



Mission Report

THE PROVISION OF TECHNICAL ASSISTANCE GDDS MODULE ON HEALTH: REPORT ON EXPERT ASSISTANCE TO LESOTHO

Prepared by:

Gareth Daniell

Prepared for:

**The Lesotho Department of Health Planning and Statistics,
Lesotho Bureau of Statistics and the World Bank, General
Data Dissemination System, Socio-Demographic Statistics
Project for Anglophone Africa**

September 2009

Table of Contents

1. INTRODUCTION	4
2. IMPLEMENTATION OF THE CONSULTANCY	4
3. ACKNOWLEDGEMENTS	4
4. PROGRAM CONTEXT	5
5. LESOTHO DEPARTMENT OF HEALTH PLANNING AND STATISTICS (HPSD) DATA WAREHOUSE NEEDS	5
6. CHALLENGES FACING THE LESOTHO DEPARTMENT OF HEALTH PLANNING AND STATISTICS (HPSD)	5
7. OVERVIEW OF MEETINGS AND DISCUSSIONS	6
8. ASSESSMENT OF CURRENT SITUATION	6
9. FUNCTIONAL SPECIFICATION	8
9.1 Introduction	8
9.2 Solution Overview	10
9.3 User Interface	12
9.3.1 Overview	12
9.3.2 Import Tool	12
9.3.3 Reporting	13
10. ACTIVITIES & FURTHER RECOMMENDATIONS	13
10.1 Short term activities	13
10.2 Further Recommendations	14
10.2.1 Staff Training	14
10.3 Potential Future Actions	14
11. RISKS	15
11.1 Source databases and applications	15
11.2 IT department	15
11.3 Training	16

12.	TRAVEL AND ACCOMMODATION ARRANGEMENTS	16
13.	FINAL REFLECTIONS	16
14.	ANNEXURES	17
14.1	Annexure 1: Overview of Meetings Held	17
14.2	Annexure 2: Terms of Reference	18
14.3	Annexure 3: Functional Specification	19
14.4	Annexure 4: Source Code and Technical Manual	20

1. INTRODUCTION

The Lesotho Bureau of Statistics and the Department of Health Planning and Statistics are under constant pressure to produce relevant up to date Health Statistics. They have identified a fully functional and operational health data warehouse as a requirement in order to store and retrieve relevant health data necessary for data analysis and dissemination.

To this end the Department of Health Planning and Statistics requested Technical Assistance for the creation of a data warehouse.

During 2008 and 2009 several consultancies were undertaken in order to design and assist with the creation of a data warehouse in Lesotho. This was completed successfully, but during the GDDS closing workshop the Lesotho delegation identified that they had a few hurdles which they could not overcome on their own. A follow-up mission was therefore conducted in South Africa with the technical specialist from Lesotho.

The specific objectives and deliverables for this mission were detailed in the Terms of Reference (refer to Annexure 2: Terms of Reference).

The basic objectives and deliverables for this mission are however stated below.

Specific issues to be covered during the consultancy were the following:

Objectives / activities of mission

- To finalize the data warehouse
- Transfer skills to Lesotho IT staff
- Create a technical manual for the data warehouse

Specific outputs for the consultancy were the following

- Report in detail on all relevant conditions identified.
- Health Statistics Data Warehouse
- Custom Developed Import Tools
- Data warehouse manual

2. IMPLEMENTATION OF THE CONSULTANCY

This follow-up mission was implemented by Mr. Gareth Daniell from **GeoSpace International**. Fifteen (15) work days were allocated for the mission, and ran from the 31st August 2009 to 15th September 2009.

3. ACKNOWLEDGEMENTS

The consultant would like to thank the Lesotho GDDS Coordinator, 'M'e Malehloa Molato and the head of the IT Department, Mr Mokoena Lesenyeho and Chief Statistician, Ms. Mahlape Ramoseme for their time and support during all the consultancies.

Special thanks Mr. Teboho Koma for his enthusiasm, assistance and positive attitude.

The consultant would also like to thank the General Data Dissemination Project of the World Bank for sponsoring the consultancy.

4. PROGRAM CONTEXT

With financial support from the Department for International Development (DFID) of the United Kingdom, the World Bank is implementing a project to assist 21 Anglophone Africa countries to participate in the General Data Dissemination System (GDDS). Participating countries are being assisted to participate in the GDDS through two separate, but linked projects both financed by DFID. The IMF is providing project management and technical support in the area of economic and financial statistics. The World Bank is providing technical support in the area of socio-demographic statistics.

Technical assistance is being provided through the World Bank to help countries implement plans for improvement in population, health, agriculture, labor market, justice and security, management of statistical systems, GIS and small area statistics. The GDDS framework developed by the IMF provides the framework for the detailed elaboration of long-term statistical development strategies. Participating countries have already expressed their requests for technical assistance and both the IMF and the World Bank have developed their assistance strategies.

Lesotho was one of the countries which asked for technical assistance in the field of Health Statistics.

5. LESOTHO DEPARTMENT OF HEALTH PLANNING AND STATISTICS (HPSD) DATA WAREHOUSE NEEDS

Information Technology is currently playing a crucial role in the creation of Statistical Health data in Lesotho. Information technology is used as a data capturing and analysis tool. The use of Information Technology as a data capturing and data creation tool mean that a vast amount of data is created over a very short space of time. The Department of Health and Statistics has been utilizing the current data capturing tools for more than a year and a lot of relevant data has therefore been created on daily, weekly and monthly basis. The data is stored and analyzed in separate database files and Microsoft Excel worksheets. The HPSD does not want the entire data capturing process or applications to be redesigned, but assistance is required assimilating this data into a central data store and then producing accurate and timely reports from this central store. A fully functional data warehouse is therefore a necessity for the Lesotho HPSD to successfully store and retrieve the data.

6. CHALLENGES FACING THE LESOTHO DEPARTMENT OF HEALTH PLANNING AND STATISTICS (HPSD)

The HPSD faced the following challenges regarding Health Statistics.

- The design and creation of an integrated and properly designed central database
- Creation of pre-determined data analysis structure
- Creation of data communication protocol
- Insufficient IT staff quota

The department was aware of these challenges and the request for technical assistance was the first step in addressing them.

Currently, the only major challenge still faced by the HPSD is the problem of insufficient staff.

7. OVERVIEW OF MEETINGS AND DISCUSSIONS

Please refer to Annexure 1: Overview of Meetings Held for the meetings and discussions that took place.

8. ASSESSMENT OF CURRENT SITUATION

A successful operational data warehouse consists of the following components:

- Data
- IT Infrastructure
 - Hardware
 - Software
 - Communication
- People
- Method

During earlier missions an assessment was done based on the above mentioned components, and a proposed data warehouse was designed.

Data

Relevant Statistical Health data is captured on a daily, weekly and monthly basis.

The following data is collected:

Inpatient.mdb

- Inpatient data
- Outpatient data
- Mental patient data

Dental.mdb

- Dental patient data

MaternalRegister.mdb

- Maternity patient data
- Antenatal Care patient data

TBQuarterly.mdb

- Tuberculosis patient data

HIVSummary.mdb

- HIV patient data

The data is captured at facility and hospital level on a daily basis by Nurses or doctors. A monthly summary sheet is then amalgamated from the daily tally-sheets by a clerk or nurse and submitted to the office of the health information officers (Statisticians) for the 10 different Districts. The District Health Information Officers use Microsoft Access to capture the data. Several front end applications were developed by the Lesotho HPSD to facilitate the capturing process. Different front end applications are used to capture the various data sets. The data is sent to the Statistics Unit at central level in Maseru on a monthly basis. Analysis is done at District level as well as at Headquarters.

IT Infrastructure

A full assessment could not be made of all 10 Districts as a site visit was only possible to one of the districts. According to verbal commitments from the MOHSW staff though, the other districts operate on a very similar basis to the one that was visited. Adequate hardware and software is in place for the data capturing process at District Level as well as head office. All the District Officers do have a workable computer, Microsoft Access 2003 with different front-end applications is used to capture the data. Most of the District Officers however do not have access to Email and Internet at the moment and the data is transferred to head-office through manual means.

The Department of Communications is in the process of rolling-out a Government Network that will connect all Government Departments in Maseru as well as the Districts. Internet Access will also be provided.

The department does not have any immediate plans in place to upgrade to the latest versions of Microsoft Access and Excel and the data warehouse will be implemented using Microsoft SQL Express.

People

An assessment of the current staff component could only be made at the head-office. The District Health Officers that capture the data on Microsoft Access have been trained by Head Office and are constantly given additional training and updates on their performance when they deliver the data to Head office. The capturing of new data, maintenance of the data as well as the analysis of the data does not seem to be a problem at District Level or at Head Office. The current IT specialists do have an abundance of skills. The existing front end applications were developed and are currently being updated and maintained by them. The biggest problem regarding IT support is a shortage of staff. Only four IT specialists are currently employed, one of which is at management level and are therefore not able to commit his time to day to day IT support.

Method

Method requires that once the people, hardware, software and data is in place, the following must be implemented or should exist for the warehouse to achieve its full potential and provide relevant deliverables:

- In the short term, method will imply the specific data creation, collection and warehousing strategy
- In the long term, method will also include the specific data analysis, data dissemination and data access strategies and protocols

As the purpose of this mission is to implement a data warehouse, very little of the method is currently in place.

9. FUNCTIONAL SPECIFICATION

Please refer to *Annexure 3: Functional Specification* for the full specification

Please refer to *Annexure 4: Source Code and Technical Manual* for a technical discussion of how the data warehouse is put together and which sections of source code requires amendment should the data sources change. This document may also be used by other countries electing to make use of the Lesotho data warehouse source code as a framework for their own data warehouse.

9.1 Introduction

The current data and workflow processes did not allow for efficient and effective data and statistics to be distributed by the HPSD.

The various districts each collect their own data and the main objective of the data warehouse is to collate and store this data in a central repository at the HPSD. It is important to note that some of the source databases are in a current state of development and maintenance. Therefore instead of a typically rigid “star design” for the data warehouse, a more flexible “snowflake design” was used.

Star vs. Snowflake schema

In both the star and snowflake schema the focus is on the speedy retrieval of data for reporting and analysis purposes rather than focusing on the efficiency of data manipulation.

A typical star design consists of a single fact table and several dimension tables with just a single relationship joining the main fact table with the dimension table. The star schema is especially suited for speedy retrieval of data since there are less “joins” required when writing the query that retrieves the data

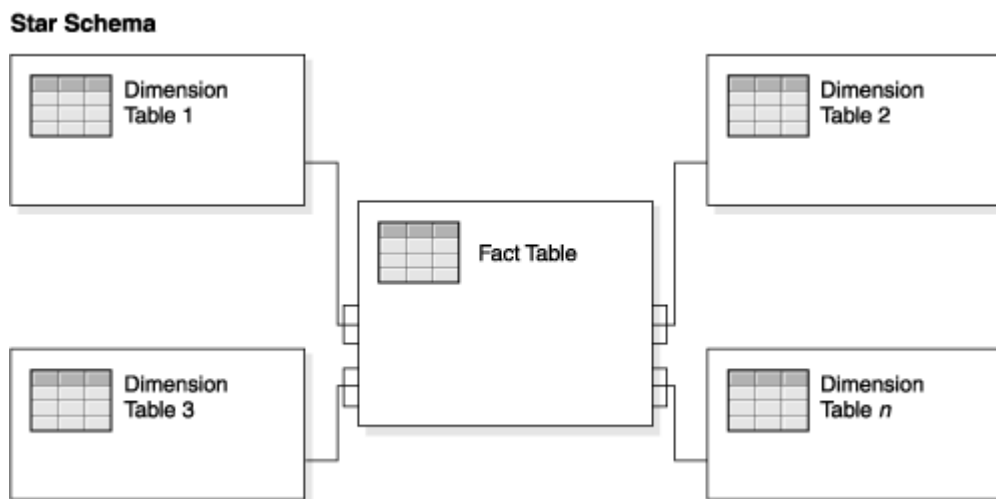


Figure 1 A Typical Star Schema

A snowflake design on the other hand still has a fact table as the central focus, but several levels of dimension tables are implemented and is more reminiscent of a normalized data base. Normalization splits up data to avoid redundancy (duplication) by moving commonly repeating groups of data into a new table. Normalization therefore tends to increase the number of tables that need to be joined in order to perform a given query, but reduces the space required to hold the data and the number of places where it needs to be updated if the data changes.

Snowflake Schema

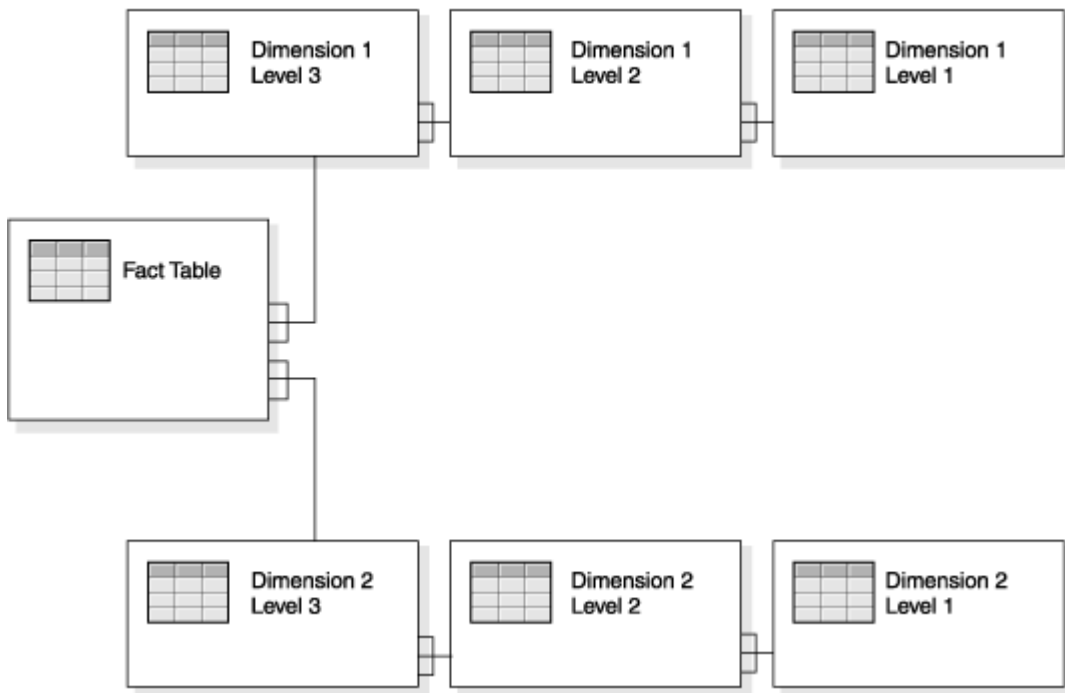


Figure 2 A Typical Snowflake Schema

The final design is based on the following constraints:

- Use the current data capturing tools as source
- Use open source and/or freeware as far as possible

A successful implementation of a data warehouse results in the following goals being achieved:

- Improve quality of data across all districts
- Eliminate inconsistent reports and reporting methods between districts.
- Provide capability for data sharing.
- Make aggregated data available in report format to interested parties for further analysis and/or integration.

9.2 Solution Overview

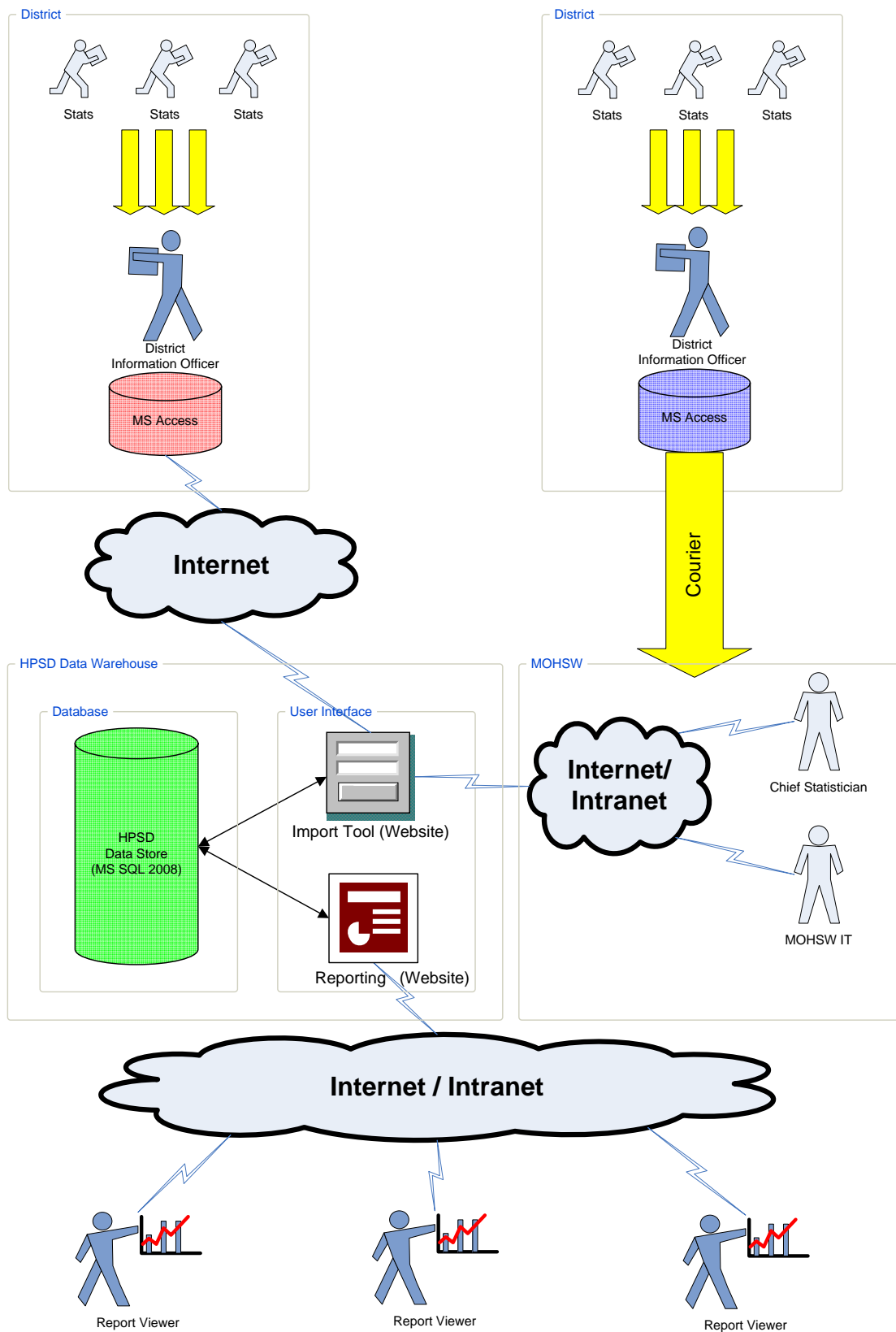


Figure 3 - Data Warehouse User Overview

User / Role Player	Description
Nurse or Clerk	<p>These users won't interact with the data warehouse directly. However, the process starts with them, and for this purpose they are included in the overview for completeness.</p> <p>This person is responsible for capturing the relevant stats on tally-sheets on a regular basis and passing it on to the District Information Officer.</p>
District Information Officer	<p>The District Information Officer receives a summary of the monthly data. This data is captured into the relevant MS Access database by means of a front-end application.</p> <p>Once completed the District Information Officer uploads the data for the attention of the Chief Statistician at the MOHSW Head Office.</p> <p>Where districts don't have Internet access, the data is transported by motor vehicle on a disk. The disk is given to the IT Officer who then uploads the MS Access database via the import tool on behalf of the relevant district.</p>
Chief Statistician	<p>The statistician responsible for receiving the data analyses the database for correctness, and requests amendments if not satisfied.</p> <p>If satisfied, the database is approved (via the Import Tool).</p> <p>Once approved, the IT Officer is able to affect the import (via the Import Tool).</p>
IT Officer	<p>The IT Officer uploads the MS Access database into the data warehouse by means of the import tool. If the Chief statistician has not yet approved the database, the upload option is not available to the IT Officer.</p> <p>Access to this functionality is limited to certain users only. Typically these users would be IT officers at the MOHSW.</p>
Report Viewer / End User	<p>Anybody that has been granted access to the reports (potentially including other role players).</p> <p>Access to reports is controlled via a username and password. Based on the username, the report viewer may have access to report on 1 or more districts' data.</p>

Table 1 – Data warehouse users

9.3 User Interface

9.3.1 Overview

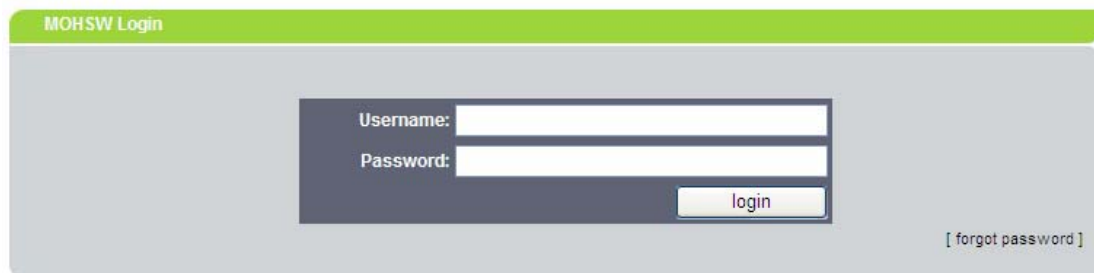
Data warehouse users interact with the data warehouse by means of a website that is hosted at the MOHSW. This website may be on the same server as the data warehouse itself or on a separate web server.

The web site prompts the user for a username and password, which will be used to identify if the user is a district-level user or MOHSW user.

9.3.2 Import Tool

Ideally the district-level data is uploaded to the MOHSW via a web site. When the district does not have Internet access, the data is transported to the MOHSW by disk and a MOHSW IT Officer uploads the data on behalf of the district.

Access to the web site is controlled via a username and password. Upon logging in to the system, the relevant security permissions is determined and access to screens and functions is controlled based on the user's assigned roles and rights.



The image shows a login form titled "MOHSW Login". It features a dark grey background with a light green header. The form contains two input fields: "Username:" and "Password:". To the right of the "Password:" field is a "login" button. Below the "login" button is a link that says "[forgot password]".

Figure 4 – Import Tool login

District level users are only able to see the data files they have already submitted to the MOHSW and upload new ones. If a file has not yet been imported into the data warehouse, the district-level user is allowed to delete it and replace it with another file.

Statistics Data Files

File Description	File Type	Date Uploaded	Delete
May 2009	InPatient	2 Jun, 2009 09h34	Data Already Imported
June 2009	Dental	3 Jul, 2009 08h12	Data Already Imported
July 2009	HIV	1 Aug, 2009 09h31	<input type="button" value="delete"/>
August 2009	TB	6 Sep, 2009 08h12	<input type="button" value="delete"/>

Upload New File		
File Description	File Type	File
<input type="text"/>	Dental <input type="button" value="v"/>	<input type="text"/> <input type="button" value="Browse..."/> <input type="button" value="Upload"/>

Figure 5 – Import Tool user interface for district-level users

MOHSW users are able to see all districts' uploads as well as action the import. In addition, the Chief Statistician is able to download the file for verification before importing it. Once a data file has been imported into the MOHSW data warehouse, the district user will no longer be able to delete it.

Statistics Data Files

District	File Description	File Type	Date Uploaded	Download	Verify	Delete	Import
Maseru	May 2009	Dental	2 Jun, 2009 09h34	<input type="button" value="download"/>	Verified	Data Already Imported	Data Already Imported
Maseru	June 2009	Dental	3 Jul, 2009 08h12	<input type="button" value="download"/>	Verified	Data Already Imported	Data Already Imported
Maseru	July 2009	Inpatient	1 Aug, 2009 09h31	<input type="button" value="download"/>	<input type="button" value="verify"/>	<input type="button" value="delete"/>	<input type="checkbox"/>
Maseru	August 2009	OutPatient	6 Sep, 2009 08h12	<input type="button" value="download"/>	<input type="button" value="verify"/>	<input type="button" value="delete"/>	<input type="checkbox"/>
Mafeteng	May 2009	Dental	2 Jun, 2009 09h34	<input type="button" value="download"/>	Verified	Data Already Imported	Data Already Imported
Mafeteng	June 2009	HIV	3 Jul, 2009 08h12	<input type="button" value="download"/>	Verified	Data Already Imported	Data Already Imported
Mafeteng	July 2009	TB	1 Aug, 2009 09h31	<input type="button" value="download"/>	<input type="button" value="verify"/>	<input type="button" value="delete"/>	<input type="checkbox"/>
Mafeteng	August 2009	Inpatient	6 Sep, 2009 08h12	<input type="button" value="download"/>	<input type="button" value="verify"/>	<input type="button" value="delete"/>	<input type="checkbox"/>
							<input type="button" value="Import"/>
Upload New File							
District	File Description	File Type	File				
Maseru	<input type="text"/>	Dental	<input type="text"/>	<input type="button" value="Browse..."/>	<input type="button" value="Upload"/>		

Figure 6 – Import Tool user interface for MOHSW users

9.3.3 Reporting

Along with uploading/importing data files, users are also able to generate and view reports from the data warehouse user interface.

District-level users are only able to view reports that relate to their data, while MOHSW users are able to view reports for the entire Lesotho.

10. ACTIVITIES & FURTHER RECOMMENDATIONS

The following short term and long term activities are recommended for the operation of a data warehouse.

10.1 Short term activities

The main goal for all the short term activities is the establishment of a data warehouse at the Head office to enable the Department to store, update, query, retrieve and disseminate the data. The establishment of the data warehouse consists of the following activities:

- Project management
- User Analysis or Scoping of requirements
- Functional Specification
- System Design
- Implementation
- Installation
- Testing
- Training

Project Management

Completed

User Analysis & Functional specification

Completed – refer to *Annexure 3: Functional Specification*

System Design, Implementation, Installation, Testing and Delivery

Completed – refer to *Annexure 4: Source Code and Technical Manual*

10.2 Further Recommendations

10.2.1 Staff Training

The IT department has been strongly advised to invest in formal training in order to improve the skills of their IT staff. This is in the best interest of the long-term success of the data warehouse.

10.3 Potential Future Actions

Once the issues in 10.2 (Further Recommendations) and 11 (RISKS) have been addressed the success and future of the data warehouse will be in very good standing.

As per the initial report by Mr. E. de Fortier of August 2008, the HPSD then has the option of extending the data warehouse to introduce some or all of the following actions.

Rolling out of data warehouse at District Level

Microsoft SQL Express can be deployed at District Level to import the information captured in Microsoft Access and export the merged data for analysis purposes. Every district will therefore have a separate data warehouse that is updated with the latest Microsoft Access captured data on a monthly basis. The different district Microsoft SQL Express data warehouses will then be sent to the head office to be imported into the central level Statistics Unit data warehouse based on Microsoft SQL Server in Maseru.

Estimated Duration for Development: ***5 days per District***

Front end data capturing application

The existing front end application based on Microsoft Access can be changed so that the data is captured into a central Microsoft SQL Express data store at each district. The front end application design should incorporate future plans like data capturing via intra and internet. It would be advised if this action is to be considered, that the front-end applications be re-written so that they are web-based and can be deployed over the Internet.

Estimated duration for front end application development: ***Several Weeks (in-house development and/or outsourced assistance)***

Ongoing Training

Extra training in the new data warehouse, front end data capturing application and data analysis at District level is essential.

The IT Personnel together with the relevant personnel of the Ministry of Health and Social Welfare must ensure that all District staff members are well versed in the use of the new data warehouse, front end data capturing application and data analysis software.

Estimated duration for Training: ***5 days (continuous process)***.

Protocols

As with any data set, pre-determined protocols regarding data access need to be established. This is very important especially when the data warehouse is made available first via the Departments own network and intranet but also later on the internet.

Estimated duration for Protocol Setup: ***5 days (continuous process)***.

11. RISKS

There are certain critical issues which needs to be addressed for the data warehouse to reach its full potential, if they are not addressed the data warehouse faces the risk of failure. These are:

11.1 Source databases and applications

The source databases and corresponding front-end applications are not designed according to best practices. They are not correctly normalized and some databases include redundant tables etc. There is a risk of data inconsistency once imported. However, the risk of data inconsistency appears to be minor since the users are well trained in preventing bad data being captured. There is also a chief statistician in the process double-checking the data before it's imported.

Many of the source databases are still "works in progress". Most are however either very close to their final design or haven't changed in a long time. The TB data capturing system has already been redesigned though. The old TB database was used for the specification of the data warehouse as the user says that just the front-end capturing tool is changing to .Net and the DB is remaining the same.

11.2 IT department

The IT department support staff quota is not optimal at the moment. The IT software specialists currently employed by the Department do have the required skills necessary for the creation of the data warehouse; however the lack of sufficient IT resources poses a serious risk for the maintenance and support of the data warehouse. The data warehouse creation activities required the services of a dedicated IT specialist for the duration of the project. This was not the case, and the implementation and installation of the data warehouse fell behind schedule.

The IT support shortage could be solved with the use of external assistance, however it is not the ideal solution and the Department should ideally look to solve the problem in-house. One of the MOHSW's goals from the start is that they gain the necessary skills to maintain and enhance the data warehouse themselves, and this will not be the case if the development is outsourced completely. The creation of a data warehouse with the added functionality is currently putting extra strain on the IT department though.

The maintenance and support of the data warehouses at District Level and at Headquarters is of utmost importance. The central data warehouse as well as the subsequent analysis and reporting depends on the constant updating of the warehouses with new data from the Districts. Maintenance and support is therefore not limited to headquarters only, but must also be provided at District level. The potential lack of IT support poses a serious risk to the success of the data warehouse in the short term but also to the successful upkeep of the warehouse in the long run. The shortage of IT support staff therefore poses a short term as well as a long term risk for the successful creation and maintenance of the data warehouse.

The Department is well aware of the IT staff shortage. Sustainable plans must now be put in place in order for the risk to me minimized.

11.3 Training

The recommendation for the creation of a data warehouse and future development at District and headquarter level is based on a phased approach. New technology and method will be deployed at District level. The District Health Information officers are responsible for the capturing, analysis and reporting at District Level. Training of the District Health Information Officer responsible for the data capturing, analysis and reporting is thus very important to ensure that they keep up with the changing data warehouse technology and method. The IT staff also need be constantly trained on new technology and developments in order to ensure that the data warehouse and the related functionality are updated accordingly. A training program should be developed as part of the data warehouse roll-out.

12. TRAVEL AND ACCOMMODATION ARRANGEMENTS

There were no problems regarding the travel and accommodation arrangements.

13. FINAL REFLECTIONS

The HPSD made sure that, through all their efforts, the consultancies could be completed as effectively as possible. The challenge is to ensure that issues raised in 10.2 (Further Recommendations) and 11 (RISKS) are addressed and any long term plans for the HPSD data warehouse are set in motion. The professionalism in all aspects of their work is noteworthy and the future of Lesotho Health is in very capable hands.

14. ANNEXURES

14.1 Annexure 1: Overview of Meetings Held

When	7 th September
Attendees	Mr. T. Koma - IT Specialist
Overview	Introduction and discuss general mission objectives

Between the 7th and 11th September, Mr T. Koma shadowed the consultant in all aspects of the finalization of the data warehouse and although no official meetings were held, every day took on the form of an informal meeting or workshop.

When	11 th September
Attendees	Mr. T. Koma - IT Specialist
Overview	Review of objectives and final debriefing

14.2 Annexure 2: Terms of Reference

Refer to:

- *TOR_Lesotho_Geospace_Health_Mission5.doc*

14.3 Annexure 3: Functional Specification

Refer to:

- *Lesotho_Health_Data_Warehouse_Functional_Specification_Apr09.doc*

14.4 Annexure 4: Source Code and Technical Manual

Refer to:

- *Lesotho_Health_Data_Warehouse_Technical_Manual_Sep09.doc*

General Data Dissemination System, (GDDS phase 2)

Socio-Demographic Statistics Project for Anglophone Africa

Provision of Technical Assistance as the expert for:

Module: Health Statistics to Lesotho, Central Statistics Office

Date: July 2009

Background

With financial support from the Department for International Development (DFID) of the United Kingdom, the World Bank is implementing a project to assist 21 Anglophone Africa countries to participate in the General Data Dissemination System (GDDS). Participating countries are being assisted to participate in the GDDS through two separate, but linked projects both financed by DFID. The IMF is providing project management and technical support in the area of economic and financial statistics. The World Bank is providing technical support in the area of socio-demographic statistics.

Technical Assistance (TA)

Technical assistance is being provided through the World Bank to help countries implement plans for improvement in population, health, agriculture, labor market, justice and security, management of statistical systems, GIS and small area statistics. The GDDS framework developed by the IMF provides the framework for the detailed elaboration of long-term statistical development strategies. Participating countries have already expressed their requests for technical assistance and both the IMF and the World Bank have developed their assistance strategies.

Lesotho was one of the countries which asked for technical assistance in the field of Health Statistics

Terms of Reference

Background

Lesotho attended the GDDS 2 Launch Workshop on Health Statistics in **Gaborone, Botswana, in October 2007** where they drew up their Country Work Plan regarding the deliverance of three TA missions covering the country identified priorities. These priorities are part of the Work Plan Structure document which acts as a living document for the duration of the technical assistance and serves as an information base from which the TOR for every mission can be drawn up.

The general objective is to have GDDS assistance in developing their system for Health Statistics (JSS). Their country team would do this in collaboration with the statistical expert.

Purpose of the assignment

The purpose of this assignment would be to complete the fifth technical assistance mission for Lesotho in order to finalize the data warehouse and all necessary skill transfer.

	Lesotho	Remarks
Priority 1	Development of a District data warehouse	See annex
Priority 2	Skill transfer to the Lesotho IT staff	See annex
Priority 3	Creation of a data warehouse manual	See annex

This mission will deal with Priority 1, 2 and 3 as follows:

Total consultant time for the mission is planned to be 15 days divided as follow:

- 10 days actual mission time
- 3 days creation of data warehouse time
- 2 day report writing

Main Tasks to be performed

The fifth mission will deal with priority 1 and 2 in the following way:

This mission is the fifth and final mission out of a series of several missions.

1. System design, Implementation, testing and training

The User Analysis and Functional Specification has been completed in previous missions. The fifth and final mission would be to ensure that the system design, implementation, testing and training is completed in full. This entails the following sub functions:

- Ensure that all the necessary import tools and relationships required to import the transactional data into the warehouse data are completed
- Design, create and implement a reporting framework
- Implement Data warehouse
- Implement Reporting framework
- Testing
- Training

The Lesotho Department of Health IT technician will receive hands on training in the use, maintenance and future developments of the Data Warehouse to ensure continuity.

The fifth mission will deal with priority 3 in the following way:

2. Data warehouse manual

A general manual will be designed in the general specification, design and use of the data warehouse. 2 Days for a general manual and 1 day for a country specific manual.

3. Draft documentation.

The main topics discussed should be documented. Also a system of documentation for this project needs to be designed.

Deliverables for this mission:

- Agenda with a list of meetings and activities
- Mission report for the file of the country on the discussions held in the appropriate format.
- Mission report for the World Bank according to the format provided, with annexes

Skill requirements

The consultant would need relevant experience of understanding the IT needs to design a data ware house approach for Health Statistics and its use for policy research, and understanding the African context.

Communications:

The expert will meet at the start of the visit with the Head of the organization and GDDS coordinator, if possible, and you will report briefly to them at the end of the visit.

Duration

As noted, the total consultant time for the mission is 15 days with 10 days actual mission time, 3 days data warehouse manual creation and 2 days report writing.

Timing

The fifth consultancy for Lesotho would be completed by the end of September 2009.

Annexes: Context information.

A: Work plan for Lesotho in the field of Health Statistics.

B: Main activities to be considered during the discussions of the framework, possibly in relation with the IMF DQAF.

Annex A

Work and Action Plan Priority

Country: Lesotho

Priority 1 – Development of District data Warehouse

1. Problem being addressed

- Unavailability of district data warehouse

1. Strategic Objectives

- To build a data warehouse that will contain data from all data sources to facilitate data sharing and easy retrieval of data by users from all corners

2. Activities required

- Technical Protocol for establishing data warehouse – software, hardware and contents
- Train relevant maintenance personnel
- Standardization and connectivity

3. Input required international

- Technical assistance

4. Own preparations required

- Ensure that all stakeholders are informed of the meeting
- Compile all the documents needed

5. Output planned

- Consensus on the kind of information to be included and where the warehouse will be housed (ministerial)

6. Changes anticipated

- Increased demand and use of information

7. Linking with further activities

- This will assist in the use of data by different stakeholders without the hassle of going to different departments/ministries to get the information

8. Other donors supporting this topic

- WHO/HMN, WB, Irish Aid, MCC

9. Timing

- September 2009



Functional Specification

Lesotho Health Data Warehouse

Prepared by:

Gareth Daniell

Prepared for:

**The Lesotho Department of Health Planning and Statistics,
Lesotho Bureau of Statistics and the World Bank, General
Data Dissemination System, Socio-Demographic Statistics
Project for Anglophone Africa**

November 2008

Table of Contents

1. INTRODUCTION	4
1.1 Purpose And Objectives	4
1.2 Who should read this document	4
1.3 Related Documents	4
1.4 Other	4
2. USER ANALYSIS	5
3. DATA WAREHOUSE DETAIL	6
3.1 Table Definitions	6
3.1.1 Facility	6
3.1.1.1 Agency Table	6
3.1.1.2 Area Table	6
3.1.1.3 Facility Table	7
3.1.2 ANC	7
3.1.2.1 ANCRRegister Table	8
3.1.3 Dental	10
3.1.3.1 DentalDiagnosis Table	11
3.1.3.2 DentalProcedure Table	11
3.1.3.3 DentalRegister Table	11
3.1.4 Inpatient	12
3.1.4.1 InpatientDiagnosis Table	13
3.1.4.2 InpatientRegister Table	13
3.1.5 Maternity	14
3.1.5.1 DeliveryChild Table	15
3.1.5.2 DeliveryRegister Table	15
3.1.6 Mental	17
3.1.6.1 MentalStage Table	17
3.1.6.2 MentalCondition Table	17
3.1.6.3 MentalRegister Table	18
3.1.7 Outpatient	19
3.1.7.1 OutpatientCondition Table	19
3.1.7.2 OutpatientRegister Table	20
3.1.8 TB Details	21
3.1.8.1 TBLaboratoryActivites Table	21
3.1.8.2 TBDetails Table	22
3.1.8.3 TBHIVActivitiesTBName Table	22
3.1.8.4 TBHIVActivities Table	22
3.1.9 TB Outcome	23
3.1.9.1 TBTreatmentOutcomeCaseType Table	24
3.1.9.2 TBTreatmentOutcome Table	24
3.1.9.3 TBTreatmentOutcomeHIVActivitiesTable	25
3.1.10 TB Sputum	26
3.1.10.1 TBSputumCohnversion Table	26
4. REPORTING	27

1. INTRODUCTION

1.1 Purpose And Objectives

As part of the drive to create accurate and accessible Statistical Health Data, the Lesotho Department of Health Planning and Statistics requires a central store for all data collected at the district level.

The purpose of this document is therefore to provide, in detail, the layout and definitions of the tables, joins, look-up tables etc. required for the design and implementation of a data warehouse.

1.2 Who should read this document

IT Managers and Software developers responsible for the implementation of the data warehouse and corresponding import tools.

1.3 Related Documents

Terms of Reference:

TOR_Lesotho_Geospace_Health_DWS1.doc

Technical Proposal:

TECHNICAL PROPOSAL_GDDS2_SECOND_MISSION_LESOTHO_HEALTH_GEOSPACE.pdf

1.4 Other

It is important to note that some of the source databases are in a current state of development and maintenance. Changes to these source databases will possibly result in changes to the design of the data warehouse. Therefore instead of a typically rigid "star design" for the data warehouse, a more flexible "snowflake design" was used.

It is also anticipated that Mother Child Health (MCH) data (currently in a FoxPro format) may also be added to the data warehouse at a later date. The data warehouse is therefore designed with a bit of flexibility in mind rather than focusing purely on performance.

If the MS SQL Server license that the department possesses at the moment is not sufficient, IT is recommended that SQL Express be used for the implementation of the data warehouse. This document however does not require a specific database though, and options such as MySQL may be evaluated as well for the implementation.

2. USER ANALYSIS

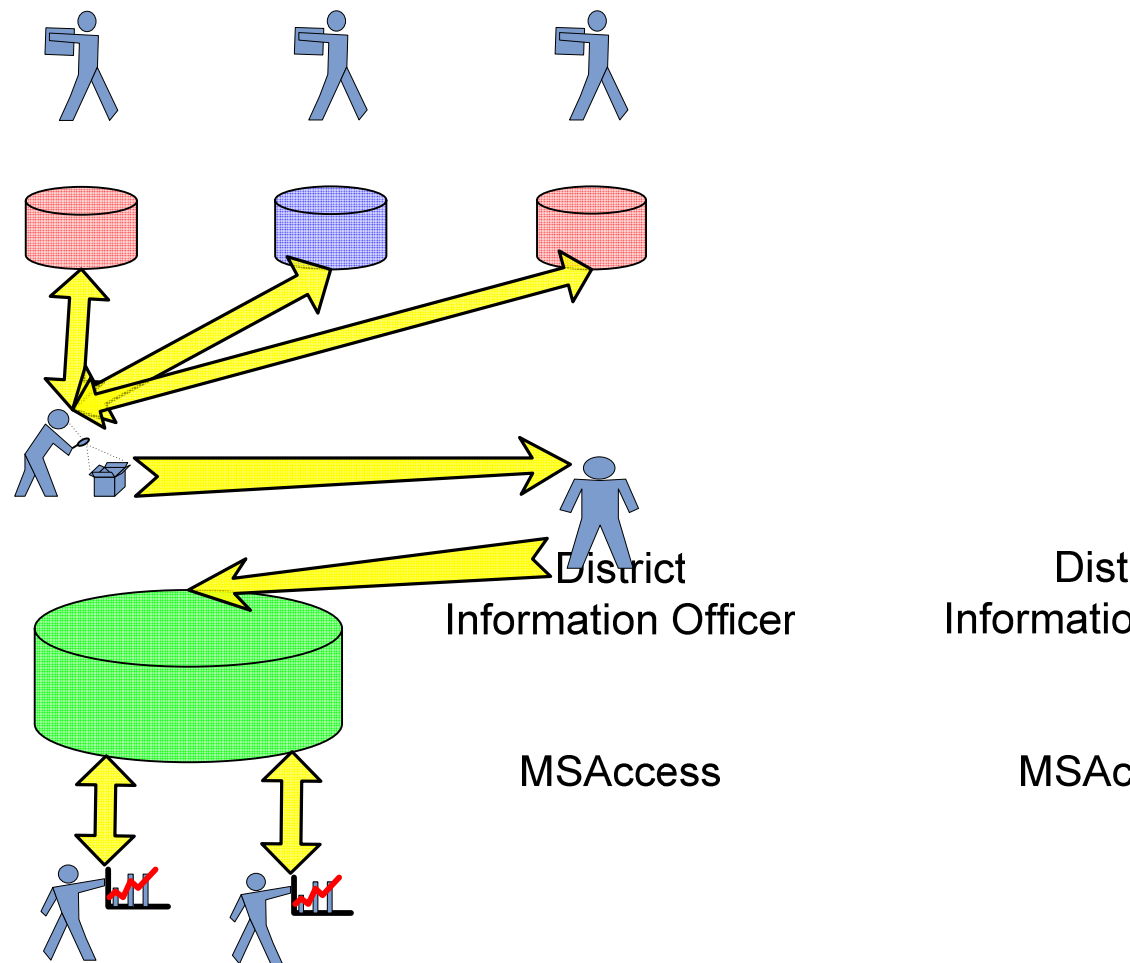


Figure 1 - Data Warehouse User Overview

User / Role Player	Description
District Information Officer	The District Information Officers receive raw data on hand written forms. This data is captured into the relevant MS Access database by means of a front-end application. Once completed the District Information Officer takes the data to the Chief Statistician in at Head Office. Since most districts don't have Internet access yet, the data is transported by motor vehicle on a disk.
Chief Statistician	The statistician responsible for receiving the data analyses the database for correctness, and requests amendments if not satisfied. If satisfied the databases are given to the IT Officer.
IT Officer	The IT Officer uploads the database into the data warehouse by means of an importing tool.
Report Viewer / End User	Anybody with access to the reports, including other role players.

Data Store

3. DATA WAREHOUSE DETAIL

3.1 Table Definitions

3.1.1 Facility

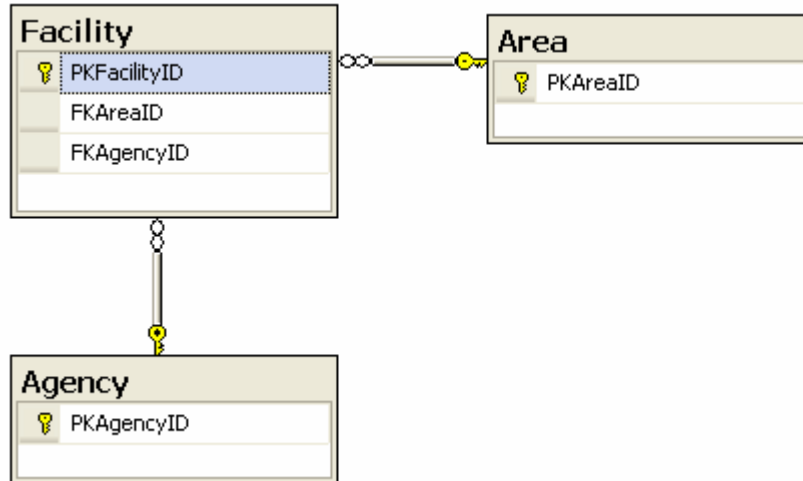


Figure 2 – Facility Data Relationships

3.1.1.1 Agency Table

Columns

Column Name	Data Type	Allow Nulls
PKAgencyID	int	False
Code	varchar(50)	False
Description	varchar(255)	False

Indexes

Name	Clustered	Columns
IX_Agency	False	Code
PK_Agency	True	PKAgencyID

3.1.1.2 Area Table

Columns

Column Name	Data Type	Allow Nulls
PKAreaID	int	False
Code	varchar(50)	False
Description	varchar(255)	False

Indexes

Name	Clustered	Columns
IX_Area	False	Code

PK_Area	True	PKAreaID
---------	------	----------

3.1.1.3 Facility Table

Columns

Column Name	Data Type	Allow Nulls
PKFacilityID	int	False
FKAreaID	int	True
FKAgencyID	int	True
Code	varchar(50)	True
FacilityName	varchar(255)	True
District	varchar(255)	True

Indexes

Name	Clustered	Columns
IX_Facility	False	FKAreaID
IX_Facility_1	False	FKAgencyID
IX_Facility_2	False	Code
IX_Facility_3	False	District
PK_Facility_1	True	PKFacilityID

Foreign Key Constraints

Name	Columns	Referenced Table	Referenced Columns
FK_Facility_Agency	FKAgencyID	Agency	PKAgencyID
FK_Facility_Area	FKAreaID	Area	PKAreaID

3.1.2 ANC

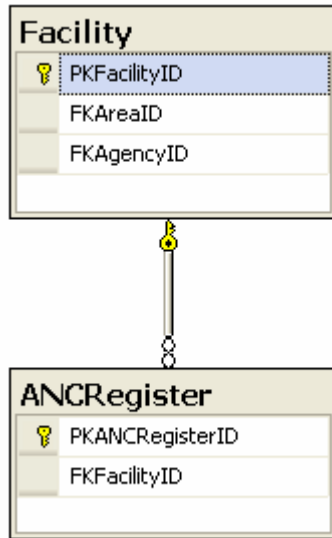


Figure 3 – ANC Data Relationships

See 3.1.1.3 for Facility table definition

3.1.2.1 ANCRRegister Table

Columns

Column Name	Data Type	Allow Nulls
PKANCRRegisterID	int	False
FKFacilityID	int	False
RegisterUser	varchar(255)	True
ReportMonth	smallint	False
ReportYear	smallint	False
field1	int	True
field2	int	True
field3	int	True
field4	int	True
field5	int	True
field6	int	True
field7	int	True
field8	int	True
field9	int	True
field10	int	True
field11	int	True
field12	int	True
field13	int	True
field14	int	True

field15	int	True
field16	int	True
field17	int	True
field18	int	True
field19	int	True
field20	int	True
field21	int	True
field22	int	True
field23	int	True
field24	int	True
field25	int	True
field26	int	True
field27	int	True
field28	int	True
field29	int	True
field30	int	True
field31	int	True
field32	int	True
field33	int	True
field34	int	True
field35	int	True
field36	int	True
field37	int	True
field38	int	True
field39	int	True
field40	int	True
field41	int	True
field42	int	True
field43	int	True
field44	int	True
field45	int	True
field46	int	True
field47	int	True
field48	int	True
field49	int	True

field50	int	True
field51	int	True
field52	int	True
field53	int	True
field54	int	True
field55	int	True
field56	int	True
field57	int	True
field58	int	True
field59	int	True
field60	int	True
field61	int	True
field62	int	True
field63	int	True
field64	int	True

Indexes

Name	Clustered	Columns
IX_ANCRegister	True	FKFacilityID
IX_ANCRegister_1	False	ReportMonth
IX_ANCRegister_2	False	ReportYear
PK_ANCRegister	False	PKANCRegisterID

Foreign Key Constraints

Name	Columns	Referenced Table	Referenced Columns
FK_ANCRegister_Facility	FKFacilityID	Facility	PKFacilityID

3.1.3 Dental

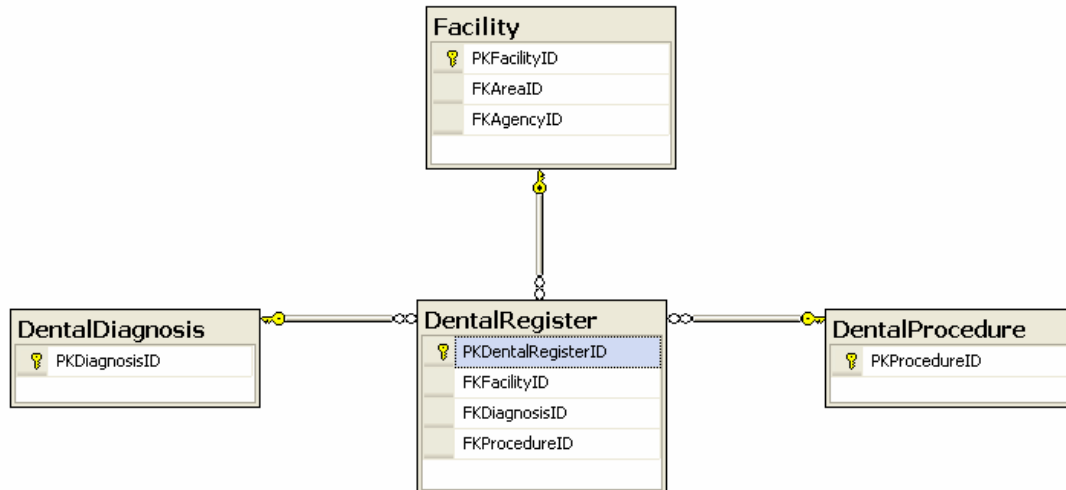


Figure 4 - Dental Data Relationships

See 3.1.1.3 for Facility table definition

3.1.3.1 DentalDiagnosis Table

Columns

Column Name	Data Type	Allow Nulls
PKDiagnosisID	int	False
Description	varchar(255)	True

Indexes

Name	Clustered	Columns
PK_DentalDiagnosis	True	PKDiagnosisID

3.1.3.2 DentalProcedure Table

Columns

Column Name	Data Type	Allow Nulls
PKProcedureID	int	False
Description	varchar(255)	True

Indexes

Name	Clustered	Columns
PK_DentalProcedure	True	PKProcedureID

3.1.3.3 DentalRegister Table

Columns

Column Name	Data	Allow
-------------	------	-------

	Type	Nulls
PKDentalRegisterID	int	False
FKFacilityID	int	False
RegisterUser	varchar(255)	True
FormNumber	int	True
Month	smallint	True
Year	smallint	True
Date	datetime	True
Age	float	True
Sex	char(1)	True
FKDiagnosisID	int	True
FKProcedureID	int	True
Repeat	bit	True

Indexes

Name	Clustered	Columns
IX_DentalRegister	True	FKFacilityID
IX_DentalRegister_1	False	Month
IX_DentalRegister_2	False	Year
IX_DentalRegister_3	False	Date
IX_DentalRegister_4	False	FKDiagnosisID
IX_DentalRegister_5	False	FKProcedureID
PK_DentalRegister	False	PKDentalRegisterID

Foreign Key Constraints

Name	Columns	Referenced Table	Referenced Columns
FK_DentalRegister_DentalDiagnosis	FKDiagnosisID	DentalDiagnosis	PKDiagnosisID
FK_DentalRegister_DentalProcedure	FKProcedureID	DentalProcedure	PKProcedureID
FK_DentalRegister_Facility	FKFacilityID	Facility	PKFacilityID

3.1.4 Inpatient

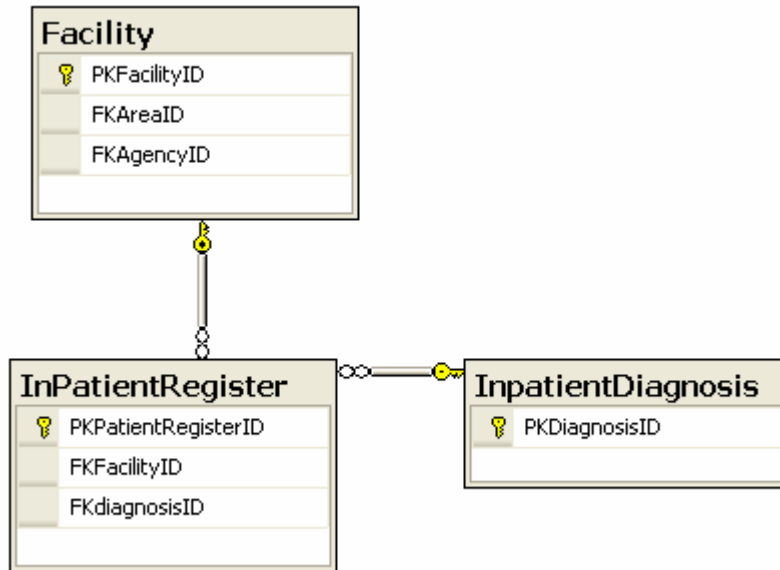


Figure 5 – Inpatient Data Relationships

See 3.1.1.3 for Facility table definition

3.1.4.1 InpatientDiagnosis Table

Columns

Column Name	Data Type	Allow Nulls
PKDiagnosisID	int	False
Code	varchar(50)	True
Description	varchar(255)	True

Indexes

Name	Clustered	Columns
IX_InpatientDiagnosis	False	Code
PK_Diagnosis	True	PKDiagnosisID

3.1.4.2 InpatientRegister Table

Columns

Column Name	Data Type	Allow Nulls
PKPatientRegisterID	int	False
FKFacilityID	int	False
FormNumber	int	False
RegisterUser	varchar(255)	True
Ward	varchar(50)	True
AdmissionDate	datetime	True

DischargeDate	datetime	True
Sex	char(1)	True
Age	float	True
AgeModulo	varchar(50)	True
LSCitizen	bit	True
FKdiagnosisID	int	True
Outcome	smallint	True

Indexes

Name	Clustered	Columns
IX_InPatientRegister	False	AdmissionDate
IX_InPatientRegister_1	False	DischargeDate
IX_InPatientRegister_2	False	FKdiagnosisID
PK_InPatientRegister	True	PKPatientRegisterID

Foreign Key Constraints

Name	Columns	Referenced Table	Referenced Columns
FK_InPatientRegister_Facility	FKFacilityID	Facility	PKFacilityID
FK_InPatientRegister_InpatientDiagnosis	FKdiagnosisID	InpatientDiagnosis	PKDiagnosisID

3.1.5 Maternity

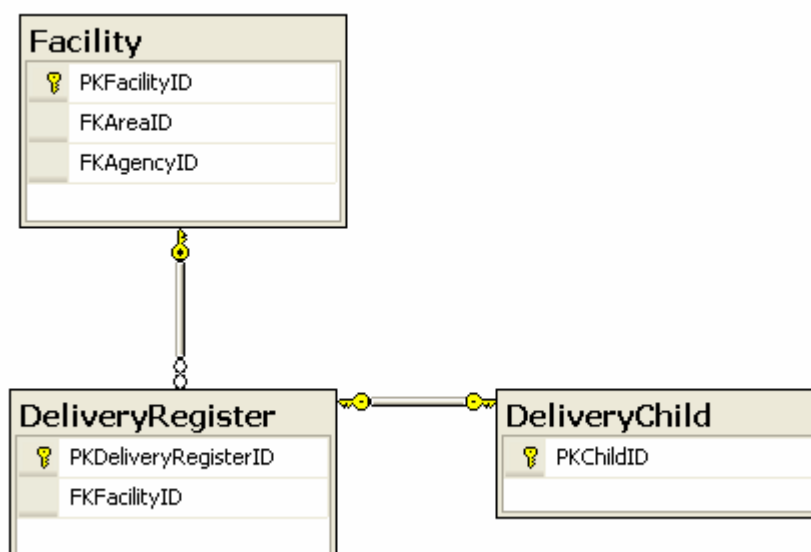


Figure 6 - Maternity Data Relationships

See 3.1.1.3 for Facility table definition

3.1.5.1 DeliveryChild Table

Columns

Column Name	Data Type	Allow Nulls
PKChildID	int	False
FKDeliveryRegisterID	int	False
APGAR1MIN	int	True
APGAR5MIN	int	True
Sex	varchar(50)	True
ChildWeight	float	True
NewbornMaturity	int	True
Malformations	varchar(50)	True
BirthStatus	varchar(50)	True
StillBirth	varchar(50)	True
arvPROPHGIVEN	int	True
arvPROPHDISCHGIVEN	varchar(50)	True
Immunization	int	True
feedingOptBirth	varchar(50)	True
InfantStatusDisch1	varchar(50)	True
InfantStatusDisch2	int	True

Indexes

Name	Clustered	Columns
IX_DeliveryChild	False	FKDeliveryRegisterID
PK_Child	True	PKChildID

3.1.5.2 DeliveryRegister Table

Columns

Column Name	Data Type	Allow Nulls
PKDeliveryRegisterID	int	False
FKFacilityID	int	False
RegisterUser	varchar(255)	True
FormNumber	int	False
FormIndex	int	False
ReportingDate	datetime	False
AdmissionDate	datetime	False

Age	int	False
ANCAttended	bit	False
Parity	int	False
GestationalAge	smallint	False
pmtctTestHIVSTATUS	varchar(50)	True
pmtctTestHAARTSTATUS	varchar(50)	True
pmtctPreTestMaternity	varchar(50)	True
pmtctTESTHIVMATERNITY	varchar(50)	True
pmtctTESTRESULTMART	varchar(50)	True
pmtctARVPROHPREG	int	True
pmtctARVPROPHB4DELIVERY	int	True
pmtctARVLABOR	int	True
pmtctARVPROPHNumHRS	varchar(50)	True
DeliveryDate	datetime	True
DeliveryMode	int	True
DeliveryNumFetus	int	True
MaternCI	varchar(255)	True
MaternDeathOccur	varchar(50)	True
MaternDeathCause	int	True
postPartumMotherStatusDisch	varchar(50)	True
postPartMotherRefferd	int	True
postPartMotherCouns	varchar(50)	True
postPartarvDisch	varchar(50)	True
dischargeDate	datetime	True
postPartDelivBy	varchar(50)	True

Indexes

Name	Clustered	Columns
IX_DeliveryRegister	True	FKFacilityID
IX_DeliveryRegister_1	False	ReportingDate
IX_DeliveryRegister_2	False	AdmissionDate
PK_DeliveryRegister	False	PKDeliveryRegisterID

Foreign Key Constraints

Name	Columns	Referenced Table	Referenced Columns
------	---------	------------------	--------------------

FK_DeliveryRegister_DeliveryChild	PKDeliveryRegisterID	DeliveryChild	PKChildID
FK_DeliveryRegister_Facility	FKFacilityID	Facility	PKFacilityID

3.1.6 Mental

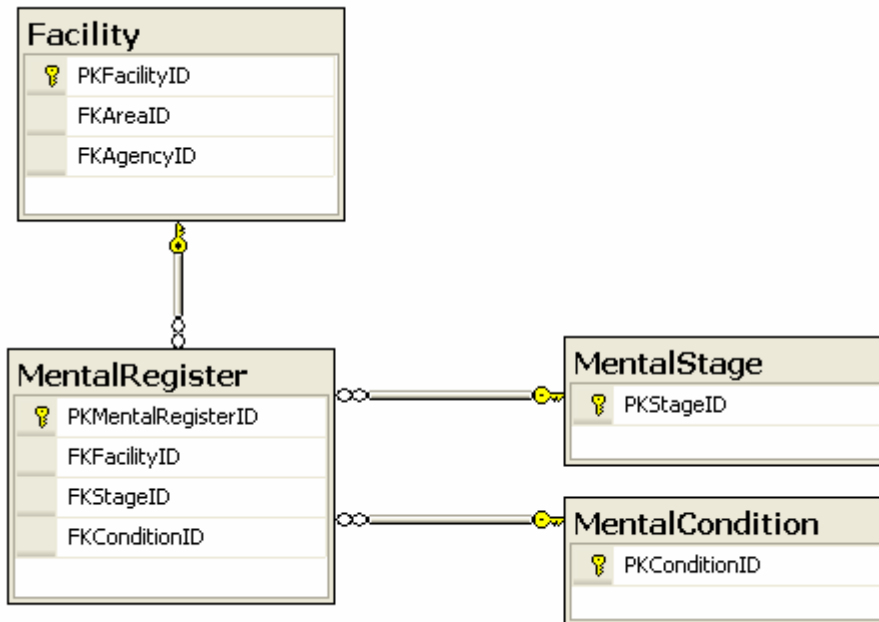


Figure 7 – Mental Data Relationships

See 3.1.1.3 for Facility table definition

3.1.6.1 MentalStage Table

Columns

Column Name	Data Type	Allow Nulls
PKStageID	int	False
Description	varchar(50)	False

Indexes

Name	Clustered	Columns
PK_MentalStage	True	PKStageID

3.1.6.2 MentalCondition Table

Columns

Column Name	Data Type	Allow Nulls
PKConditionID	int	False
Description	varchar(50)	False

Indexes

Name	Clustered	Columns
PK_MentalCondition	True	PKConditionID

3.1.6.3 MentalRegister Table

Columns

Column Name	Data Type	Allow Nulls
PKMentalRegisterID	int	False
FKFacilityID	int	False
RegisterUser	varchar(255)	True
FormNumber	int	False
Date	datetime	True
FKStageID	int	True
FKConditionID	int	True
M_Age_0_9	int	True
M_Age_10_19	int	True
M_Age_20_29	int	True
M_Age_30_39	int	True
M_Age_40_49	int	True
M_Age_50_59	int	True
M_Age_60_64	int	True
M_Age_65	int	True
F_Age_0_9	int	True
F_Age_10_19	int	True
F_Age_20_29	int	True
F_Age_30_39	int	True
F_Age_40_49	int	True
F_Age_50_59	int	True
F_Age_60_64	int	True
F_Age_65	int	True

Indexes

Name	Clustered	Columns
IX_MentalRegister	True	FKFacilityID
IX_MentalRegister_1	False	Date

IX_MentalRegister_2	False	FKStageID
IX_MentalRegister_3	False	FKConditionID
PK_MentalRegister	False	PKMentalRegisterID

Foreign Key Constraints

Name	Columns	Referenced Table	Referenced Columns
FK_MentalRegister_Facility	FKFacilityID	Facility	PKFacilityID
FK_MentalRegister_MentalCondition	FKConditionID	MentalCondition	PKConditionID
FK_MentalRegister_MentalStage	FKStageID	MentalStage	PKStageID

3.1.7 Outpatient

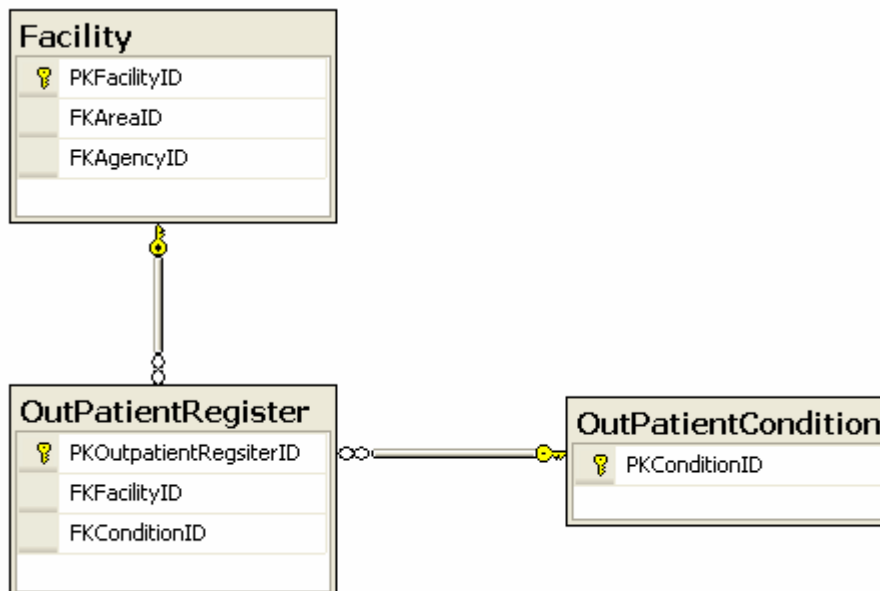


Figure 8 – Outpatient Data Relationships

See 3.1.1.3 for Facility table definition

3.1.7.1 OutpatientCondition Table

Columns

Column Name	Data Type	Allow Nulls
PKConditionID	int	False
Description	int	True

Indexes

Name	Clustered	Columns
PK_OutPatientCondition	True	PKConditionID

3.1.7.2 OutpatientRegister Table

Columns

Column Name	Data Type	Allow Nulls
PKOutpatientRegisterID	int	False
RegisterUser	varchar(255)	True
FKFacilityID	int	False
FormNumber	int	False
FKConditionID	int	True
M_Age_0_9	int	True
M_Age_10_19	int	True
M_Age_20_29	int	True
M_Age_30_39	int	True
M_Age_40_49	int	True
M_Age_50_59	int	True
M_Age_60_64	int	True
M_Age_65	int	True
F_Age_0_9	int	True
F_Age_10_19	int	True
F_Age_20_29	int	True
F_Age_30_39	int	True
F_Age_40_49	int	True
F_Age_50_59	int	True
F_Age_60_64	int	True
F_Age_65	int	True

Indexes

Name	Clustered	Columns
IX_OutPatientRegister	False	FKFacilityID
IX_OutPatientRegister_1	False	FKConditionID
PK_OutPatientRegister	False	PKOutpatientRegisterID

Foreign Key Constraints

Name	Columns	Referenced Table	Referenced Columns
FK_OutPatientRegister_Facility	FKFacilityID	Facility	PKFacilityID
FK_OutPatientRegister_OutPatientCondition	FKConditionID	OutPatientCondition	PKConditionID

3.1.8 TB Details

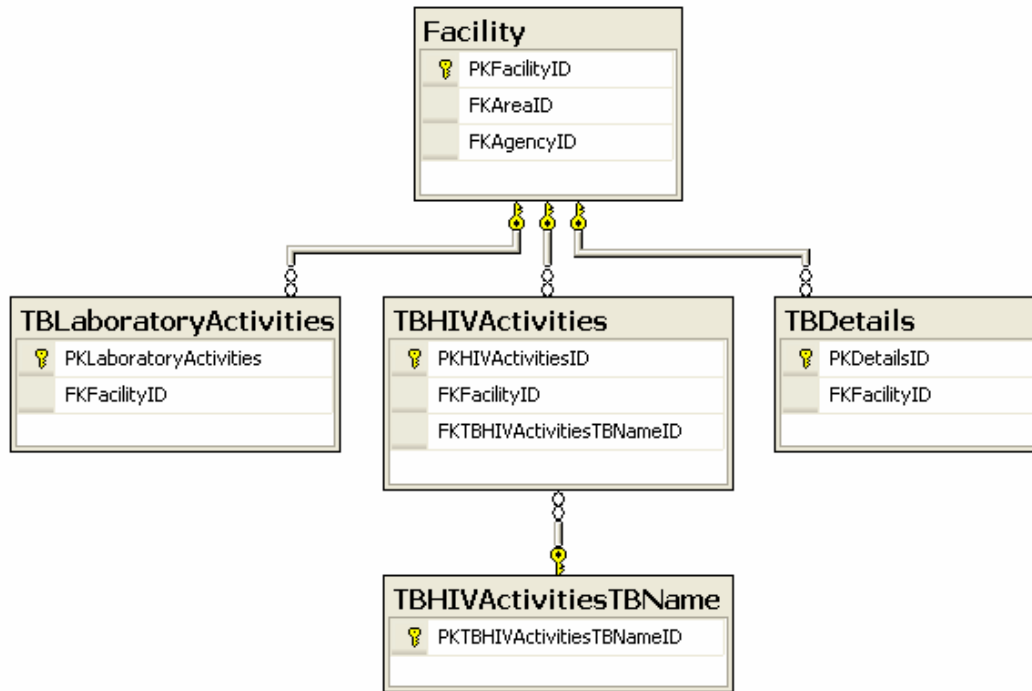


Figure 9 – TB Details Data Relationships

See 3.1.1.3 for Facility table definition

3.1.8.1 TBLaboratoryActivites Table

Columns

Column Name	Data Type	Allow Nulls
PKLaboratoryActivities	int	False
FKFacilityID	int	False
ReportDate	datetime	False
TotalExamined	int	True
SmearPositive	int	True

Indexes

Name	Clustered	Columns
IX_TBLaboratoryActivities	True	FKFacilityID
IX_TBLaboratoryActivities_1	False	ReportDate
PK_TBLaboratoryActivities	False	PKLaboratoryActivities

Foreign Key Constraints

Name	Columns	Referenced Table	Referenced Columns
FK_TBLaboratoryActivities_Facility	FKFacilityID	Facility	PKFacilityID

3.1.8.2 TBDetails Table

Columns

Column Name	Data Type	Allow Nulls
PKDetailsID	int	False
FKFacilityID	int	False
ReportDate	datetime	False
TBType	varchar(50)	True
TBName	varchar(50)	True
Sex	char(1)	True
AgeGroup	int	True
Cases	int	True

Indexes

Name	Clustered	Columns
IX_TBDetails	True	FKFacilityID
IX_TBDetails_1	False	ReportDate
IX_TBDetails_2	False	TBName
IX_TBDetails_3	False	TBType
PK_TBDetails	False	PKDetailsID

Foreign Key Constraints

Name	Columns	Referenced Table	Referenced Columns
FK_TBDetails_Facility	FKFacilityID	Facility	PKFacilityID

3.1.8.3 TBHIVActivitiesTBName Table

Columns

Column Name	Data Type	Allow Nulls
PKTBHIVActivitiesTBNameID	int	False
Description	varchar(50)	True

Indexes

Name	Clustered	Columns
PK_TBHIVActivitiesName	True	PKTBHIVActivitiesTBNameID

3.1.8.4 TBHIVActivities Table

Columns

Column Name	Data Type	Allow Nulls
PKHIVActivitiesID	int	False
FKFacilityID	int	False
ReportDate	datetime	False
FKTBHIVActivitiesTBNameID	int	False
M_Tested_HIV	int	True
M_HIV_Positive	int	True
M_On_CPT	int	True
M_On_ARV	int	True
F_Tested_HIV	int	True
F_HIV_Positive	int	True
F_On_CPT	int	True
F_On_ARV	int	True

Indexes

Name	Clustered	Columns
IX_TBHIVActivities	False	FKFacilityID
IX_TBHIVActivities_1	False	ReportDate
IX_TBHIVActivities_2	False	FKTBHIVActivitiesTBNameID
PK_TBHIVActivities	False	PKHIVActivitiesID

Foreign Key Constraints

Name	Columns	Referenced Table	Referenced Columns
FK_TBHIVActivities_Facility1	FKFacilityID	Facility	PKFacilityID
FK_TBHIVActivities_TBHIVActivitiesTBName1	FKTBHIVActivitiesTBNameID	TBHIVActivitiesTBName	PKTBHIVActivitiesTBNameID

3.1.9 TB Outcome

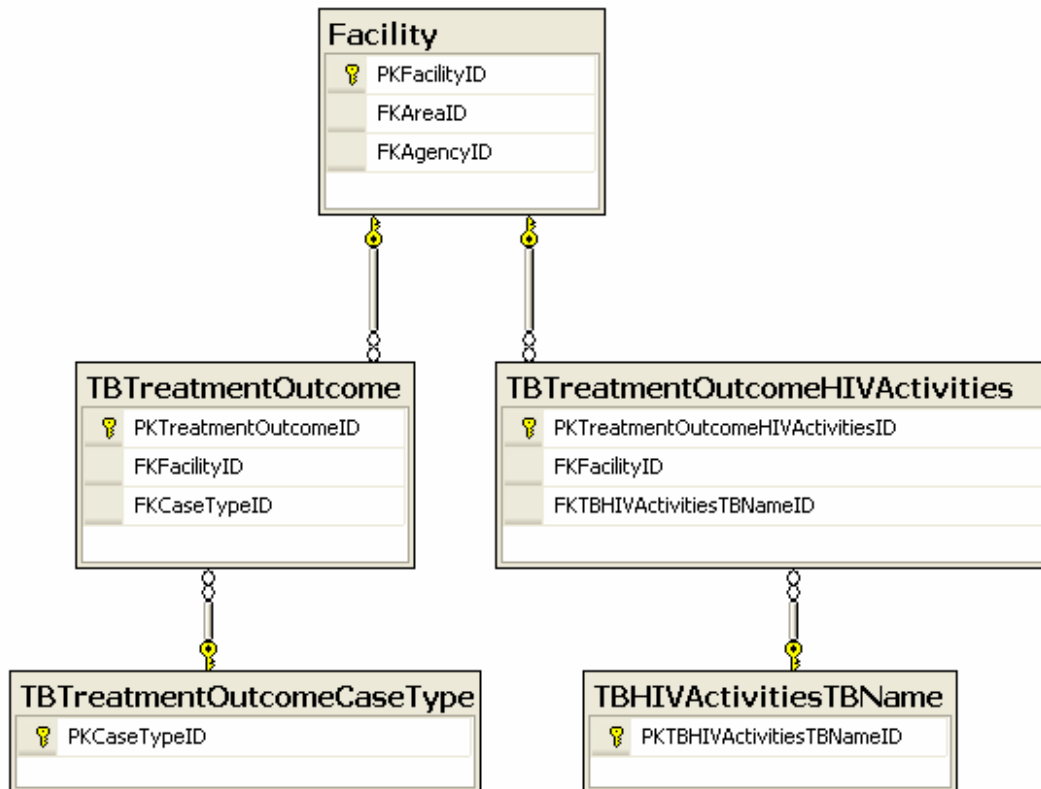


Figure 10 – TB Outcome Data Relationships

See 3.1.1.3 for Facility table definition

See 3.1.8.3 for TBHIVActivitiesTBName table definition

3.1.9.1 TBTreatmentOutcomeCaseType Table

Columns

Column Name	Data Type	Allow Nulls
PKCaseTypeID	int	False
Description	varchar(50)	False

Indexes

Name	Clustered	Columns
PK_TBTreatmentOutcomeCaseType	True	PKCaseTypeID

3.1.9.2 TBTreatmentOutcome Table

Columns

Column Name	Data Type	Allow Nulls
PKTreatmentOutcomeID	int	False
FKFacilityID	int	False

ReportDate	datetime	False
FKCaseTypeID	int	True
bHIVPositive	bit	True
Registered	int	True
Cured	int	True
Completed	int	True
Died	int	True
Failed	int	True
Defaulted	int	True
Transferred	int	True
Evaluated	int	True
NotEvaluated	int	True

Indexes

Name	Clustered	Columns
IX_TBTreatmentOutcome	True	FKFacilityID
IX_TBTreatmentOutcome_1	False	ReportDate
IX_TBTreatmentOutcome_2	False	FKCaseTypeID
PK_TBTreatment	False	PKTreatmentOutcomeID

Foreign Key Constraints

Name	Columns	Referenced Table	Referenced Columns
FK_TBTreatmentOutcome_Facility	FKFacilityID	Facility	PKFacilityID
FK_TBTreatmentOutcome_TBTreatmentOutcomeCaseType	FKCaseTypeID	TBTreatmentOutcomeCaseType	PKCaseTypeID

3.1.9.3 TBTreatmentOutcomeHIVActivitiesTable

Columns

Column Name	Data Type	Allow Nulls
PKTreatmentOutcomeHIVActivitiesID	int	False
FKFacilityID	int	False
ReportDate	datetime	False
FKTBHIVActivitiesTBNameID	int	False
M_Tested_HIV	int	True
M_HIV_Positive	int	True
M_On_CPT	int	True
M_On_ARV	int	True

F_Test_HIV	int	True
F_HIV_Positive	int	True
F_On_CPT	int	True
F_On_ARV	int	True

Indexes

Name	Clustered	Columns
IX_TBTreatmentOutcomeHIVActivities	False	FKFacilityID
IX_TBTreatmentOutcomeHIVActivities_1	False	ReportDate
IX_TBTreatmentOutcomeHIVActivities_2	False	FKTBHIVActivitiesTBNameID
PK_TBTreatmentOutcomeHIVActivities	False	PKTreatmentOutcomeHIVActivitiesID

Foreign Key Constraints

Name	Columns	Referenced Table	Referenced Columns
FK_TBTreatmentOutcomeHIVActivities_Facility	FKFacilityID	Facility	PKFacilityID
FK_TBTreatmentOutcomeHIVActivities_TBHIVActivitiesTBName1	FKTBHIVActivitiesTBNameID	TBHIVActivitiesTBName	PKTBHIVActivitiesTBNameID

3.1.10 TB Sputum

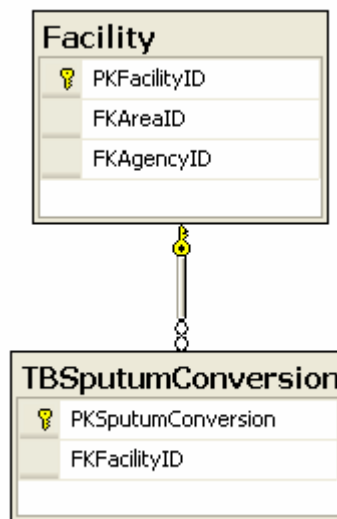


Figure 11 – TB Sputum Data Relationships

See 3.1.1.3 for Facility table definition

3.1.10.1 TBSputumCohnversion Table

Columns

Column Name	Data Type	Allow Nulls
PKSputumConversion	int	False
FKFacilityID	int	False

ReportDate	datetime	False
NewSmearPositiveCases	int	True
SmearNotDoneAt2or3Months	int	True
SputumConversionAt2Months	int	True
SputumConversionAt3Months	int	True
TotalConverted2or3Months	int	True
ConversionRate	float	True

Indexes

Name	Clustered	Columns
IX_TBSpulumConversion	True	FKFacilityID
IX_TBSpulumConversion_1	False	ReportDate
PK_TBSpulumConversion	False	PKSpulumConversion

Foreign Key Constraints

Name	Columns	Referenced Table	Referenced Columns
FK_TBSpulumConversion_Facility	FKFacilityID	Facility	PKFacilityID

4. REPORTING

It is suggested that Crystal Reports or Reporting Services (if using SQL Express) be used for the reporting requirements. The report parameters and requirements will be defined in the design documentation.



Technical Manual

Lesotho Health Data Warehouse

Prepared by:

Gareth Daniell

Prepared for:

**The Lesotho Department of Health Planning and Statistics,
Lesotho Bureau of Statistics and the World Bank, General
Data Dissemination System, Socio-Demographic Statistics
Project for Anglophone Africa**

September 2009

TABLE OF CONTENTS

1. INTRODUCTION	3
1.1.1 DOCUMENT PURPOSE AND OBJECTIVES	3
1.1.2 WHO SHOULD READ THIS DOCUMENT	3
2. DATABASES	3
2.1.1 CREATING THE DISTRICTDATA DATABASE	4
2.1.2 DEFINING USERS	4
2.1.3 CREATING THE MOHSW_DATAWAREHOUSE DATABASE	4
3. IMPORT TOOL(S)	5
3.1.1 UPLOADING	5
3.1.2 IMPORTING	7
3.1.3 IMPORTING DETAIL	8
3.1.4 IMPORTING DETAIL – DATA CONVERSION REQUIRED	10
4. ANNEXURES	16
4.1.1 ANNEXURE 1: DISTRICT DATABASE SQL SCRIPT	16
4.1.2 ANNEXURE 2: MOHSW_DATAWAREHOUSE DATABASE SQL SCRIPT	29

1. INTRODUCTION

1.1.1 Document Purpose and Objectives

The purpose of this document is to provide high level technical guide on the source code for the Lesotho data warehouse.

1.1.2 Who should read this document

Software developers responsible for the maintenance and support of Lesotho's health statistics data warehouse and related import tools.

This document should also be used by other countries electing to make use of the Lesotho data warehouse as a framework for their own data warehouse.

2. DATABASES

The system utilizes 2 MS SQL databases (MOHSW_DistrictData and MOHSW_Datawarehouse).

- **DistrictData** contains users, rights, roles and districts, and it tracks the status of the MS Access file, whether it is uploaded, verified, deleted or imported. This is used by the website to logon to the application so that users may perform their tasks based on their roles and rights. The DistrictData database contains a typical set of roles and rights and may be used as is or with minimal changes.
- **Datawarehouse** contains the actual statistical data used for warehouse reporting and analysis. As per data warehouse design principles, the tables in this database may closely resemble the source transactional databases, but instead of emphasis being placed on database normalization, attention is instead placed on timely reporting. In the event that the required warehouse design skills are not available it is possible for the warehouse database to mimic the transaction source databases exactly.
- To create the DistrictData database, copy the script and execute it in MS SQL Server Management Studio:

2.1.1 Creating the DistrictData Database

Refer to *Annexure 1: District Database SQL Script* for the SQL script to create the DistrictData database. In the event that your country is divided into provinces or states etc. instead of districts, the text [district] may be safely substituted as needed.

2.1.2 Defining Users

The website uses role base authorization. You will need to define the user with the necessary privileges to create additional users and perform any additional tasks in the MS SQL database.

- Creating the default admin user:
 - Define a role in the Role table, e.g. Admin;
 - Define all roles as in UserRole enum in Enumerations file
 - Define a right in the Right table, e.g. View_All_Districts
 - Define all rights as in Right enum in PageBase file
 - Create a record in the RoleRight table that gives the created role, e.g. Admin, the right to view all district data, e.g. View_All_Districts
 - Create a user, e.g. Admin1, in ImportUser and set the password to **7c6a180b36896a0a8c02787eeafb0e4c** (decoded as the string "password1")
 - Create a record in ImportUserRole to give the Admin user the Admin role

If everything is in order, run the application and see if you can log on to the system using the newly created Admin user.

2.1.3 Creating the MOHSW_DataWarehouse Database

Refer to *Annexure 2: MOHSW_Datawarehouse Database SQL Script* for the SQL script to create the database that will contain the statistical data. The script in *Annexure 2: MOHSW_Datawarehouse Database SQL Script* refers specifically to Lesotho's data warehouse layout as at the time of design. Additional source database need to be added to this script as and when they become necessary. As per data warehouse design principles, the tables in this database may closely resemble the source transactional databases, but instead of emphasis being placed on database normalization, attention is instead placed on timely reporting. In the event that the required warehouse design skills are not available it is possible for the warehouse database to mimic the transaction source databases exactly.

3. IMPORT TOOL(S)

3.1.1 Uploading

The system is dependent on the MS Access database file that the district uploads to the system.

- The user selects the district for which they are uploading the file.
 - Districts are defined in the District table in MS SQL database.
- Then selects the file type
 - These are defined in ImportType table.
- Selects the date.
 - This date will be used when it comes to the import process.
- User then clicks the Browse button to select the file (this is the MS Access database file to be used to import from)
- User clicks upload

This is what is happening when the Upload Button is clicked:

```
----- code snippet start -----
string UploadLocationURL =
System.Configuration.ConfigurationManager.AppSettings["UploadLocation"];
    ◦ Get the uploadlocation from the web.config file
actualFileName = distName + selectedDate.Month.ToString() +
selectedDate.Year.ToString() + "IPD" + MyNewFileUpload.FileName;
    ◦ Construct the file name
MyNewFileUpload.SaveAs(UploadLocationURL + actualFileName);
    ◦ Save the file to the server
    if (!(validateUploadFile(actualFileName)))
    {
        // Delete the database from the upload location.
        fi.Delete();
        return;
    }
    ◦ Check if the file contains the month and the year specified in the
    selected date.
    ◦ Delete the file in the data does not exist

uploadData();
    ◦ Connect to SQL Server database and insert data into
    ImportDataFiles.
----- code snippet end -----
```

Figure 1 - Upload Button Code Snippet

The method used to perform the upload of the database from the user's machine to the server:

```
----- code snippet start -----
private void uploadData()
{
    try
    {
```

```

        DateTime selectedDate =
Convert.ToDateTime(textBoxCalendar.Text.ToString());
        Database db =
DatabaseFactory.CreateDatabase("DataWareHouse");
        string sqlCommand = string.Empty;
        DbCommand dbCommand = null;
        SessionInfo si = (SessionInfo)Session["SessionInfo"];

        string uploadFileLoc =
System.Configuration.ConfigurationManager.AppSettings["virtualUploadL
ocation"].ToString();
        uploadFileLoc += tbFileDescription.Text.ToString();

        sqlCommand = "spAddImportDataFiles";
        dbCommand = db.GetStoredProcCommand(sqlCommand);

        if (!si.HasAccess(Right.View_All_Districts))
        {
            db.AddInParameter(dbCommand, "@FKDistrictID",
DbType.Int32, si.DistrictID);
        }
        else if (si.HasAccess(Right.View_All_Districts))
        {
            string selectedDist =
DropDownListDistrict.SelectedValue.ToString();
            int intdistID = LoadUserDistrictID(selectedDist);
            db.AddInParameter(dbCommand, "@FKDistrictID",
DbType.Int32, intdistID);
        }
        db.AddInParameter(dbCommand, "@FKImportTypeID",
DbType.Int32, CentralDropDownImportType.SelectedValue);
        db.AddInParameter(dbCommand, "@FileLocation",
DbType.StringFixedLength, uploadFileLoc);
        db.AddInParameter(dbCommand, "@FileDescription",
DbType.StringFixedLength, tbFileDescription.Text.ToString());
        db.AddInParameter(dbCommand, "@UploadDate_Intended",
DbType.DateTime, selectedDate);
        db.AddInParameter(dbCommand, "@DateUploaded",
DbType.DateTime, DateTime.Now);
        db.AddInParameter(dbCommand, "@Verified",
DbType.Boolean, false);
        db.AddInParameter(dbCommand, "@ImportedToDW",
DbType.Boolean, false);

        db.ExecuteNonQuery(dbCommand);
    }
    catch (Exception ex)
    {
        // todo log error
        throw;
    }
}
----- code snippet end -----

```

Figure 2 – UploadData() Method - Upload Button Code Snippet contd.

3.1.2 Importing

The following instructions assume that the file has been uploaded successfully.

- Get the data to be imported from MS Access File e.g.:
 - query = "Select * from Register"; → *this is the query used to get the data from MS Access file (Butha Buthe92008IPDInpatient.mdb), this file will be in the folder virtualUploadLocation ("/DataWareHouse/Uploaded") defined in the appSettings in the web.config file*
 - dbAdapter = new OleDbDataAdapter(query, accConn);
dbAdapter.Fill(dbTable); → *User OleDbDataAdapter to fill the data table in memory with the data from the MS Access file*
 - DatabaseImport databaseImport = new DatabaseImport();
databaseImport.CopyInpatientData(sourceTable); → *this is where the import is happening*

Following is the complete code that copies the data from the filled datatable (dbTable, MS Access database file) to the MS SQL database

```
----- code snippet start -----  
  
public void CopyInpatientData(DataTable sourceTable)  
{  
    try  
    {  
        Database db =  
DatabaseFactory.CreateDatabase("DataWareHouseStore");  
        string sqlCommand = string.Empty;  
        DbCommand dbCommand = null;  
        foreach (DataRow drow in sourceTable.Rows)  
        {  
            sqlCommand = "spAddInpatientRegister";  
            dbCommand = db.GetStoredProcCommand(sqlCommand);  
            db.AddInParameter(dbCommand, "@FKFacilityID",  
DbType.Int32,  
Convert.ToInt32(hashFacility[drow["Hopital"].ToString().Trim()]));  
            db.AddInParameter(dbCommand, "@FormNumber",  
DbType.Int32,  
Convert.ToInt32(drow["FormNumber"].ToString().Trim()));  
            db.AddInParameter(dbCommand, "@RegisterUser",  
DbType.StringFixedLength,  
drow["User"].ToString().Trim());  
            db.AddInParameter(dbCommand, "@Ward",  
DbType.StringFixedLength, drow["Ward"].ToString().Trim());  
            db.AddInParameter(dbCommand, "@AdmissionDate",  
DbType.DateTime,  
Convert.ToDateTime(drow["AdmissionDate"].ToString().Trim()));  
            db.AddInParameter(dbCommand, "@DischargeDate",  
DbType.DateTime,  

```

```

Convert.ToDateTime(drow["DischargeDate"].ToString().Trim());
    db.AddInParameter(dbCommand, "@Sex",
DbType.StringFixedLength, drow["Sex"].ToString().Trim());
    db.AddInParameter(dbCommand, "@Age", DbType.Double,
Convert.ToDouble(drow["Age"].ToString().Trim());
    db.AddInParameter(dbCommand, "@AgeModulo",
DbType.StringFixedLength,
drow["AgeModulo"].ToString().Trim());
    db.AddInParameter(dbCommand, "@LSCitizen",
DbType.Boolean,
Convert.ToBoolean(drow["LsCitizen"].ToString().Trim());
    db.AddInParameter(dbCommand, "@FKdiagnosisID",
DbType.Int32,
Convert.ToInt32(hashDiagnosisID[drow["Code"].ToString().Trim()]));
    db.AddInParameter(dbCommand, "@Outcome",
DbType.Int32,
Convert.ToInt32(drow["Outcome"].ToString().Trim()));
    db.ExecuteNonQuery(dbCommand);
    }
    catch (Exception ex)
    {
        throw new Exception("Failed to import Inpatient Register
data due to : " + ex.Message.Trim());
    }
----- code snippet end -----

```

Figure 3 – Importing: Copy () Method

3.1.3 Importing Detail

Database db = DatabaseFactory.CreateDatabase("DataWarehouseStore");
- Create the database object using the DataWarehouseStore connection string, making use of the Enterprise library.

```

foreach (DataRow drow in sourceTable.Rows)
{
    ...
}

```

- Go through all the rows in the sourceTable (table created from MS Access file)

```

sqlCommand = "spAddInpatientRegister";
dbCommand = db.GetStoredProcCommand(sqlCommand);

```

- Get the stored procedure to be used to insert the data to the database

```

db.AddInParameter(dbCommand, "@FormNumber", DbType.Int32,
Convert.ToInt32(drow["FormNumber"].ToString().Trim()));

```

- Add the values of the parameter in the stored procedure

```
db.ExecuteNonQuery(dbCommand);
- Run the stored procedure on the database
```

```
----- code snippet start -----

CREATE PROCEDURE [dbo].[spAddInpatientRegister]
    @FKFacilityID int = NULL,
    @FormNumber int = NULL,
    @RegisterUser varchar(255) = NULL ,
    @Ward varchar(50) = NULL,
    @AdmissionDate datetime = NULL,
    @DischargeDate datetime = NULL,
    @Sex char(1) = NULL,
    @Age float = NULL,
    @AgeModulo varchar(50) = NULL,
    @LSCitizen bit = NULL,
    @FKdiagnosisID int = NULL,
    @Outcome smallint = NULL

AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

    INSERT INTO dbo.InPatientRegister
        (FKFacilityID , FormNumber ,
        RegisterUser , Ward , AdmissionDate , DischargeDate ,
        Sex ,Age , AgeModulo , LSCitizen , FKdiagnosisID ,
Outcome )
    VALUES
        ( @FKFacilityID , @FormNumber ,
        @RegisterUser , @Ward , @AdmissionDate , @DischargeDate
        ,
        @Sex ,@Age , @AgeModulo , @LSCitizen , @FKdiagnosisID ,
@Outcome )
    END
----- code snippet end -----
```

Figure 4 – Insert Stored Procedure Example

```
----- code snippet start -----

CREATE TABLE [dbo].[InPatientRegister](
    [PKPatientRegisterID] [int] IDENTITY(1,1) NOT NULL,
    [FKFacilityID] [int] NOT NULL,
    [FormNumber] [int] NULL,
    [RegisterUser] [varchar](255) NULL,
    [Ward] [varchar](50) NULL,
    [AdmissionDate] [datetime] NULL,
    [DischargeDate] [datetime] NULL,
    [Sex] [char](1) NULL,
    [Age] [float] NULL,
    [AgeModulo] [varchar](50) NULL,
    [LSCitizen] [bit] NULL,
    [FKdiagnosisID] [int] NULL,
    [Outcome] [smallint] NULL,
    CONSTRAINT [PK_InPatientRegister] PRIMARY KEY CLUSTERED
```

```

(
    [PKPatientRegisterID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
) ON [PRIMARY]

GO

SET ANSI_PADDING OFF
GO

ALTER TABLE [dbo].[InPatientRegister] WITH CHECK ADD CONSTRAINT
[FK_InPatientRegister_Facility] FOREIGN KEY([FKFacilityID])
REFERENCES [dbo].[Facility] ([PKFacilityID])
GO

ALTER TABLE [dbo].[InPatientRegister] CHECK CONSTRAINT
[FK_InPatientRegister_Facility]
GO

ALTER TABLE [dbo].[InPatientRegister] WITH CHECK ADD CONSTRAINT
[FK_InPatientRegister_InpatientDiagnosis] FOREIGN
KEY([FKdiagnosisID])
REFERENCES [dbo].[InpatientDiagnosis] ([PKDiagnosisID])
GO

ALTER TABLE [dbo].[InPatientRegister] CHECK CONSTRAINT
[FK_InPatientRegister_InpatientDiagnosis]
GO
----- code snippet end -----

```

Figure 5 – Warehouse Database Table Example

3.1.4 Importing Detail – data conversion required

Some imports may be more difficult than others, like the OPD whose datawarehouse tables have been optimised for reporting (so the structure of the MS Access source tables and MS SQL Table are different to one another).

To import the OPD data from the file the same logical process is followed:

- Get the data to be imported from MS Access file
- Copy the data to MS SQL database

The following method can be found in the DatabaseImport.cs file

```

----- code snippet start -----

public void CopyOPDMonthlyData(DataTable sourceTable)
{
    IList<OutPatientRegister> outPatientRegisters =
GetPatientRegisters(sourceTable);
    Database db =
DatabaseFactory.CreateDatabase("DataWareHouseStore");
    DbCommand command =
db.GetStoredProcCommand("spAddOutPatientRegister");

```

```

        db.AddInParameter(command, "@RegisterUser", DbType.String);
        db.AddInParameter(command, "@FKFacilityID", DbType.Int32);
        db.AddInParameter(command, "@FKConditionID", DbType.Int32);
        db.AddInParameter(command, "@RegisterDate",
DbType.DateTime);
        db.AddInParameter(command, "@M_Age_0_4", DbType.Int32);
        db.AddInParameter(command, "@M_Age_5_9", DbType.Int32);
        ...
        db.AddInParameter(command, "@F_Age_65", DbType.Int32);

        foreach (var outPatientRegister in outPatientRegisters)
        {
            db.SetParameterValue(command, "@RegisterUser",
outPatientRegister.RegisterUser);
            db.SetParameterValue(command, "@FKFacilityID",
outPatientRegister.FKFacilityID);
            db.SetParameterValue(command, "@FKConditionID",
outPatientRegister.FKConditionID);
            db.SetParameterValue(command, "@RegisterDate",
outPatientRegister.RegisterDate);
            db.SetParameterValue(command, "@M_Age_0_4",
outPatientRegister.M_Age_0_4);
            ...
            db.SetParameterValue(command, "@F_Age_65",
outPatientRegister.F_Age_65);

            db.ExecuteNonQuery(command);
        }
    }
}

```

----- code snippet end-----

Figure 6 – Importing Detail: Copy () Method

The following snippet gets the data from the sourceTable (Data read from MS Access file)

----- code snippet start -----

```

private IList<OutPatientRegister> GetPatientRegisters(DataTable sourceTable)
{
    LoadFacilityByName();
    IList<OutPatientRegister> registers = new List<OutPatientRegister>();
    var provider = new MappingsProvider(MappingType.OPD);

    foreach (DataRow row in sourceTable.Rows)
    {
        string registerUser = row["User"].ToString().Trim();
        int facility =
int.Parse(hashFacilityByName[row["Facility"].ToString().Trim()].ToString());
        DateTime registerDate = Convert.ToDateTime(row["WeekDate"].ToString());
        string gender = row["Sex"].ToString().Trim();
        string age = row["Age"].ToString().Trim().Replace("+", "").Trim();
        string fieldName = (gender == "Female" ? "F_" : "M_") + "Age_" + age;
    }
}

```

```

        CreateAllConditionsForCurrentAgeGroup(registerUser, age, facility, registerDate,
gender, registers);
        UpdateCreatedRegisters(row, registers, facility, fieldName, provider);
    }

    return registers;
}
----- code snippet end-----

```

Figure 7 – Importing Detail: Copy () Method

Populating the hash table with all the facility names in the facility table in the MS SQL database

```

----- code snippet start -----

    public void LoadFacilityByName()
    {
        try
        {
            if (hashFacilityByName == null)
            {
                hashFacilityByName = new Hashtable();
            }

            Database db =
DatabaseFactory.CreateDatabase("DataWareHouseStore");
            string sqlCommand = string.Empty;
            DbCommand dbCommand = null;

            sqlCommand = "spGETFacility";
            dbCommand = db.GetStoredProcCommand(sqlCommand);

            using (IDataReader dataReader =
db.ExecuteReader(dbCommand))
            {
                while (dataReader.Read())
                {
                    int intFacilityID =
Convert.ToInt32(dataReader["PKFacilityID"]);
                    string facility =
dataReader["FacilityName"].ToString().Trim();
                    if (!hashFacilityByName.ContainsKey(facility))
                    {
                        hashFacilityByName.Add(facility,
intFacilityID);
                    }
                }
            }
        }
    }
----- code snippet end-----

```

Figure 8 – Importing Detail: Populating a hash table

Because the table is so complex we need to come up with the way to get the name of the column from the condition name.

```
----- code snippet start -----  
  
public interface IMappingsProvider  
{  
    string GetFieldCaption(string fieldName);  
    string[] GetAllCaptions();  
    string[] GetAllConditionNames();  
}  
----- code snippet end -----
```

Figure 9 – Importing Detail: Mapping Provider Interface

```
----- code snippet start -----  
  
public MappingsProvider(MappingType mappingType)  
{  
    string folder =  
    HostingEnvironment.ApplicationPhysicalPath + @"\common\";  
    filename = folder + (mappingType == MappingType.OPD ?  
    "OPDMappings.xml" : "MentalMappings.xml");  
    PopulatefieldMappings();  
}  
----- code snippet end-----
```

Figure 10 – Importing Detail: Mapping Provider Constructor

This takes the MappingType and based on the type, read the file which contains the mappings defined:

```
----- code snippet start -----  
  
<?xml version="1.0" encoding="utf-8" ?>  
<mappings>  
    <Column name="field2" map="Cholera" />  
    .....  
</mappings>  
----- code snippet end-----
```

Figure 11 – Importing Detail: Mapping Type Configuration

This is the xml file that maps the condition name to the field it maps to the source table.

- o **name**: the name of the column in the source table
- o **map**: the name of the condition

This loops on all the conditions and create an [OutPatientRegister](#) object if it does not exists, then adds it to the list of [OutPatientRegister](#).

```
----- code snippet start -----  
  
private void CreateAllConditionsForCurrentAgeGroup(...)  
{  
    foreach (var condition in hashConditions.Keys)
```

```

        {
            int conditionId =
int.Parse(hashConditions[condition].ToString());
            string formNumber = string.Format("{0}_{1}", facility,
conditionId);
            bool exists = false;

            if (registers.Count > 0)
            {
                exists = registers.Where(x =>
x.FormNumber.Equals(formNumber)).Count() > 0;
            }

            if (!exists)
            {
                OutPatientRegister register = new
OutPatientRegister();
                register.RegisterUser = registerUser;
                ...
                registers.Add(register);
            }
        }
    }
}

```

----- code snippet end-----

Figure 12 – Importing Detail: CreateAllConditionsForCurrentAgeGroup() Method

----- code snippet start -----

```

private void UpdateCreatedRegisters(DataRow currentSourceRow,
IList<OutPatientRegister> registers, int facility, string
fieldName, MappingsProvider provider)
{
    string[] conditions = provider.GetAllConditionNames();

    foreach (var condition in conditions)
    {
        int conditionId =
int.Parse(hashConditions[condition.Trim()].ToString());
        string formNumber = string.Format("{0}_{1}", facility,
conditionId);
        int fieldValue =
Convert.ToInt32(currentSourceRow[provider.GetFieldCaption(condition)].To
String());
        var groupToUpdate = registers.Where(x =>
x.FormNumber.Equals(formNumber));

        foreach (var update in groupToUpdate)
        {
            SetFieldValue(update, fieldName, fieldValue);
        }
    }
}

```

----- code snippet end-----

Figure 13 – Importing Detail: UpdateCreatedRegisters() Method

```

----- code snippet start -----
private void SetFieldValue(OutPatientRegister register, string fieldName, int value)
{
    Type type = register.GetType();

    PropertyInfo propertyInfo = type.GetProperty(fieldName);
    propertyInfo.SetValue(register, value, null);
}
----- code snippet end-----

```

Figure 14 – Importing Detail: SetFieldValue () Method

- This get all the condition names from the xml and loops through, get the current object to be updated using the constructed form number
- Get the field value from the source table using the mappings
int fieldValue =
Convert.ToInt32(currentSourceRow[provider.GetFieldCaption(condition)].ToString());
- Use reflection to update the correct field (SetFieldValue) in the OutPatientRegister object.
- We then get the Database using the Enterprise Library
Database db = DatabaseFactory.CreateDatabase("DataWarehouseStore");
- Get the stored procedure
db.GetStoredProcCommand("spAddOutPatientRegister");
- Add Parameter to the stored procedure command
db.AddInParameter(command, "@RegisterUser", DbType.String);
- Set Parameter added
db.SetParameterValue(command, "@RegisterUser", outPatientRegister.RegisterUser);
- Execute the stored procedure
db.ExecuteNonQuery(command);

4. ANNEXURES

4.1.1 Annexure 1: District Database SQL Script

----- [START COPY HERE] -----

```
/****** Object: Table [dbo].[District]   Script Date: 09/15/2009 11:59:25 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[District](
    [PKDistrictID] [int] NOT NULL,
    [DistrictName] [varchar](255) NOT NULL,
    CONSTRAINT [PK_District] PRIMARY KEY CLUSTERED
(
    [PKDistrictID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object: Table [dbo].[ImportType]   Script Date: 09/15/2009 11:59:29 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[ImportType](
    [PKImportTypeID] [int] NOT NULL,
    [Description] [varchar](255) NOT NULL,
    CONSTRAINT [PK_ImportType] PRIMARY KEY CLUSTERED
(
    [PKImportTypeID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object: Table [dbo].[ImportUser]   Script Date: 09/15/2009 11:59:30 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
```

```

CREATE TABLE [dbo].[ImportUser](
    [PKImportUserID] [int] IDENTITY(1,1) NOT NULL,
    [Username] [varchar](50) NOT NULL,
    [Password] [varchar](50) NOT NULL,
    [FirstNames] [varchar](50) NOT NULL,
    [Surname] [varchar](50) NOT NULL,
    [Email] [varchar](50) NULL,
    CONSTRAINT [PK_ImportUser] PRIMARY KEY NONCLUSTERED
(
    [PKImportUserID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: StoredProcedure [dbo].[spAddImportDataFiles]  Script Date: 09/15/2009
11:59:20 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spAddImportDataFiles]
    @FKDistrictID int = NULL,
    @FKImportTypeID int = NULL,
    @FileLocation varchar(1024) = NULL,
    @FileDescription varchar(255) = NULL ,
    @UploadDate_Intended datetime = NULL,
    @DateUploaded datetime = NULL,
    @Verified bit = NULL,
    @ImportedToDW bit = NULL

AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

    INSERT INTO dbo.ImportDataFiles
        (FKDistrictID , FKImportTypeID ,
        FileLocation , FileDescription , UploadDate_Intended, DateUploaded , Verified ,
        ImportedToDW)
        VALUES
        (@FKDistrictID , @FKImportTypeID ,
        @FileLocation , @FileDescription , @UploadDate_Intended, @DateUploaded , @Verified ,
        @ImportedToDW)

    --          AND (@intFKStateID IS NULL OR U.intFKStateID = @intFKStateID)
    -- Insert statements for procedure here
END
GO
/***** Object: StoredProcedure [dbo].[spSETImportedToDW]  Script Date: 09/15/2009
11:59:23 *****/
SET ANSI_NULLS ON

```

```

GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spSETImportedToDW]
@PKImportDataFileID int = NULL,
@ImportedToDW bit = NULL
AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
    UPDATE dbo.ImportDataFiles
    SET ImportedToDW = @ImportedToDW

    WHERE
        @PKImportDataFileID IS NULL OR PKImportDataFileID = @PKImportDataFileID
END
GO
/***** Object: StoredProcedure [dbo].[spSETVerified]  Script Date: 09/15/2009 11:59:23
*****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spSETVerified]
@PKImportDataFileID int = NULL,
@Verified bit = NULL
AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
    UPDATE dbo.ImportDataFiles
    SET Verified = @Verified

    WHERE
        @PKImportDataFileID IS NULL OR PKImportDataFileID = @PKImportDataFileID
END
GO
/***** Object: StoredProcedure [dbo].[spDELETERecord]  Script Date: 09/15/2009 11:59:21
*****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spDELETERecord]
@PKImportDataFileID int = NULL
AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
    DELETE FROM dbo.ImportDataFiles

    WHERE

```

PKImportDataFileID = @PKImportDataFileID

```
END
GO
/***** Object: Table [dbo].[Audit]  Script Date: 09/15/2009 11:59:25 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Audit](
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [UserId] [int] NOT NULL,
    [ActionPerformed] [varchar](250) NOT NULL,
    [DateActionPerformed] [datetime] NOT NULL,
    CONSTRAINT [PK_Audit] PRIMARY KEY CLUSTERED
(
    [ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[Role]  Script Date: 09/15/2009 11:59:33 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Role](
    [PKRoleID] [int] NOT NULL,
    [Description] [varchar](50) NOT NULL,
    [Comments] [varchar](255) NULL,
    CONSTRAINT [PK_Role] PRIMARY KEY CLUSTERED
(
    [PKRoleID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[Right]  Script Date: 09/15/2009 11:59:32 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Right](
```

```

        [PKRightID] [int] NOT NULL,
        [Description] [varchar](50) NOT NULL,
        [Comments] [varchar](255) NULL,
    CONSTRAINT [PK_Right] PRIMARY KEY CLUSTERED
    (
        [PKRightID] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
    ) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[ImportDataFiles]  Script Date: 09/15/2009 11:59:28 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[ImportDataFiles](
    [PKImportDataFileID] [int] IDENTITY(1,1) NOT NULL,
    [FKDistrictID] [int] NOT NULL,
    [FKImportTypeID] [int] NOT NULL,
    [FileLocation] [varchar](1024) NOT NULL,
    [FileDescription] [varchar](255) NOT NULL,
    [UploadDate_Intended] [datetime] NOT NULL,
    [DateUploaded] [datetime] NOT NULL CONSTRAINT
[DF_ImportDataFiles_DateUploaded] DEFAULT (getdate()),
    [Verified] [bit] NOT NULL,
    [ImportedToDW] [bit] NOT NULL,
    CONSTRAINT [PK_ImportDataFiles] PRIMARY KEY NONCLUSTERED
    (
        [PKImportDataFileID] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
    ) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[DistrictImportUser]  Script Date: 09/15/2009 11:59:26 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[DistrictImportUser](
    [FKDistrictID] [int] NOT NULL,
    [FKImportUserID] [int] NOT NULL
) ON [PRIMARY]
GO
/***** Object: Table [dbo].[ImportUserRole]  Script Date: 09/15/2009 11:59:31 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON

```

```

GO
CREATE TABLE [dbo].[ImportUserRole](
    [FKImportUserID] [int] NOT NULL,
    [FKRoleID] [int] NOT NULL
) ON [PRIMARY]
GO
/***** Object: Table [dbo].[RoleRight]  Script Date: 09/15/2009 11:59:33 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[RoleRight](
    [FKRoleID] [int] NOT NULL,
    [FKRightID] [int] NOT NULL
) ON [PRIMARY]
GO
/***** Object: StoredProcedure [dbo].[spGETUserRights]  Script Date: 09/15/2009 11:59:22 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spGETUserRights]
    @UserID      int
AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
    SELECT DISTINCT
        R.PKRightID,
        R.Description
    FROM
        [ImportUser] IU
        JOIN [ImportUserRole] IUR ON IU.PKImportUserID = IUR.FKImportUserID
        JOIN [RoleRight] RR ON IUR.FKRoleID = RR.FKRoleID
        JOIN [Right] R ON R.PKRightID = RR.FKRightID
    WHERE
        IU.PKImportUserID = @UserID
END
GO
/***** Object: StoredProcedure [dbo].[spINSERTUser]  Script Date: 09/15/2009 11:59:23 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spINSERTUser]
    @name varchar(50),
    @surname varchar(50),
    @username varchar(50),
    @password varchar(50),
    @role int,
    @emailAddress varchar(50) = NULL,

```

```

        @errorMessage varchar(100) = null out
AS
BEGIN

    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

    declare @userCount int

    select @userCount = count(PKImportUserId)
    from importUser
    where username = @username

    if (@userCount > 0)
    begin
        set @errorMessage = 'Username exists';
    end
    else
    begin
        INSERT INTO ImportUser
            ([Username]
            ,[Password]
            ,[FirstNames]
            ,[Surname]
            ,[Email])
        VALUES
            (@username
            ,@password
            ,@name
            ,@surname
            ,@emailAddress)

        declare @userID int;
        select @userID = PKImportUserID from ImportUser where username =
@username

        insert into importUserRole
        values (
                    @userID,
                    @role
                )

    end
END
GO
/***** Object: StoredProcedure [dbo].[spDELETEUser]  Script Date: 09/15/2009 11:59:21
*****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spDELETEUser]
    @userId int
AS

```

```

BEGIN
    delete from ImportUserRole
    where FKImportUserID = @userId

    delete from ImportUser
    where PKImportUserId = @userId

END
GO
/***** Object: StoredProcedure [dbo].[spGETDistrict]   Script Date: 09/15/2009 11:59:21
*****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spGETDistrict]
    @PKDistrictID int = NULL,
    @DistrictName varchar(255) = NULL
AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

    SELECT
        PKDistrictID,
        DistrictName
    FROM
        [District]
    WHERE
        (@PKDistrictID IS NULL OR PKDistrictID = @PKDistrictID)
        AND (@DistrictName IS NULL OR DistrictName = @DistrictName)

    ORDER BY PKDistrictID

END
GO
/***** Object: StoredProcedure [dbo].[spGETUserDistrictInfo]   Script Date: 09/15/2009
11:59:22 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spGETUserDistrictInfo]
    @PKImportUserID int = NULL,
    @UserName varchar(50) = NULL,
    @Password varchar(50) = NULL
AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

    SELECT
        IU.PKImportUserID,
        DIU.FKDistrictID,

```

```

        D.DistrictName
FROM
    [ImportUser] IU
    JOIN [DistrictImportUser] DIU ON DIU.FKImportUserID = IU.PKImportUserID
    JOIN [District] D on DIU.FKDistrictID = D.PKDistrictID
WHERE
    (@PKImportUserID IS NULL OR IU.PKImportUserID = @PKImportUserID)
--
    AND (@intFKStateID IS NULL OR U.intFKStateID = @intFKStateID)
-- Insert statements for procedure here
END
GO
/***** Object: StoredProcedure [dbo].[spGETImportDataFiles]  Script Date: 09/15/2009
11:59:21 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spGETImportDataFiles]
    @PKImportDataFileID int = NULL,
    @FKDistrictID int = NULL,
    @FKImportTypeID int = NULL,
    @FileLocation varchar(1024) = NULL,
    @FileDescription varchar(255) = NULL,
    @UploadDate_Intended datetime = NULL,
    @DateUploaded datetime = NULL,
    @Verified bit = NULL,
    @ImportedToDW bit = NULL
AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

    SELECT
        IDF.PKImportDataFileID,
        IDF.FKDistrictID,
        D.DistrictName,
        IDF.FKImportTypeID,
        IT.Description AS ImportTypeDescription,
        IDF.FileLocation,
        IDF.FileDescription,
        IDF.UploadDate_Intended,
        IDF.DateUploaded,
        IDF.Verified,
        IDF.ImportedToDW
    FROM
        [ImportDataFiles] IDF
        JOIN [District] D ON D.PKDistrictID = IDF.FKDistrictID
        JOIN [ImportType] IT ON IT.PKImportTypeID = IDF.FKImportTypeID
    WHERE
        (@PKImportDataFileID IS NULL OR IDF.PKImportDataFileID =
@PKImportDataFileID)
        AND (@FKDistrictID IS NULL OR IDF.FKDistrictID = @FKDistrictID)
        AND (@FKImportTypeID IS NULL OR IDF.FKImportTypeID = @FKImportTypeID)

```

```

        AND (@FileLocation IS NULL OR IDF.FileLocation = @FileLocation)
        AND (@FileDescription IS NULL OR IDF.FileDescription = @FileDescription)
        AND (@UploadDate_Intended IS NULL OR IDF.UploadDate_Intended =
@UploadDate_Intended)
        AND (@DateUploaded IS NULL OR IDF.DateUploaded = @DateUploaded)
        AND (@Verified IS NULL OR IDF.Verified = @Verified)
        AND (@ImportedToDW IS NULL OR IDF.ImportedToDW = @ImportedToDW)

        ORDER BY IDF.DateUploaded DESC
    END
GO
/***** Object: StoredProcedure [dbo].[spGETImportType]  Script Date: 09/15/2009 11:59:22
*****/
SET ANSI_NULLS OFF
GO
SET QUOTED_IDENTIFIER OFF
GO
CREATE PROCEDURE [dbo].[spGETImportType]
    @PKImportTypeID int = NULL,
    @Description varchar(255) = NULL
AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

    SELECT
        PKImportTypeID,
        Description
    FROM
        [ImportType]
    WHERE
        (@PKImportTypeID IS NULL OR PKImportTypeID = @PKImportTypeID)
        AND (@Description IS NULL OR Description = @Description)

    END
GO
/***** Object: StoredProcedure [dbo].[spUPDATEResetUserPassword]  Script Date:
09/15/2009 11:59:23 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spUPDATEResetUserPassword]
    @userId int,
    @newPassword varchar(50)
AS
BEGIN
    Update ImportUser
    set password = @newPassword
    where PKImportuserId = @userId
END
GO
/***** Object: StoredProcedure [dbo].[spUPDATEUser]  Script Date: 09/15/2009 11:59:24
*****/

```

```

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spUPDATEUser]
    @userId int,
    @name varchar(50),
    @surname varchar(50),
    @emailAddress varchar(50) = NULL
AS
BEGIN

    UPDATE ImportUser
    SET [FirstNames] = @name
        ,[Surname] = @surname
        ,[Email] = @emailAddress
    WHERE pkImportUserId = @userId

END
GO
/***** Object: StoredProcedure [dbo].[spGETUserInfo]  Script Date: 09/15/2009 11:59:22
*****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spGETUserInfo]
    @PKImportUserID int = NULL,
    @UserName varchar(50) = NULL,
    @Password varchar(50) = NULL
--    @intFKStateID int = NULL -- state not yet implemented
AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

    SELECT
        IU.PKImportUserID,
        IU.Username,
        IU.Password,
        IU.Firstnames,
        IU.Surname,
        IU.Email
    FROM
        [ImportUser] IU
    WHERE
        (@PKImportUserID IS NULL OR IU.PKImportUserID = @PKImportUserID)
        AND (@UserName IS NULL OR IU.Username = @UserName)
        AND (@Password IS NULL OR IU.Password = @Password)
--        AND (@intFKStateID IS NULL OR U.intFKStateID = @intFKStateID)
END
GO

```

```
/****** Object: StoredProcedure [dbo].[spINSERTAudit] Script Date: 09/15/2009 11:59:22
*****/
```

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spINSERTAudit]
    @UserId int,
    @ActionPerformed varchar(250)
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [Audit]
        ([UserId],[ActionPerformed],[DateActionPerformed])
    VALUES
        (@UserId,@ActionPerformed, getdate())
END
GO
```

```
/****** Object: ForeignKey [FK_DistrictUser_District] Script Date: 09/15/2009 11:59:26
*****/
```

```
ALTER TABLE [dbo].[DistrictImportUser] WITH CHECK ADD CONSTRAINT
[FK_DistrictUser_District] FOREIGN KEY([FKDistrictID])
REFERENCES [dbo].[District] ([PKDistrictID])
GO
```

```
ALTER TABLE [dbo].[DistrictImportUser] CHECK CONSTRAINT [FK_DistrictUser_District]
GO
```

```
/****** Object: ForeignKey [FK_DistrictUser_ImportUser] Script Date: 09/15/2009 11:59:26
*****/
```

```
ALTER TABLE [dbo].[DistrictImportUser] WITH CHECK ADD CONSTRAINT
[FK_DistrictUser_ImportUser] FOREIGN KEY([FKImportUserID])
REFERENCES [dbo].[ImportUser] ([PKImportUserID])
GO
```

```
ALTER TABLE [dbo].[DistrictImportUser] CHECK CONSTRAINT [FK_DistrictUser_ImportUser]
GO
```

```
/****** Object: ForeignKey [FK_ImportDataFiles_District] Script Date: 09/15/2009 11:59:28
*****/
```

```
ALTER TABLE [dbo].[ImportDataFiles] WITH CHECK ADD CONSTRAINT
[FK_ImportDataFiles_District] FOREIGN KEY([FKDistrictID])
REFERENCES [dbo].[District] ([PKDistrictID])
GO
```

```
ALTER TABLE [dbo].[ImportDataFiles] CHECK CONSTRAINT [FK_ImportDataFiles_District]
GO
```

```
/****** Object: ForeignKey [FK_ImportDataFiles_ImportType] Script Date: 09/15/2009
11:59:28 *****/
```

```
ALTER TABLE [dbo].[ImportDataFiles] WITH CHECK ADD CONSTRAINT
[FK_ImportDataFiles_ImportType] FOREIGN KEY([FKImportTypeID])
REFERENCES [dbo].[ImportType] ([PKImportTypeID])
GO
```

```
ALTER TABLE [dbo].[ImportDataFiles] CHECK CONSTRAINT [FK_ImportDataFiles_ImportType]
GO
```

```
/****** Object: ForeignKey [FK_ImportUserRole_ImportUser] Script Date: 09/15/2009
11:59:31 *****/
```

```

ALTER TABLE [dbo].[ImportUserRole] WITH CHECK ADD CONSTRAINT
[FK_ImportUserRole_ImportUser] FOREIGN KEY([FKImportUserID])
REFERENCES [dbo].[ImportUser] ([PKImportUserID])
GO
ALTER TABLE [dbo].[ImportUserRole] CHECK CONSTRAINT [FK_ImportUserRole_ImportUser]
GO
/***** Object: ForeignKey [FK_ImportUserRole_Role] Script Date: 09/15/2009 11:59:31
*****/
ALTER TABLE [dbo].[ImportUserRole] WITH CHECK ADD CONSTRAINT
[FK_ImportUserRole_Role] FOREIGN KEY([FKRoleID])
REFERENCES [dbo].[Role] ([PKRoleID])
GO
ALTER TABLE [dbo].[ImportUserRole] CHECK CONSTRAINT [FK_ImportUserRole_Role]
GO
/***** Object: ForeignKey [FK_RoleRight_Right] Script Date: 09/15/2009 11:59:34 *****/
ALTER TABLE [dbo].[RoleRight] WITH CHECK ADD CONSTRAINT [FK_RoleRight_Right]
FOREIGN KEY([FKRightID])
REFERENCES [dbo].[Right] ([PKRightID])
GO
ALTER TABLE [dbo].[RoleRight] CHECK CONSTRAINT [FK_RoleRight_Right]
GO
/***** Object: ForeignKey [FK_RoleRight_Role] Script Date: 09/15/2009 11:59:34 *****/
ALTER TABLE [dbo].[RoleRight] WITH CHECK ADD CONSTRAINT [FK_RoleRight_Role] FOREIGN
KEY([FKRoleID])
REFERENCES [dbo].[Role] ([PKRoleID])
GO
ALTER TABLE [dbo].[RoleRight] CHECK CONSTRAINT [FK_RoleRight_Role]
GO
----- [END COPY HERE] -----

```

4.1.2 Annexure 2: MOHSW_Datawarehouse Database SQL Script

----- [START COPY HERE] -----

```
/****** Object: User [admin] Script Date: 09/16/2009 10:23:14 *****/
CREATE USER [admin] WITHOUT LOGIN WITH DEFAULT_SCHEMA=[dbo]
GO
/****** Object: Schema [ICT\ASPNET] Script Date: 09/16/2009 10:23:08 *****/
CREATE SCHEMA [ICT\ASPNET] AUTHORIZATION [ICT\ASPNET]
GO
/****** Object: Table [dbo].[Agency] Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Agency](
    [PKAgencyID] [int] NOT NULL,
    [Code] [varchar](50) NOT NULL,
    [Description] [varchar](255) NOT NULL,
    CONSTRAINT [PK_Agency] PRIMARY KEY CLUSTERED
(
    [PKAgencyID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object: Table [dbo].[Area] Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Area](
    [PKAreaID] [int] NOT NULL,
    [Code] [varchar](50) NOT NULL,
    [Description] [varchar](255) NOT NULL,
    CONSTRAINT [PK_Area] PRIMARY KEY CLUSTERED
(
    [PKAreaID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object: Table [dbo].[ReportType] Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
```

```

GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[ReportType](
    [PKReportTypeID] [int] NOT NULL,
    [Description] [varchar](255) NOT NULL,
    CONSTRAINT [PK_ReportType] PRIMARY KEY CLUSTERED
(
    [PKReportTypeID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[MentalStage] Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[MentalStage](
    [PKStageID] [int] NOT NULL,
    [Description] [varchar](50) NOT NULL,
    CONSTRAINT [PK_MentalStage] PRIMARY KEY CLUSTERED
(
    [PKStageID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[ReportName] Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[ReportName](
    [PKReportNameID] [int] NOT NULL,
    [Description] [varchar](255) NOT NULL,
    CONSTRAINT [PK_ReportName] PRIMARY KEY CLUSTERED
(
    [PKReportNameID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

```

```

SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[TBTreatmentOutcomeCaseType]  Script Date: 09/16/2009
10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[TBTreatmentOutcomeCaseType](
    [PKCaseTypeID] [int] NOT NULL,
    [Description] [varchar](50) NOT NULL,
    CONSTRAINT [PK_TBTreatmentOutcomeCaseType] PRIMARY KEY CLUSTERED
(
    [PKCaseTypeID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[OutPatientCondition]  Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[OutPatientCondition](
    [PKConditionID] [int] IDENTITY(1,1) NOT NULL,
    [Description] [varchar](150) NOT NULL,
    CONSTRAINT [PK_OutPatientCondition] PRIMARY KEY CLUSTERED
(
    [PKConditionID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: StoredProcedure [dbo].[spAddOutPatientRegister]  Script Date: 09/16/2009
10:23:12 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spAddOutPatientRegister]
    @RegisterUser varchar(50),
    @FKFacilityID int,
    @FKConditionID int,
    @RegisterDate datetime,
    @M_Age_0_4 int,

```

```
@M_Age_5_9 int,  
@M_Age_10_14 int,  
@M_Age_15_19 int,  
@M_Age_20_24 int,  
@M_Age_25_29 int,  
@M_Age_30_34 int,  
@M_Age_35_39 int,  
@M_Age_40_44 int,  
@M_Age_45_49 int,  
@M_Age_50_54 int,  
@M_Age_55_59 int,  
@M_Age_60_64 int,  
@M_Age_65 int,  
@F_Age_0_4 int,  
@F_Age_5_9 int,  
@F_Age_10_14 int,  
@F_Age_15_19 int,  
@F_Age_20_24 int,  
@F_Age_25_29 int,  
@F_Age_30_34 int,  
@F_Age_35_39 int,  
@F_Age_40_44 int,  
@F_Age_45_49 int,  
@F_Age_50_54 int,  
@F_Age_55_59 int,  
@F_Age_60_64 int,  
@F_Age_65 int
```

```
AS  
BEGIN
```

```
-- SET NOCOUNT ON added to prevent extra result sets from  
-- interfering with SELECT statements.  
SET NOCOUNT ON;
```

```
INSERT INTO [MOHSW_Datawarehouse].[dbo].[OutPatientRegister]  
([RegisterUser]  
,[FKFacilityID]  
,[FormNumber]  
,[FKConditionID]  
,[RegisterDate]  
,[M_Age_0_4]  
,[M_Age_5_9]  
,[M_Age_10_14]  
,[M_Age_15_19]  
,[M_Age_20_24]  
,[M_Age_25_29]  
,[M_Age_30_34]  
,[M_Age_35_39]  
,[M_Age_40_44]  
,[M_Age_45_49]  
,[M_Age_50_54]  
,[M_Age_55_59]
```

```

,[M_Age_60_64]
,[M_Age_65]
,[F_Age_0_4]
,[F_Age_5_9]
,[F_Age_10_14]
,[F_Age_15_19]
,[F_Age_20_24]
,[F_Age_25_29]
,[F_Age_30_34]
,[F_Age_35_39]
,[F_Age_40_44]
,[F_Age_45_49]
,[F_Age_50_54]
,[F_Age_55_59]
,[F_Age_60_64]
,[F_Age_65])
VALUES
(@RegisterUser
,@FKFacilityID
,1
,@FKConditionID
,@RegisterDate
,@M_Age_0_4
,@M_Age_5_9
,@M_Age_10_14
,@M_Age_15_19
,@M_Age_20_24
,@M_Age_25_29
,@M_Age_30_34
,@M_Age_35_39
,@M_Age_40_44
,@M_Age_45_49
,@M_Age_50_54
,@M_Age_55_59
,@M_Age_60_64
,@M_Age_65
,@F_Age_0_4
,@F_Age_5_9
,@F_Age_10_14
,@F_Age_15_19
,@F_Age_20_24
,@F_Age_25_29
,@F_Age_30_34
,@F_Age_35_39
,@F_Age_40_44
,@F_Age_45_49
,@F_Age_50_54
,@F_Age_55_59
,@F_Age_60_64
,@F_Age_65)

```

```

END
GO

```

```
/****** Object: StoredProcedure [dbo].[spGETMentalConditions] Script Date: 09/16/2009
10:23:12 *****/
```

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spGETMentalConditions]
AS
BEGIN
    SET NOCOUNT ON;

    SELECT [PKConditionID],[Description]
    FROM [MOHSW_Datawarehouse].[dbo].[MentalCondition]
END
GO
```

```
/****** Object: StoredProcedure [dbo].[spINSERTMentalRegister] Script Date: 09/16/2009
10:23:12 *****/
```

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spINSERTMentalRegister]
    @RegisterUser varchar(50),
    @FKFacilityID int,
    @FormNumber int,
    @nDate datetime,
    @FKStageID int,
    @FKConditionID int,
    @M_Age_0_9 int,
    @M_Age_10_19 int,
    @M_Age_20_29 int,
    @M_Age_30_39 int,
    @M_Age_40_49 int,
    @M_Age_50_59 int,
    @M_Age_60_64 int,
    @M_Age_65 int,
    @F_Age_0_9 int,
    @F_Age_10_19 int,
    @F_Age_20_29 int,
    @F_Age_30_39 int,
    @F_Age_40_49 int,
    @F_Age_50_59 int,
    @F_Age_60_64 int,
    @F_Age_65 int
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;
INSERT INTO [MOHSW_Datawarehouse].[dbo].[MentalRegister]
    ([RegisterUser]
    ,[FKFacilityID]
    ,[FormNumber]
```

```

,[Date]
,[FKStageID]
,[FKConditionID]
,[M_Age_0_9]
,[M_Age_10_19]
,[M_Age_20_29]
,[M_Age_30_39]
,[M_Age_40_49]
,[M_Age_50_59]
,[M_Age_60_64]
,[M_Age_65]
,[F_Age_0_9]
,[F_Age_10_19]
,[F_Age_20_29]
,[F_Age_30_39]
,[F_Age_40_49]
,[F_Age_50_59]
,[F_Age_60_64]
,[F_Age_65])
VALUES
(@RegisterUser,
@FKFacilityID,
@FormNumber,
@nDate,
@FKStageID,
@FKConditionID,
@M_Age_0_9,
@M_Age_10_19,
@M_Age_20_29,
@M_Age_30_39,
@M_Age_40_49,
@M_Age_50_59,
@M_Age_60_64,
@M_Age_65,
@F_Age_0_9,
@F_Age_10_19,
@F_Age_20_29,
@F_Age_30_39,
@F_Age_40_49,
@F_Age_50_59,
@F_Age_60_64,
@F_Age_65)
END
GO
/***** Object: Table [dbo].[MentalCondition]   Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[MentalCondition](
    [PKConditionID] [int] IDENTITY(1,1) NOT NULL,

```

```

        [Description] [varchar](50) NOT NULL,
    CONSTRAINT [PK_MentalCondition] PRIMARY KEY CLUSTERED
    (
        [PKConditionID] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
    ) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[TBHIVActivitiesTBName]  Script Date: 09/16/2009 10:23:14
*****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[TBHIVActivitiesTBName](
    [PKTBHIVActivitiesTBNameID] [int] NOT NULL,
    [Description] [varchar](50) NULL,
    CONSTRAINT [PK_TBHIVActivitiesName] PRIMARY KEY CLUSTERED
    (
        [PKTBHIVActivitiesTBNameID] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
    ) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[DentalDiagnosis]  Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[DentalDiagnosis](
    [PKDiagnosisID] [int] NOT NULL,
    [Description] [varchar](255) NULL,
    CONSTRAINT [PK_DentalDiagnosis] PRIMARY KEY CLUSTERED
    (
        [PKDiagnosisID] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
    ) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[DentalProcedure]  Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON

```

```

GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[DentalProcedure](
    [PKProcedureID] [int] NOT NULL,
    [Description] [varchar](255) NULL,
    CONSTRAINT [PK_DentalProcedure] PRIMARY KEY CLUSTERED
(
    [PKProcedureID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[InpatientDiagnosis] Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[InpatientDiagnosis](
    [PKDiagnosisID] [int] IDENTITY(1,1) NOT NULL,
    [Code] [varchar](50) NULL,
    [Description] [varchar](255) NULL,
    CONSTRAINT [PK_Diagnosis] PRIMARY KEY CLUSTERED
(
    [PKDiagnosisID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: StoredProcedure [dbo].[spAddInpatientRegister] Script Date: 09/16/2009
10:23:12 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spAddInpatientRegister]
    @FKFacilityID int = NULL,
    @FormNumber int = NULL,
    @RegisterUser varchar(255) = NULL ,
    @Ward varchar(50) = NULL,
    @AdmissionDate datetime = NULL,
    @DischargeDate datetime = NULL,
    @Sex char(1) = NULL,
    @Age float = NULL,
    @AgeModulo varchar(50) = NULL,
    @LSCitizen bit = NULL,
    @FKdiagnosisID int = NULL,

```

```

@Outcome smallint = NULL

AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

    INSERT INTO dbo.InPatientRegister
        (FKFacilityID , FormNumber ,
        RegisterUser , Ward , AdmissionDate , DischargeDate ,
        Sex ,Age , AgeModulo , LSCitizen , FKdiagnosisID , Outcome )
    VALUES
        ( @FKFacilityID , @FormNumber ,
        @RegisterUser , @Ward , @AdmissionDate , @DischargeDate ,
        @Sex ,@Age , @AgeModulo , @LSCitizen , @FKdiagnosisID , @Outcome )
--
-- AND (@intFKStateID IS NULL OR U.intFKStateID = @intFKStateID)
-- Insert statements for procedure here
END
GO
/***** Object: Table [dbo].[ReportTypeName]  Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[ReportTypeName](
    [FKReportTypeID] [int] NOT NULL,
    [FKReportNameID] [int] NOT NULL
) ON [PRIMARY]
GO
/***** Object: Table [dbo].[MentalRegister]  Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[MentalRegister](
    [PKMentalRegisterID] [int] IDENTITY(1,1) NOT NULL,
    [RegisterUser] [varchar](255) NULL,
    [FKFacilityID] [int] NOT NULL,
    [FormNumber] [int] NOT NULL,
    [Date] [datetime] NULL,
    [FKStageID] [int] NULL,
    [FKConditionID] [int] NULL,
    [M_Age_0_9] [int] NULL,
    [M_Age_10_19] [int] NULL,
    [M_Age_20_29] [int] NULL,
    [M_Age_30_39] [int] NULL,
    [M_Age_40_49] [int] NULL,
    [M_Age_50_59] [int] NULL,
    [M_Age_60_64] [int] NULL,
    [M_Age_65] [int] NULL,
    [F_Age_0_9] [int] NULL,

```

```

[F_Age_10_19] [int] NULL,
[F_Age_20_29] [int] NULL,
[F_Age_30_39] [int] NULL,
[F_Age_40_49] [int] NULL,
[F_Age_50_59] [int] NULL,
[F_Age_60_64] [int] NULL,
[F_Age_65] [int] NULL,
CONSTRAINT [PK_MentalRegister] PRIMARY KEY CLUSTERED
(
    [PKMentalRegisterID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[Facility]    Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Facility](
    [PKFacilityID] [int] IDENTITY(1,1) NOT NULL,
    [FKAreaID] [int] NULL,
    [FKAgencyID] [int] NULL,
    [Code] [varchar](50) NULL,
    [FacilityName] [varchar](255) NULL,
    [District] [varchar](255) NULL,
    CONSTRAINT [PK_Facility_1] PRIMARY KEY CLUSTERED
(
    [PKFacilityID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[TBTreatmentOutcome]    Script Date: 09/16/2009 10:23:14
*****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[TBTreatmentOutcome](
    [PKTreatmentOutcomeID] [int] NOT NULL,
    [FKFacilityID] [int] NOT NULL,
    [ReportDate] [datetime] NOT NULL,
    [FKCaseTypeID] [int] NULL,
    [bHIVPositive] [bit] NULL,
    [Registered] [int] NULL,
    [Cured] [int] NULL,

```

```

    [Completed] [int] NULL,
    [Died] [int] NULL,
    [Failed] [int] NULL,
    [Defaulted] [int] NULL,
    [Transferred] [int] NULL,
    [Evaluated] [int] NULL,
    [NotEvaluated] [int] NULL,
    CONSTRAINT [PK_TBTreatment] PRIMARY KEY CLUSTERED
(
    [PKTreatmentOutcomeID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/***** Object: Table [dbo].[OutPatientRegister]    Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[OutPatientRegister](
    [PKOutpatientRegisterID] [int] IDENTITY(1,1) NOT NULL,
    [RegisterUser] [varchar](255) NULL,
    [FKFacilityID] [int] NOT NULL,
    [FormNumber] [int] NOT NULL,
    [RegisterDate] [datetime] NOT NULL,
    [FKConditionID] [int] NULL,
    [M_Age_0_4] [int] NULL,
    [M_Age_5_9] [int] NULL,
    [M_Age_10_14] [int] NULL,
    [M_Age_15_19] [int] NULL,
    [M_Age_20_24] [int] NULL,
    [M_Age_25_29] [int] NULL,
    [M_Age_30_34] [int] NULL,
    [M_Age_35_39] [int] NULL,
    [M_Age_40_44] [int] NULL,
    [M_Age_45_49] [int] NULL,
    [M_Age_50_54] [int] NULL,
    [M_Age_55_59] [int] NULL,
    [M_Age_60_64] [int] NULL,
    [M_Age_65] [int] NULL,
    [F_Age_0_4] [int] NULL,
    [F_Age_5_9] [int] NULL,
    [F_Age_10_14] [int] NULL,
    [F_Age_15_19] [int] NULL,
    [F_Age_20_24] [int] NULL,
    [F_Age_25_29] [int] NULL,
    [F_Age_30_34] [int] NULL,
    [F_Age_35_39] [int] NULL,
    [F_Age_40_44] [int] NULL,
    [F_Age_45_49] [int] NULL,
    [F_Age_50_54] [int] NULL,

```

```

        [F_Age_55_59] [int] NULL,
        [F_Age_60_64] [int] NULL,
        [F_Age_65] [int] NULL,
    CONSTRAINT [PK_OutPatientRegister] PRIMARY KEY CLUSTERED
    (
        [PKOutpatientRegsiterID] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
    ) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[TBTreatmentOutcomeHIVActivities]    Script Date: 09/16/2009
10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[TBTreatmentOutcomeHIVActivities](
    [PKTreatmentOutcomeHIVActivitiesID] [int] NOT NULL,
    [FKFacilityID] [int] NOT NULL,
    [ReportDate] [datetime] NOT NULL,
    [FKTBHIVActivitiesTBNameID] [int] NOT NULL,
    [M_Testesd_HIV] [int] NULL,
    [M_HIV_Positive] [int] NULL,
    [M_On_CPT] [int] NULL,
    [M_On_ARV] [int] NULL,
    [F_Testesd_HIV] [int] NULL,
    [F_HIV_Positive] [int] NULL,
    [F_On_CPT] [int] NULL,
    [F_On_ARV] [int] NULL,
    CONSTRAINT [PK_TBTreatmentOutcomeHIVActivities] PRIMARY KEY CLUSTERED
    (
        [PKTreatmentOutcomeHIVActivitiesID] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
    ) ON [PRIMARY]
GO
/***** Object: Table [dbo].[TBHIVActivities]    Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[TBHIVActivities](
    [PKHIVActivitiesID] [int] NOT NULL,
    [FKFacilityID] [int] NOT NULL,
    [ReportDate] [datetime] NOT NULL,
    [FKTBHIVActivitiesTBNameID] [int] NOT NULL,
    [M_Testesd_HIV] [int] NULL,
    [M_HIV_Positive] [int] NULL,
    [M_On_CPT] [int] NULL,
    [M_On_ARV] [int] NULL,
    [F_Testesd_HIV] [int] NULL,

```

```

        [F_HIV_Positive] [int] NULL,
        [F_On_CPT] [int] NULL,
        [F_On_ARV] [int] NULL,
    CONSTRAINT [PK_TBHIVActivities] PRIMARY KEY CLUSTERED
    (
        [PKHIVActivitiesID] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
    ) ON [PRIMARY]
GO
/***** Object: Table [dbo].[DeliveryChild]   Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[DeliveryChild](
    [PKChildID] [int] IDENTITY(1,1) NOT NULL,
    [FKDeliveryRegisterID] [int] NOT NULL,
    [APGAR1MIN] [int] NULL,
    [APGAR5MIN] [int] NULL,
    [Sex] [varchar](50) NULL,
    [ChildWeight] [float] NULL,
    [newbornMaturity] [int] NULL,
    [Malformations] [varchar](50) NULL,
    [BirthStatus] [varchar](50) NULL,
    [StillBirth] [varchar](50) NULL,
    [arvPROPHGIVEN] [int] NULL,
    [arvPROPHDISCHGIVEN] [varchar](50) NULL,
    [Immunization] [int] NULL,
    [feedingOptBirth] [varchar](50) NULL,
    [infantStatusDisch1] [varchar](50) NULL,
    [infantStatusDisch2] [int] NULL,
    CONSTRAINT [PK_Child] PRIMARY KEY CLUSTERED
    (
        [PKChildID] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
    ) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[DentalRegister]   Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[DentalRegister](
    [PKDentalRegisterID] [int] NOT NULL,
    [FKFacilityID] [int] NOT NULL,

```

```

    [RegisterUser] [varchar](255) NULL,
    [FormNumber] [int] NULL,
    [Month] [smallint] NULL,
    [Year] [smallint] NULL,
    [Date] [datetime] NULL,
    [Age] [float] NULL,
    [Sex] [char](1) NULL,
    [FKDiagnosisID] [int] NULL,
    [FKProcedureID] [int] NULL,
    [Repeat] [bit] NULL,
CONSTRAINT [PK_DentalRegister] PRIMARY KEY CLUSTERED
(
    [PKDentalRegisterID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[InPatientRegister]  Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[InPatientRegister](
    [PKPatientRegisterID] [int] IDENTITY(1,1) NOT NULL,
    [FKFacilityID] [int] NOT NULL,
    [FormNumber] [int] NULL,
    [RegisterUser] [varchar](255) NULL,
    [Ward] [varchar](50) NULL,
    [AdmissionDate] [datetime] NULL,
    [DischargeDate] [datetime] NULL,
    [Sex] [char](1) NULL,
    [Age] [float] NULL,
    [AgeModulo] [varchar](50) NULL,
    [LSCitizen] [bit] NULL,
    [FKdiagnosisID] [int] NULL,
    [Outcome] [smallint] NULL,
CONSTRAINT [PK_InPatientRegister] PRIMARY KEY CLUSTERED
(
    [PKPatientRegisterID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[TBLaboratoryActivities]  Script Date: 09/16/2009 10:23:14
*****/
SET ANSI_NULLS ON
GO

```

```

SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[TBLaboratoryActivities](
    [PKLaboratoryActivities] [int] NOT NULL,
    [FKFacilityID] [int] NOT NULL,
    [ReportDate] [datetime] NOT NULL,
    [TotalExamined] [int] NULL,
    [SmearPositive] [int] NULL,
    CONSTRAINT [PK_TBLaboratoryActivities] PRIMARY KEY CLUSTERED
(
    [PKLaboratoryActivities] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/***** Object: Table [dbo].[TBDetails] Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[TBDetails](
    [PKDetailsID] [int] NOT NULL,
    [FKFacilityID] [int] NOT NULL,
    [ReportDate] [datetime] NOT NULL,
    [TBType] [varchar](50) NULL,
    [TBName] [varchar](50) NULL,
    [Sex] [char](1) NULL,
    [AgeGroup] [int] NULL,
    [Cases] [int] NULL,
    CONSTRAINT [PK_TBDetails] PRIMARY KEY CLUSTERED
(
    [PKDetailsID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[HIV_HTC] Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[HIV_HTC](
    [PKHTCID] [int] NOT NULL,
    [FKFacilityID] [int] NOT NULL,
    [HTCUser] [varchar](255) NULL,
    [Couple] [int] NULL,
    [Age] [varchar](50) NULL,

```

```

        [IndicatorID] [nchar](10) NULL,
    CONSTRAINT [PK_HIV_HTC] PRIMARY KEY CLUSTERED
    (
        [PKHTCID] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
    ) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[DeliveryRegister]    Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[DeliveryRegister](
    [PKDeliveryRegisterID] [int] NOT NULL,
    [FKFacilityID] [int] NOT NULL,
    [RegisterUser] [varchar](255) NULL,
    [FormNumber] [int] NOT NULL,
    [FormIndex] [int] NOT NULL,
    [ReportingDate] [datetime] NOT NULL,
    [AdmissionDate] [datetime] NOT NULL,
    [Age] [int] NOT NULL,
    [ANCAttended] [bit] NOT NULL,
    [Parity] [int] NOT NULL,
    [GestationalAge] [smallint] NOT NULL,
    [pmtctTestHIVSTATUS] [varchar](50) NULL,
    [pmtctTestHAARTSTATUS] [varchar](50) NULL,
    [pmtctPreTestMaternity] [varchar](50) NULL,
    [pmtctTESTHIVMATERNITY] [varchar](50) NULL,
    [pmtctTESTRESULTMART] [varchar](50) NULL,
    [pmtctARVPROHPREG] [int] NULL,
    [pmtctARVPROPHB4DELIVERY] [int] NULL,
    [pmtctARVLABOR] [int] NULL,
    [pmtctARVPROPHNumHRS] [varchar](50) NULL,
    [DeliveryDate] [datetime] NULL,
    [DeliveryMode] [int] NULL,
    [DeliveryNumFetus] [int] NULL,
    [maternCI] [varchar](255) NULL,
    [maternDeathOccur] [varchar](50) NULL,
    [maternDeathCause] [int] NULL,
    [postPartumMotherStatusDisch] [varchar](50) NULL,
    [postPartMotherRefferd] [int] NULL,
    [postPartMotherCouns] [varchar](50) NULL,
    [postPartarvDisch] [varchar](50) NULL,
    [dischargeDate] [datetime] NULL,
    [postPartDelivBy] [varchar](50) NULL,
    CONSTRAINT [PK_DeliveryRegister] PRIMARY KEY CLUSTERED
    (
        [PKDeliveryRegisterID] ASC

```

```

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[TBSputumConversion]   Script Date: 09/16/2009 10:23:14
*****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[TBSputumConversion](
    [PKSputumConversion] [int] NOT NULL,
    [FKFacilityID] [int] NOT NULL,
    [ReportDate] [datetime] NOT NULL,
    [NewSmearPositiveCases] [int] NULL,
    [SmearNotDoneAt2or3Months] [int] NULL,
    [SputumConversionAt2Months] [int] NULL,
    [SputumConversionAt3Months] [int] NULL,
    [TotalConverted2or3Months] [int] NULL,
    [ConversionRate] [float] NULL,
    CONSTRAINT [PK_TBSpoutumConversion] PRIMARY KEY CLUSTERED
(
    [PKSputumConversion] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/***** Object: Table [dbo].[ANCRegister]   Script Date: 09/16/2009 10:23:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[ANCRegister](
    [PKANCRegisterID] [int] NOT NULL,
    [FKFacilityID] [int] NOT NULL,
    [RegisterUser] [varchar](255) NULL,
    [ReportMonth] [smallint] NOT NULL,
    [ReportYear] [smallint] NOT NULL,
    [field1] [int] NULL,
    [field2] [int] NULL,
    [field3] [int] NULL,
    [field4] [int] NULL,
    [field5] [int] NULL,
    [field6] [int] NULL,
    [field7] [int] NULL,
    [field8] [int] NULL,
    [field9] [int] NULL,
    [field10] [int] NULL,
    [field11] [int] NULL,

```

[field12] [int] NULL,
[field13] [int] NULL,
[field14] [int] NULL,
[field15] [int] NULL,
[field16] [int] NULL,
[field17] [int] NULL,
[field18] [int] NULL,
[field19] [int] NULL,
[field20] [int] NULL,
[field21] [int] NULL,
[field22] [int] NULL,
[field23] [int] NULL,
[field24] [int] NULL,
[field25] [int] NULL,
[field26] [int] NULL,
[field27] [int] NULL,
[field28] [int] NULL,
[field29] [int] NULL,
[field30] [int] NULL,
[field31] [int] NULL,
[field32] [int] NULL,
[field33] [int] NULL,
[field34] [int] NULL,
[field35] [int] NULL,
[field36] [int] NULL,
[field37] [int] NULL,
[field38] [int] NULL,
[field39] [int] NULL,
[field40] [int] NULL,
[field41] [int] NULL,
[field42] [int] NULL,
[field43] [int] NULL,
[field44] [int] NULL,
[field45] [int] NULL,
[field46] [int] NULL,
[field47] [int] NULL,
[field48] [int] NULL,
[field49] [int] NULL,
[field50] [int] NULL,
[field51] [int] NULL,
[field52] [int] NULL,
[field53] [int] NULL,
[field54] [int] NULL,
[field55] [int] NULL,
[field56] [int] NULL,
[field57] [int] NULL,
[field58] [int] NULL,
[field59] [int] NULL,
[field60] [int] NULL,
[field61] [int] NULL,
[field62] [int] NULL,
[field63] [int] NULL,
[field64] [int] NULL,

```

CONSTRAINT [PK_ANCRegister] PRIMARY KEY CLUSTERED
(
    [PKANCRegisterID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: StoredProcedure [dbo].[spGETReportType]  Script Date: 09/16/2009 10:23:12
*****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spGETReportType]
    @PKReportTypeID int = NULL
AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

    SELECT
        RT.PKReportTypeID,
        RT.Description
    FROM
        [ReportType] RT
    WHERE
        (@PKReportTypeID IS NULL OR RT.PKReportTypeID = @PKReportTypeID)
--      AND (@intFKStateID IS NULL OR U.intFKStateID = @intFKStateID)
-- Insert statements for procedure here

    ORDER BY RT.PKReportTypeID;
END
GO
/***** Object: StoredProcedure [dbo].[spGETReportName]  Script Date: 09/16/2009
10:23:12 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spGETReportName]
    @PKReportTypeID int = NULL
AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

    SELECT
        RT.PKReportTypeID,
        RN.PKReportNameID,
        RN.Description

```

```

FROM
    [ReportType] RT
    JOIN [ReportTypeName] RTN ON RTN.FKReportTypeID = RT.PKReportTypeID
    JOIN [ReportName] RN on RTN.FKReportNameID = RN.PKReportNameID
WHERE
    (@PKReportTypeID IS NULL OR RT.PKReportTypeID = @PKReportTypeID)
--
    AND (@intFKStateID IS NULL OR U.intFKStateID = @intFKStateID)
-- Insert statements for procedure here
END
GO
/***** Object: StoredProcedure [dbo].[spAddMaternalChildData]   Script Date: 09/16/2009
10:23:12 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spAddMaternalChildData]
    @FKDeliveryRegisterID int,
    @APGAR1MIN int = NULL,
    @APGAR5MIN int = NULL,
    @Sex varchar(50) = NULL,
    @ChildWeight float = NULL,
    @newbornMaturity int = NULL,
    @Malformations varchar(50) = NULL,
    @BirthStatus varchar(50) = NULL,
    @StillBirth varchar(50) = NULL,
    @arvPROPHGIVEN int = NULL,
    @arvPROPHDISCHGIVEN varchar(50) = NULL,
    @Immunization int = NULL,
    @feedingOptBirth varchar(50) = NULL,
    @infantStatusDisch1 varchar(50) = NULL,
    @infantStatusDisch2 int = NULL

AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

    INSERT INTO dbo.DeliveryChild
        ( FKDeliveryRegisterID , APGAR1MIN,
          APGAR5MIN, Sex , ChildWeight,
          newbornMaturity , Malformations , BirthStatus , StillBirth, arvPROPHGIVEN ,
          arvPROPHDISCHGIVEN, Immunization , feedingOptBirth , infantStatusDisch1,
          infantStatusDisch2
        )
    VALUES
        (@FKDeliveryRegisterID, @APGAR1MIN , @APGAR5MIN,@Sex,
         @ChildWeight, @newbornMaturity, @Malformations, @BirthStatus,
         @StillBirth, @arvPROPHGIVEN, @arvPROPHDISCHGIVEN, @Immunization,
         @feedingOptBirth, @infantStatusDisch1, @infantStatusDisch2
        )
--
    AND (@intFKStateID IS NULL OR U.intFKStateID = @intFKStateID)
-- Insert statements for procedure here

```

```

END
GO
/***** Object: StoredProcedure [dbo].[spGETConditions]  Script Date: 09/16/2009 10:23:12
*****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spGETConditions]
AS
BEGIN
    SET NOCOUNT ON;

    SELECT [PKConditionID],[Description]
    FROM [MOHSW_Datawarehouse].[dbo].[OutPatientCondition]
END
GO
/***** Object: StoredProcedure [dbo].[spAddMaternalData]  Script Date: 09/16/2009
10:23:12 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spAddMaternalData]
    @PKDeliveryRegisterID int = NULL,
    @FKFacilityID int = NULL,
    @RegisterUser varchar(255) = NULL,
    @FormNumber int = NULL,
    @FormIndex int = NULL,
    @ReportingDate datetime = NULL,
    @AdmissionDate datetime = NULL,
    @Age int = NULL,
    @ANCAttended bit = NULL,
    @Parity int = NULL,
    @GestationalAge smallint = NULL,
    @pmtctTestHIVSTATUS varchar(50) = NULL,
    @pmtctTestHAARTSTATUS varchar(50) = NULL,
    @pmtctPreTestMaternity varchar(50) = NULL,
    @pmtctTESTHIVMATERNITY varchar(50) = NULL,
    @pmtctTESTRESULTMART varchar(50) = NULL,
    @pmtctARVPROHPREG int = NULL,
    @pmtctARVPROPHB4DELIVERY int = NULL,
    @pmtctARVLABOR int = NULL,
    @pmtctARVPROPHNumHRS varchar(50) = NULL,
    @DeliveryDate datetime = NULL,
    @DeliveryMode int = NULL,
    @DeliveryNumFetus int = NULL,
    @maternCI varchar(255) = NULL,
    @maternDeathOccur varchar(50) = NULL,
    @maternDeathCause int = NULL,
    @postPartumMotherStatusDisch varchar(50) = NULL,
    @postPartMotherRefferd int = NULL,
    @postPartMotherCouns varchar(50) = NULL,

```

```

@postPartarvDisch varchar(50) = NULL,
@dischargeDate datetime = NULL,
@postPartDelivBy varchar(50) = NULL

AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

    INSERT INTO dbo.DeliveryRegister
    ( PKDeliveryRegisterID , FKFacilityID,
      RegisterUser, FormNumber, FormIndex , ReportingDate,
      AdmissionDate , Age , ANCAttended , Parity , GestationalAge,
      pmtctTestHIVSTATUS , pmtctTestHAARTSTATUS , pmtctPreTestMaternity ,
      pmtctTESTHIVMATERNITY , pmtctTESTRESULTMART , pmtctARVPROHPREG ,
      pmtctARVPROPHB4DELIVERY , pmtctARVLAVOR , pmtctARVPROPHNumHRS,
      DeliveryDate , DeliveryMode , DeliveryNumFetus , maternCI ,
      maternDeathOccur , maternDeathCause , postPartumMotherStatusDisch ,
      postPartMotherRefferd , postPartMotherCouns , postPartarvDisch ,
      dischargeDate , postPartDelivBy
    )
    VALUES
    ( @PKDeliveryRegisterID , @FKFacilityID,
      @RegisterUser, @FormNumber, @FormIndex , @ReportingDate,
      @AdmissionDate , @Age , @ANCAttended , @Parity , @GestationalAge,
      @pmtctTestHIVSTATUS , @pmtctTestHAARTSTATUS ,
      @pmtctPreTestMaternity ,
      @pmtctTESTHIVMATERNITY , @pmtctTESTRESULTMART ,
      @pmtctARVPROHPREG ,
      @pmtctARVPROPHB4DELIVERY , @pmtctARVLAVOR ,
      @pmtctARVPROPHNumHRS,
      @DeliveryDate , @DeliveryMode , @DeliveryNumFetus , @maternCI ,
      @maternDeathOccur , @maternDeathCause , @postPartumMotherStatusDisch
    ,
      @postPartMotherRefferd , @postPartMotherCouns , @postPartarvDisch ,
      @dischargeDate , @postPartDelivBy
    )
    -- AND (@intFKStateID IS NULL OR U.intFKStateID = @intFKStateID)
    -- Insert statements for procedure here
END
GO
/***** Object: StoredProcedure [dbo].[spGETInpatientReportChp6Table5] Script Date:
09/16/2009 10:23:12 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spGETInpatientReportChp6Table5]
    @FacilityName varchar(255) = NULL,
    @StartingMonth int = NULL,
    @EndMonth int = NULL,
    @SelYear int = NULL

```

```

AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

SELECT
    FAC.FacilityName,
    IPD.SEX,
    Round(IPD.Age/365, 1) AS Age,

    'AgeRange' = CASE
        WHEN Round(IPD.Age/365, 1) = 998 THEN 'Unknwown
Children'
        WHEN Round(IPD.Age/365, 1) = 999 THEN 'Unknwown
Adults'
        WHEN Round(IPD.Age/365, 1) >= 0 AND
Round(IPD.Age/365, 1) < 5 THEN '0-4'
        WHEN Round(IPD.Age/365, 1) >= 5 AND
Round(IPD.Age/365, 1) < 10 THEN '05-9'
        WHEN Round(IPD.Age/365, 1) >= 10 AND
Round(IPD.Age/365, 1) < 15 THEN '10-14'
        WHEN Round(IPD.Age/365, 1) >= 15 AND
Round(IPD.Age/365, 1) < 20 THEN '15-19'
        WHEN Round(IPD.Age/365, 1) >= 20 AND
Round(IPD.Age/365, 1) < 25 THEN '20-24'
        WHEN Round(IPD.Age/365, 1) >= 25 AND
Round(IPD.Age/365, 1) < 30 THEN '25-29'
        WHEN Round(IPD.Age/365, 1) >= 30 AND
Round(IPD.Age/365, 1) < 35 THEN '30-34'
        WHEN Round(IPD.Age/365, 1) >= 35 AND
Round(IPD.Age/365, 1) < 40 THEN '35-39'
        WHEN Round(IPD.Age/365, 1) >= 40 AND
Round(IPD.Age/365, 1) < 45 THEN '40-44'
        WHEN Round(IPD.Age/365, 1) >= 45 AND
Round(IPD.Age/365, 1) < 50 THEN '45-49'
        WHEN Round(IPD.Age/365, 1) >= 50 AND
Round(IPD.Age/365, 1) < 55 THEN '50-54'
        WHEN Round(IPD.Age/365, 1) >= 55 AND
Round(IPD.Age/365, 1) < 60 THEN '55-59'
        WHEN Round(IPD.Age/365, 1) >= 60 AND
Round(IPD.Age/365, 1) < 65 THEN '60-64'
        WHEN Round(IPD.Age/365, 1) >= 65 AND
Round(IPD.Age/365, 1) < 150 THEN '65+'
        -- ELSE 'Invalid Age'
    END ,

    ID.Description AS FinalDiagnosis,
    COUNT(IPD.FKdiagnosisID) AS DiagFreq,
    IPD.Outcome

FROM ([InPatientRegister] IPD JOIN
    InpatientDiagnosis ID ON IPD.FKdiagnosisID = ID.PKDiagnosisID) JOIN
    [Facility] FAC ON IPD.FKFacilityID = FAC.PKFacilityID

```

```

WHERE
    (IPD.Outcome = 4) AND
    (@SelYear IS NULL OR YEAR(IPD.DischargeDate) = @SelYear) AND
    (@StartingMonth IS NULL OR @EndMonth IS NULL) OR ((MONTH(IPD.DischargeDate)
    >= @StartingMonth AND MONTH(IPD.DischargeDate) <= @EndMonth)) AND
    (@FacilityName IS NULL OR FAC.FacilityName = @FacilityName) -- Outcome = 4 for
deaths

GROUP BY FAC.FacilityName, IPD.Age, IPD.Sex, IPD.Outcome, ID.Description, IPD.FKdiagnosisID

END
GO
/***** Object: StoredProcedure [dbo].[spGETDiagnosisID]  Script Date: 09/16/2009 10:23:12
*****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spGETDiagnosisID]
AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
    SELECT DISTINCT
        PKDiagnosisID,
        Code,
        Description
    FROM
        [InpatientDiagnosis]
END
GO
/***** Object: StoredProcedure [dbo].[spGETInpatientReportChp6Table2]  Script Date:
09/16/2009 10:23:12 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spGETInpatientReportChp6Table2]
    @FacilityName varchar(255) = NULL,
    @StartingMonth int = NULL,
    @EndMonth int = NULL,
    @SelYear int = NULL
AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

SELECT
    FAC.FacilityName,
    IPD.FKFacilityID,

```

```

IPD.AdmissionDate,
IPD.DischargeDate,
IPD.Sex,
Round(IPD.Age/365, 1) AS Age,

'AgeRange' = CASE
    WHEN Round(IPD.Age/365, 1) = 998 THEN 'Unknwown
Children'
    WHEN Round(IPD.Age/365, 1) = 999 THEN 'Unknwown
Adults'
    WHEN Round(IPD.Age/365, 1) >= 0 AND
Round(IPD.Age/365, 1) < 5 THEN '0-4'
    WHEN Round(IPD.Age/365, 1) >= 5 AND
Round(IPD.Age/365, 1) < 10 THEN '05-9'
    WHEN Round(IPD.Age/365, 1) >= 10 AND
Round(IPD.Age/365, 1) < 15 THEN '10-14'
    WHEN Round(IPD.Age/365, 1) >= 15 AND
Round(IPD.Age/365, 1) < 20 THEN '15-19'
    WHEN Round(IPD.Age/365, 1) >= 20 AND
Round(IPD.Age/365, 1) < 25 THEN '20-24'
    WHEN Round(IPD.Age/365, 1) >= 25 AND
Round(IPD.Age/365, 1) < 30 THEN '25-29'
    WHEN Round(IPD.Age/365, 1) >= 30 AND
Round(IPD.Age/365, 1) < 35 THEN '30-34'
    WHEN Round(IPD.Age/365, 1) >= 35 AND
Round(IPD.Age/365, 1) < 40 THEN '35-39'
    WHEN Round(IPD.Age/365, 1) >= 40 AND
Round(IPD.Age/365, 1) < 45 THEN '40-44'
    WHEN Round(IPD.Age/365, 1) >= 45 AND
Round(IPD.Age/365, 1) < 50 THEN '45-49'
    WHEN Round(IPD.Age/365, 1) >= 50 AND
Round(IPD.Age/365, 1) < 55 THEN '50-54'
    WHEN Round(IPD.Age/365, 1) >= 55 AND
Round(IPD.Age/365, 1) < 60 THEN '55-59'
    WHEN Round(IPD.Age/365, 1) >= 60 AND
Round(IPD.Age/365, 1) < 65 THEN '60-64'
    WHEN Round(IPD.Age/365, 1) >= 65 AND
Round(IPD.Age/365, 1) < 150 THEN '65+'
--
    ELSE 'Invalid Age'
END ,

IPD.AgeModulo,
IPD.FKdiagnosisID,
IPD.Outcome

FROM [InPatientRegister] IPD LEFT JOIN
    [Facility] FAC ON IPD.FKFacilityID = FAC.PKFacilityID

WHERE
    (@SelYear IS NULL OR YEAR(IPD.DischargeDate) = @SelYear) AND
    (@StartingMonth IS NULL OR @EndMonth IS NULL) OR ((MONTH(IPD.DischargeDate)
>= @StartingMonth AND MONTH(IPD.DischargeDate) <= @EndMonth)) AND
    (@FacilityName IS NULL OR FAC.FacilityName = @FacilityName)

```

```

--          AND (@intFKStateID IS NULL OR U.intFKStateID = @intFKStateID)
-- Insert statements for procedure here
END
GO
/***** Object: StoredProcedure [dbo].[spGETInpatientReportChp6Table1]  Script Date:
09/16/2009 10:23:12 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spGETInpatientReportChp6Table1]
    @FacilityName varchar(255) = NULL,
    @StartingMonth int = NULL,
    @EndMonth int = NULL,
    @SelYear int = NULL

AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

SELECT
    FAC.FacilityName,
    IPD.FKFacilityID,
    IPD.AdmissionDate,
    IPD.DischargeDate,
    IPD.Sex,
    Round(IPD.Age/365, 1) AS Age,

    'AgeRange' = CASE
        WHEN Round(IPD.Age/365, 1) = 998 THEN 'Unknwown
Children'
        WHEN Round(IPD.Age/365, 1) = 999 THEN 'Unknwown
Adults'
        WHEN Round(IPD.Age/365, 1) >= 0 AND
Round(IPD.Age/365, 1) < 5 THEN '0-4'
        WHEN Round(IPD.Age/365, 1) >= 5 AND
Round(IPD.Age/365, 1) < 10 THEN '05-9'
        WHEN Round(IPD.Age/365, 1) >= 10 AND
Round(IPD.Age/365, 1) < 15 THEN '10-14'
        WHEN Round(IPD.Age/365, 1) >= 15 AND
Round(IPD.Age/365, 1) < 20 THEN '15-19'
        WHEN Round(IPD.Age/365, 1) >= 20 AND
Round(IPD.Age/365, 1) < 25 THEN '20-24'
        WHEN Round(IPD.Age/365, 1) >= 25 AND
Round(IPD.Age/365, 1) < 30 THEN '25-29'
        WHEN Round(IPD.Age/365, 1) >= 30 AND
Round(IPD.Age/365, 1) < 35 THEN '30-34'
        WHEN Round(IPD.Age/365, 1) >= 35 AND
Round(IPD.Age/365, 1) < 40 THEN '35-39'
        WHEN Round(IPD.Age/365, 1) >= 40 AND
Round(IPD.Age/365, 1) < 45 THEN '40-44'

```

```

        WHEN Round(IPD.Age/365, 1) >= 45 AND
Round(IPD.Age/365, 1) < 50 THEN '45-49'
        WHEN Round(IPD.Age/365, 1) >= 50 AND
Round(IPD.Age/365, 1) < 55 THEN '50-54'
        WHEN Round(IPD.Age/365, 1) >= 55 AND
Round(IPD.Age/365, 1) < 60 THEN '55-59'
        WHEN Round(IPD.Age/365, 1) >= 60 AND
Round(IPD.Age/365, 1) < 65 THEN '60-64'
        WHEN Round(IPD.Age/365, 1) >= 65 AND
Round(IPD.Age/365, 1) < 150 THEN '65+'
--
        ELSE 'Invalid Age'
        END ,
    IPD.AgeModulo,
    IPD.FKdiagnosisID,
    IPD.Outcome

FROM [InPatientRegister] IPD LEFT JOIN
    [Facility] FAC ON IPD.FKFacilityID = FAC.PKFacilityID

WHERE
    (@SelYear IS NULL OR YEAR(IPD.AdmissionDate) = @SelYear) AND
    (@StartingMonth IS NULL OR @EndMonth IS NULL) OR ((MONTH(IPD.AdmissionDate)
BETWEEN @StartingMonth AND @EndMonth)) AND
    (@FacilityName IS NULL OR FAC.FacilityName = @FacilityName)

--
    AND (@intFKStateID IS NULL OR U.intFKStateID = @intFKStateID)
-- Insert statements for procedure here
END
GO
/***** Object: StoredProcedure [dbo].[spGETInpatientData]    Script Date: 09/16/2009
10:23:12 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[spGETInpatientData]
    @FacilityName varchar(255) = NULL,
    @StartingMonth int = NULL,
    @EndMonth int = NULL,
    @SelYear int = NULL

AS
BEGIN
    SET NOCOUNT ON
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

SELECT
    FAC.FacilityName,
    IPD.FKFacilityID,
    IPD.AdmissionDate,
    IPD.DischargeDate,
    IPD.Sex,
    Round(IPD.Age/365, 1) AS Age,

```

```

'AgeRange' = CASE
    WHEN Round(IPD.Age/365, 1) = 998 THEN 'Unknwown
Children'
    WHEN Round(IPD.Age/365, 1) = 999 THEN 'Unknwown
Adults'
    WHEN Round(IPD.Age/365, 1) > 0 AND
Round(IPD.Age/365, 1) < 5 THEN '0-4'
    WHEN Round(IPD.Age/365, 1) >= 5 AND
Round(IPD.Age/365, 1) < 10 THEN '05-9'
    WHEN Round(IPD.Age/365, 1) >= 10 AND
Round(IPD.Age/365, 1) < 15 THEN '10-14'
    WHEN Round(IPD.Age/365, 1) >= 15 AND
Round(IPD.Age/365, 1) < 20 THEN '15-19'
    WHEN Round(IPD.Age/365, 1) >= 20 AND
Round(IPD.Age/365, 1) < 25 THEN '20-24'
    WHEN Round(IPD.Age/365, 1) >= 25 AND
Round(IPD.Age/365, 1) < 30 THEN '25-29'
    WHEN Round(IPD.Age/365, 1) >= 30 AND
Round(IPD.Age/365, 1) < 35 THEN '30-34'
    WHEN Round(IPD.Age/365, 1) >= 35 AND
Round(IPD.Age/365, 1) < 40 THEN '35-39'
    WHEN Round(IPD.Age/365, 1) >= 40 AND
Round(IPD.Age/365, 1) < 45 THEN '40-44'
    WHEN Round(IPD.Age/365, 1) >= 45 AND
Round(IPD.Age/365, 1) < 50 THEN '45-49'
    WHEN Round(IPD.Age/365, 1) >= 50 AND
Round(IPD.Age/365, 1) < 55 THEN '50-54'
    WHEN Round(IPD.Age/365, 1) >= 55 AND
Round(IPD.Age/365, 1) < 60 THEN '55-59'
    WHEN Round(IPD.Age/365, 1) >= 60 AND
Round(IPD.Age/365, 1) < 65 THEN '60-64'
    WHEN Round(IPD.Age/365, 1) >= 65 AND
Round(IPD.Age/365, 1) < 150 THEN '65+'
--
    ELSE 'Invalid Age'
END ,

    IPD.AgeModulo,
    IPD.FKdiagnosisID,
    IPD.Outcome

FROM [InPatientRegister] IPD JOIN
    [Facility] FAC ON IPD.FKFacilityID = FAC.PKFacilityID

WHERE
    (@SelYear IS NULL OR YEAR(IPD.DischargeDate) = @SelYear) AND
    (@StartingMonth IS NULL OR @EndMonth IS NULL) OR ((MONTH(IPD.DischargeDate)
>= @StartingMonth AND MONTH(IPD.DischargeDate) <= @EndMonth)) AND
    (@FacilityName IS NULL OR FAC.FacilityName = @FacilityName)

--
    AND (@intFKStateID IS NULL OR U.intFKStateID = @intFKStateID)
-- Insert statements for procedure here
END
GO

```

```
/****** Object: StoredProcedure [dbo].[spGETFacility] Script Date: 09/16/2009 10:23:12
*****/
```

```
SET ANSI_NULLS ON
```

```
GO
```

```
SET QUOTED_IDENTIFIER ON
```

```
GO
```

```
CREATE PROCEDURE [dbo].[spGETFacility]
```

```
    @PKFacilityID int = NULL,  
    @FKAreaID int = NULL,  
    @FKAgencyID int = NULL,  
    @Code varchar(50) = NULL,  
    @FacilityName varchar(255) = NULL,  
    @District varchar(255) = NULL
```

```
AS
```

```
BEGIN
```

```
    SET NOCOUNT ON
```

```
    SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
```

```
    SELECT
```

```
        PKFacilityID,  
        FKAreaID,  
        FKAgencyID,  
        Code,  
        FacilityName,  
        District
```

```
    FROM
```

```
        [Facility]
```

```
    WHERE
```

```
        (@PKFacilityID IS NULL OR PKFacilityID = @PKFacilityID)  
        AND (@FKAreaID IS NULL OR FKAreaID = @FKAreaID)  
        AND (@FKAgencyID IS NULL OR FKAgencyID = @FKAgencyID)  
        AND (@Code IS NULL OR Code = @Code)  
        AND (@FacilityName IS NULL OR FacilityName LIKE '%@FacilityName%')  
        AND (@District IS NULL OR District = @District)
```

```
    END
```

```
GO
```

```
/****** Object: ForeignKey [FK_ANCRegister_Facility] Script Date: 09/16/2009 10:23:14
*****/
```

```
ALTER TABLE [dbo].[ANCRegister] WITH CHECK ADD CONSTRAINT [FK_ANCRegister_Facility]  
FOREIGN KEY([FKFacilityID])
```

```
REFERENCES [dbo].[Facility] ([PKFacilityID])
```

```
GO
```

```
ALTER TABLE [dbo].[ANCRegister] CHECK CONSTRAINT [FK_ANCRegister_Facility]
```

```
GO
```

```
/****** Object: ForeignKey [FK_DeliveryChild_DeliveryRegister] Script Date: 09/16/2009  
10:23:14 *****/
```

```
ALTER TABLE [dbo].[DeliveryChild] WITH CHECK ADD CONSTRAINT  
[FK_DeliveryChild_DeliveryRegister] FOREIGN KEY([FKDeliveryRegisterID])
```

```
REFERENCES [dbo].[DeliveryRegister] ([PKDeliveryRegisterID])
```

```
ON DELETE CASCADE
```

```
GO
```

```
ALTER TABLE [dbo].[DeliveryChild] CHECK CONSTRAINT [FK_DeliveryChild_DeliveryRegister]
```

```
GO
```

/***** Object: ForeignKey [FK_DeliveryRegister_Facility] Script Date: 09/16/2009 10:23:14
*****/

```
ALTER TABLE [dbo].[DeliveryRegister] WITH CHECK ADD CONSTRAINT  
[FK_DeliveryRegister_Facility] FOREIGN KEY([FKFacilityID])  
REFERENCES [dbo].[Facility] ([PKFacilityID])
```

GO

```
ALTER TABLE [dbo].[DeliveryRegister] CHECK CONSTRAINT [FK_DeliveryRegister_Facility]
```

GO

/***** Object: ForeignKey [FK_DentalRegister_DentalDiagnosis] Script Date: 09/16/2009
10:23:14 *****/

```
ALTER TABLE [dbo].[DentalRegister] WITH CHECK ADD CONSTRAINT  
[FK_DentalRegister_DentalDiagnosis] FOREIGN KEY([FKDiagnosisID])  
REFERENCES [dbo].[DentalDiagnosis] ([PKDiagnosisID])
```

GO

```
ALTER TABLE [dbo].[DentalRegister] CHECK CONSTRAINT [FK_DentalRegister_DentalDiagnosis]
```

GO

/***** Object: ForeignKey [FK_DentalRegister_DentalProcedure] Script Date: 09/16/2009
10:23:14 *****/

```
ALTER TABLE [dbo].[DentalRegister] WITH CHECK ADD CONSTRAINT  
[FK_DentalRegister_DentalProcedure] FOREIGN KEY([FKProcedureID])  
REFERENCES [dbo].[DentalProcedure] ([PKProcedureID])
```

GO

```
ALTER TABLE [dbo].[DentalRegister] CHECK CONSTRAINT [FK_DentalRegister_DentalProcedure]
```

GO

/***** Object: ForeignKey [FK_DentalRegister_Facility] Script Date: 09/16/2009 10:23:14
*****/

```
ALTER TABLE [dbo].[DentalRegister] WITH CHECK ADD CONSTRAINT  
[FK_DentalRegister_Facility] FOREIGN KEY([FKFacilityID])  
REFERENCES [dbo].[Facility] ([PKFacilityID])
```

GO

```
ALTER TABLE [dbo].[DentalRegister] CHECK CONSTRAINT [FK_DentalRegister_Facility]
```

GO

/***** Object: ForeignKey [FK_Facility_Agency] Script Date: 09/16/2009 10:23:14 *****/

```
ALTER TABLE [dbo].[Facility] WITH CHECK ADD CONSTRAINT [FK_Facility_Agency] FOREIGN  
KEY([FKAgencyID])  
REFERENCES [dbo].[Agency] ([PKAgencyID])
```

GO

```
ALTER TABLE [dbo].[Facility] CHECK CONSTRAINT [FK_Facility_Agency]
```

GO

/***** Object: ForeignKey [FK_Facility_Area] Script Date: 09/16/2009 10:23:14 *****/

```
ALTER TABLE [dbo].[Facility] WITH CHECK ADD CONSTRAINT [FK_Facility_Area] FOREIGN  
KEY([FKAreaID])  
REFERENCES [dbo].[Area] ([PKAreaID])
```

GO

```
ALTER TABLE [dbo].[Facility] CHECK CONSTRAINT [FK_Facility_Area]
```

GO

/***** Object: ForeignKey [FK_HIV_HTC_Facility] Script Date: 09/16/2009 10:23:14
*****/

```
ALTER TABLE [dbo].[HIV_HTC] WITH CHECK ADD CONSTRAINT [FK_HIV_HTC_Facility]  
FOREIGN KEY([FKFacilityID])  
REFERENCES [dbo].[Facility] ([PKFacilityID])
```

GO

```
ALTER TABLE [dbo].[HIV_HTC] CHECK CONSTRAINT [FK_HIV_HTC_Facility]
```

```

GO
/***** Object: ForeignKey [FK_InPatientRegister_Facility]  Script Date: 09/16/2009 10:23:14
*****/
ALTER TABLE [dbo].[InPatientRegister] WITH CHECK ADD CONSTRAINT
[FK_InPatientRegister_Facility] FOREIGN KEY([FKFacilityID])
REFERENCES [dbo].[Facility] ([PKFacilityID])
GO
ALTER TABLE [dbo].[InPatientRegister] CHECK CONSTRAINT [FK_InPatientRegister_Facility]
GO
/***** Object: ForeignKey [FK_InPatientRegister_InpatientDiagnosis]  Script Date:
09/16/2009 10:23:14 *****/
ALTER TABLE [dbo].[InPatientRegister] WITH CHECK ADD CONSTRAINT
[FK_InPatientRegister_InpatientDiagnosis] FOREIGN KEY([FKdiagnosisID])
REFERENCES [dbo].[InpatientDiagnosis] ([PKDiagnosisID])
GO
ALTER TABLE [dbo].[InPatientRegister] CHECK CONSTRAINT
[FK_InPatientRegister_InpatientDiagnosis]
GO
/***** Object: ForeignKey [FK_MentalRegister_Facility]  Script Date: 09/16/2009 10:23:14
*****/
ALTER TABLE [dbo].[MentalRegister] WITH CHECK ADD CONSTRAINT
[FK_MentalRegister_Facility] FOREIGN KEY([FKFacilityID])
REFERENCES [dbo].[Facility] ([PKFacilityID])
GO
ALTER TABLE [dbo].[MentalRegister] CHECK CONSTRAINT [FK_MentalRegister_Facility]
GO
/***** Object: ForeignKey [FK_MentalRegister_MentalCondition]  Script Date: 09/16/2009
10:23:14 *****/
ALTER TABLE [dbo].[MentalRegister] WITH CHECK ADD CONSTRAINT
[FK_MentalRegister_MentalCondition] FOREIGN KEY([FKConditionID])
REFERENCES [dbo].[MentalCondition] ([PKConditionID])
GO
ALTER TABLE [dbo].[MentalRegister] CHECK CONSTRAINT [FK_MentalRegister_MentalCondition]
GO
/***** Object: ForeignKey [FK_MentalRegister_MentalStage]  Script Date: 09/16/2009
10:23:14 *****/
ALTER TABLE [dbo].[MentalRegister] WITH CHECK ADD CONSTRAINT
[FK_MentalRegister_MentalStage] FOREIGN KEY([FKStageID])
REFERENCES [dbo].[MentalStage] ([PKStageID])
GO
ALTER TABLE [dbo].[MentalRegister] CHECK CONSTRAINT [FK_MentalRegister_MentalStage]
GO
/***** Object: ForeignKey [FK_OutPatientRegister_Facility]  Script Date: 09/16/2009
10:23:14 *****/
ALTER TABLE [dbo].[OutPatientRegister] WITH CHECK ADD CONSTRAINT
[FK_OutPatientRegister_Facility] FOREIGN KEY([FKFacilityID])
REFERENCES [dbo].[Facility] ([PKFacilityID])
GO
ALTER TABLE [dbo].[OutPatientRegister] CHECK CONSTRAINT [FK_OutPatientRegister_Facility]
GO
/***** Object: ForeignKey [FK_OutPatientRegister_OutPatientCondition]  Script Date:
09/16/2009 10:23:14 *****/

```

```

ALTER TABLE [dbo].[OutPatientRegister] WITH CHECK ADD CONSTRAINT
[FK_OutPatientRegister_OutPatientCondition] FOREIGN KEY([FKConditionID])
REFERENCES [dbo].[OutPatientCondition] ([PKConditionID])
GO
ALTER TABLE [dbo].[OutPatientRegister] CHECK CONSTRAINT
[FK_OutPatientRegister_OutPatientCondition]
GO
/***** Object: ForeignKey [FK_ReportTypeName_ReportName]  Script Date: 09/16/2009
10:23:14 *****/
ALTER TABLE [dbo].[ReportTypeName] WITH CHECK ADD CONSTRAINT
[FK_ReportTypeName_ReportName] FOREIGN KEY([FKReportNameID])
REFERENCES [dbo].[ReportName] ([PKReportNameID])
GO
ALTER TABLE [dbo].[ReportTypeName] CHECK CONSTRAINT
[FK_ReportTypeName_ReportName]
GO
/***** Object: ForeignKey [FK_ReportTypeName_ReportType]  Script Date: 09/16/2009
10:23:14 *****/
ALTER TABLE [dbo].[ReportTypeName] WITH CHECK ADD CONSTRAINT
[FK_ReportTypeName_ReportType] FOREIGN KEY([FKReportTypeID])
REFERENCES [dbo].[ReportType] ([PKReportTypeID])
GO
ALTER TABLE [dbo].[ReportTypeName] CHECK CONSTRAINT [FK_ReportTypeName_ReportType]
GO
/***** Object: ForeignKey [FK_TBDetails_Facility]  Script Date: 09/16/2009 10:23:14
*****/
ALTER TABLE [dbo].[TBDetails] WITH CHECK ADD CONSTRAINT [FK_TBDetails_Facility]
FOREIGN KEY([FKFacilityID])
REFERENCES [dbo].[Facility] ([PKFacilityID])
GO
ALTER TABLE [dbo].[TBDetails] CHECK CONSTRAINT [FK_TBDetails_Facility]
GO
/***** Object: ForeignKey [FK_TBHIVActivities_Facility1]  Script Date: 09/16/2009 10:23:14
*****/
ALTER TABLE [dbo].[TBHIVActivities] WITH CHECK ADD CONSTRAINT
[FK_TBHIVActivities_Facility1] FOREIGN KEY([FKFacilityID])
REFERENCES [dbo].[Facility] ([PKFacilityID])
GO
ALTER TABLE [dbo].[TBHIVActivities] CHECK CONSTRAINT [FK_TBHIVActivities_Facility1]
GO
/***** Object: ForeignKey [FK_TBHIVActivities_TBHIVActivitiesTBName1]  Script Date:
09/16/2009 10:23:14 *****/
ALTER TABLE [dbo].[TBHIVActivities] WITH CHECK ADD CONSTRAINT
[FK_TBHIVActivities_TBHIVActivitiesTBName1] FOREIGN KEY([FKTBHIVActivitiesTBNameID])
REFERENCES [dbo].[TBHIVActivitiesTBName] ([PKTBHIVActivitiesTBNameID])
GO
ALTER TABLE [dbo].[TBHIVActivities] CHECK CONSTRAINT
[FK_TBHIVActivities_TBHIVActivitiesTBName1]
GO
/***** Object: ForeignKey [FK_TBLaboratoryActivities_Facility]  Script Date: 09/16/2009
10:23:14 *****/
ALTER TABLE [dbo].[TBLaboratoryActivities] WITH CHECK ADD CONSTRAINT
[FK_TBLaboratoryActivities_Facility] FOREIGN KEY([FKFacilityID])

```

```

REFERENCES [dbo].[Facility] ([PKFacilityID])
GO
ALTER TABLE [dbo].[TBLaboratoryActivities] CHECK CONSTRAINT
[FK_TBLaboratoryActivities_Facility]
GO
/***** Object: ForeignKey [FK_TBSputumConversion_Facility]  Script Date: 09/16/2009
10:23:14 *****/
ALTER TABLE [dbo].[TBSputumConversion] WITH CHECK ADD CONSTRAINT
[FK_TBSputumConversion_Facility] FOREIGN KEY([FKFacilityID])
REFERENCES [dbo].[Facility] ([PKFacilityID])
GO
ALTER TABLE [dbo].[TBSputumConversion] CHECK CONSTRAINT
[FK_TBSputumConversion_Facility]
GO
/***** Object: ForeignKey [FK_TBTreatmentOutcome_Facility]  Script Date: 09/16/2009
10:23:14 *****/
ALTER TABLE [dbo].[TBTreatmentOutcome] WITH CHECK ADD CONSTRAINT
[FK_TBTreatmentOutcome_Facility] FOREIGN KEY([FKFacilityID])
REFERENCES [dbo].[Facility] ([PKFacilityID])
GO
ALTER TABLE [dbo].[TBTreatmentOutcome] CHECK CONSTRAINT
[FK_TBTreatmentOutcome_Facility]
GO
/***** Object: ForeignKey [FK_TBTreatmentOutcome_TBTreatmentOutcomeCaseType]
Script Date: 09/16/2009 10:23:14 *****/
ALTER TABLE [dbo].[TBTreatmentOutcome] WITH CHECK ADD CONSTRAINT
[FK_TBTreatmentOutcome_TBTreatmentOutcomeCaseType] FOREIGN KEY([FKCaseTypeID])
REFERENCES [dbo].[TBTreatmentOutcomeCaseType] ([PKCaseTypeID])
GO
ALTER TABLE [dbo].[TBTreatmentOutcome] CHECK CONSTRAINT
[FK_TBTreatmentOutcome_TBTreatmentOutcomeCaseType]
GO
/***** Object: ForeignKey [FK_TBTreatmentOutcomeHIVActivities_Facility]  Script Date:
09/16/2009 10:23:14 *****/
ALTER TABLE [dbo].[TBTreatmentOutcomeHIVActivities] WITH CHECK ADD CONSTRAINT
[FK_TBTreatmentOutcomeHIVActivities_Facility] FOREIGN KEY([FKFacilityID])
REFERENCES [dbo].[Facility] ([PKFacilityID])
GO
ALTER TABLE [dbo].[TBTreatmentOutcomeHIVActivities] CHECK CONSTRAINT
[FK_TBTreatmentOutcomeHIVActivities_Facility]
GO
/***** Object: ForeignKey [FK_TBTreatmentOutcomeHIVActivities_TBHIVActivitiesTBName1]
Script Date: 09/16/2009 10:23:14 *****/
ALTER TABLE [dbo].[TBTreatmentOutcomeHIVActivities] WITH CHECK ADD CONSTRAINT
[FK_TBTreatmentOutcomeHIVActivities_TBHIVActivitiesTBName1] FOREIGN
KEY([FKTBHIVActivitiesTBNameID])
REFERENCES [dbo].[TBHIVActivitiesTBName] ([PKTBHIVActivitiesTBNameID])
GO
ALTER TABLE [dbo].[TBTreatmentOutcomeHIVActivities] CHECK CONSTRAINT
[FK_TBTreatmentOutcomeHIVActivities_TBHIVActivitiesTBName1]
GO

```

----- [END COPY HERE] -----

