monthNum</td>
<td>int(11)</td>
</tr>
<tr>
<td>5</td>
<td>yearNum</td>
<td>int(11)</td>
</tr>
<tr>
<td>6</td>
<td>invoiceNo</td>
<td>int(11)</td>
</tr>
</tbody>
</table>

### Table cn2011 (Credit Notes Prior Year)

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>creditNoteNumber</td>
<td>varchar(15)</td>
</tr>
<tr>
<td>2</td>
<td>creditNoteDate</td>
<td>date</td>
</tr>
<tr>
<td>3</td>
<td>customerID</td>
<td>int(11)</td>
</tr>
<tr>
<td>4</td>
<td>customerName</td>
<td>text</td>
</tr>
<tr>
<td>5</td>
<td>creditNoteAmount</td>
<td>double</td>
</tr>
<tr>
<td>6</td>
<td>creditNoteMonth</td>
<td>int(11)</td>
</tr>
</tbody>
</table>

### Table cn2012 (Credit Notes Current Year)

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>creditNoteNumber</td>
<td>varchar(15)</td>
</tr>
<tr>
<td>2</td>
<td>creditNoteDate</td>
<td>date</td>
</tr>
<tr>
<td>3</td>
<td>postDate</td>
<td>date</td>
</tr>
<tr>
<td>4</td>
<td>customerID</td>
<td>int(11)</td>
</tr>
<tr>
<td>5</td>
<td>customerName</td>
<td>text</td>
</tr>
<tr>
<td>6</td>
<td>creditNoteAmount</td>
<td>double</td>
</tr>
<tr>
<td>7</td>
<td>creditNoteMonth</td>
<td>int(11)</td>
</tr>
<tr>
<td>8</td>
<td>quarter</td>
<td>int(11)</td>
</tr>
</tbody>
</table>
Table des2011 (Despatch Notes Prior Year)

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>despatchNote</td>
<td>varchar(15)</td>
</tr>
<tr>
<td>2</td>
<td>despatchDate</td>
<td>date</td>
</tr>
<tr>
<td>3</td>
<td>postDate</td>
<td>date</td>
</tr>
<tr>
<td>4</td>
<td>customerID</td>
<td>int(11)</td>
</tr>
<tr>
<td>5</td>
<td>customerName</td>
<td>text</td>
</tr>
<tr>
<td>6</td>
<td>quantity</td>
<td>int(11)</td>
</tr>
<tr>
<td>7</td>
<td>netSale</td>
<td>double</td>
</tr>
<tr>
<td>8</td>
<td>monthNum</td>
<td>int(11)</td>
</tr>
</tbody>
</table>

Table des2012 (Despatch Notes Current Year)

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>despatchNote</td>
<td>varchar(15)</td>
</tr>
<tr>
<td>2</td>
<td>despatchDate</td>
<td>date</td>
</tr>
<tr>
<td>3</td>
<td>postDate</td>
<td>date</td>
</tr>
<tr>
<td>4</td>
<td>customerID</td>
<td>int(11)</td>
</tr>
<tr>
<td>5</td>
<td>customerName</td>
<td>text</td>
</tr>
<tr>
<td>6</td>
<td>quantity</td>
<td>int(11)</td>
</tr>
<tr>
<td>7</td>
<td>netAmount</td>
<td>double</td>
</tr>
<tr>
<td>8</td>
<td>outOfSequence</td>
<td>int(11)</td>
</tr>
<tr>
<td>9</td>
<td>duplicate</td>
<td>int(11)</td>
</tr>
<tr>
<td>10</td>
<td>gaps</td>
<td>int(11)</td>
</tr>
<tr>
<td>11</td>
<td>gap2</td>
<td>int(11)</td>
</tr>
<tr>
<td>12</td>
<td>monthNum</td>
<td>int(11)</td>
</tr>
</tbody>
</table>

Table dr2011 (Debtors Prior Year)

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>debtorID</td>
<td>int(11)</td>
</tr>
<tr>
<td>2</td>
<td>debtorName</td>
<td>text</td>
</tr>
<tr>
<td>3</td>
<td>currentBal</td>
<td>double</td>
</tr>
<tr>
<td>4</td>
<td>creditLimit</td>
<td>int(11)</td>
</tr>
</tbody>
</table>
Table dr2012 (Debtors Current Year)

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>debtorID</td>
<td>int(11)</td>
</tr>
<tr>
<td>2</td>
<td>debtorName</td>
<td>text</td>
</tr>
<tr>
<td>3</td>
<td>currentBal</td>
<td>double</td>
</tr>
<tr>
<td>4</td>
<td>creditLimit</td>
<td>int(11)</td>
</tr>
</tbody>
</table>

Table sales2011 (Sales Prior Year)

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>invoiceNo</td>
<td>int(11)</td>
</tr>
<tr>
<td>2</td>
<td>invoiceDate</td>
<td>date</td>
</tr>
<tr>
<td>3</td>
<td>dueDate</td>
<td>date</td>
</tr>
<tr>
<td>4</td>
<td>customerID</td>
<td>int(11)</td>
</tr>
<tr>
<td>5</td>
<td>customerName</td>
<td>text</td>
</tr>
<tr>
<td>6</td>
<td>dateTrans</td>
<td>date</td>
</tr>
<tr>
<td>7</td>
<td>postDate</td>
<td>date</td>
</tr>
<tr>
<td>8</td>
<td>description</td>
<td>varchar(15)</td>
</tr>
<tr>
<td>9</td>
<td>grossSale</td>
<td>double</td>
</tr>
<tr>
<td>10</td>
<td>vat</td>
<td>double</td>
</tr>
<tr>
<td>11</td>
<td>netSale</td>
<td>double</td>
</tr>
<tr>
<td>12</td>
<td>quantity</td>
<td>int(11)</td>
</tr>
<tr>
<td>13</td>
<td>branch</td>
<td>text</td>
</tr>
<tr>
<td>14</td>
<td>despatchNote</td>
<td>varchar(15)</td>
</tr>
<tr>
<td>15</td>
<td>despatchDate</td>
<td>date</td>
</tr>
<tr>
<td>16</td>
<td>monthNum</td>
<td>int(11)</td>
</tr>
</tbody>
</table>
Table sales2012 (Sales Current Year)

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>invoiceNo</td>
<td>int(11)</td>
</tr>
<tr>
<td>2</td>
<td>invoiceDate</td>
<td>date</td>
</tr>
<tr>
<td>3</td>
<td>dueDate</td>
<td>date</td>
</tr>
<tr>
<td>4</td>
<td>customerId</td>
<td>int(11)</td>
</tr>
<tr>
<td>5</td>
<td>customerName</td>
<td>text</td>
</tr>
<tr>
<td>6</td>
<td>dateTrans</td>
<td>date</td>
</tr>
<tr>
<td>7</td>
<td>postDate</td>
<td>date</td>
</tr>
<tr>
<td>8</td>
<td>description</td>
<td>varchar(15)</td>
</tr>
<tr>
<td>9</td>
<td>grossSale</td>
<td>double</td>
</tr>
<tr>
<td>10</td>
<td>vat</td>
<td>double</td>
</tr>
<tr>
<td>11</td>
<td>netSalo</td>
<td>double</td>
</tr>
<tr>
<td>12</td>
<td>quantity</td>
<td>int(11)</td>
</tr>
<tr>
<td>13</td>
<td>branch</td>
<td>text</td>
</tr>
<tr>
<td>14</td>
<td>despatchNote</td>
<td>varchar(15)</td>
</tr>
<tr>
<td>15</td>
<td>amountPaid</td>
<td>double</td>
</tr>
<tr>
<td>16</td>
<td>outstanding</td>
<td>int(11)</td>
</tr>
<tr>
<td>17</td>
<td>paidDate</td>
<td>date</td>
</tr>
<tr>
<td>18</td>
<td>agingAmount</td>
<td>double</td>
</tr>
<tr>
<td>19</td>
<td>payStatusEnd</td>
<td>int(11)</td>
</tr>
<tr>
<td>20</td>
<td>payStatusMid</td>
<td>int(11)</td>
</tr>
<tr>
<td>21</td>
<td>aging</td>
<td>varchar(5)</td>
</tr>
<tr>
<td>22</td>
<td>monthNum</td>
<td>int(11)</td>
</tr>
<tr>
<td>23</td>
<td>quarter</td>
<td>int(11)</td>
</tr>
<tr>
<td>24</td>
<td>despatchDate</td>
<td>date</td>
</tr>
</tbody>
</table>
3.3. **PHP Interface**

In order to set up the user interface you should first off all locate your server’s root folder. This folder could be found within your server folders. For example, if you installed your server onto your C drive and called the folder “Apache2.2”, then the root folder will be within that folder. The root folder is called “htdocs” and all files that contain code should be copied into that folder. This means that all HTML, PHP, CSS and JavaScript files used in this application, must be present in the “htdocs” folder in order for the application to function properly.

Copy the files from the “Files” folder on the CD provided into this folder.
You can view your files stored within the “htdocs” folder by entering “localhost” into your browser. Your result should resemble the image shown below. The files stored on your “localhost” can only be deleted within the “htdocs” folder.

![Image of localhost browser page showing file index]

You will be required to modify the “detail.php” file to match the connection details of your database. The code for this process is displayed below. You will need to change the $database, $user and $password parameters.

```php
<?php
$host = "127.0.0.1";
$database = "fyp";
$user = "root";
$password = "yee4Tech";
mysql_connect("$host", $user, $password);
mysql_select_db("$database");
?>
```
4. TESTING

Testing should be carried out to ensure that all the required software is properly installed and connected.

Server

To test if your server has been properly installed, you should type in “localhost/index.html” into your search box in your browser. If the server has been installed successfully you will be displayed with the image below.

![It works!](image1)

PHPMyAdmin

Your entire PHPMyAdmin folder should be copied into the “htdocs” folder like the rest of your files. To test if the installation has been successful, type in “localhost/phpmyadmin/” into the search box in your browser. If the installation was completed correctly, you will be displayed with the log in page of PHPMyAdmin.

![PHPMyAdmin](image2)
Microsoft Expression Web 4

Prior to testing the connection of Microsoft Expression Web 4 and MySQL, you should make sure that all the details within the “detail.php” file are matching the connection details of your database. When this has been completed, you should type “localhost/testconnect.php” into the search box in your browser. If the connection to the database has been established, you will be presented with the image below.

If the connection was unsuccessful you will be presented with an explanation of the error that has occurred. This will help you identify the mistake within the “detail.php” file.

When the software and the server connections have been established successfully, you may begin testing various functions within the application to ensure that it still operates correctly.
5. MODIFICATIONS

You may modify the code as desired. The electronic version of the entire code required for this application is available on the CD attached.
6. DATABASE BACKUP

Since one database will be constantly updated for each KPMG client, you should create a backup every time the database is updated to avoid data loss. The backup of a database is performed in PHPMyAdmin.

1. Log into PHPMyAdmin.
2. Select the database you wish to backup.
3. Click the export button on the top panel.
4. Select the “Custom – display all possible options” option.
5. Select to save to a file and the format of the file to CSV or SQL.
6. Leave the rest of the options as default.
7. Click “Go”. The download of the backup will begin.