**Name:** Yuanlong Zhang
**Date:** March. 20, 2022
**Course:** BIDD 320: Data Migration Techniques (Etl Processing)

# Final Project
## ETL Processing Manual

**Version: 1.0**

**Yuanlong Zhang**

## *Introduction*

This manual describes the essential components of our ETL process. The comprehensive ETL solution includes the following components:

- *Microsoft SSIS to provide visualization, a user-friendly interface, and simplify the ETL processes;*
- *Python for data pre-processing;*
- *MongoDB for reporting file upload;*
- *SQL agent jobs for automation;*
- *Microsoft SSRS for ETL/SSIS job reporting.*

The ETL solution provides some automation for a small medical clinic. It allows the company to automatically import the Patient and visit data from CSV files to the patient database manually import doctor schedule files. All the data will be organized in the centralized data warehouse for other reporting purposes.

**Our ETL Process Manual consists of 7 sections (see below).**

- *You can click the text to jump to the topic you are interested in.*
- *You can also use "bookmark" in the PDF reader to jump to the topic.*

# Contents

*Last Updated: Mar.20, 2022*

# 1. Background and Solution Overview

The ETL solution is designed for a small medical clinic for daily operations.

## 1.1. Background

**Currently**, the business has individual clinics send data to a corporate office by uploading CSV files each day. **Those files are then added to one of the two databases**. The current ETL process is **entirely manual**.

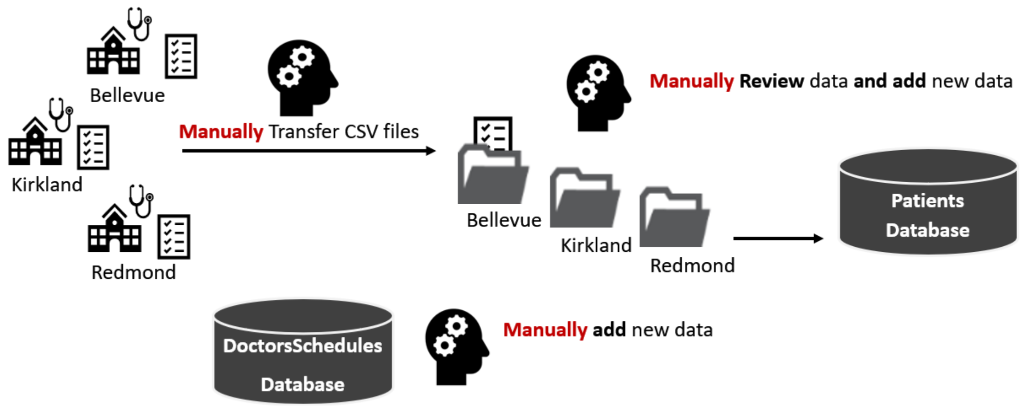
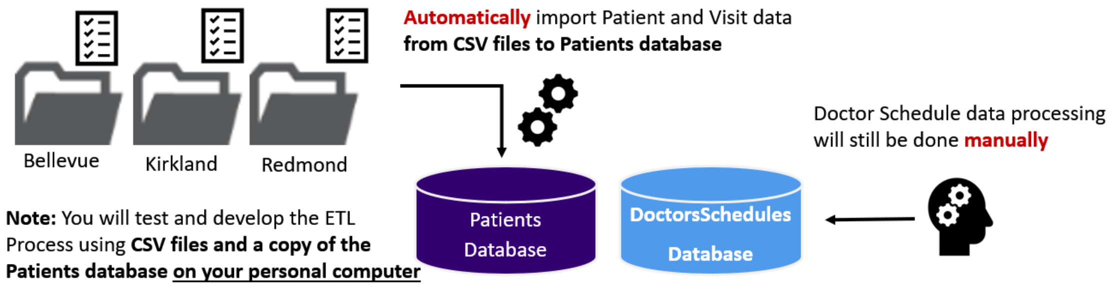

Bellevue
Kirkland
Redmond
Manually Transfer CSV files
Bellevue
Kirkland
Redmond
Manually Review data and add new data
Patients Database
DoctorsSchedules Database
Manually add new data

*Figure 1 General Processes of ETL (before)*

The **new** ETL process will **include some automation**. If all goes well, they may fully automate in the future.


Bellevue
Kirkland
Redmond
Automatically import Patient and Visit data from CSV files to Patients database
Doctor Schedule data processing will still be done manually
Patients Database
DoctorsSchedules Database

**Note:** You will test and develop the ETL Process using **CSV files and a copy of the Patients database** **on your personal computer**

*Figure 2 General Processes of ETL (After)*

## 1.2. ETL Solution Overview

Our ETL solutions consist of four milestones:

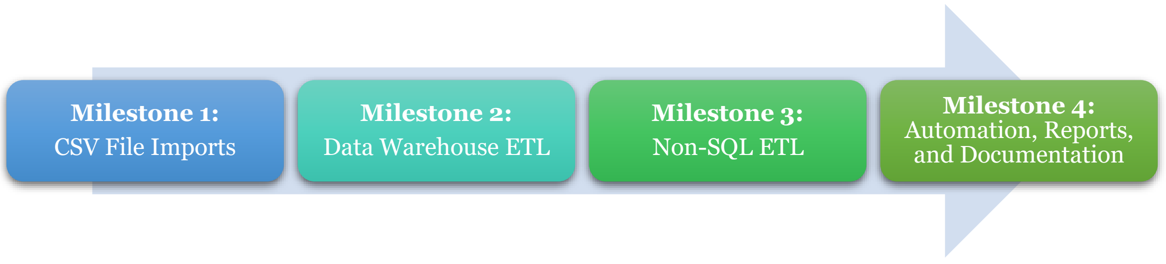

Milestone 1: CSV File Imports
Milestone 2: Data Warehouse ETL
Milestone 3: Non-SQL ETL
Milestone 4: Automation, Reports, and Documentation

*Figure 3 Overall Processes of the ETL Solution*

### *1.2.1. The detailed development processes*

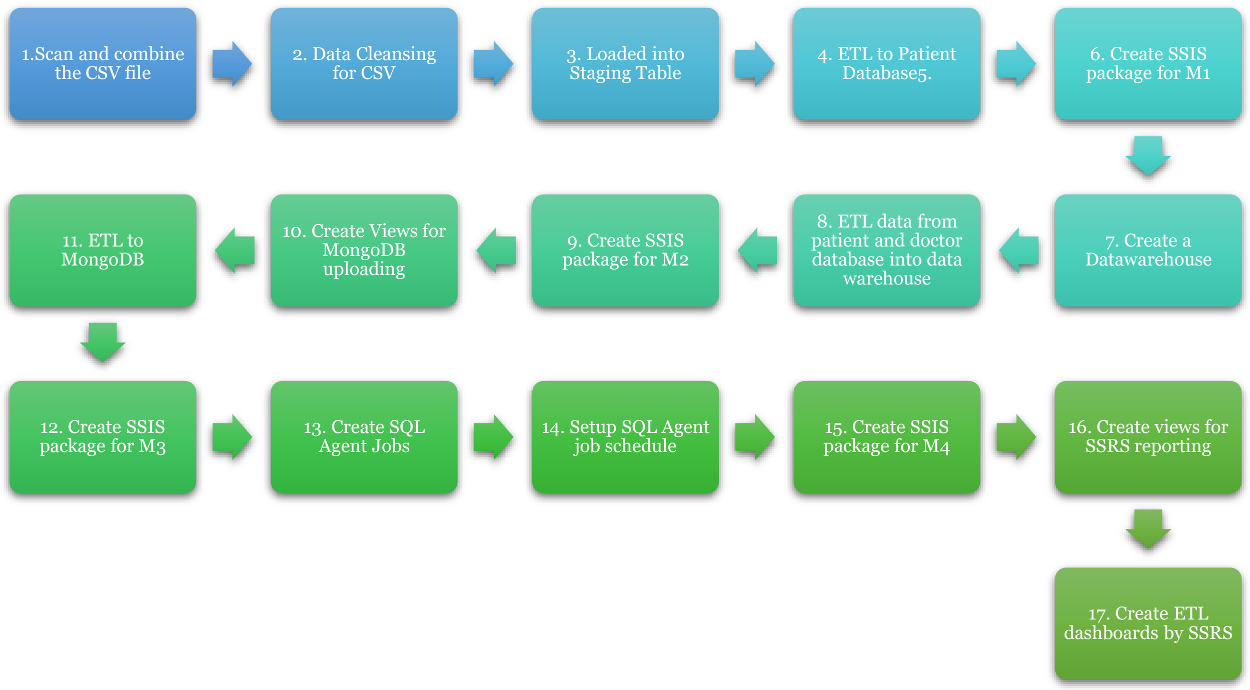

1.Scan and combine the CSV file
2. Data Cleansing for CSV
3. Loaded into Staging Table
4. ETL to Patient Database5.
6. Create SSIS package for M1
7. Create a Datawarehouse
8. ETL data from patient and doctor database into data warehouse
9. Create SSIS package for M2
10. Create Views for MongoDB uploading
11. ETL to MongoDB
12. Create SSIS package for M3
13. Create SQL Agent Jobs
14. Setup SQL Agent job schedule
15. Create SSIS package for M4
16. Create views for SSRS reporting
17. Create ETL dashboards by SSRS

*Figure 4 Detailed Steps in the ETL Solution*

### *1.2.2. The ETL files—SSIS ETL Objects*

| Object Name | Milestone | Description | Location |
|---|---|---|---|
| ETLFiletoDatabases.dtsx | 1 | Set of tasks that move data from files to staging tables | _BISolutions\ETLFinal_YuanlongZhang \ETLPackages |
| DWClinicReportDataETL.dtsx | 2 | Set of tasks that move data from database to data warehouse | _BISolutions\ETLFinal_YuanlongZhang \ETLPackages |
| ETLClincReportsDocumentData.dtsx | 3 | Set of tasks that move data from the data warehouse to MongoDB | _BISolutions\ETLFinal_YuanlongZhang \ETLPackages |
| ETLJob.dtsx | 4 | Set of tasks that will call all other three packages for SSIS job | _BISolutions\ETLFinal_YuanlongZhang \ETLPackages |

### *1.2.3. The ETL files—Non-SSIS ETL Objects*

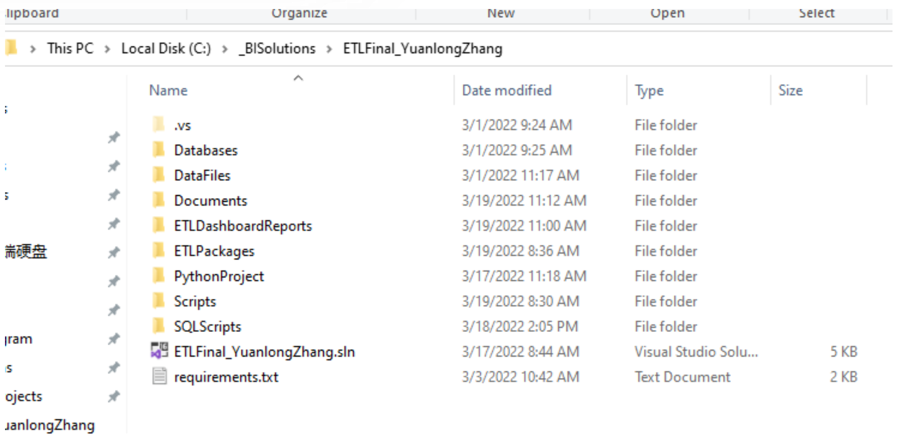
| Object Name | Milestone | Description | Location |
|---|---|---|---|
| PythonETL.py | 1 | Script that moves CSV report data to a Staging Database | _BISolutions\ETLFinal_YuanlongZhang\Scripts |
| MongoDBETL.py | 3 | Script that moves DW data to MongoDB | _BISolutions\ETLFinal_YuanlongZhang\Scripts |
| ETLDetails.rdl | 4 | Paginated report for ETL details information | _BISolutions\ETLFinal_YuanlongZhang\ETLDashboardReports |
| Maindashboard.rdl | 4 | Paginated report for the landing page (high-level summary) | _BISolutions\ETLFinal_YuanlongZhang\ETLDashboardReports |
| SSISJobDetails.rdl | 4 | Paginated report for SSIS Job information | _BISolutions\ETLFinal_YuanlongZhang\ETLDashboardReports |

# 2. Checklist for ETL Solutions

To make sure you can use the ETL solutions, we recommend you perform the following steps:

## 2.1. General Instructions

- All the ETL Solution files should be unzipped into C:_BISolutions\ETLFinal_YuanlongZhang

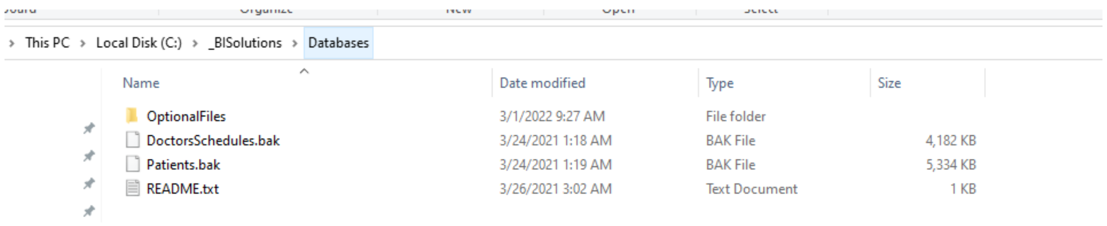


This PC › Local Disk (C:) › _BISolutions › ETLFinal_YuanlongZhang

| Name | Date modified | Type | Size |
|---|---|---|---|
| .vs | 3/1/2022 9:24 AM | File folder | |
| Databases | 3/1/2022 9:25 AM | File folder | |
| DataFiles | 3/1/2022 11:17 AM | File folder | |
| Documents | 3/19/2022 11:12 AM | File folder | |
| ETLDashboardReports | 3/19/2022 11:00 AM | File folder | |
| ETLPackages | 3/19/2022 8:36 AM | File folder | |
| PythonProject | 3/17/2022 11:18 AM | File folder | |
| Scripts | 3/19/2022 8:30 AM | File folder | |
| SQLScripts | 3/18/2022 2:05 PM | File folder | |
| ETLFinal_YuanlongZhang.sln | 3/17/2022 8:44 AM | Visual Studio Solu... | 5 KB |
| requirements.txt | 3/3/2022 10:42 AM | Text Document | 2 KB |

*Figure 5 Screenshot for the ETL Solution Folder*

- The database files (in \Databases) should be copied to C:_BISolutions\Databases; you should **restore** the Patients.bak in SSMS. The Patient database will be restored to the server.



This PC › Local Disk (C:) › _BISolutions › Databases

| Name | Date modified | Type | Size |
|---|---|---|---|
| OptionalFiles | 3/1/2022 9:27 AM | File folder | |
| DoctorsSchedules.bak | 3/24/2021 1:18 AM | BAK File | 4,182 KB |
| Patients.bak | 3/24/2021 1:19 AM | BAK File | 5,334 KB |
| README.txt | 3/26/2021 3:02 AM | Text Document | 1 KB |

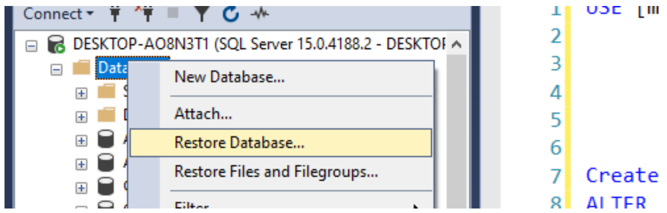
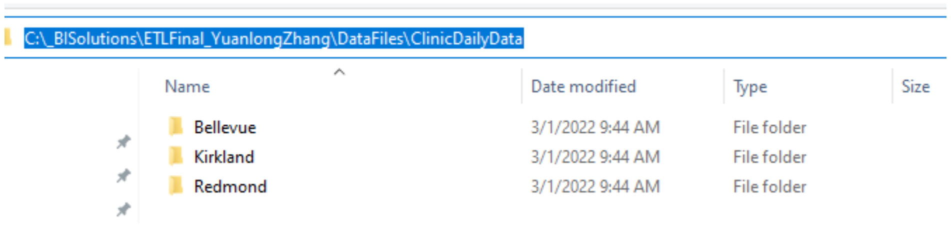

Connect
DESKTOP-AO8N3T1 (SQL Server 15.0.4188.2 - DESKTOP
Data
New Database...
Attach...
Restore Database...
Restore Files and Filegroups...
Create
ALTER

*Figure 6 Screenshot for Restore Patient Database*

## 2.2. Checklist for Milestone 1

- Before running Milestone 1 (**ETLFilesToDatabases.dtsx**), please load the updated datafile into C:_BISolutions\ETLFinal_YuanlongZhang\DataFiles\ClinicDailyData


C:_BISolutions\ETLFinal_YuanlongZhang\DataFiles\ClinicDailyData

| Name | Date modified | Type | Size |
|---|---|---|---|
| Bellevue | 3/1/2022 9:44 AM | File folder | |
| Kirkland | 3/1/2022 9:44 AM | File folder | |
| Redmond | 3/1/2022 9:44 AM | File folder | |

*Figure 7 Screenshot for the data source (csv)*

- Make sure you DISABLED the "Pre-load (Databases) Sequence Container **after the first execution**:

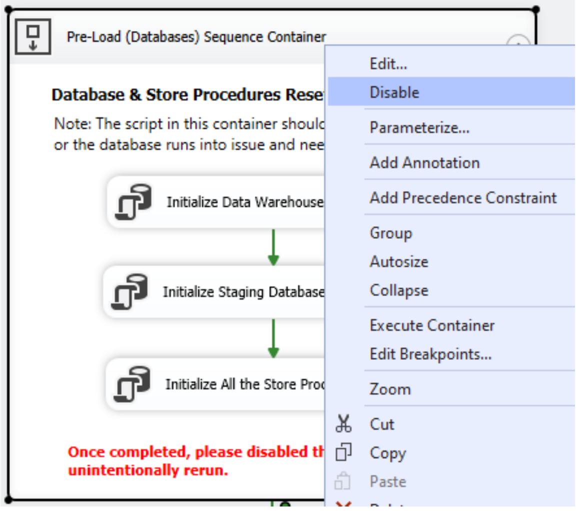

Pre-Load (Databases) Sequence Container
Database & Store Procedures Rese
Note: The script in this container shoulc
or the database runs into issue and nee
Initialize Data Warehouse
Initialize Staging Database
Initialize All the Store Proc
Once completed, please disabled th
unintentionally rerun.
Edit...
Disable
Parameterize...
Add Annotation
Add Precedence Constraint
Group
Autosize
Collapse
Execute Container
Edit Breakpoints...
Zoom
Cut
Copy
Paste

*Figure 8 Screenshot for how to disable containers*

## 2.3. Checklist for Milestone 2

- In Milestone 2 (**DWClinicReportDataETL.dtsx**), before execution, please make sure you have a stable internet connection and have correctly set up the VPN connection to UW:

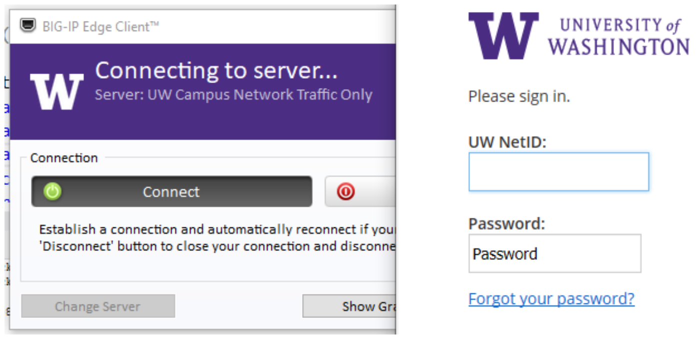
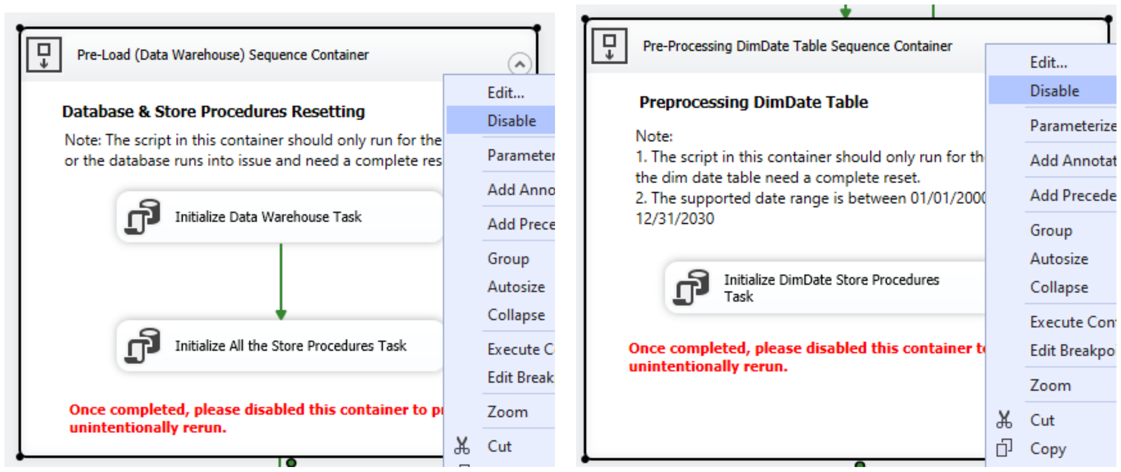


*Figure 9 Screenshot for how to connect to UW VPN*

- The Pre-Load (Data Warehouse) Sequence Container and the Pre-Processing DimDate Table Sequence Container should be run ONLY ONCE. After that, they should be DISABLED.


Pre-Load (Data Warehouse) Sequence Container

**Database & Store Procedures Resetting**

Note: The script in this container should only run for the
or the database runs into issue and need a complete res

Initialize Data Warehouse Task

Initialize All the Store Procedures Task

**Once completed, please disabled this container to p**
**unintentionally rerun.**

Pre-Processing DimDate Table Sequence Container

**Preprocessing DimDate Table**

Note:
1. The script in this container should only run for th
the dim date table need a complete reset.
2. The supported date range is between 01/01/2000
12/31/2030

Initialize DimDate Store Procedures Task

**Once completed, please disabled this container t**
**unintentionally rerun.**

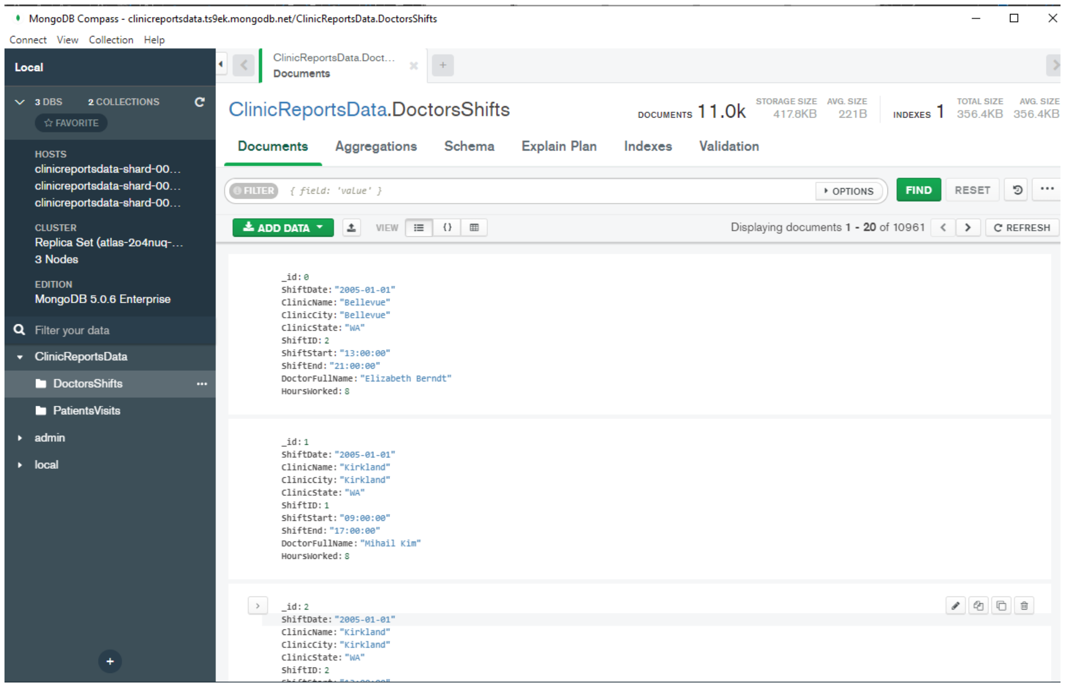
*Figure 10 Screenshot for how to disable containers*

## *2.4. Checklist for Milestone 3*

- Before execution of Milestone 3 (**ETLClinicReportsDocumentData.dtsx**), make sure you **have stable internet connection**.
- We've included the MongoDB connection string into the python script. ***In case you want to use MongoDB Compass***, the connection string is as follows:

```
mongodb+srv://BICert:BICert@clinicreportsdata.ts9ek.mongodb.net/test?retryWrites=true&w=majority
```

- If you set it up correctly, you should be able to access the MongoDB data through **MongoDB Compass** as follows after executing Milestone 3.


MongoDB Compass - clinicreportsdata.ts9ek.mongodb.net/ClinicReportsData.DoctorsShifts

ClinicReportsData.DoctorsShifts

DOCUMENTS 11.0k STORAGE SIZE 417.8KB AVG. SIZE 221B INDEXES 1 TOTAL SIZE 356.4KB AVG. SIZE 356.4KB

HOSTS
clinicreportsdata-shard-00...
clinicreportsdata-shard-00...
clinicreportsdata-shard-00...

CLUSTER
Replica Set (atlas-2o4nuq-...
3 Nodes

EDITION
MongoDB 5.0.6 Enterprise

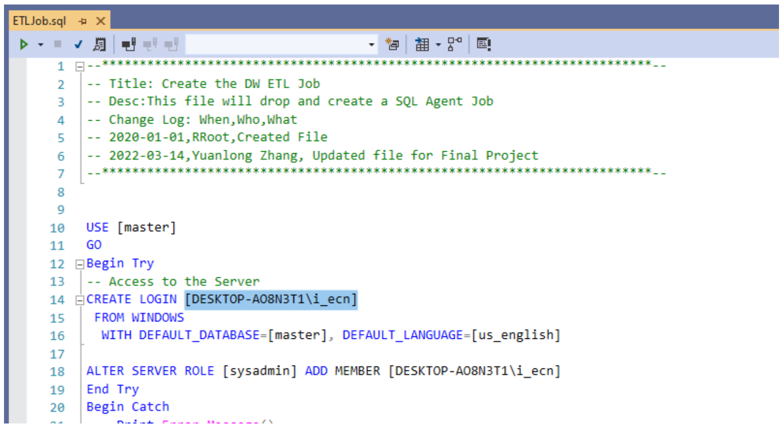
Displaying documents 1 - 20 of 10961

```
_id: 0
ShiftDate: "2005-01-01"
ClinicName: "Bellevue"
ClinicCity: "Bellevue"
ClinicState: "WA"
ShiftID: 2
ShiftStart: "13:00:00"
ShiftEnd: "21:00:00"
DoctorFullName: "Elizabeth Berndt"
HoursWorked: 8

_id: 1
ShiftDate: "2005-01-01"
ClinicName: "Kirkland"
ClinicCity: "Kirkland"
ClinicState: "WA"
ShiftID: 1
ShiftStart: "09:00:00"
ShiftEnd: "17:00:00"
DoctorFullName: "Mihail Kim"
HoursWorked: 8

_id: 2
ShiftDate: "2005-01-01"
ClinicName: "Kirkland"
ClinicCity: "Kirkland"
ClinicState: "WA"
ShiftID: 2
```

*Figure 11 Screenshot for MongoDB Compass*

## *2.5. Checklist for Milestone 4*

- Before running the **ETLJob.sql**, please make sure you have updated the user name to match your local configuration:



```
--**************************************************************************--
-- Title: Create the DW ETL Job
-- Desc:This file will drop and create a SQL Agent Job
-- Change Log: When,Who,What
-- 2020-01-01,RRoot,Created File
-- 2022-03-14,Yuanlong Zhang, Updated file for Final Project
--**************************************************************************--


USE [master]
GO
Begin Try
-- Access to the Server
CREATE LOGIN [DESKTOP-AO8N3T1\i_ecn]
 FROM WINDOWS
  WITH DEFAULT_DATABASE=[master], DEFAULT_LANGUAGE=[us_english]

ALTER SERVER ROLE [sysadmin] ADD MEMBER [DESKTOP-AO8N3T1\i_ecn]
End Try
Begin Catch
```

*Figure 12 Screenshot for SQL Script (ETLJob.sql)*

- Before executing the **ETLJob.sql**, please delete all the SQL Agent Jobs, Credentials, and SSIS Proxies you've ***created from the other students' script***.
- You can execute the SQL Agent Job in SSMS. Before execution, please make sure you run the SQL Agent Service:

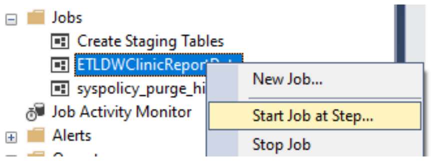
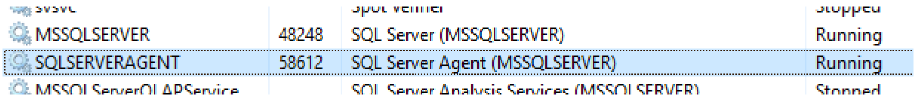


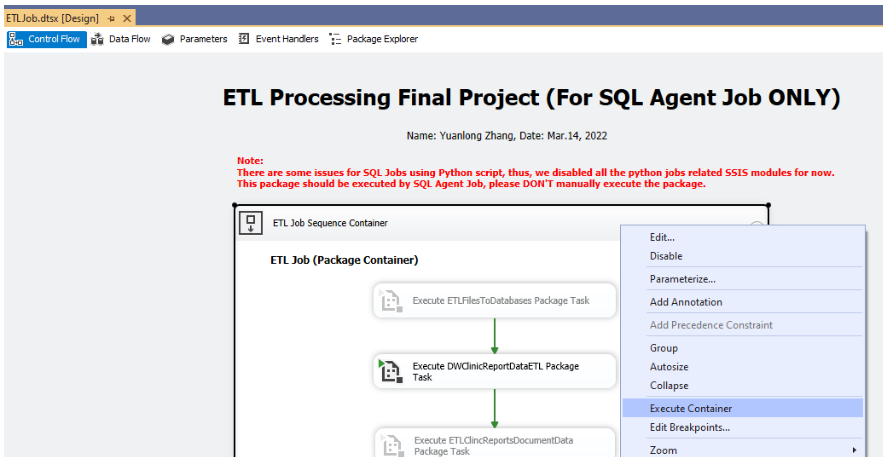
*Figure 13 Screenshot for how to start an SQL Agent Job manually*



| | | | |
|---|---|---|---|
| MSSQLSERVER | 48248 | SQL Server (MSSQLSERVER) | Running |
| SQLSERVERAGENT | 58612 | SQL Server Agent (MSSQLSERVER) | Running |

*Figure 14 Screenshot for how to check the service status for SQL Agent Service*

- In case you **<u>have any problems with the SQL Agent Job</u>**, you can **manually execute** the **ETLJob.dtsx**


ETLJob.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
ETL Job Sequence Container
ETL Job (Package Container)
Execute ETLFilesToDatabases Package Task
Execute DWClinicReportDataETL Package Task
Execute ETLClincReportsDocumentData Package Task
Edit...
Disable
Parameterize...
Add Annotation
Add Precedence Constraint
Group
Autosize
Collapse
Execute Container
Edit Breakpoints...
Zoom

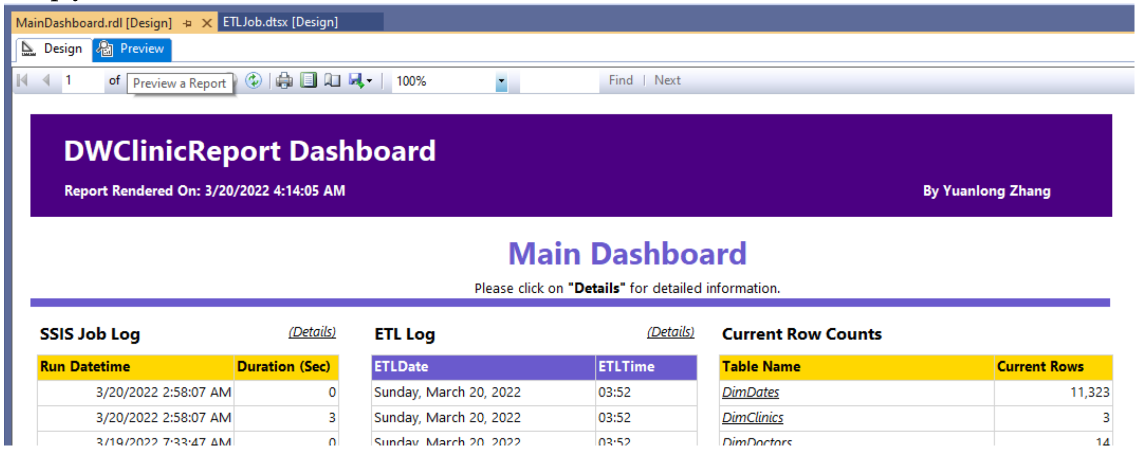
**ETL Processing Final Project (For SQL Agent Job ONLY)**

Name: Yuanlong Zhang, Date: Mar.14, 2022

**Note:**
**There are some issues for SQL Jobs using Python script, thus, we disabled all the python jobs related SSIS modules for now.**
**This package should be executed by SQL Agent Job, please DON'T manually execute the package.**

*Figure 15 Screenshot for Milestone 4: ETLJob.dtsx*

- Unless you've correctly installed and configured the SSRS server on your computer, you should directly open the SSRS reports in Visual Studio.
- Simply click on "MainDashboard.rdl" and switch to the "Preview" tab:



**DWClinicReport Dashboard**

Report Rendered On: 3/20/2022 4:14:05 AM

By Yuanlong Zhang

**Main Dashboard**

Please click on **"Details"** for detailed information.

**SSIS Job Log** (Details)

| Run Datetime | Duration (Sec) |
|---|---|
| 3/20/2022 2:58:07 AM | 0 |
| 3/20/2022 2:58:07 AM | 3 |
| 3/19/2022 7:33:47 AM | 0 |

**ETL Log** (Details)

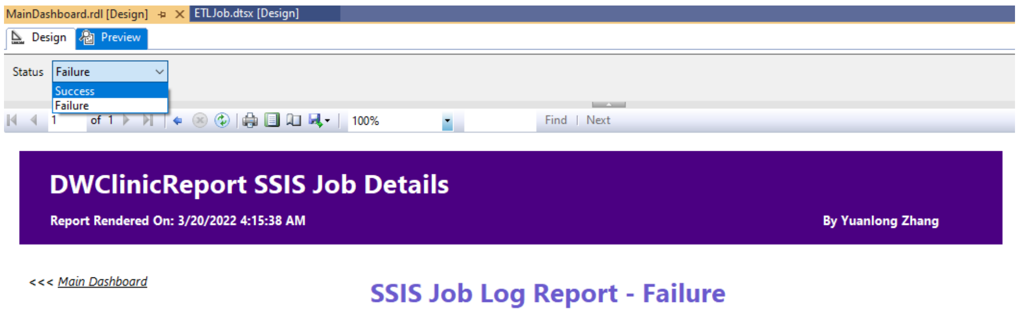
| ETLDate | ETLTime |
|---|---|
| Sunday, March 20, 2022 | 03:52 |
| Sunday, March 20, 2022 | 03:52 |
| Sunday, March 20, 2022 | 03:52 |

**Current Row Counts**

| Table Name | Current Rows |
|---|---|
| DimDates | 11,323 |
| DimClinics | 3 |
| DimDoctors | 14 |

*Figure 16 Screenshot for MainDashboard SSRS Paginated Report*

- You can jump to **SSISJobDetails.rdl** and **ETLDetails.rdl** by clicking "Details."
- In the SSIS Job Details dashboard, we ***display all the failure records (if they exist) by default***. If you want to view the **successful one**, use the slider on the top left corner and click "view report" on the top right corner:



**DWClinicReport SSIS Job Details**

Report Rendered On: 3/20/2022 4:15:38 AM

By Yuanlong Zhang

<<< Main Dashboard

**SSIS Job Log Report - Failure**

*Figure 17 Screenshot for SSISJobDetails SSRS Paginated Report (selected "failure")*

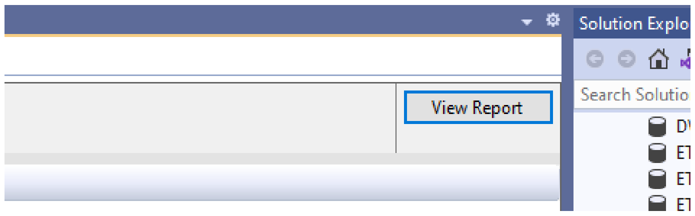


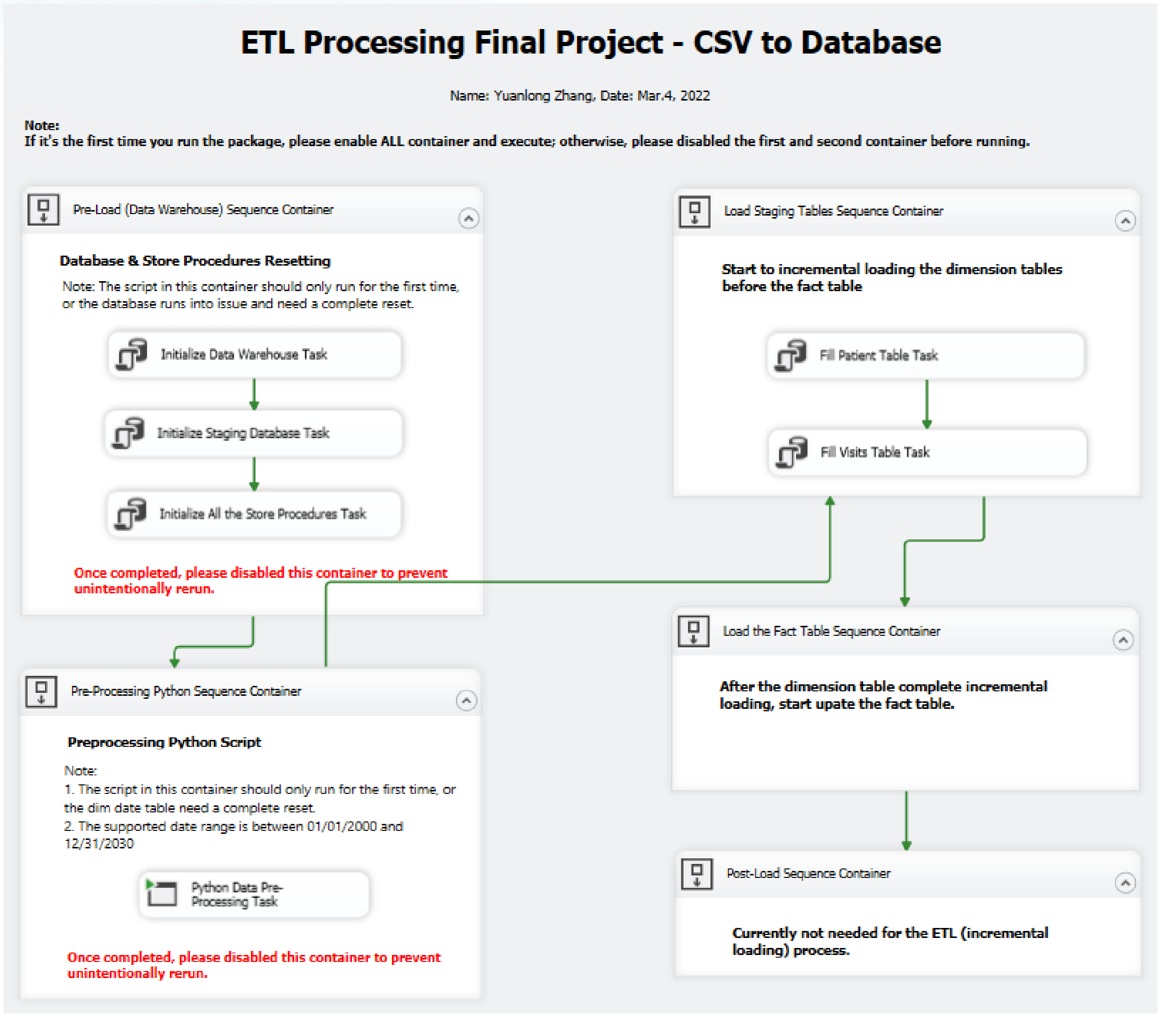
*Figure 18 Screenshot for "view report" button in SSRS preview mode*

# *3. ETL Process Overview*

There is 4 ETL package in the solution:

## *3.1. SSIS for Milestone 1*

Here's the overall structure of the first ETL processes designed for Milestone 1: CSV file imports process.



**ETL Processing Final Project - CSV to Database**

Name: Yuanlong Zhang, Date: Mar.4, 2022

**Note:**
**If it's the first time you run the package, please enable ALL container and execute; otherwise, please disabled the first and second container before running.**

Pre-Load (Data Warehouse) Sequence Container

**Database & Store Procedures Resetting**

Note: The script in this container should only run for the first time, or the database runs into issue and need a complete reset.

**Once completed, please disabled this container to prevent unintentionally rerun.**

Pre-Processing Python Sequence Container

**Preprocessing Python Script**

Note:
1. The script in this container should only run for the first time, or the dim date table need a complete reset.
2. The supported date range is between 01/01/2000 and 12/31/2030

**Once completed, please disabled this container to prevent unintentionally rerun.**

Load Staging Tables Sequence Container

**Start to incremental loading the dimension tables before the fact table**

Load the Fact Table Sequence Container

**After the dimension table complete incremental loading, start upate the fact table.**

Post-Load Sequence Container

**Currently not needed for the ETL (incremental loading) process.**

Initialize Data Warehouse Task
Initialize Staging Database Task
Initialize All the Store Procedures Task
Python Data Pre-Processing Task
Fill Patient Table Task
Fill Visits Table Task

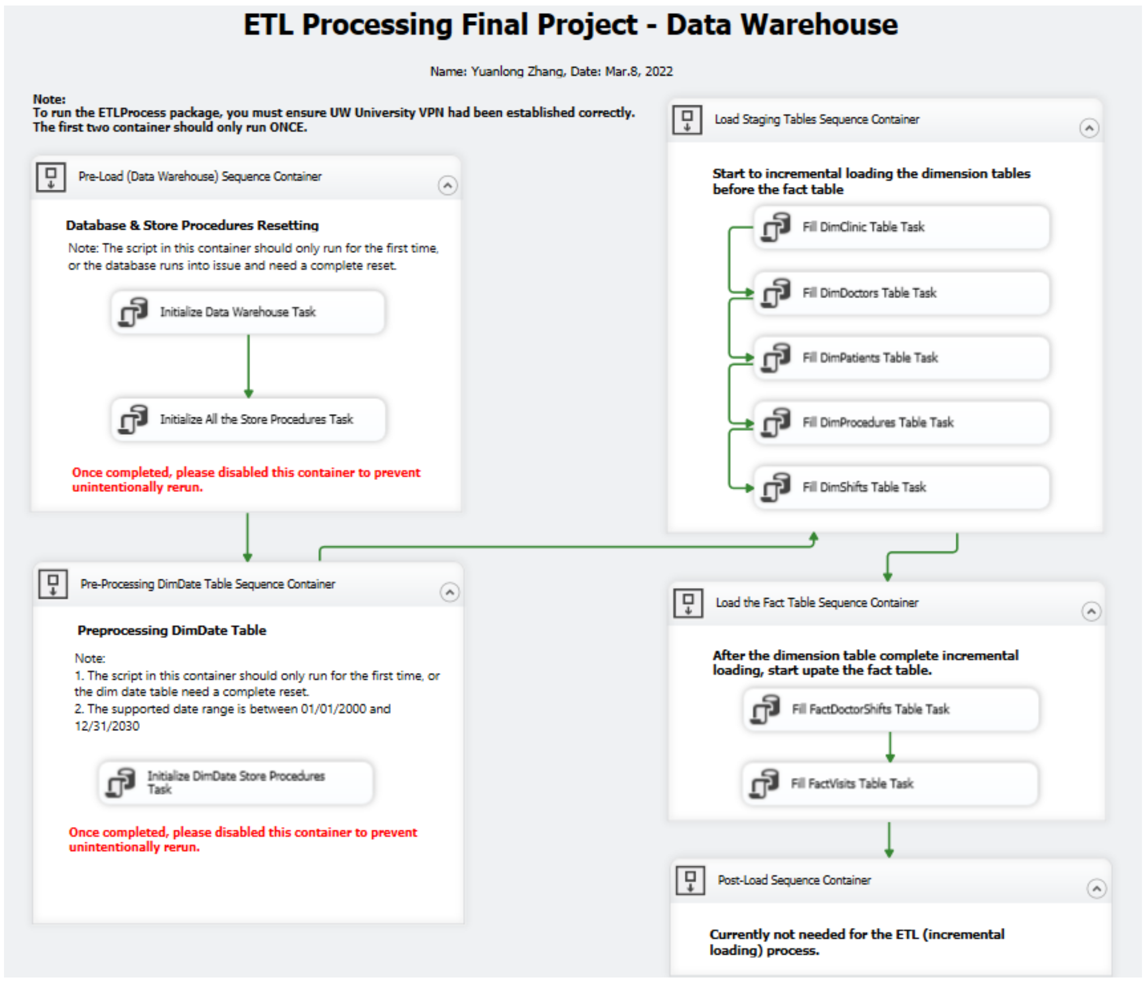
*Figure 19 Screenshot for SSIS Package for Milestone 1*

**Note:**

- We use several Sequence containers to execute the overall processes.
- To the detailed internal logic, we will discuss in the following part.

## *3.2. SSIS for Milestone 2*

Here's the overall structure of the second ETL process, which was designed for Milestone 2: Data Warehouse ETL.



**ETL Processing Final Project - Data Warehouse**

Name: Yuanlong Zhang, Date: Mar.8, 2022

**Note:**
**To run the ETLProcess package, you must ensure UW University VPN had been established correctly.**
**The first two container should only run ONCE.**

Pre-Load (Data Warehouse) Sequence Container

**Database & Store Procedures Resetting**
Note: The script in this container should only run for the first time, or the database runs into issue and need a complete reset.

Initialize Data Warehouse Task

Initialize All the Store Procedures Task

**Once completed, please disabled this container to prevent unintentionally rerun.**

Pre-Processing DimDate Table Sequence Container

**Preprocessing DimDate Table**
Note:
1. The script in this container should only run for the first time, or the dim date table need a complete reset.
2. The supported date range is between 01/01/2000 and 12/31/2030

Initialize DimDate Store Procedures Task

**Once completed, please disabled this container to prevent unintentionally rerun.**

Load Staging Tables Sequence Container

**Start to incremental loading the dimension tables before the fact table**

Fill DimClinic Table Task

Fill DimDoctors Table Task

Fill DimPatients Table Task

Fill DimProcedures Table Task

Fill DimShifts Table Task

Load the Fact Table Sequence Container

**After the dimension table complete incremental loading, start upate the fact table.**

Fill FactDoctorShifts Table Task

Fill FactVisits Table Task

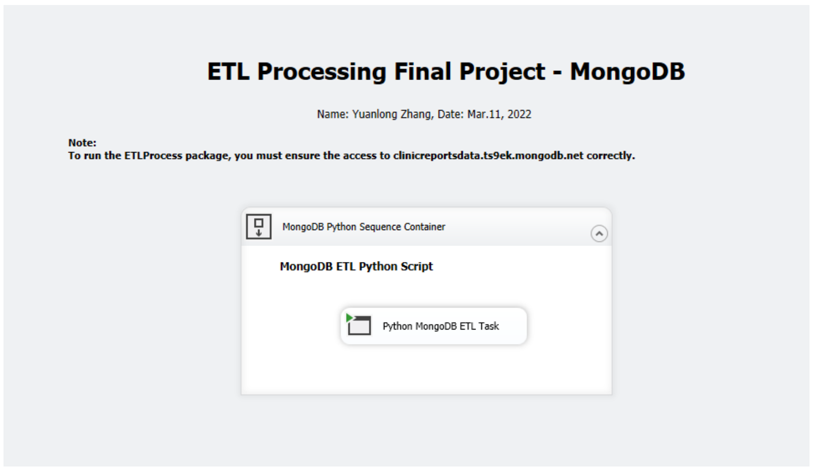
Post-Load Sequence Container

**Currently not needed for the ETL (incremental loading) process.**

*Figure 20 Screenshot for SSIS Package for Milestone 2*

## *3.3. SSIS for Milestone 3*

Here's the overall structure of the third ETL process, which was designed for Milestone 3: Non-SQL ETL.



**ETL Processing Final Project - MongoDB**

Name: Yuanlong Zhang, Date: Mar.11, 2022

**Note:**
**To run the ETLProcess package, you must ensure the access to clinicreportsdata.ts9ek.mongodb.net correctly.**

MongoDB Python Sequence Container

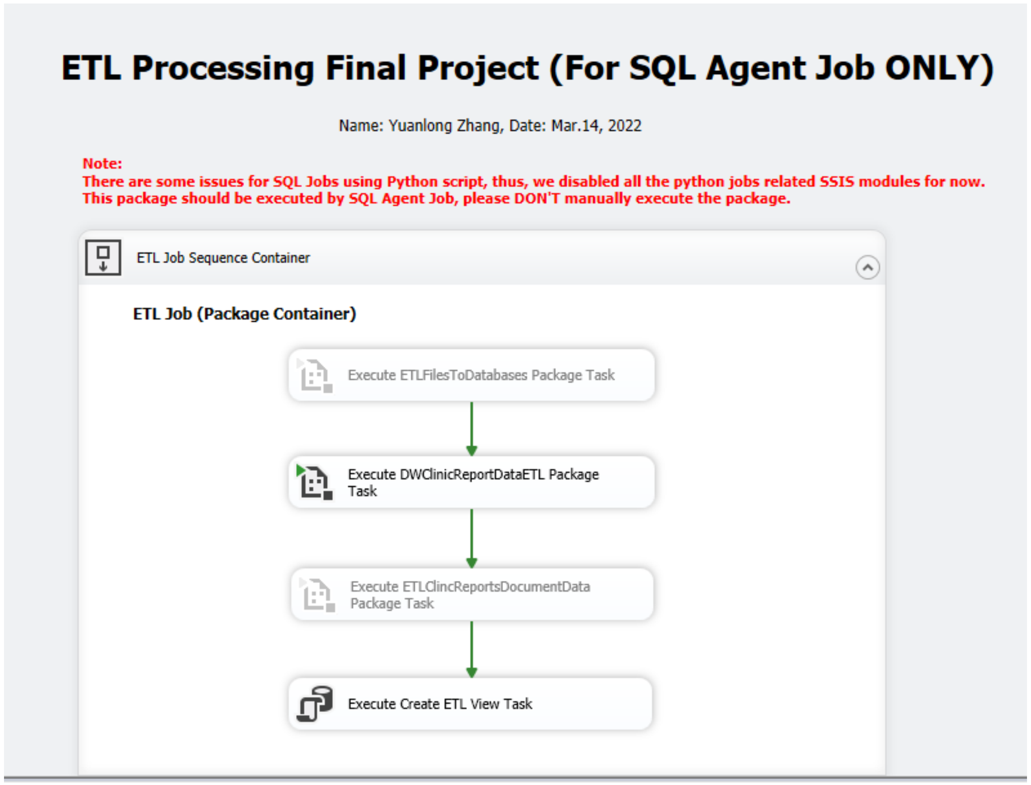
**MongoDB ETL Python Script**

Python MongoDB ETL Task

*Figure 21 Screenshot for SSIS Package for Milestone 3*

## *3.4. SSIS for Milestone 4*

Here's the overall structure of the fourth ETL process, which was designed for Milestone 4: Automation, reports, and documentation.



**ETL Processing Final Project (For SQL Agent Job ONLY)**

Name: Yuanlong Zhang, Date: Mar.14, 2022

**Note:**
**There are some issues for SQL Jobs using Python script, thus, we disabled all the python jobs related SSIS modules for now.**
**This package should be executed by SQL Agent Job, please DON'T manually execute the package.**

ETL Job Sequence Container
ETL Job (Package Container)
Execute ETLFilesToDatabases Package Task
Execute DWClinicReportDataETL Package Task
Execute ETLClincReportsDocumentData Package Task
Execute Create ETL View Task

*Figure 22 Screenshot for SSIS Package for Milestone 4*

# 4. ETL Process Objects

Our ETL process uses Microsoft SSIS, consisting of 4 package files.

## 4.1. SSIS Package for Milestone 1

Here's the list of files for our ETL processes for Milestone 1:

| | File Name | Description |
|---|---|---|
| 1 | ETLFilesToDatabases.dtsx | Set of tasks that move data from files to staging tables.<br>*Note: _BISolutions\ETLFinal_YuanlongZhang\ETLPackages* |
| 2 | PythonETL.py | Script that moves CSV report data to a Staging Database.<br>*Note: _BISolutions\ETLFinal_YuanlongZhang\Scripts* |
| 3 | ETLLoading.sql | SQL Script for staging table/store procedures definition and transformation to update patient data.<br>*Note: _BISolutions\ETLFinal_YuanlongZhang\SQLScripts* |

## 4.2. ETL Visualization for Milestone 1

We use Microsoft's SQL Server Integration Services (SSIS) to visualize our ETL process. SSIS includes a set of commands represented by visual objects, called tasks, constraints, containers, or transformations. The most common objects used in our ETL process are Sequence Containers, Execute SQL tasks, Precedent Constraints, and Connections.

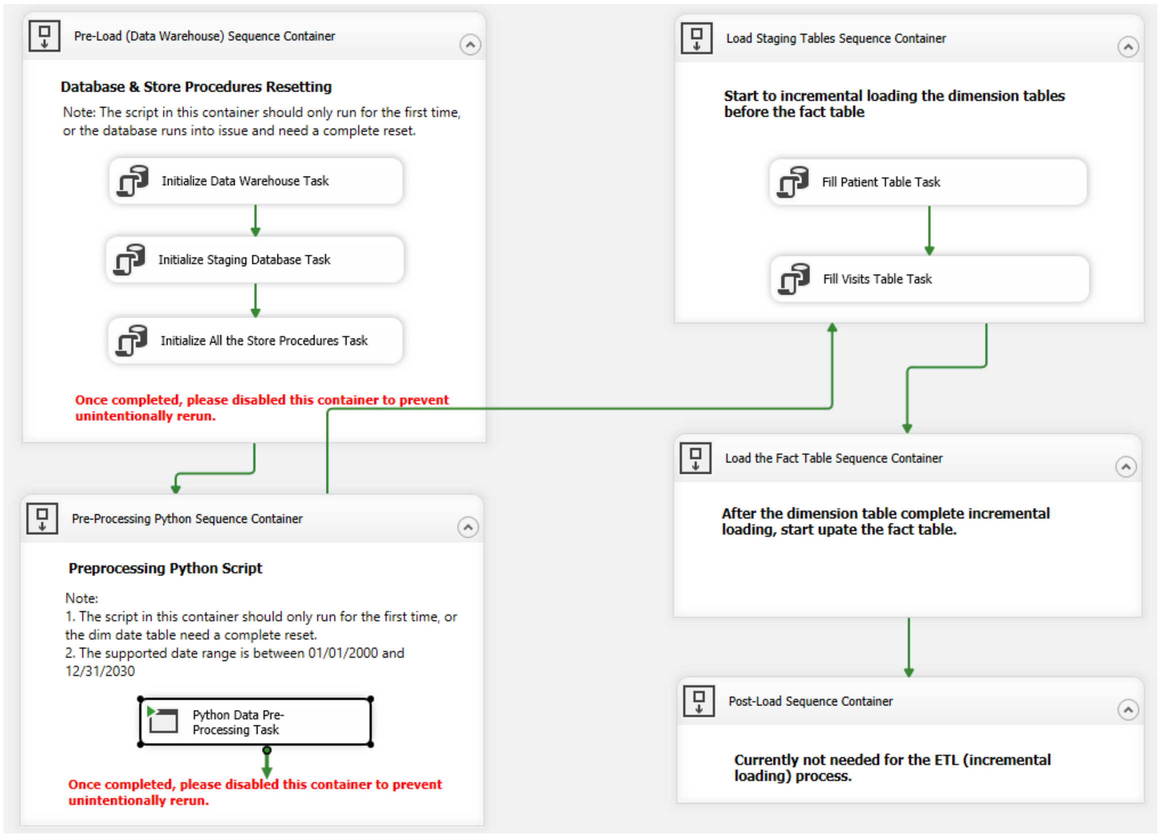

Pre-Load (Data Warehouse) Sequence Container
Database & Store Procedures Resetting
Note: The script in this container should only run for the first time, or the database runs into issue and need a complete reset.
Initialize Data Warehouse Task
Initialize Staging Database Task
Initialize All the Store Procedures Task
Once completed, please disabled this container to prevent unintentionally rerun.
Pre-Processing Python Sequence Container
Preprocessing Python Script
Note:
1. The script in this container should only run for the first time, or the dim date table need a complete reset.
2. The supported date range is between 01/01/2000 and 12/31/2030
Python Data Pre-Processing Task
Once completed, please disabled this container to prevent unintentionally rerun.
Load Staging Tables Sequence Container
Start to incremental loading the dimension tables before the fact table
Fill Patient Table Task
Fill Visits Table Task
Load the Fact Table Sequence Container
After the dimension table complete incremental loading, start upate the fact table.
Post-Load Sequence Container
Currently not needed for the ETL (incremental loading) process.

*Figure 23 Overview of SSIS package: Control Flow*

**Note:**

- These visualization objects are stored and configured in an SSIS Package file.

### *4.2.1. Connections*

We use a Connection object that connects to the original data source file, invalid phone number file, and data warehouse that holds our SQL ETL objects.

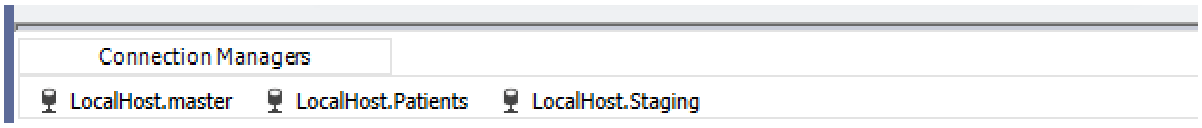

Connection Managers
LocalHost.master LocalHost.Patients LocalHost.Staging

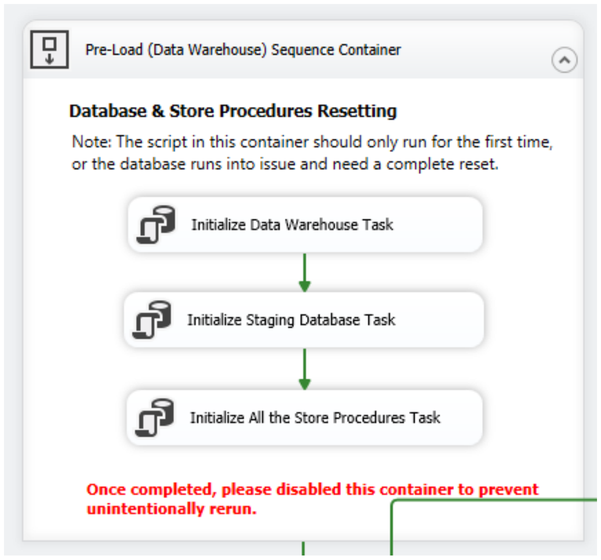
*Figure 24 Connections for Milestone 1*

We used OLE DB connections; here's the detailed information:

| No. | Name | Description |
|---|---|---|
| 1 | localhost.master | This is an OLE DB connection.<br>Will connect to localhost\master |
| 2 | localhost.patients | This is an OLE DB connection.<br>Will connect to localhost\patients |
| 3 | localhost.staging | This is an OLE DB connection.<br>Will connect to localhost\staging |

### *4.2.2. Pre-Load (Data Warehouse) Sequence Container*

There are three components inside the sequence container.


Pre-Load (Data Warehouse) Sequence Container

**Database & Store Procedures Resetting**

Note: The script in this container should only run for the first time, or the database runs into issue and need a complete reset.

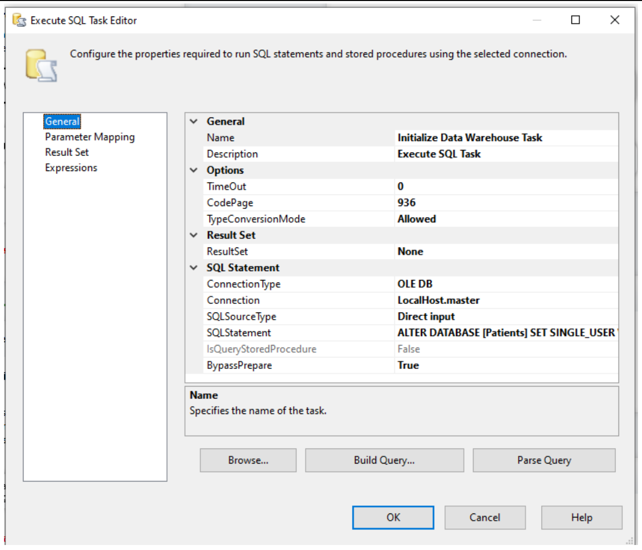
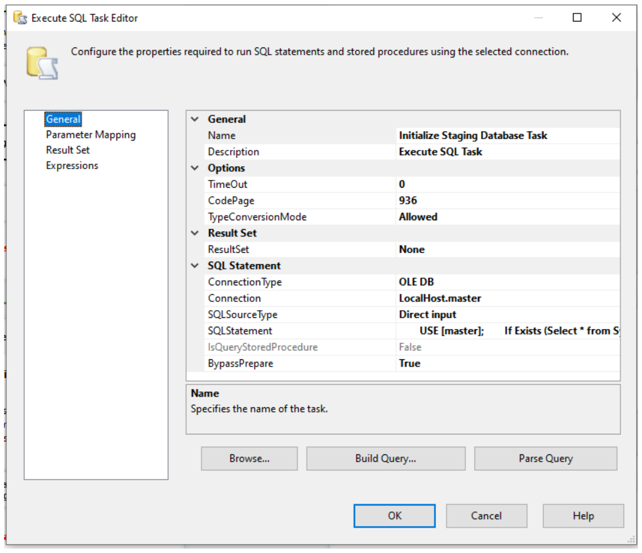
Initialize Data Warehouse Task
Initialize Staging Database Task
Initialize All the Store Procedures Task

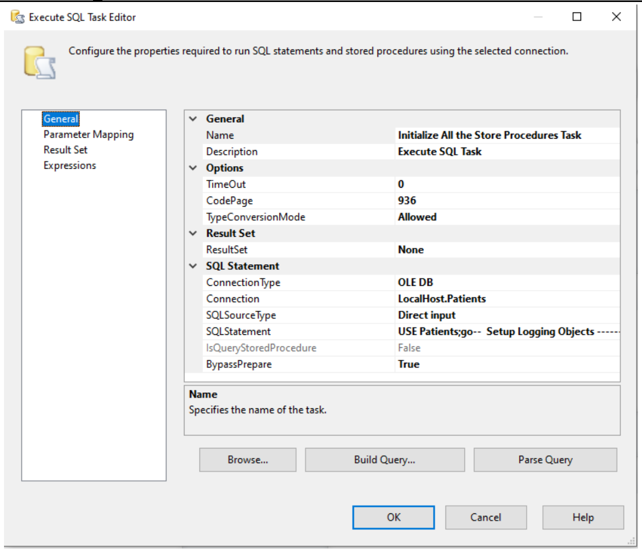
**Once completed, please disabled this container to prevent unintentionally rerun.**

*Figure 25 Pre-Load Sequence Container in Milestone 1*

**Note:**
- This container should only run ONCE.

| No. | Name | Description |
|---|---|---|
| 1 | Initialize Data Warehouse Task Initialize Data Warehouse Task | This uses the script to initialize all the database files:<br>It creates the local copy for the patient database. |

| No. | Name | Description |
|---|---|---|
| | |  Execute SQL Task Editor<br>Configure the properties required to run SQL statements and stored procedures using the selected connection.<br>General<br>Parameter Mapping<br>Result Set<br>Expressions<br>**General**<br>Name: Initialize Data Warehouse Task<br>Description: Execute SQL Task<br>**Options**<br>TimeOut: 0<br>CodePage: 936<br>TypeConversionMode: Allowed<br>**Result Set**<br>ResultSet: None<br>**SQL Statement**<br>ConnectionType: OLE DB<br>Connection: LocalHost.master<br>SQLSourceType: Direct input<br>SQLStatement: ALTER DATABASE [Patients] SET SINGLE_USER<br>IsQueryStoredProcedure: False<br>BypassPrepare: True<br>**Name**<br>Specifies the name of the task.<br>Browse... Build Query... Parse Query<br>OK Cancel Help |
| 2 | Initialize Staging Database Task | Used to create all the staging tables for ETL. The tables are designed to temporarily load the data and transform it from CSV files before loading it into the database.<br>The script is available in ETLLoading.sql  Execute SQL Task Editor<br>Configure the properties required to run SQL statements and stored procedures using the selected connection.<br>General<br>Parameter Mapping<br>Result Set<br>Expressions<br>**General**<br>Name: Initialize Staging Database Task<br>Description: Execute SQL Task<br>**Options**<br>TimeOut: 0<br>CodePage: 936<br>TypeConversionMode: Allowed<br>**Result Set**<br>ResultSet: None<br>**SQL Statement**<br>ConnectionType: OLE DB<br>Connection: LocalHost.master<br>SQLSourceType: Direct input<br>SQLStatement: USE [master]; If Exists (Select * from S<br>IsQueryStoredProcedure: False<br>BypassPrepare: True<br>**Name**<br>Specifies the name of the task.<br>Browse... Build Query... Parse Query<br>OK Cancel Help |
| 3 | Initialize all the store procedures task | Used to create all the views and store procedures for ETL processes.<br>To be detailed, we created views for each table and the store procedures to load the data from views.<br>The script is available in ETLLoading.sql |

| No. | Name | Description |
|---|---|---|
| | |  We will discuss the transformations in the later part. |

Execute SQL Task Editor

Configure the properties required to run SQL statements and stored procedures using the selected connection.

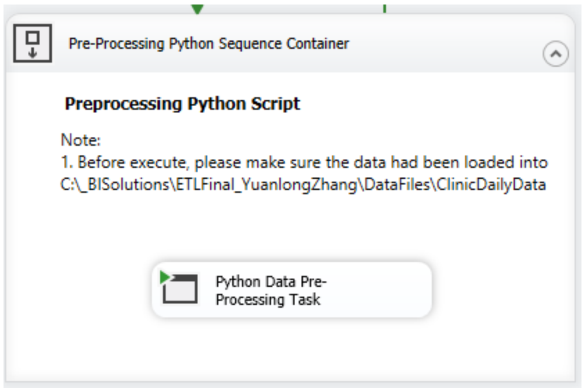
General
Parameter Mapping
Result Set
Expressions

| | |
|---|---|
| **General** | |
| Name | Initialize All the Store Procedures Task |
| Description | Execute SQL Task |
| **Options** | |
| TimeOut | 0 |
| CodePage | 936 |
| TypeConversionMode | Allowed |
| **Result Set** | |
| ResultSet | None |
| **SQL Statement** | |
| ConnectionType | OLE DB |
| Connection | LocalHost.Patients |
| SQLSourceType | Direct input |
| SQLStatement | USE Patients;go-- Setup Logging Objects ------ |
| IsQueryStoredProcedure | False |
| BypassPrepare | True |

**Name**
Specifies the name of the task.

## *4.2.3. Pre-Processing Python Sequence Container*

There is one component in the container.



Pre-Processing Python Sequence Container

**Preprocessing Python Script**

Note:
1. Before execute, please make sure the data had been loaded into C:_BISolutions\ETLFinal_YuanlongZhang\DataFiles\ClinicDailyData

*Figure 26 Preprocessing Python Script in Milestone 1*

**Note:**
- The data file should be loaded into c:_BISolutions\ETLFinal_YuanlongZhang\DataFiles\ClinicDailyData
- We will analyze the python script in a later part.

## *4.2.4. Load Staging Tables Sequence Container*

There are two components in the container.

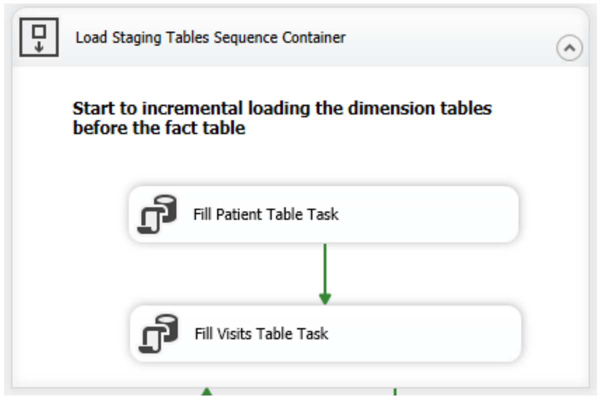
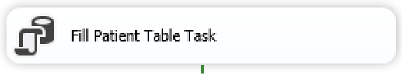
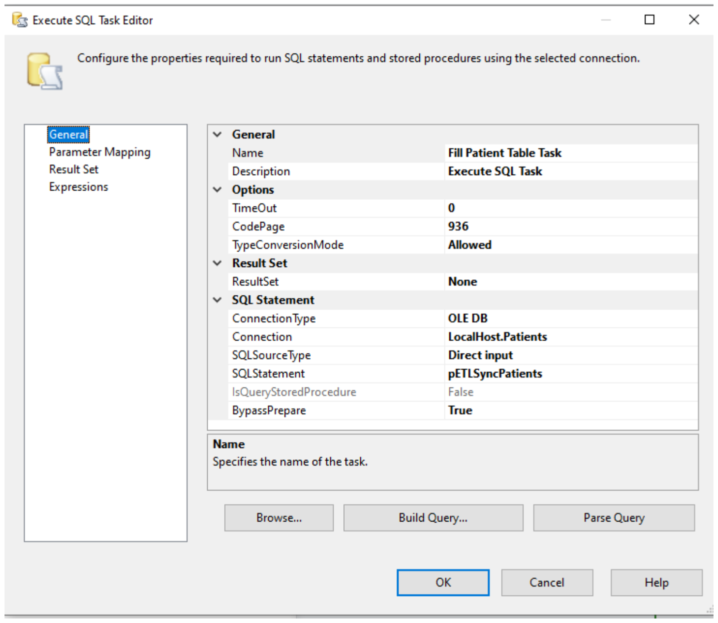
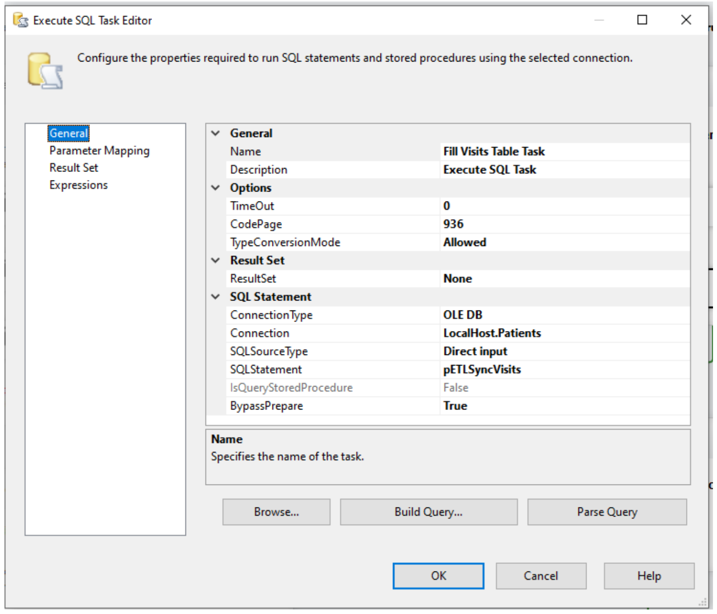

Load Staging Tables Sequence Container
Start to incremental loading the dimension tables before the fact table
Fill Patient Table Task
Fill Visits Table Task

*Figure 27 Load Staging Tables Sequence Container in Milestone 1*

| No | Name | Description |
|---|---|---|
| 1 | Fill Patient Table Task  | Used to fill data into the patient table.<br>It executes pETLSyncPatients store-procedure<br><br>Execute SQL Task Editor<br>Configure the properties required to run SQL statements and stored procedures using the selected connection.<br>**General**: Name: **Fill Patient Table Task**; Description: **Execute SQL Task**<br>**Options**: TimeOut: **0**; CodePage: **936**; TypeConversionMode: **Allowed**<br>**Result Set**: ResultSet: **None**<br>**SQL Statement**: ConnectionType: **OLE DB**; Connection: **LocalHost.Patients**; SQLSourceType: **Direct input**; SQLStatement: **pETLSyncPatients**; IsQueryStoredProcedure: False; BypassPrepare: **True**<br>**Name** Specifies the name of the task.<br>We will provide more details about the script in the next section. |
| 2 | Fill Visits Table Task | Used to fill data into visit table.<br>It executes pETLSyncVisits store-procedure |

| No | Name | Description |
|---|---|---|
| | |  Execute SQL Task Editor — Configure the properties required to run SQL statements and stored procedures using the selected connection. General: Name = Fill Visits Table Task; Description = Execute SQL Task. Options: TimeOut = 0; CodePage = 936; TypeConversionMode = Allowed. Result Set: ResultSet = None. SQL Statement: ConnectionType = OLE DB; Connection = LocalHost.Patients; SQLSourceType = Direct input; SQLStatement = pETLSyncVisits; IsQueryStoredProcedure = False; BypassPrepare = True. Name: Specifies the name of the task. We will provide more details about the script in the next section. |

## 4.3. SSIS Package for Milestone 2

Here's the list of files for our ETL processes for Milestone 2:

| | File Name | Description |
|---|---|---|
| 1 | DWClinicReportDataETL.dtsx | Set of tasks that move data from database to data warehouse. *Note: _BISolutions\ETLFinal_YuanlongZhang\ETLPackages* |
| 2 | Create DWClinicReportData.sql | SQL Script that creates the data warehouse. *Note: _BISolutions\ETLFinal_YuanlongZhang\SQLScripts* |
| 3 | DWETLObjects.sql | SQL Script for views/store procedures definition and transformation to ETL the data from database to data warehouse. *Note: _BISolutions\ETLFinal_YuanlongZhang\SQLScripts* |

## *4.4. Data Warehouse Design for Milestone 2*

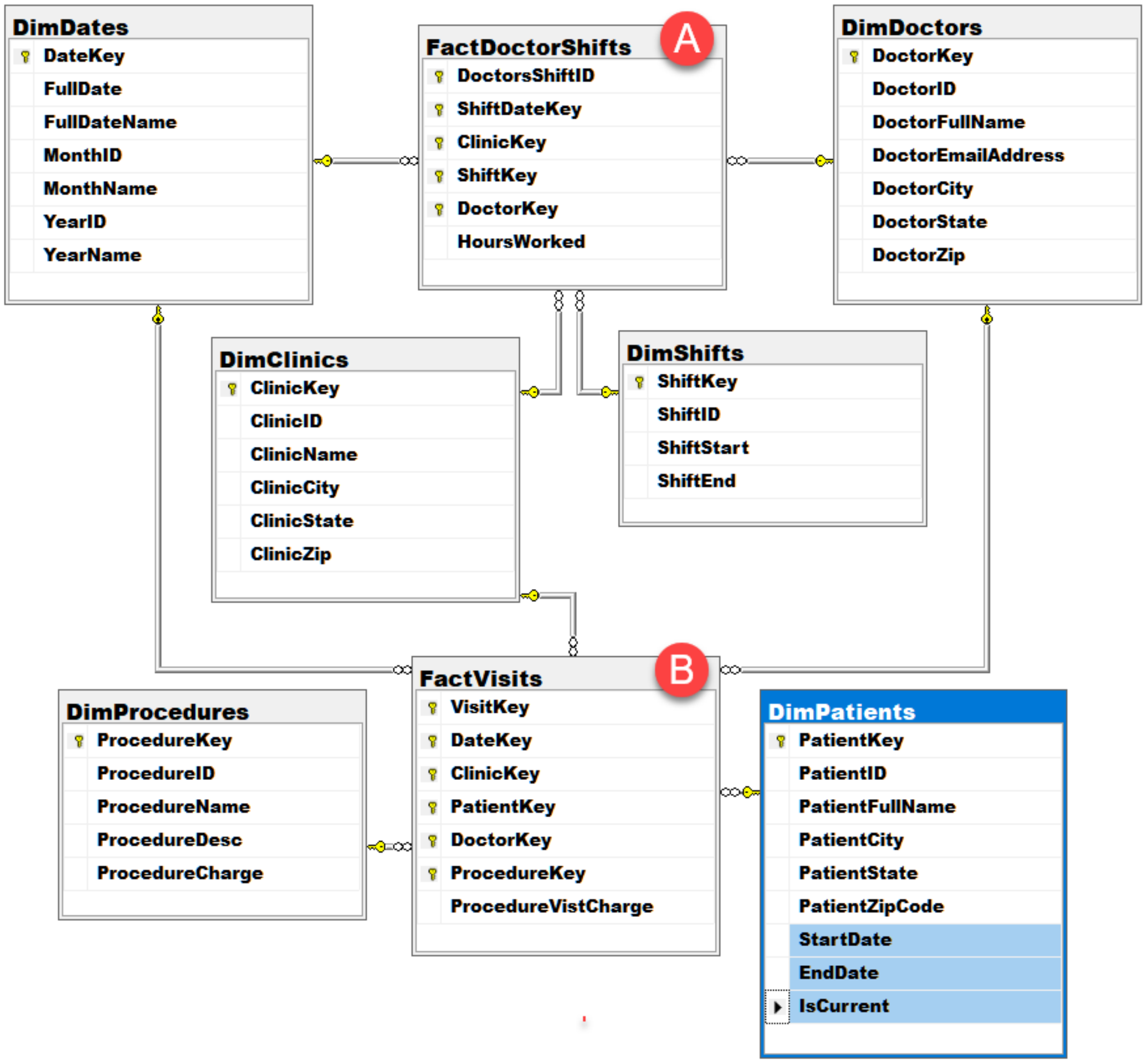
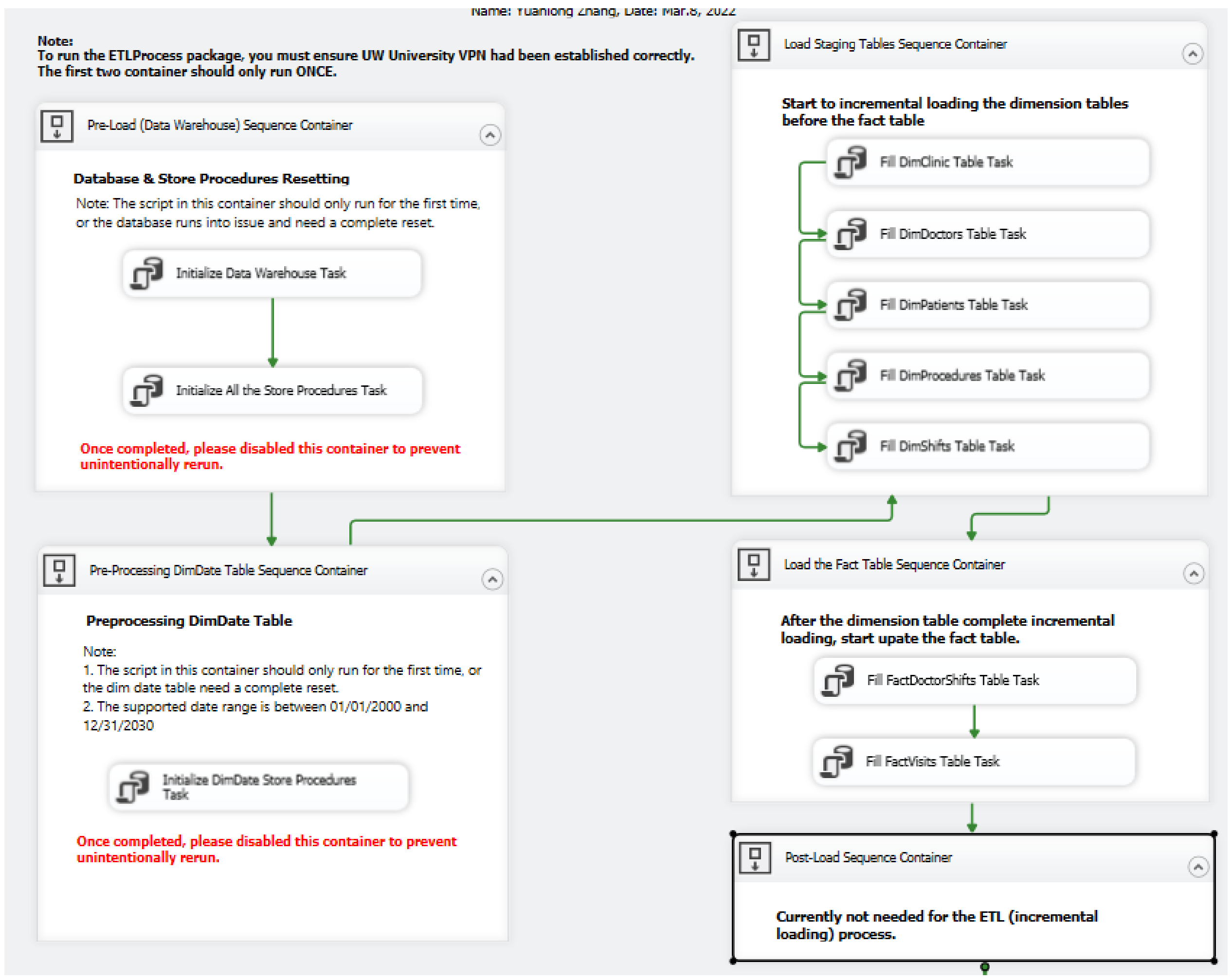


*Figure 28 Diagram for Data Warehouse in Milestone 2*

**Note:**
- All the tables (except DimPatients) are a Type-1 SCD design.
- DimPatients is a Type-2 SCD design.

## *4.5. ETL Visualization for Milestone 2*

We use Microsoft's SQL Server Integration Services (SSIS) to visualize our ETL process. SSIS includes a set of commands represented by visual objects, called tasks, constraints, containers, or transformations. The most common objects used in our ETL process are Sequence Containers, Execute SQL tasks, Precedent Constraints, and Connections.


Name: Yuanlong Zhang, Date: Mar.8, 2022

**Note:**
**To run the ETLProcess package, you must ensure UW University VPN had been established correctly.**
**The first two container should only run ONCE.**

Pre-Load (Data Warehouse) Sequence Container

**Database & Store Procedures Resetting**
Note: The script in this container should only run for the first time, or the database runs into issue and need a complete reset.

Initialize Data Warehouse Task
Initialize All the Store Procedures Task

**Once completed, please disabled this container to prevent unintentionally rerun.**

Pre-Processing DimDate Table Sequence Container

**Preprocessing DimDate Table**
Note:
1. The script in this container should only run for the first time, or the dim date table need a complete reset.
2. The supported date range is between 01/01/2000 and 12/31/2030

Initialize DimDate Store Procedures Task

**Once completed, please disabled this container to prevent unintentionally rerun.**

Load Staging Tables Sequence Container

**Start to incremental loading the dimension tables before the fact table**

Fill DimClinic Table Task
Fill DimDoctors Table Task
Fill DimPatients Table Task
Fill DimProcedures Table Task
Fill DimShifts Table Task

Load the Fact Table Sequence Container

**After the dimension table complete incremental loading, start upate the fact table.**

Fill FactDoctorShifts Table Task
Fill FactVisits Table Task

Post-Load Sequence Container

**Currently not needed for the ETL (incremental loading) process.**

*Figure 29 Overview of SSIS package in Milestone 2*

**Note:**
- These visualization objects are stored and configured in an SSIS Package file.

## *4.5.1. SSIS Connections*
We use a Connection object that connects to the original data source file, invalid phone number file, and data warehouse that holds our SQL ETL objects.

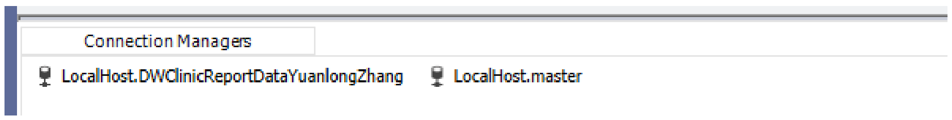

Connection Managers
LocalHost.DWClinicReportDataYuanlongZhang LocalHost.master

*Figure 30 Connections in Milestone 2*

We used OLE DB connections; here's the detailed information:

| No. | Name | Description |
|---|---|---|
| 1 | localhost.master | This is an OLE DB connection.<br>Will connect to localhost\master |
| 2 | localhost.DWClinicReportDataYuanlongZhang | This is an OLE DB connection. |

| No. | Name | Description |
|---|---|---|
| | | Will connect to localhost\DWClinicReportDataYuanlongZhang |

### *4.5.2. Database Connections*

There are two external connections in milestone 2:

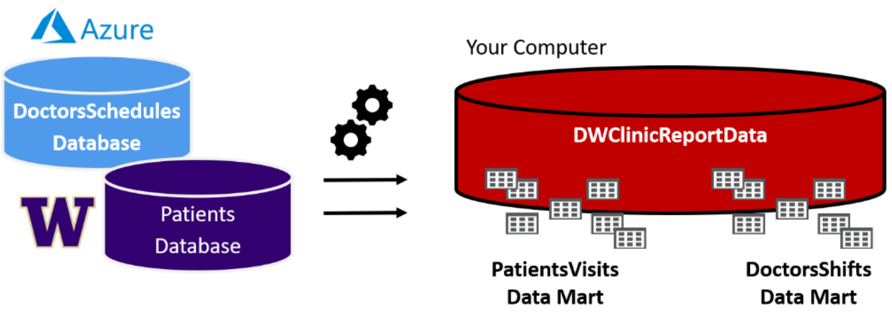


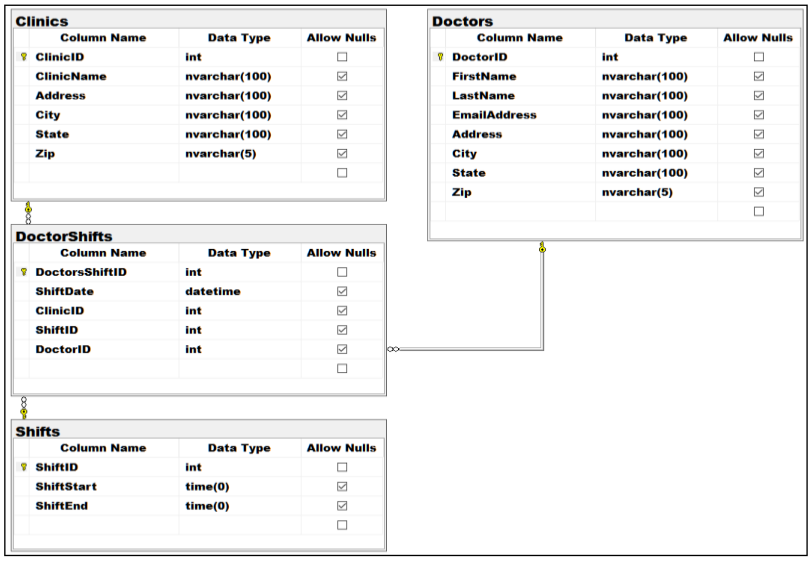
*Figure 31 High-level Diagram for Milestone 2*

- *The **DoctorsSchedule** database is on our class's **Azure cloud Server***
  - Server: **continuumsql.westus2.cloudapp.azure.com**
  - Login: BICert
  - Password: BICert
- *The **Patients** database is on **our class's UW Server***
  - Server: **is-root01.ischool.uw.edu\BI**
  - Login: BICert
  - Password: BICert

**Note:**

- To access my UW SQL Server, **you must connect to the UW network using "OnNet" Virtual Private Network software:** Husky-OnNet

The database diagrams are as follows:



**Clinics**

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| ClinicID | int | ☐ |
| ClinicName | nvarchar(100) | ☑ |
| Address | nvarchar(100) | ☑ |
| City | nvarchar(100) | ☑ |
| State | nvarchar(100) | ☑ |
| Zip | nvarchar(5) | ☑ |
| | | ☐ |

**Doctors**

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| DoctorID | int | ☐ |
| FirstName | nvarchar(100) | ☑ |
| LastName | nvarchar(100) | ☑ |
| EmailAddress | nvarchar(100) | ☑ |
| Address | nvarchar(100) | ☑ |
| City | nvarchar(100) | ☑ |
| State | nvarchar(100) | ☑ |
| Zip | nvarchar(5) | ☑ |
| | | ☐ |

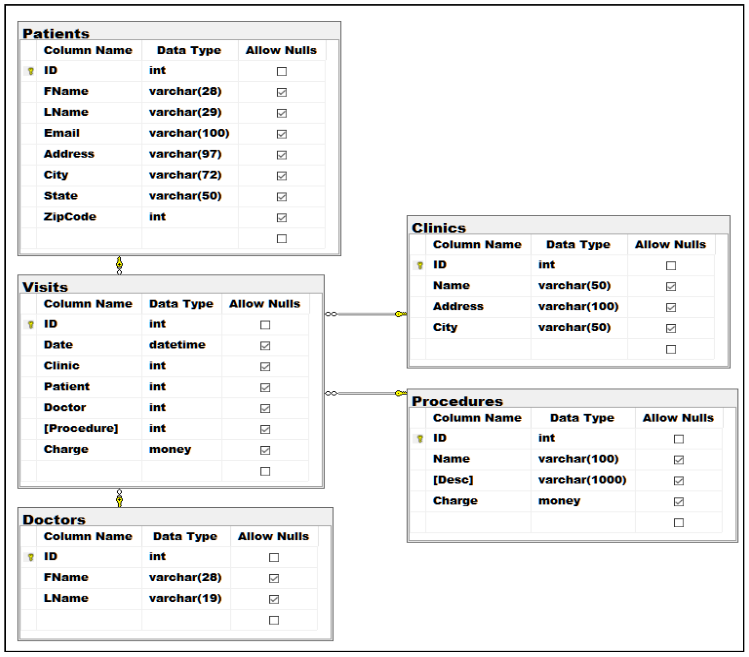
**DoctorShifts**

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| DoctorsShiftID | int | ☐ |
| ShiftDate | datetime | ☑ |
| ClinicID | int | ☑ |
| ShiftID | int | ☑ |
| DoctorID | int | ☑ |
| | | ☐ |

**Shifts**

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| ShiftID | int | ☐ |
| ShiftStart | time(0) | ☑ |
| ShiftEnd | time(0) | ☑ |
| | | ☐ |

*Figure 32 Database Diagram for DoctorsSchedule*



**Patients**

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| ID | int | ☐ |
| FName | varchar(28) | ☑ |
| LName | varchar(29) | ☑ |
| Email | varchar(100) | ☑ |
| Address | varchar(97) | ☑ |
| City | varchar(72) | ☑ |
| State | varchar(50) | ☑ |
| ZipCode | int | ☑ |
| | | ☐ |

**Visits**

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| ID | int | ☐ |
| Date | datetime | ☑ |
| Clinic | int | ☑ |
| Patient | int | ☑ |
| Doctor | int | ☑ |
| [Procedure] | int | ☑ |
| Charge | money | ☑ |
| | | ☐ |

**Doctors**

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| ID | int | ☐ |
| FName | varchar(28) | ☑ |
| LName | varchar(19) | ☑ |
| | | ☐ |

**Clinics**

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| ID | int | ☐ |
| Name | varchar(50) | ☑ |
| Address | varchar(100) | ☑ |
| City | varchar(50) | ☑ |
| | | ☐ |

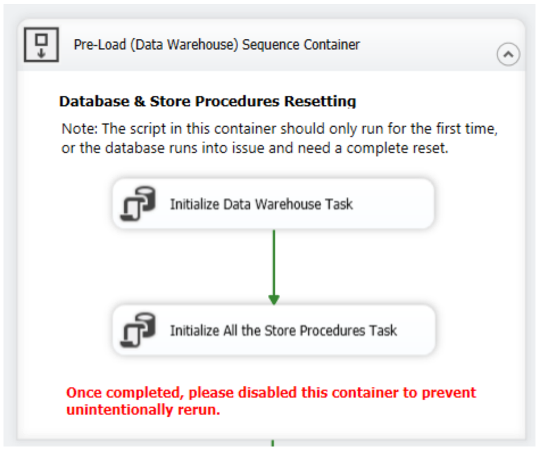
**Procedures**

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| ID | int | ☐ |
| Name | varchar(100) | ☑ |
| [Desc] | varchar(1000) | ☑ |
| Charge | money | ☑ |
| | | ☐ |

*Figure 33 Database Diagram for Patients*

### 4.5.3. Pre-Load (Data Warehouse) Sequence Container

There are three components inside the sequence container.



Pre-Load (Data Warehouse) Sequence Container

**Database & Store Procedures Resetting**

Note: The script in this container should only run for the first time, or the database runs into issue and need a complete reset.

Initialize Data Warehouse Task

Initialize All the Store Procedures Task

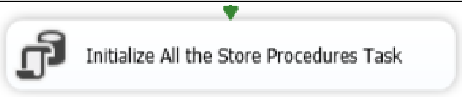
**Once completed, please disabled this container to prevent unintentionally rerun.**

*Figure 34 Pre-Load Sequence Container for Milestone 2*

**Note:**

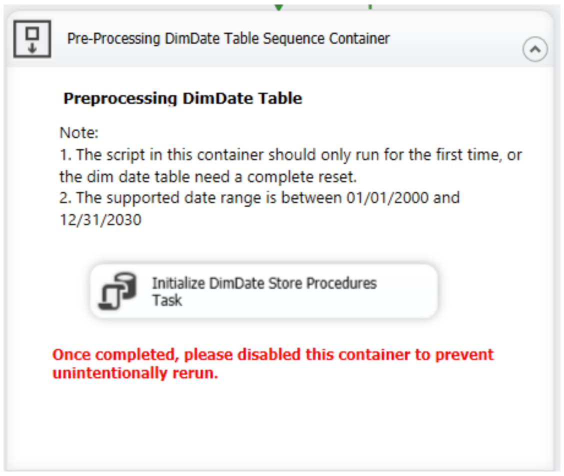
- This container should only run ONCE.

| No. | Name | Description |
|---|---|---|
| 1 | Initialize Data Warehouse Task Initialize Data Warehouse Task | This uses the script to initialize all the data warehouse tables and constraints: It creates the local copy for the data warehouse. The script is available in Create DWClinicReportData.sql |
| 2 | Initialize All the Store Procedures Task | Used to create all the views and store procedures definition and transformation to ETL the data from database to data warehouse. |

| No. | Name | Description |
|---|---|---|
| |  Initialize All the Store Procedures Task | The script is available in DWETLObjects.sql |

### *4.5.4. Pre-Load DimDate Table Sequence Container*

There is one component inside the sequence container.


Pre-Processing DimDate Table Sequence Container

**Preprocessing DimDate Table**

Note:
1. The script in this container should only run for the first time, or the dim date table need a complete reset.
2. The supported date range is between 01/01/2000 and 12/31/2030

Initialize DimDate Store Procedures Task

**Once completed, please disabled this container to prevent unintentionally rerun.**

*Figure 35 Preprocessing DimDate Table Sequence Container in Milestone 2*

**Note:**
- This container should only run ONCE.
- The date range is from 01/01/2000 ~ 12/31/2030. Please adjust the time range if needed.

### *4.5.5. Loading Staging Table Sequence Container*

There are five components inside the sequence container.

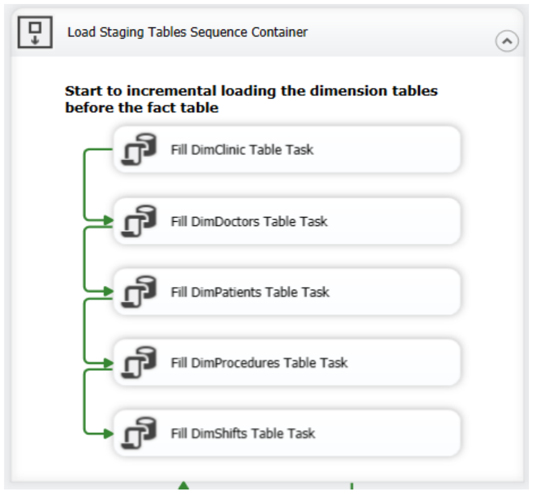

Load Staging Tables Sequence Container
Start to incremental loading the dimension tables before the fact table
Fill DimClinic Table Task
Fill DimDoctors Table Task
Fill DimPatients Table Task
Fill DimProcedures Table Task
Fill DimShifts Table Task

*Figure 36 Load Staging Tables Sequence Container in Milestone 2*

**Note:**

- These items are executed in the store-procedures created in the Pre-Load stage. It had a similar logic as we explained in milestone 1.
- For any transformation we performed, please refer to the later section.

| No. | Name | Executed Store Procedure | SCD Design |
|---|---|---|---|
| 1 | Fill DimClinic Table Task | pETLSyncDimClinics | Type 1 |
| 2 | Fill DimDoctors Table Task | pETLSyncDimDoctors | Type 1 |
| 3 | Fill DimPatients Table Task | pETLSyncDimPatients | Type 2 |
| 4 | Fill DimProcedures Table Task | pETLSyncDimProcedures | Type 1 |
| 5 | Fill DimShifts Table Task | pETLSyncDimShifts | Type 1 |

### *4.5.6. Loading Fact Table Sequence Container*

There are two components inside the sequence container.

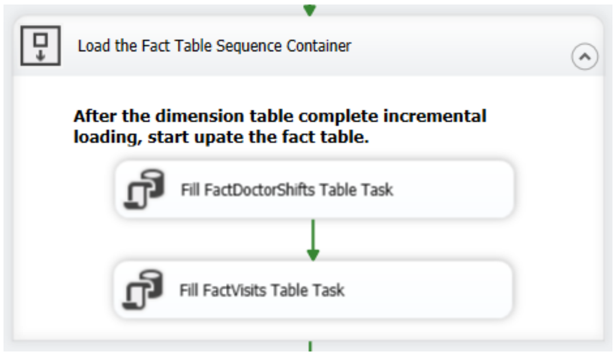

Load the Fact Table Sequence Container
After the dimension table complete incremental loading, start upate the fact table.
Fill FactDoctorShifts Table Task
Fill FactVisits Table Task

*Figure 37 Load the Fact Table Sequence Container in Milestone 2*

**Note:**

- These items are executed in the store-procedures created in the Pre-Load stage. It had a similar logic as we explained in milestone 1.
- We will explain all the complex transformations in the later section.

| No. | Name | Executed Store Procedure | SCD Design |
|---|---|---|---|
| 1 | Fill FactDoctorShifts Table Task | pETLSyncFactDoctorShifts | Type 1 |
| 2 | Fill FactVisits Table Task | pETLSyncFactVisits | Type 1 |

## *4.6. SSIS Package for Milestone 3*

Here's the list of files for our ETL processes for Milestone 1:

| | File Name | Description |
|---|---|---|
| 1 | ETLClinicReportsDocumentData.dtsx | Set of tasks that define views and move data from the data warehouse to MongoDB.<br>*Note: _BISolutions\ETLFinal_YuanlongZhang\ETLPackages* |
| 2 | MongoDBETL.py | Script that defines views and moves data from the data warehouse to MongoDB.<br>*Note: _BISolutions\ETLFinal_YuanlongZhang\Scripts* |
| 3 | DWMongoDBView.sql | SQL Script for creating views used for MongoDB upload.<br>*Note: _BISolutions\ETLFinal_YuanlongZhang\SQLScripts* |

| | File Name | Description |
|---|---|---|
| | | *Note2: The script had already been included in Python Script. There's no need to run this SQL Script separately.* |

## *4.7. ETL Visualization for Milestone 2*

We use Microsoft's SQL Server Integration Services (SSIS) to visualize our ETL process. SSIS includes a set of commands represented by visual objects, called tasks, constraints, containers, or transformations. The most common objects used in our ETL process are Sequence Containers, Execute SQL tasks, Precedent Constraints, and Connections.

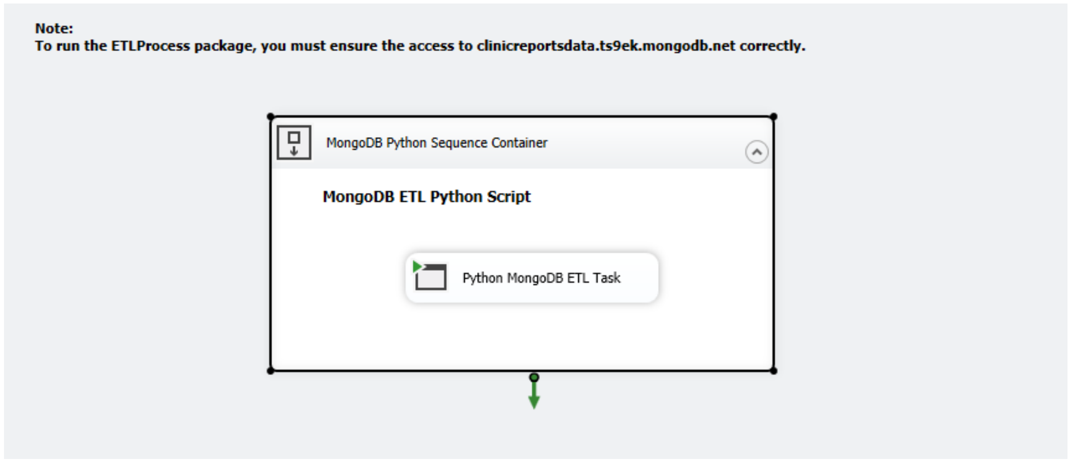

**Note:**
**To run the ETLProcess package, you must ensure the access to clinicreportsdata.ts9ek.mongodb.net correctly.**

*Figure 38 Overview of SSIS package: Control Flow*

**Note:**
- These visualization objects are stored and configured in an SSIS Package file.

### *4.7.1. External Connections*

We use external connections to MongoDB.

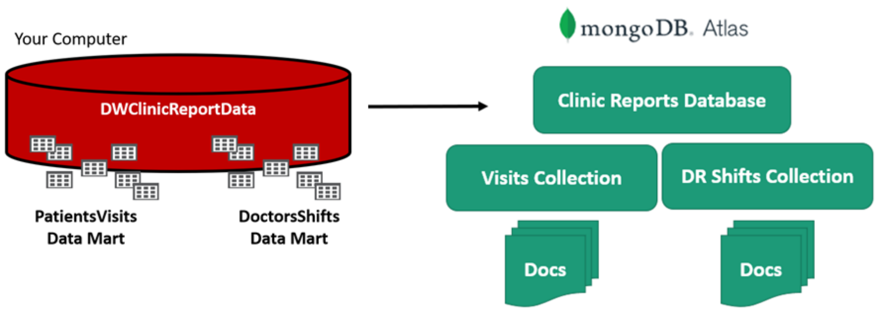
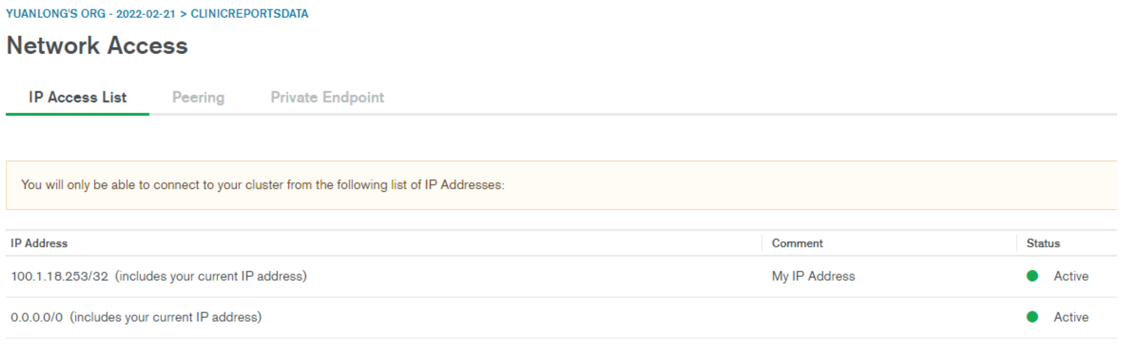


*Figure 39 High-level Diagram for Milestone 3*

The connection string is as follows:
`'mongodb+srv://BICert:BICert@clinicreportsdata.ts9ek.mongodb.net/test?retryWrites=true&w=majority'`

To ensure access, we configure the 0.0.0.0/0 in the IP Address list:



YUANLONG'S ORG - 2022-02-21 > CLINICREPORTSDATA

Network Access

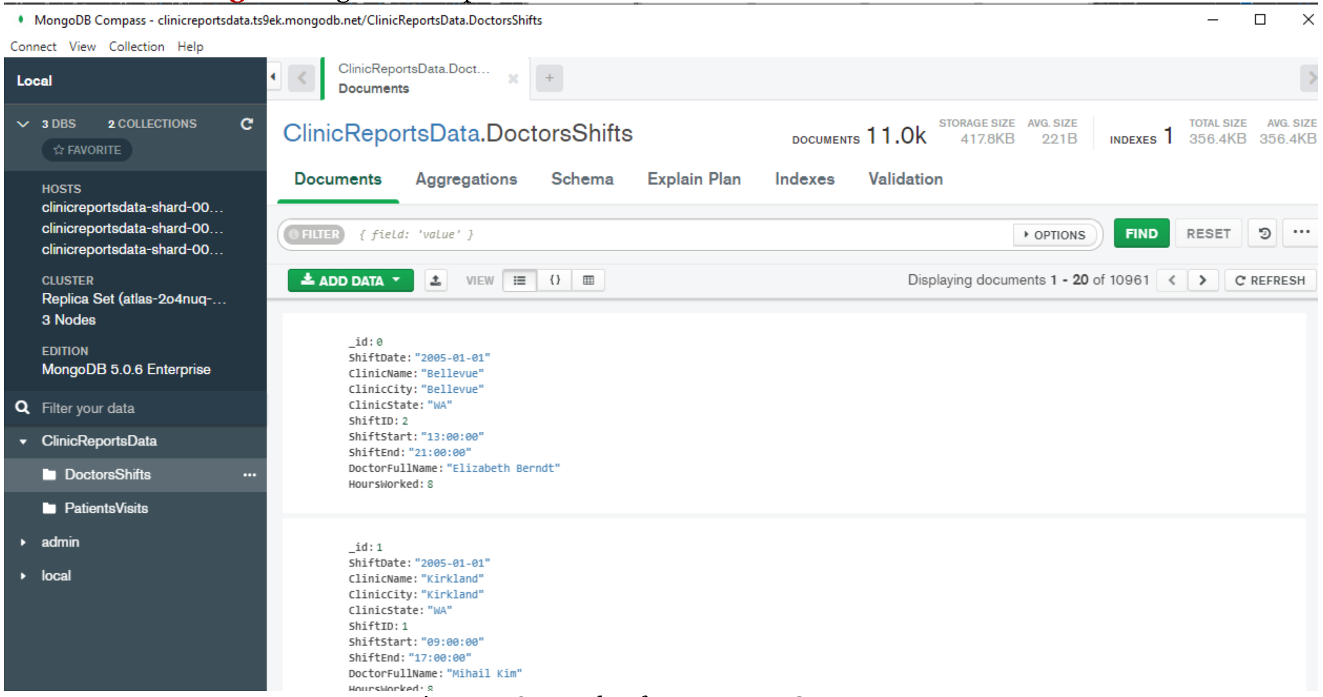
IP Access List | Peering | Private Endpoint

You will only be able to connect to your cluster from the following list of IP Addresses:

| IP Address | Comment | Status |
|---|---|---|
| 100.1.18.253/32 (includes your current IP address) | My IP Address | ● Active |
| 0.0.0.0/0 (includes your current IP address) | | ● Active |

*Figure 40 Screenshot for MongoDB IP Address Configurations*

If you configure everything correctly, after executing SSIS for milestone 3, you can access the database by **connection string** in MongoDB Compass:


MongoDB Compass - clinicreportsdata.ts9ek.mongodb.net/ClinicReportsData.DoctorsShifts
Connect View Collection Help
Local
3 DBS 2 COLLECTIONS
FAVORITE
HOSTS
clinicreportsdata-shard-00...
clinicreportsdata-shard-00...
clinicreportsdata-shard-00...
CLUSTER
Replica Set (atlas-2o4nuq-...
3 Nodes
EDITION
MongoDB 5.0.6 Enterprise
Filter your data
ClinicReportsData
DoctorsShifts
PatientsVisits
admin
local
ClinicReportsData.Doct...
Documents
ClinicReportsData.DoctorsShifts
DOCUMENTS 11.0k STORAGE SIZE 417.8KB AVG. SIZE 221B INDEXES 1 TOTAL SIZE 356.4KB AVG. SIZE 356.4KB
Documents Aggregations Schema Explain Plan Indexes Validation
FILTER { field: 'value' }
OPTIONS FIND RESET
ADD DATA VIEW
Displaying documents 1 - 20 of 10961
REFRESH

```
_id: 0
ShiftDate: "2005-01-01"
ClinicName: "Bellevue"
ClinicCity: "Bellevue"
ClinicState: "WA"
ShiftID: 2
ShiftStart: "13:00:00"
ShiftEnd: "21:00:00"
DoctorFullName: "Elizabeth Berndt"
HoursWorked: 8

_id: 1
ShiftDate: "2005-01-01"
ClinicName: "Kirkland"
ClinicCity: "Kirkland"
ClinicState: "WA"
ShiftID: 1
ShiftStart: "09:00:00"
ShiftEnd: "17:00:00"
DoctorFullName: "Mihail Kim"
HoursWorked: 8
```

*Figure 41 Screenshot for MongoDB Compass*

### *4.7.2. MongoDB Python Sequence Container*

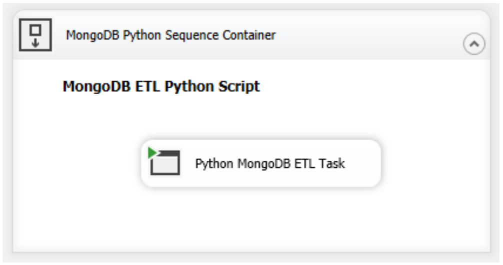


*Figure 42 MongoDB Python Sequence Container in Milestone 3*

**Note:**

- The SSIS package only has one python script task. We will explain the python script in a later section.

## *4.8. SSIS Package for Milestone 4*

Here's the list of files for our ETL processes for Milestone 4:

| | **File Name** | **Description** |
|---|---|---|
| 1 | ETLJob.dtsx | Set of tasks triggered by SQL Agent Job.<br>*Note: _BISolutions\ETLFinal_YuanlongZhang\ETLPackages* |
| 2 | ETLJob.sql | SQL Script, which defined the SQL Agent Jobs and SSIS Proxy.<br>*Note: _BISolutions\ETLFinal_YuanlongZhang\SQLScripts* |
| 3 | ETLViews.sql | SQL Script, which defined the views for SSRS paginated report.<br>*Note: _BISolutions\ETLFinal_YuanlongZhang\SQLScripts* |
| 4 | MainDashboard.rdl | Paginated report for the landing page (high-level summary)<br>*Note: _BISolutions\ETLFinal_YuanlongZhang\ETLDashboardReports* |
| 5 | ETLDetails.rdl | ReportPaginated report for ETL details information<br>*Note: _BISolutions\ETLFinal_YuanlongZhang\ETLDashboardReports* |
| 6 | SSISJobDetails.rdl | Paginated report for SSIS Job information<br>*Note: _BISolutions\ETLFinal_YuanlongZhang\ETLDashboardReports* |

## *4.9. ETL Visualization for Milestone 4*

We use Microsoft's SQL Server Integration Services (SSIS) to visualize our ETL process. SSIS includes a set of commands represented by visual objects, called tasks, constraints, containers, or transformations. The most common objects used in our ETL process are Sequence Containers, Execute SQL tasks, Precedent Constraints, and Connections.

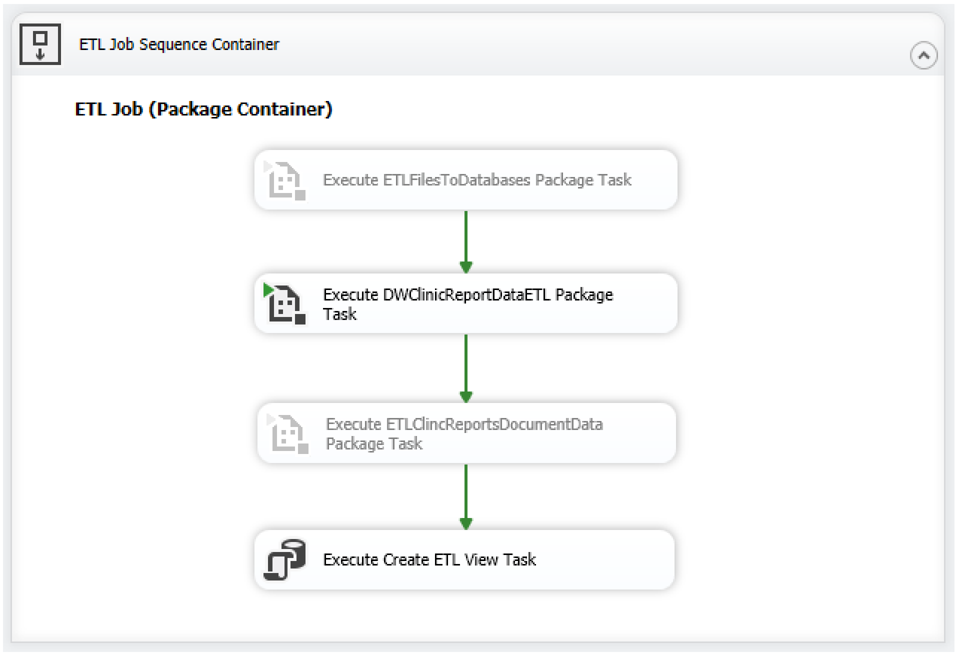

ETL Job Sequence Container
ETL Job (Package Container)
Execute ETLFilesToDatabases Package Task
Execute DWClinicReportDataETL Package Task
Execute ETLClincReportsDocumentData Package Task
Execute Create ETL View Task

*Figure 43 Overview of SSIS package: Control Flow*

**Note:**

- These visualization objects are stored and configured in an SSIS Package file.
- This package contains 3 SSIS packages we created for Milestone 1 ~ Milestone 3.
- There are technical issues when using SQL Agent Jobs to execute Python Scripts. Thus we disabled the SSIS package for milestone 1 and milestone 3.

### *4.9.1. Connections*

We use a Connection object that connects to the original data source file, invalid phone number file, and data warehouse that holds our SQL ETL objects.

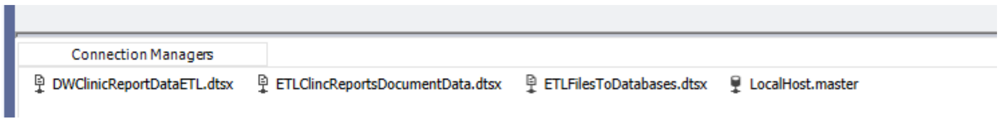


*Figure 44 Connections in Milestone 4*

We used 1 OLE DB connection and 3 file connections; here's the detailed information:

| No. | Name | Description |
|---|---|---|
| 1 | localhost.master | This is an OLE DB connection.<br>Will connect to localhost\master |
| 2 | DWClinicReportDataETL.dtsx | This is a file connection.<br>Will connect to DWClinicReportDataETL.dtsx (SSIS packge for Milestone 2) |
| 3 | ETLClinicReportsDocumentData.dtsx | This is a file connection.<br>Will connect to ETLClinicReportsDocumentData.dtsx (SSIS package for Milestone 3)<br>*Note: This component had been disabled in the SSIS package because it included the Python Script task.* |
| 4 | ETLFilesToDatabases.dtsx | This is a file connection.<br>Will connect to ETLFilesToDatabases.dtsx (SSIS package for Milestone 1)<br>*Note: This component had been disabled in the SSIS package because it included the Python Script task.* |

### *4.9.2. ETL Job Sequence Container*

There are four components inside the sequence container.

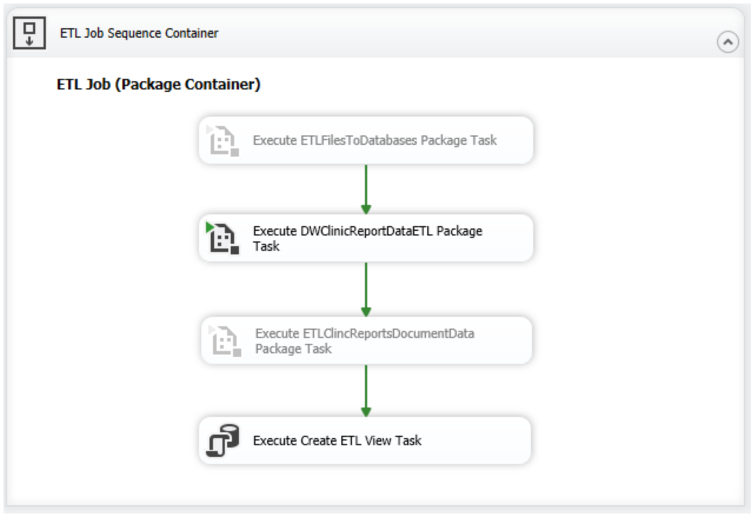
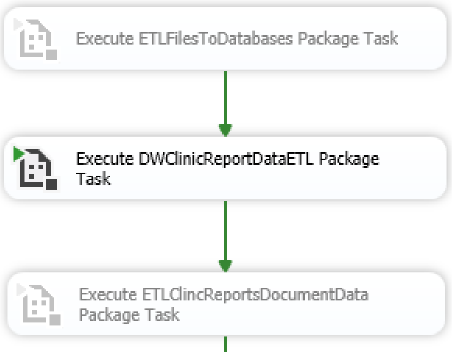
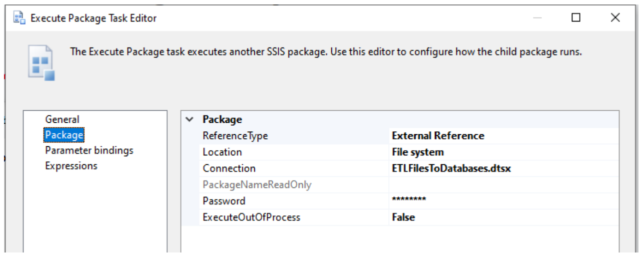
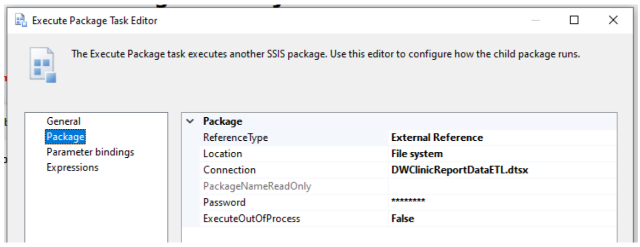
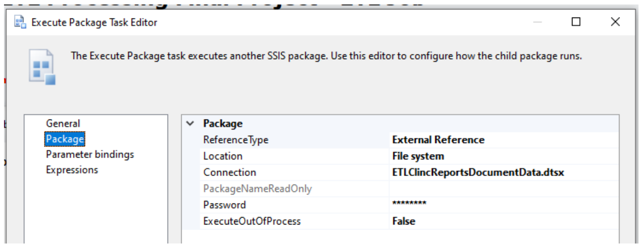
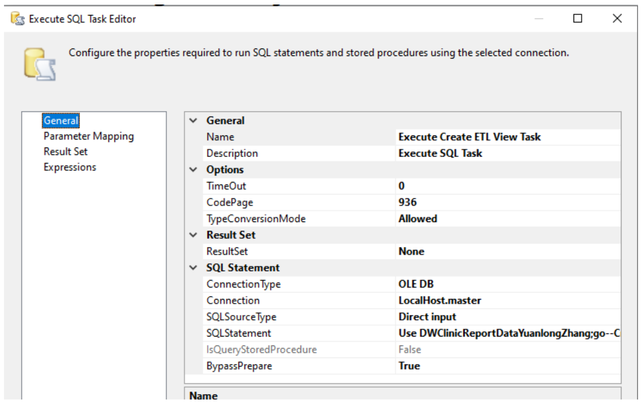

ETL Job Sequence Container
ETL Job (Package Container)
Execute ETLFilesToDatabases Package Task
Execute DWClinicReportDataETL Package Task
Execute ETLClincReportsDocumentData Package Task
Execute Create ETL View Task

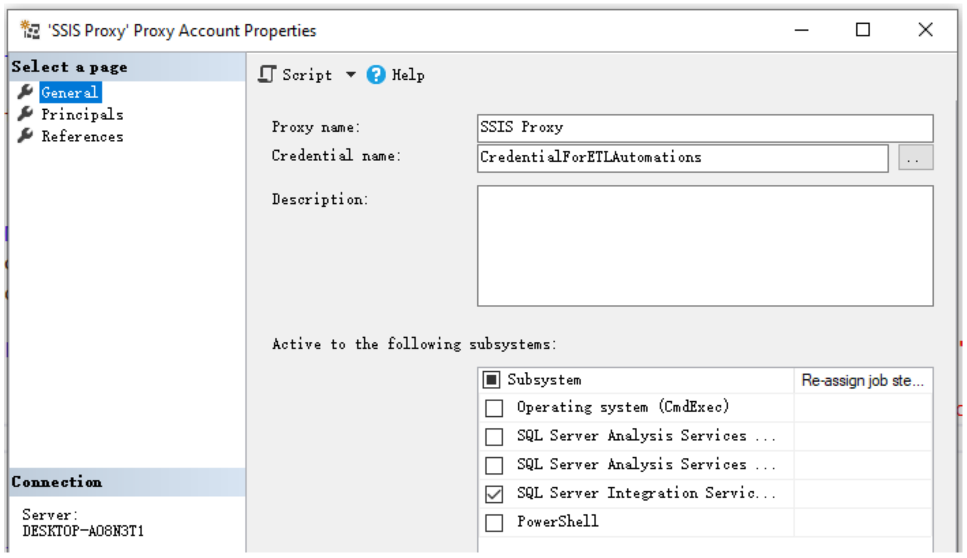
*Figure 45 Screenshot for SSIS Package for Milestone 4*

**Note:**
- This container should only run ONCE.

| No. | Name | Description |
|---|---|---|
| 1 | The first three package tasks:<br><br> Execute ETLFilesToDatabases Package Task<br>Execute DWClinicReportDataETL Package Task<br>Execute ETLClincReportsDocumentData Package Task | Which refer to the SSIS packages for Milestone 1~ 3:<br>Please refer to the previous section if you need to know more details.<br><br>Execute Package Task Editor<br>The Execute Package task executes another SSIS package. Use this editor to configure how the child package runs.<br>General / Package / Parameter bindings / Expressions<br>Package<br>ReferenceType: External Reference<br>Location: File system<br>Connection: ETLFilesToDatabases.dtsx<br>PackageNameReadOnly<br>Password: ********<br>ExecuteOutOfProcess: False<br><br>Execute Package Task Editor<br>The Execute Package task executes another SSIS package. Use this editor to configure how the child package runs.<br>General / Package / Parameter bindings / Expressions<br>Package<br>ReferenceType: External Reference<br>Location: File system<br>Connection: DWClinicReportDataETL.dtsx<br>PackageNameReadOnly<br>Password: ********<br>ExecuteOutOfProcess: False<br><br>Execute Package Task Editor<br>The Execute Package task executes another SSIS package. Use this editor to configure how the child package runs.<br>General / Package / Parameter bindings / Expressions<br>Package<br>ReferenceType: External Reference<br>Location: File system<br>Connection: ETLClincReportsDocumentData.dtsx<br>PackageNameReadOnly<br>Password: ********<br>ExecuteOutOfProcess: False<br>We configured the reference type as an External reference, ensuring SQL Agent Job can find the file and execute it correctly. |
| 2 | Execute Create ETL View Task | Used to execute the SQL Script, which defined the views for SSRS paginated report.<br>The script is available in ETLViews.sql<br><br>Execute SQL Task Editor<br>Configure the properties required to run SQL statements and stored procedures using the selected connection.<br>General / Parameter Mapping / Result Set / Expressions<br>General<br>Name: Execute Create ETL View Task<br>Description: Execute SQL Task<br>Options<br>TimeOut: 0<br>CodePage: 936<br>TypeConversionMode: Allowed<br>Result Set<br>ResultSet: None<br>SQL Statement<br>ConnectionType: OLE DB<br>Connection: LocalHost.master<br>SQLSourceType: Direct input<br>SQLStatement: Use DWClinicReportDataYuanlongZhang;go--C<br>IsQueryStoredProcedure: False<br>BypassPrepare: True<br>Name |

## *4.9.3. SQL Agent Job for Milestone 4*

To make sure the SQL Job works, we create an SSIS proxy as follows:



'SSIS Proxy' Proxy Account Properties

Proxy name: SSIS Proxy
Credential name: CredentialForETLAutomations
Description:

Active to the following subsystems:

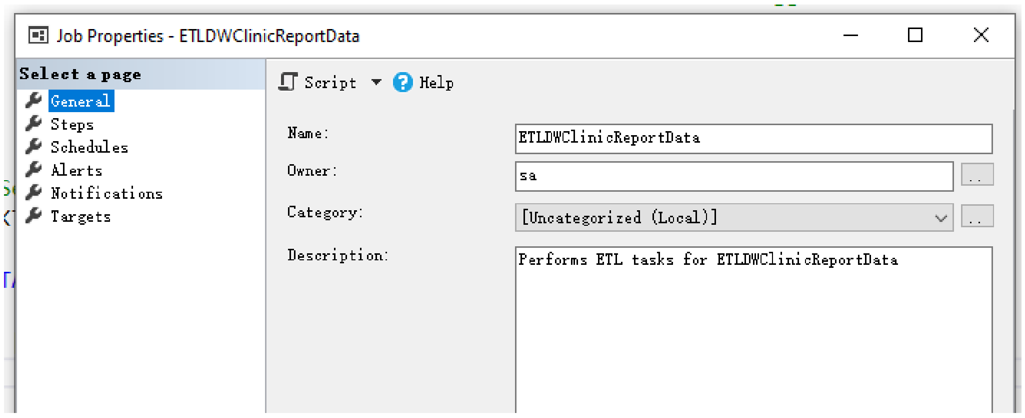
| Subsystem | Re-assign job ste... |
|---|---|
| ☐ Operating system (CmdExec) | |
| ☐ SQL Server Analysis Services ... | |
| ☐ SQL Server Analysis Services ... | |
| ✓ SQL Server Integration Servic... | |
| ☐ PowerShell | |

Connection
Server:
DESKTOP-A08N3T1

*Figure 46 Screenshot for SSIS Proxy configurations*

Then we created the SQL Agent Job by ETLJob.sql script as follows:

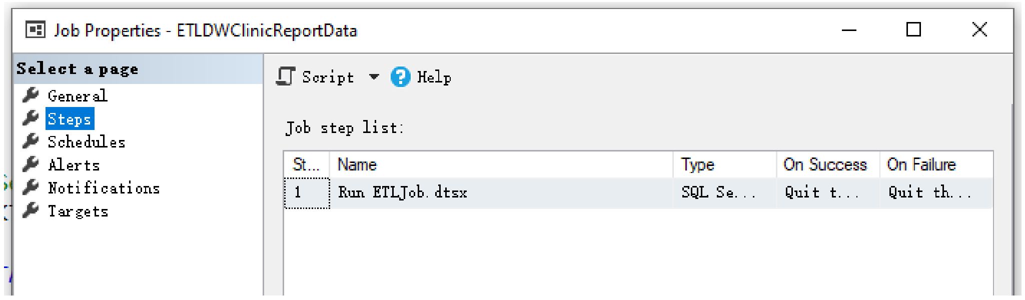
Set the Job Owner as SA:



Job Properties - ETLDWClinicReportData

Name: ETLDWClinicReportData
Owner: sa
Category: [Uncategorized (Local)]
Description: Performs ETL tasks for ETLDWClinicReportData

*Figure 47 Screenshot for ETLDWClinicReportData job configurations (General)*



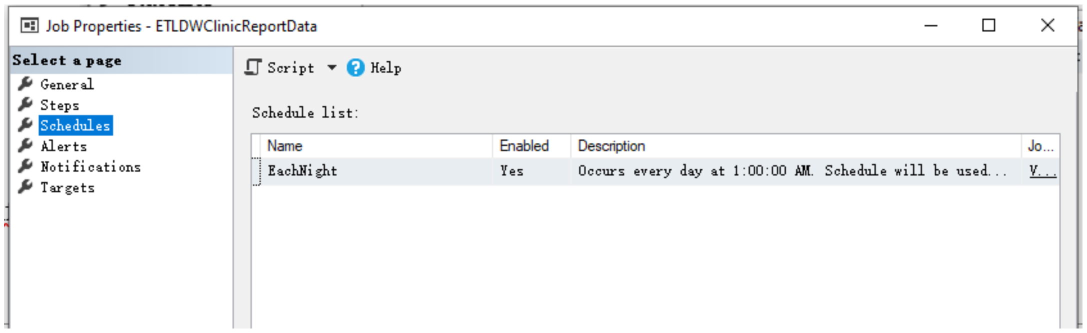
Job Properties - ETLDWClinicReportData

Job step list:

| St... | Name | Type | On Success | On Failure |
|---|---|---|---|---|
| 1 | Run ETLJob.dtsx | SQL Se... | Quit t... | Quit th... |

*Figure 48 Screenshot for ETLDWClinicReportData job configurations (Steps)*

Configured to run each night at 1 AM:



Job Properties - ETLDWClinicReportData

Schedule list:

| Name | Enabled | Description | Jo... |
|---|---|---|---|
| EachNight | Yes | Occurs every day at 1:00:00 AM. Schedule will be used... | V... |

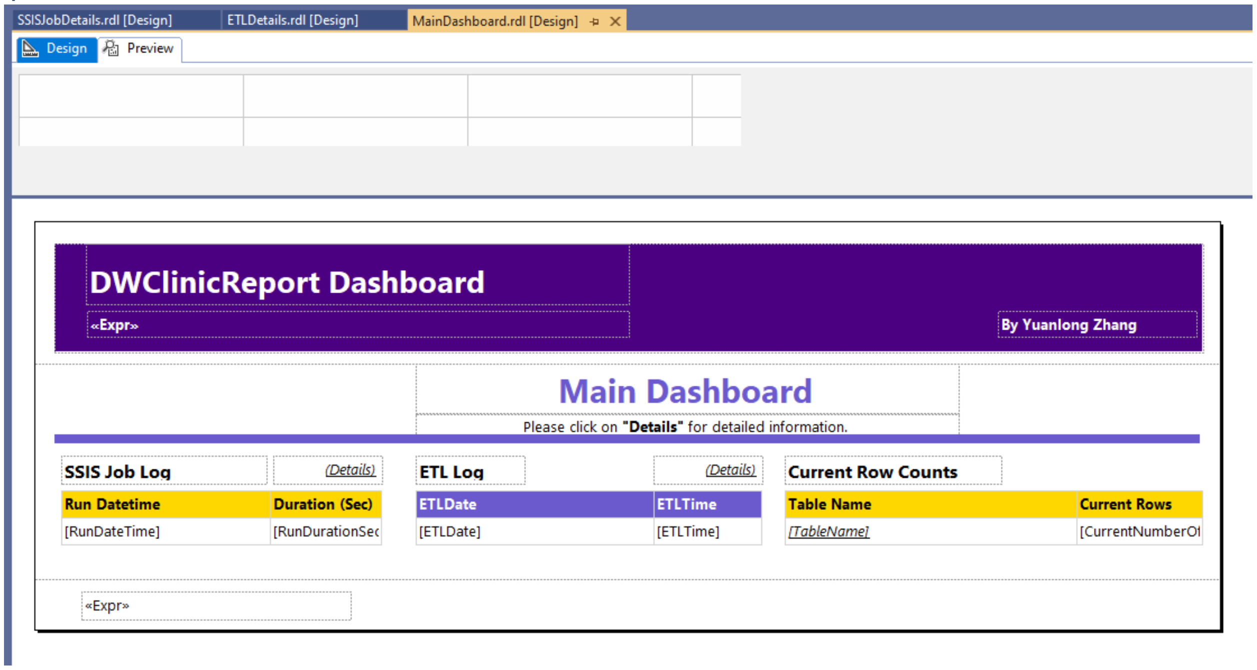
*Figure 49 Screenshot for ETLDWClinicReportData job configurations (Schedules)*

## 4.10. SSRS ETL Dashboard for Milestone 4

The package has three dashboards: MainDashboard, SSISJobDetails, and ETL Details.

You should always start from MainDashboard.

### 4.10.1. MainDashboard



DWClinicReport Dashboard
«Expr»
By Yuanlong Zhang

Main Dashboard
Please click on "**Details**" for detailed information.

SSIS Job Log (Details)

| Run Datetime | Duration (Sec) |
|---|---|
| [RunDateTime] | [RunDurationSec |

ETL Log (Details)

| ETLDate | ETLTime |
|---|---|
| [ETLDate] | [ETLTime] |

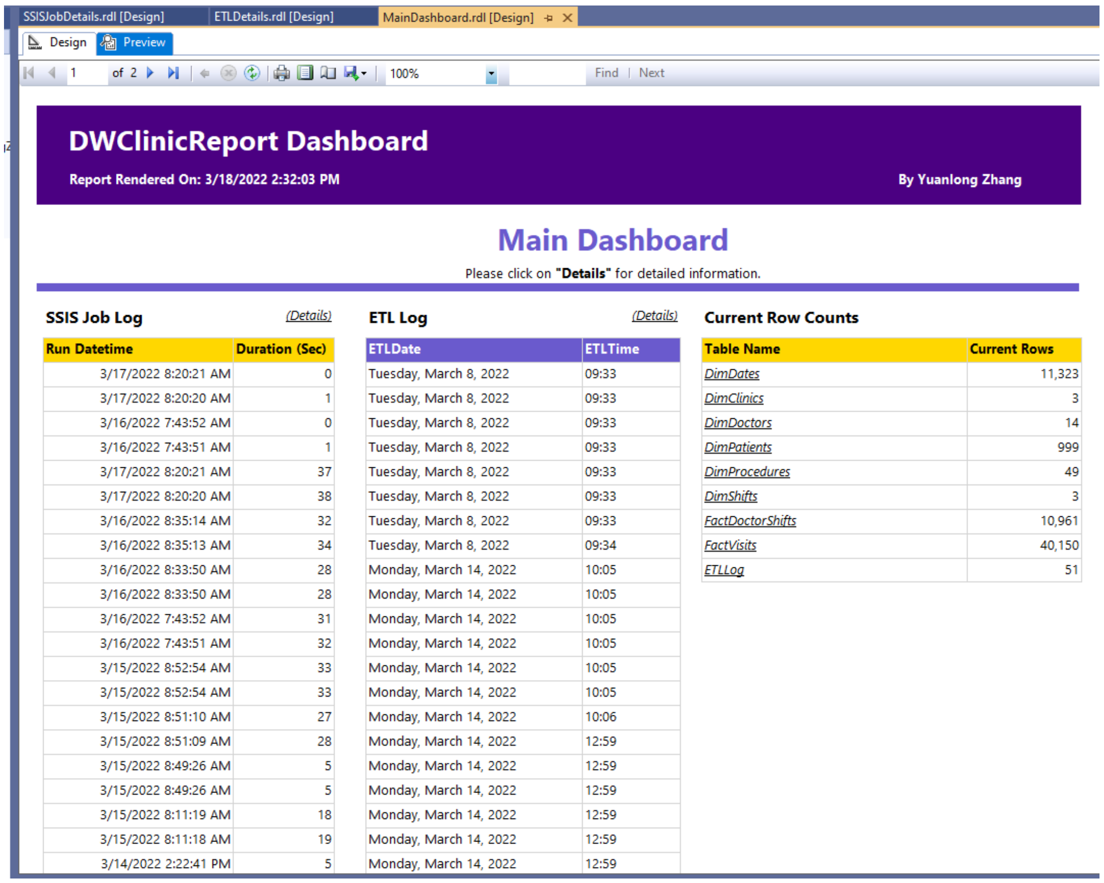
Current Row Counts

| Table Name | Current Rows |
|---|---|
| [TableName] | [CurrentNumberOf |

«Expr»

*Figure 50 Screenshot for MainDashboard.rdl (design mode)*

The dashboard is used as a landing page for all high-level summary information; it also allows you to navigate to the other two dashboards **by clicking on "(details)":**


SSISJobDetails.rdl [Design] | ETLDetails.rdl [Design] | MainDashboard.rdl [Design]

Design | Preview

1 of 2 | 100% | Find | Next

**DWClinicReport Dashboard**

**Report Rendered On: 3/18/2022 2:32:03 PM** **By Yuanlong Zhang**

**Main Dashboard**

Please click on **"Details"** for detailed information.

**SSIS Job Log** (Details)

| Run Datetime | Duration (Sec) |
|---|---|
| 3/17/2022 8:20:21 AM | 0 |
| 3/17/2022 8:20:20 AM | 1 |
| 3/16/2022 7:43:52 AM | 0 |
| 3/16/2022 7:43:51 AM | 1 |
| 3/17/2022 8:20:21 AM | 37 |
| 3/17/2022 8:20:20 AM | 38 |
| 3/16/2022 8:35:14 AM | 32 |
| 3/16/2022 8:35:13 AM | 34 |
| 3/16/2022 8:33:50 AM | 28 |
| 3/16/2022 8:33:50 AM | 28 |
| 3/16/2022 7:43:52 AM | 31 |
| 3/16/2022 7:43:51 AM | 32 |
| 3/15/2022 8:52:54 AM | 33 |
| 3/15/2022 8:52:54 AM | 33 |
| 3/15/2022 8:51:10 AM | 27 |
| 3/15/2022 8:51:09 AM | 28 |
| 3/15/2022 8:49:26 AM | 5 |
| 3/15/2022 8:49:26 AM | 5 |
| 3/15/2022 8:11:19 AM | 18 |
| 3/15/2022 8:11:18 AM | 19 |
| 3/14/2022 2:22:41 PM | 5 |

**ETL Log** (Details)

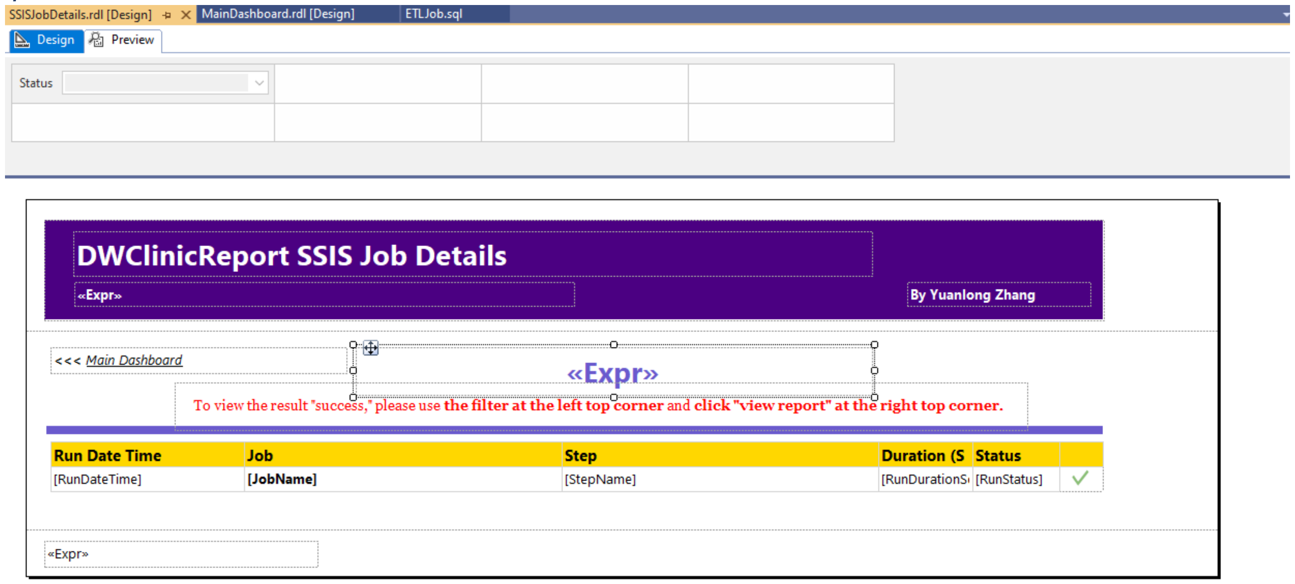
| ETLDate | ETLTime |
|---|---|
| Tuesday, March 8, 2022 | 09:33 |
| Tuesday, March 8, 2022 | 09:33 |
| Tuesday, March 8, 2022 | 09:33 |
| Tuesday, March 8, 2022 | 09:33 |
| Tuesday, March 8, 2022 | 09:33 |
| Tuesday, March 8, 2022 | 09:33 |
| Tuesday, March 8, 2022 | 09:33 |
| Tuesday, March 8, 2022 | 09:34 |
| Monday, March 14, 2022 | 10:05 |
| Monday, March 14, 2022 | 10:05 |
| Monday, March 14, 2022 | 10:05 |
| Monday, March 14, 2022 | 10:05 |
| Monday, March 14, 2022 | 10:05 |
| Monday, March 14, 2022 | 10:05 |
| Monday, March 14, 2022 | 10:06 |
| Monday, March 14, 2022 | 12:59 |
| Monday, March 14, 2022 | 12:59 |
| Monday, March 14, 2022 | 12:59 |
| Monday, March 14, 2022 | 12:59 |
| Monday, March 14, 2022 | 12:59 |
| Monday, March 14, 2022 | 12:59 |

**Current Row Counts**

| Table Name | Current Rows |
|---|---|
| DimDates | 11,323 |
| DimClinics | 3 |
| DimDoctors | 14 |
| DimPatients | 999 |
| DimProcedures | 49 |
| DimShifts | 3 |
| FactDoctorShifts | 10,961 |
| FactVisits | 40,150 |
| ETLLog | 51 |

*Figure 51 Screenshot for MainDashboard.rdl (preview mode)*

## *4.10.2. SSISJobDetails Dashboard*


SSISJobDetails.rdl [Design] | MainDashboard.rdl [Design] | ETLJob.sql

Design | Preview

Status

**DWClinicReport SSIS Job Details**

«Expr» **By Yuanlong Zhang**

<<< Main Dashboard

**«Expr»**

To view the result "success," please use **the filter at the left top corner** and **click "view report" at the right top corner.**

| Run Date Time | Job | Step | Duration (S | Status | |
|---|---|---|---|---|---|
| [RunDateTime] | **[JobName]** | [StepName] | [RunDurationS | [RunStatus] | ✓ |

«Expr»

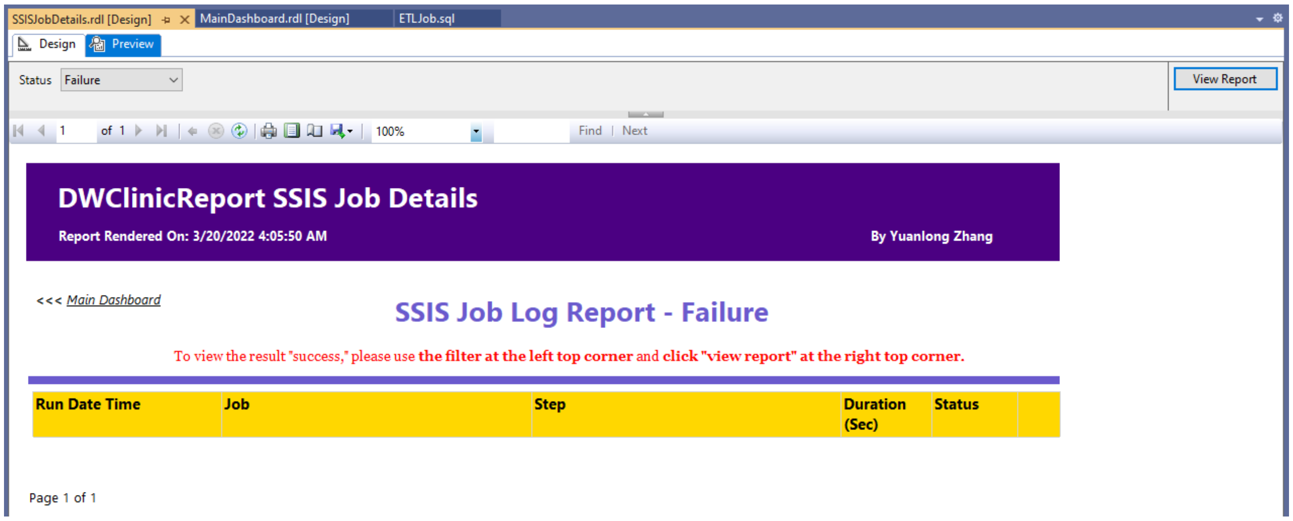
*Figure 52 Screenshot for SSISJobDetails.rdl (design mode)*

This dashboard provides a detailed view of the SSIS Job Log (which is the view from MSDB):

**Note:**

- You need to select the status (success/failure) and click on "view report."

To view the result "success," please use the filter at the left top corner and click "view report" at the right top corner.



Status: Failure

**DWClinicReport SSIS Job Details**

**Report Rendered On: 3/20/2022 4:05:50 AM** **By Yuanlong Zhang**

<<< Main Dashboard

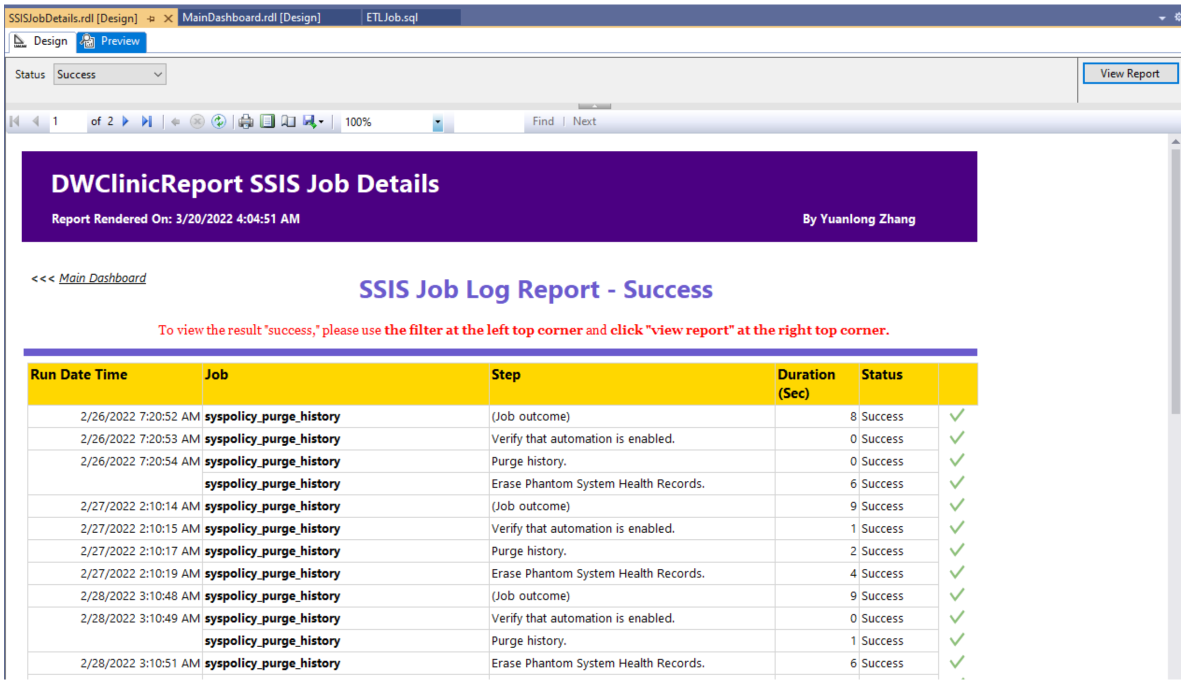
**SSIS Job Log Report - Failure**

To view the result "success," please use **the filter at the left top corner** and **click "view report" at the right top corner.**

| Run Date Time | Job | Step | Duration (Sec) | Status | |
|---|---|---|---|---|---|

Page 1 of 1

*Figure 53 Screenshot for SSISJobDetails.rdl (preview mode: Failure)*



Status: Success

**DWClinicReport SSIS Job Details**

**Report Rendered On: 3/20/2022 4:04:51 AM** **By Yuanlong Zhang**

<<< Main Dashboard

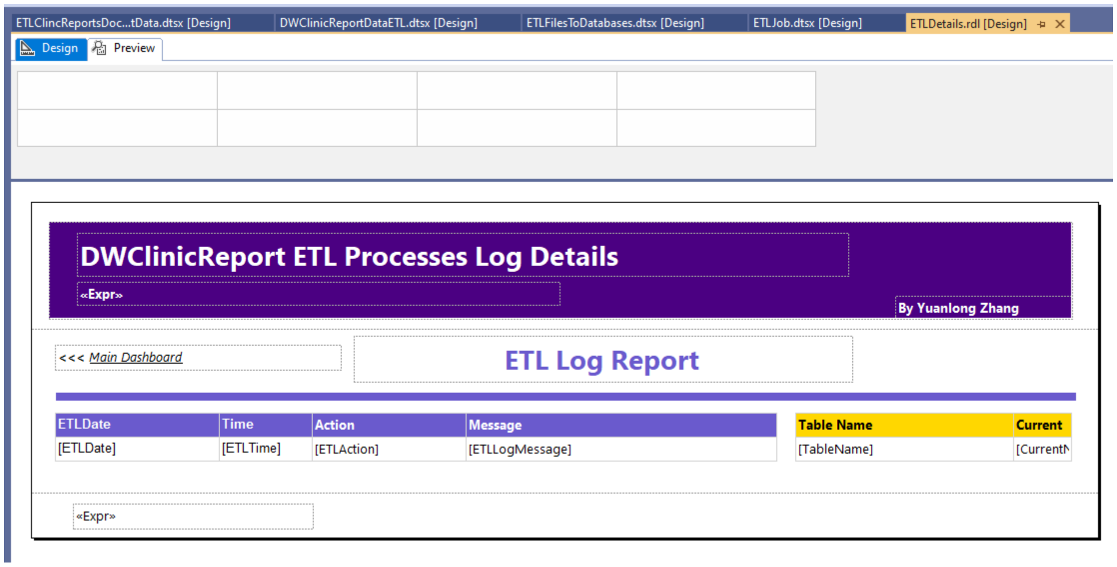
**SSIS Job Log Report - Success**

To view the result "success," please use **the filter at the left top corner** and **click "view report" at the right top corner.**

| Run Date Time | Job | Step | Duration (Sec) | Status | |
|---|---|---|---|---|---|
| 2/26/2022 7:20:52 AM | **syspolicy_purge_history** | (Job outcome) | 8 | Success | ✓ |
| 2/26/2022 7:20:53 AM | **syspolicy_purge_history** | Verify that automation is enabled. | 0 | Success | ✓ |
| 2/26/2022 7:20:54 AM | **syspolicy_purge_history** | Purge history. | 0 | Success | ✓ |
| | **syspolicy_purge_history** | Erase Phantom System Health Records. | 6 | Success | ✓ |
| 2/27/2022 2:10:14 AM | **syspolicy_purge_history** | (Job outcome) | 9 | Success | ✓ |
| 2/27/2022 2:10:15 AM | **syspolicy_purge_history** | Verify that automation is enabled. | 1 | Success | ✓ |
| 2/27/2022 2:10:17 AM | **syspolicy_purge_history** | Purge history. | 2 | Success | ✓ |
| 2/27/2022 2:10:19 AM | **syspolicy_purge_history** | Erase Phantom System Health Records. | 4 | Success | ✓ |
| 2/28/2022 3:10:48 AM | **syspolicy_purge_history** | (Job outcome) | 9 | Success | ✓ |
| 2/28/2022 3:10:49 AM | **syspolicy_purge_history** | Verify that automation is enabled. | 0 | Success | ✓ |
| | **syspolicy_purge_history** | Purge history. | 1 | Success | ✓ |
| 2/28/2022 3:10:51 AM | **syspolicy_purge_history** | Erase Phantom System Health Records. | 6 | Success | ✓ |

*Figure 54 Screenshot for SSISJobDetails.rdl (preview mode: Success)*

### 4.10.3. ETLDetails Dashboard


ETLClincReportsDoc...tData.dtsx [Design]
DWClinicReportDataETL.dtsx [Design]
ETLFilesToDatabases.dtsx [Design]
ETLJob.dtsx [Design]
ETLDetails.rdl [Design]
Design
Preview

DWClinicReport ETL Processes Log Details

«Expr»

By Yuanlong Zhang

<<< Main Dashboard

ETL Log Report

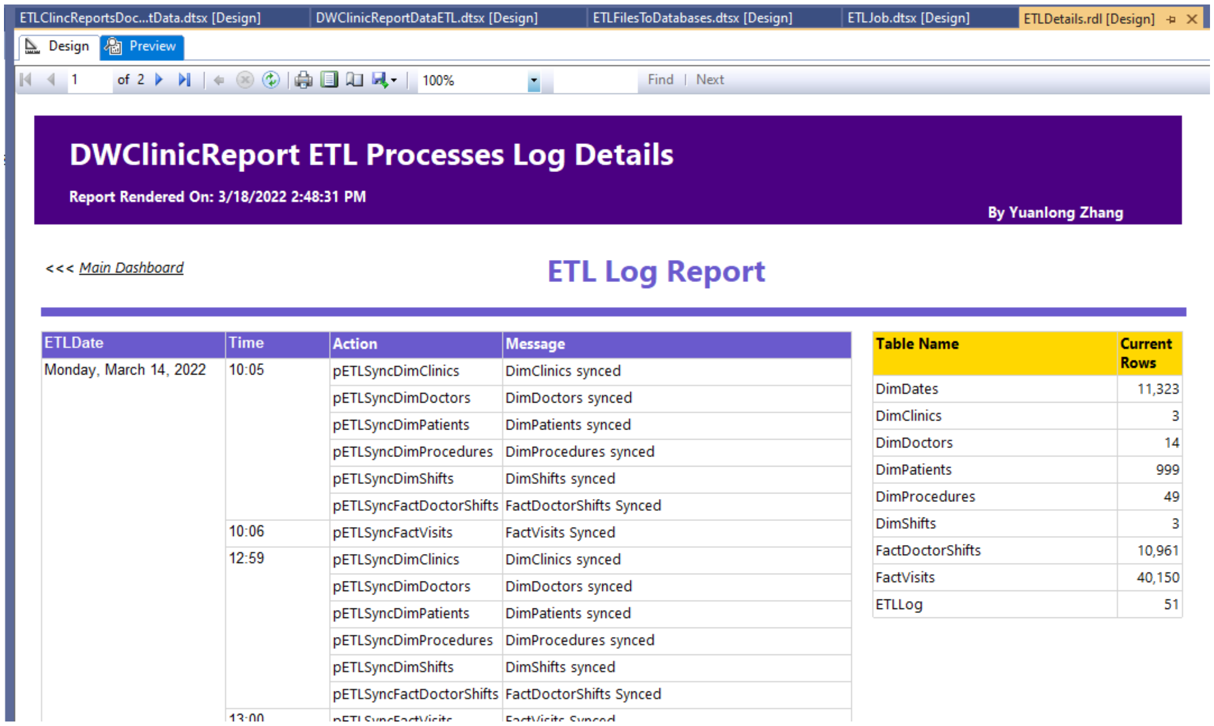
| ETLDate | Time | Action | Message |
|---|---|---|---|
| [ETLDate] | [ETLTime] | [ETLAction] | [ETLLogMessage] |

| Table Name | Current |
|---|---|
| [TableName] | [CurrentN |

«Expr»

Figure 55 Screenshot for ETLDetails.rdl (design mode)

This dashboard provides a detailed view of the ETL Log table and the row count of the table in the data warehouse:


ETLClincReportsDoc...tData.dtsx [Design]
DWClinicReportDataETL.dtsx [Design]
ETLFilesToDatabases.dtsx [Design]
ETLJob.dtsx [Design]
ETLDetails.rdl [Design]
Design
Preview
1
of 2
100%
Find | Next

DWClinicReport ETL Processes Log Details

Report Rendered On: 3/18/2022 2:48:31 PM

By Yuanlong Zhang

<<< Main Dashboard

ETL Log Report

| ETLDate | Time | Action | Message |
|---|---|---|---|
| Monday, March 14, 2022 | 10:05 | pETLSyncDimClinics | DimClinics synced |
| | | pETLSyncDimDoctors | DimDoctors synced |
| | | pETLSyncDimPatients | DimPatients synced |
| | | pETLSyncDimProcedures | DimProcedures synced |
| | | pETLSyncDimShifts | DimShifts synced |
| | | pETLSyncFactDoctorShifts | FactDoctorShifts Synced |
| | 10:06 | pETLSyncFactVisits | FactVisits Synced |
| | 12:59 | pETLSyncDimClinics | DimClinics synced |
| | | pETLSyncDimDoctors | DimDoctors synced |
| | | pETLSyncDimPatients | DimPatients synced |
| | | pETLSyncDimProcedures | DimProcedures synced |
| | | pETLSyncDimShifts | DimShifts synced |
| | | pETLSyncFactDoctorShifts | FactDoctorShifts Synced |

| Table Name | Current Rows |
|---|---|
| DimDates | 11,323 |
| DimClinics | 3 |
| DimDoctors | 14 |
| DimPatients | 999 |
| DimProcedures | 49 |
| DimShifts | 3 |
| FactDoctorShifts | 10,961 |
| FactVisits | 40,150 |
| ETLLog | 51 |

Figure 56 Screenshot for ETLDetails.rdl (preview mode)

## 5. *Data Transformation and Considerations*

We performed transformation at milestone 1 (CSV file to the database) and milestone 2 (database to the data warehouse):

### 5.1. *Files to Source Databases*

| **Source Data** | **Source Type** | **Destination** | **Destination Type** | **Transformations** |
|---|---|---|---|---|
| Folder Name | Directory Name | Staging.dbo.StagingVisits.Clinic | New Column | Extract folder name |
| Filename | Filename | Staging.dbo.StagingVisits.Date | New Column | Extract the first 8 digits of the filename (date part) |
| Email | csv | Staging.dbo.StagingNewPatient.Email | Column | Python Script to validate the valid email address and remove the invalid one |
| Staging.dbo.StagingVisits.Clinic | New Column | Patients.dbo.Patients.Date | Datetime | Combine the Date and Time column and convert it to datetime datatype |
| Staging.dbo.StagingVisits.Date | New Column | Patients.dbo.Patients.Charge | Decimal | Transform the int to decimal (18,2) |
| Staging.dbo.StagingNewPatient.Email | Column | Patients.dbo.Patients.Clinic | Int | Select the Clinic ID by joining the Patient.dbo.Clinics table |

### 5.2. *Source DB to Data Warehouse*

| **Source Data** | **Source Type** | **Destination** | **Destination Type** | **Transformations** |
|---|---|---|---|---|
| Patients.dbo.Patients.Date | Datetime | | | |
| Patients.dbo.Patients.Charge | Decimal | | | |
| Patients.dbo.Patients.Clinic | Int | | | |
| Patients.dbo.Patients.ID | int | DWClinicReportDataYuanlongZhang.dbo.DimPatients.PatientID | int | |
| Patients.dbo.Patients.FName | varchar(28) | DWClinicReportDataYuanlongZhang.dbo.DimPatients.PatientFullName | varchar(100) | Concate First Name and Last Name |
| Patients.dbo.Patients.LName | varchar(29) | DWClinicReportDataYuanlongZhang.dbo.DimPatients.PatientFullName | varchar(100) | Concate First Name and Last Name |
| Patients.dbo.Patients.Email | varchar(100) | | | |
| Patients.dbo.Patients.Address | varchar(97) | | | |
| Patients.dbo.Patients.City | varchar(72) | DWClinicReportDataYuanlongZhang.dbo.DimPatients.PatientCity | varchar(100) | |
| Patients.dbo.Patients.State | varchar(50) | DWClinicReportDataYuanlongZhang.dbo.DimPatients.PatientState | varchar(100) | |
| Patients.dbo.Patients.ZipCode | int | DWClinicReportDataYuanlongZhang.dbo.DimPatients.PatientZipCode | int | |
| Patients.dbo.Clinics.ID | int | | | |
| Patients.dbo.Clinics.Name | varchar(50) | | | |
| Patients.dbo.Clinics.Address | varchar(100) | | | |
| Patients.dbo.Clinics.City | varchar(50) | | | |
| Patients.dbo.Doctors.ID | int | | | |

| Source Data | Source Type | Destination | Destination Type | Transformations |
|---|---|---|---|---|
| Patients.dbo.Doctors.FName | varchar(28) | | | |
| Patients.dbo.Doctors.LName | varchar(19) | | | |
| Patients.dbo.Procedures.ID | int | DWClinicReportDataYuanlongZhang.dbo.DimProcedures.ProcedureID | int | |
| Patients.dbo.Procedures.Name | varchar(100) | DWClinicReportDataYuanlongZhang.dbo.DimProcedures.ProcedureName | varchar(100) | |
| Patients.dbo.Procedures.Desc | varchar(1000) | DWClinicReportDataYuanlongZhang.dbo.DimProcedures.ProcedureDesc | varchar(1000) | |
| Patients.dbo.Procedures.Charge | money | DWClinicReportDataYuanlongZhang.dbo.DimProcedures.ProcedureCharge | money | |
| Patients.dbo.Visits.ID | int | DWClinicReportDataYuanlongZhang.dbo.FactVisits.VisitKey | int | |
| Patients.dbo.Visits.Date | datetime | DWClinicReportDataYuanlongZhang.dbo.FactVisits.DateKey | int | Extact Date from datetime |
| Patients.dbo.Visits.Clinic | int | DWClinicReportDataYuanlongZhang.dbo.FactVisits.ClinicKey | int | |
| Patients.dbo.Visits.Patient | int | DWClinicReportDataYuanlongZhang.dbo.FactVisits.PatientKey | int | |
| Patients.dbo.Visits.Doctor | int | DWClinicReportDataYuanlongZhang.dbo.FactVisits.DoctorKey | int | Replace null value as 'No Doctor' |
| Patients.dbo.Visits.Procedure | int | DWClinicReportDataYuanlongZhang.dbo.FactVisits.ProcedureKey | int | |
| Patients.dbo.Visits.Charge | money | DWClinicReportDataYuanlongZhang.dbo.FactVisits.ProcedureVistCharge | money | |
| DoctorsSchedules.dbo.Clinics.ClinicID | int | DWClinicReportDataYuanlongZhang.dbo.DimClinics.ClinicID | int | Convert 1,2,3 into 100, 200, 300 |
| DoctorsSchedules.dbo.Clinics.ClinicName | nvarchar(100) | DWClinicReportDataYuanlongZhang.dbo.DimClinics.ClinicName | nvarchar(100) | |
| DoctorsSchedules.dbo.Clinics.Address | nvarchar(100) | | | |
| DoctorsSchedules.dbo.Clinics.City | nvarchar(100) | DWClinicReportDataYuanlongZhang.dbo.DimClinics.ClinicCity | nvarchar(100) | |
| DoctorsSchedules.dbo.Clinics.State | nvarchar(100) | DWClinicReportDataYuanlongZhang.dbo.DimClinics.ClinicState | nvarchar(100) | |
| DoctorsSchedules.dbo.Clinics.Zip | nvarchar(5) | DWClinicReportDataYuanlongZhang.dbo.DimClinics.ClinicZip | nvarchar(5) | |
| DoctorsSchedules.dbo.Doctors.DoctorID | int | DWClinicReportDataYuanlongZhang.dbo.DimDoctors.DoctorID | int | |
| DoctorsSchedules.dbo.Doctors.FirstName | nvarchar(100) | DWClinicReportDataYuanlongZhang.dbo.DimDoctors.DoctorFullName | int | Concate First Name and Last Name |
| DoctorsSchedules.dbo.Doctors.LastName | nvarchar(100) | DWClinicReportDataYuanlongZhang.dbo.DimDoctors.DoctorFullName | nvarchar(200) | Concate First Name and Last Name |
| DoctorsSchedules.dbo.Doctors.EmailAddress | nvarchar(100) | DWClinicReportDataYuanlongZhang.dbo.DimDoctors.DoctorEmailAddress | nvarchar(200) | |
| DoctorsSchedules.dbo.Doctors.Address | nvarchar(100) | | | |
| DoctorsSchedules.dbo.Doctors.City | nvarchar(100) | DWClinicReportDataYuanlongZhang.dbo.DimDoctors.DoctorCity | nvarchar(100) | Trim the space |
| DoctorsSchedules.dbo.Doctors.State | nvarchar(100) | DWClinicReportDataYuanlongZhang.dbo.DimDoctors.DoctorState | nvarchar(100) | Marked bad data as "(Error)" |
| DoctorsSchedules.dbo.Doctors.Zip | nvarchar(5) | DWClinicReportDataYuanlongZhang.dbo.DimDoctors.DoctorZip | nvarchar(5) | |
| DoctorsSchedules.dbo.Shifts.ShiftID | int | DWClinicReportDataYuanlongZhang.dbo.DimShifts.ShiftID | int | |
| DoctorsSchedules.dbo.Shifts.ShiftStart | time | DWClinicReportDataYuanlongZhang.dbo.DimShifts.ShiftStart | time | Transformations of time format: 12:00 to 24:00 |
| DoctorsSchedules.dbo.Shifts.ShiftEnd | time | DWClinicReportDataYuanlongZhang.dbo.DimShifts.ShiftEnd | time | Transformations of time format: 12:00 to 24:00 |

| Source Data | Source Type | Destination | Destination Type | Transformations |
|---|---|---|---|---|
| DoctorsSchedules.dbo.DoctorShifts.DoctorsShiftID | int | DWClinicReportDataYuanlongZhang.dbo.FactDoctorShifts.DoctorsShiftID | int | |
| DoctorsSchedules.dbo.DoctorShifts.ShiftDate | datetime | DWClinicReportDataYuanlongZhang.dbo.FactDoctorShifts.ShiftDateKey | int | Extract Shiftkey from DimShift |
| DoctorsSchedules.dbo.DoctorShifts.ClinicID | int | DWClinicReportDataYuanlongZhang.dbo.FactDoctorShifts.ClinicKey | int | |
| DoctorsSchedules.dbo.DoctorShifts.ShiftID | int | DWClinicReportDataYuanlongZhang.dbo.FactDoctorShifts.ShiftKey | int | Convert 1,2,3 into 100, 200, 300 |
| DoctorsSchedules.dbo.DoctorShifts.DoctorID | int | DWClinicReportDataYuanlongZhang.dbo.FactDoctorShifts.DoctorKey | int | |
| | | DWClinicReportDataYuanlongZhang.dbo.FactDoctorShifts.HoursWorked | int | Calculated based on Shiftstart and ShiftEnd |
| | | DWClinicReportDataYuanlongZhang.dbo.DimPatients.StartDate | date | |
| | | DWClinicReportDataYuanlongZhang.dbo.DimPatients.EndDate | date | |
| | | DWClinicReportDataYuanlongZhang.dbo.DimPatients.IsCurrent | int | |
| | | DWClinicReportDataYuanlongZhang.dbo.DimDates.DateKey | int | |
| | | DWClinicReportDataYuanlongZhang.dbo.DimDates.FullDate | datetime | |
| | | DWClinicReportDataYuanlongZhang.dbo.DimDates.FullDateName | nvarchar(50) | |
| | | DWClinicReportDataYuanlongZhang.dbo.DimDates.MonthID | int | |
| | | DWClinicReportDataYuanlongZhang.dbo.DimDates.MonthName | nvarchar(50) | |
| | | DWClinicReportDataYuanlongZhang.dbo.DimDates.YearID | int | |
| | | DWClinicReportDataYuanlongZhang.dbo.DimDates.YearName | nvarchar(50) | |
| | | DWClinicReportDataYuanlongZhang.dbo.ETLLog.ETLLogID | int | |
| | | DWClinicReportDataYuanlongZhang.dbo.ETLLog.ETLDateAndTime | datetime | |
| | | DWClinicReportDataYuanlongZhang.dbo.ETLLog.ETLAction | varchar(100) | |
| | | DWClinicReportDataYuanlongZhang.dbo.ETLLog.ETLLogMessage | varchar(2000) | |
| | | DWClinicReportDataYuanlongZhang.dbo.vETLLog.ETLLogID | int | |
| | | DWClinicReportDataYuanlongZhang.dbo.vETLLog.ETLDate | nvarchar(4000) | |
| | | DWClinicReportDataYuanlongZhang.dbo.vETLLog.ETLTime | nvarchar(4000) | |
| | | DWClinicReportDataYuanlongZhang.dbo.vETLLog.ETLAction | varchar(100) | |
| | | DWClinicReportDataYuanlongZhang.dbo.vETLLog.ETLLogMessage | varchar(2000) | |

# 6. ETL Scripts Analysis & Explanation

In this section, we will demonstrate and provide more details about the script file:

## 6.1. Python Script

### 6.1.1. PythonETL.py (Milestone 1)

```python

# import required module
import os
import pandas as pd
import re
import pyodbc

# assign directory
directory = 'C:\_BISolutions\ETLFinal_YuanlongZhang\DataFiles\ClinicDailyData'

StagingData = pd.DataFrame()
StagingNewPatient = pd.DataFrame()
StagingVisits = pd.DataFrame()


# iterate over files in
# that directory
for root, dirs, files in os.walk(directory):
    for filename in files:
        StagingData = pd.read_csv(root + "\\" + filename)
        StagingData["Date"] = filename[:8]
        StagingData["Clinic"] = root.split("\\")[-1]
        if filename[8:11] == "New":
            frames = [StagingNewPatient, StagingData]
            StagingNewPatient = pd.concat(frames)

        if filename[8:11] == "Vis":
            frames = [StagingVisits, StagingData]
            StagingVisits = pd.concat(frames)

dfnew = pd.DataFrame()
dferror = pd.DataFrame()
i, j = StagingNewPatient.shape
for row in range(0,i):
    temp = StagingNewPatient.iloc[row,:]
    if(re.search('^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$',temp["Email"])):
         frames = [dfnew, temp]
         dfnew = pd.concat(frames,axis=1)
    else:
        def calculate_area(row):
            return "(error)" + row['Email']
        temp["Email"] = temp["Email"] + " (error)"
        frames = [dfnew, temp]
        dfnew = pd.concat(frames, axis=1)

# print(StagingNewPatient)
df_corrected = dfnew.transpose()
df_error = dferror.transpose()

# Load the data to SQL Server
# Connect to SQL
con_str = ("Driver={SQL Server Native Client 11.0};"
           "Server=localhost;"
           "Database=master;"
           "Trusted_Connection=yes;")
con_obj = pyodbc.connect(con_str)

# Create (update or delete) uses a cursor
cusor_obj = con_obj.cursor()
# cusor_obj.execute('''
#         USE [master];
#         If Exists (Select * from Sysdatabases Where Name = 'Staging')
```

*Read the CSV file from ClinicDailyData folder*

*Combine all the data from CSV file into the dataframes*

*Split data based on new patients/visits data; Add additional columns based on file name and folder name.*

*Use regular expression to determine the valid email-address. Marked invalid email address with suffix (error).*

*Established SQL Server connections.*

```
#             Begin
#                 ALTER DATABASE [Staging] SET SINGLE_USER WITH ROLLBACK IMMEDIATE
#                 DROP DATABASE [Staging]
#             End
#         Create Database [Staging];
#         ALTER DATABASE Staging SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
#         ALTER DATABASE Staging collate SQL_Latin1_General_CP1_CI_AS;
#         ALTER DATABASE Staging SET MULTI_USER;
#         ''')
cusor_obj.execute("Use Staging;If (object_id('StagingNewPatient') is not null) Drop Table
StagingNewPatient;")
SQL_str = '''Create Table StagingNewPatient
            (FName nvarchar(50)
            ,LName nvarchar(50)
            ,Email nvarchar(100)
            ,Address nvarchar(100)
            ,City nvarchar(50)
            ,State nvarchar(50)
            ,ZipCode nvarchar(20)
            ,Date nvarchar(20)
            ,Clinic nvarchar(20) )
          '''
cusor_obj.execute(SQL_str)

cusor_obj.execute("Use Staging;If (object_id('StagingVisits') is not null) Drop Table
StagingVisits;")
SQL_str = '''Create Table StagingVisits
            (Time time
            ,Patient int
            ,Doctor int
            ,[Procedure] int
            ,Charge int
            ,Date int
            ,Clinic nvarchar(20) )
          '''
cusor_obj.execute(SQL_str)

# Insert Dataframe into SQL Server:
SQL_str = ''' INSERT INTO Staging.dbo.StagingNewPatient
            (FName, LName, Email, Address, City, State, ZipCode, Date, Clinic)
            values(?,?,?,?,?,?,?,?,?) '''
for index, row in df_corrected.iterrows():
    cusor_obj.execute(SQL_str, row.FName, row.LName, row.Email, row.Address, row.City,
row.State,
    str(row.ZipCode), str(row.Date), row.Clinic)

con_obj.commit()

# Insert Dataframe into SQL Server:
SQL_str = ''' INSERT INTO Staging.dbo.StagingVisits
            (Time, Patient, Doctor, [Procedure], Charge, Date, Clinic)
            values(?,?,?,?,?,?,?) '''
for index, row in StagingVisits.iterrows():
    cusor_obj.execute(SQL_str, row.Time, row.Patient, row.Doctor, row.Procedure,
row.Charge, row.Date, row.Clinic)

con_obj.commit()
cusor_obj.close()

# Save ErrorReport data to a new file.
# df_error.to_csv('C:\_BISolutions\ETLFinal_YuanlongZhang\ErrorReport\ErrorReport.csv')

# print(df_corrected)
# print(df_error)
# print(StagingVisits)
```

*To avoid dirty read, we reset the table every time when ETL process execute.*

*Insert the records from dataframe into staging database*

### *6.1.2. MongoDBETL.py (Milestone 3)*

```
'''*************************************************************************--
-- Desc: This script connects to MongoDB Atlas cloud and Perform ETL procedures
-- Change Log: When,Who,What
-- 2020-05-15,RRoot,Created File
-- 2022-02-21,Yuanlong Zhang, Updated the File to add new functions for ETL.
-- 2022-03-09,Yuanlong Zhang, Updated the file for the final project.
--*************************************************************************'''

# Please install the package:
# Python -m pip install pyodbc

import pymongo
import pandas as pd
import pyodbc

try:
        # Connect to SQL
        con_str = ("Driver={SQL Server Native Client 11.0};"
                   "Server=localhost;"
                   "Database=DWClinicReportDataYuanlongZhang;"
                   "Trusted_Connection=yes;")
        con_obj = pyodbc.connect(con_str)

        # Create (update or delete) uses a cursor
        cusor_obj = con_obj.cursor()
        cusor_obj.execute("IF (Object_ID('vRptDoctorShifts') is not null) Drop View 
vRptDoctorShifts;")
        SQL_str = '''Create View vRptDoctorShifts
                        AS
                        Select
                        [ShiftDate] = Cast(Cast([FullDate] as date) as varchar(100))
                        ,[ClinicName] = dc.ClinicName
                        ,[ClinicCity] = dc.ClinicCity
                        ,[ClinicState] = dc.ClinicState
                        ,[ShiftID] = ds.ShiftID
                        ,[ShiftStart] = ds.ShiftStart
                        ,[ShiftEnd] = ds.ShiftEnd
                        ,[DoctorFullName] = ddo.DoctorFullName
                        ,[HoursWorked] = fds.HoursWorked
                        FROM
                        dbo.FactDoctorShifts fds
                        JOIN dbo.DimDates dd
                        ON fds.ShiftDateKey = dd.DateKey
                        JOIN dbo.DimClinics dc
                        ON fds.ClinicKey = dc.ClinicID
                        JOIN dbo.DimShifts ds
                        ON fds.ShiftKey = ds.ShiftID
                        JOIN dbo.DimDoctors ddo
                        ON fds.DoctorKey = ddo.DoctorKey;'''
        cusor_obj.execute(SQL_str)

        # Select from SQL Server using the Pandas module!
        vRptDoctorShifts = pd.read_sql("select * from vRptDoctorShifts;", con_obj)
        #print(vRptDoctorShifts)

        cusor_obj.execute("IF (Object_ID('vRptPatientVisits') is not null) Drop View 
vRptPatientVisits;")
        SQL_str = '''Create View vRptPatientVisits
                        AS
                        Select
                        [VisitDate] = Cast(Cast([FullDate] as date) as varchar(100))
                        ,[ClinicName] = dc.ClinicName
                        ,[ClinicCity] = dc.ClinicCity
                        ,[ClinicState] = dc.ClinicState
                        ,[ProcedureName] = dp.ProcedureName
                        ,[PatientFullName] = dpt.PatientFullName
                        ,[PatientCity] = dpt.PatientCity
                        ,[PatientState] =dpt.PatientState
                        ,[DoctorFullName] = ddo.DoctorFullName
```

*Establish connections to Data Warehouse*

*To avoid dirty read, reset and recreate the view every time*

*Load the data from SQL view into the dataframe.*

```
                        ,[ProcedureVisitCharge] = fv.ProcedureVistCharge
                        FROM
                        dbo.FactVisits fv
                        JOIN dbo.DimDates dd
                        ON fv.DateKey = dd.DateKey
                        JOIN dbo.DimClinics dc
                        ON fv.ClinicKey = dc.ClinicID
                        JOIN dbo.DimProcedures dp
                        ON fv.ProcedureKey = dp.ProcedureID
                        JOIN dbo.DimDoctors ddo
                        ON fv.DoctorKey = ddo.DoctorID
                        JOIN dbo.DimPatients dpt
                        ON fv.PatientKey = dpt.PatientID;'''
        cusor_obj.execute(SQL_str)

        # Select from SQL Server using the Pandas module!
        vRptPatientVisits = pd.read_sql("select * from vRptPatientVisits;", con_obj)
        #print(vRptPatientVisits)

        # Always clean up your objects when done!
        cusor_obj.close()
        con_obj.close()


vRptPatientVisits.to_csv('C:\\_BISolutions\\ETLFinal_YuanlongZhang\\DataFiles\\Staging\\vRptPatie
ntVisitStaging.csv')

vRptDoctorShifts.to_csv('C:\\_BISolutions\\ETLFinal_YuanlongZhang\\DataFiles\\Staging\\vRptDoctor
ShiftsStaging.csv')

        MongoRptPatient =
pd.read_csv('C:\\_BISolutions\\ETLFinal_YuanlongZhang
g.csv')
        MongoRptDoctor =
pd.read_csv('C:\\_BISolutions\\ETLFinal_YuanlongZhang\\DataFiles\\Staging\\vRptDoctorShiftsStagin
g.csv')

        # 1. Whitelist your IP address and create a user on
        # ( https://account.mongodb.com/account/login )
        # 2. Create a connection string
        # Note: the connection string can change without notice!
        strCon =
'mongodb+srv://BICert:BICert@clinicreportsdata.ts9ek.mongodb.net/test?retryWrites=true&w=majority
'

        objCon = pymongo.MongoClient(strCon)
        db = objCon["ClinicReportsData"]
        db.drop_collection('DoctorsShifts')
        objCol = db.create_collection('DoctorsShifts')
        MongoRptDoctor['_id'] = MongoRptDoctor['Unnamed: 0']
        MongoRptDoctor = MongoRptDoctor.drop(columns=['Unnamed: 0'])
        MongoRptDoctor = MongoRptDoctor.to_dict('records')
        objCol.insert_many(MongoRptDoctor)

        db.drop_collection('PatientsVisits')
        objCol = db.create_collection('PatientsVisits')
        MongoRptPatient['_id'] = MongoRptPatient['Unnamed: 0']
        MongoRptPatient = MongoRptPatient.drop(columns=['Unnamed: 0'])
        MongoRptPatient = MongoRptPatient.to_dict('records')
        objCol.insert_many(MongoRptPatient)

except Exception as e:
        print(e)
```

*Export the data into the external CSV file and preparing for uploading.*

*Read the CSV file into dataframe. (This step can solve the data type issue)*

*MongoDB connection string*

*Drop the collection every time before upload*

*Reset the id column and mapping to dataframe id*

*Transform the data type and insert into MongoDB.*

## *6.2. SQL Script*

**Note:**

- The SQL Script is very long. Thus, we hide the repeatable parts, help files, and the code provided by Randal.

### *6.2.1. ETLLoading.sql (Milestone 1)*

We use vETLNewPatient as an example:

*Define ETL Log Table*

```sql
--**************************************************************************--
-- Title: DWFinal-ETL Loading
-- Author: Yuanlong Zhang
-- Desc: This file used to load the new data into the database
-- Change Log: When,Who,What
-- 2021-01-17,RRoot,Created File
-- 2022-01-24,Yuanlong Zhang, Completed File
-- 2022-03-01,Yuanlong Zhang, Updated the file for Final Projects

--***************************************************************************--


USE Patients;
go

--  Setup Logging Objects ----------------------------------------------------

If NOT Exists(Select * From Sys.tables where Name = 'ETLLog')
  Create -- Drop
  Table ETLLog
  (ETLLogID int identity Primary Key
  ,ETLDateAndTime datetime Default GetDate()
  ,ETLAction varchar(100)
  ,ETLLogMessage varchar(2000)
  );
go

Create or Alter View vETLLog
As
  Select
   ETLLogID
  ,ETLDate = Format(ETLDateAndTime, 'D', 'en-us')
  ,ETLTime = Format(Cast(ETLDateAndTime as datetime2), 'HH:mm', 'en-us')
  ,ETLAction
  ,ETLLogMessage
  From ETLLog;
go


Create or Alter Proc pInsETLLog
 (@ETLAction varchar(100), @ETLLogMessage varchar(2000))
--**************************************************************************--
-- Desc: This Sproc creates an admin table for logging ETL metadata.
-- Change Log: When,Who,What
-- 2020-01-01,RRoot,Created Sproc
--**************************************************************************--
As
Begin
  Declare @RC int = 0;
  Begin Try
    Begin Tran;
      Insert Into ETLLog
       (ETLAction,ETLLogMessage)
      Values
       (@ETLAction,@ETLLogMessage)
    Commit Tran;
    Set @RC = 1;
  End Try
  Begin Catch
    If @@TRANCOUNT > 0 Rollback Tran;
    Set @RC = -1;
  End Catch
  Return @RC;
End
Go


--**************************************************************************--
-- A) Drop the FOREIGN KEY CONSTRAINTS and Clear the tables
 -- NOT NEEDED FOR INCREMENTAL LOADING:
--**************************************************************************--


--**************************************************************************--
-- B) Synchronize the Tables
--**************************************************************************--
```

*Created view for staging purposes.*

```
/****** [dbo].[DimCustomers] ******/
go
Create or Alter View vETLNewPatient
/* Author: Yuanlong Zhang
** Desc: Extracts and transforms data for DimCustomers
** Change Log: When,Who,What
** 2022-01-25,Yuanlong Zhang,Created Sproc (MERGE).
*/
As
  Select [FName] = [FName]
        ,[LName] = [LName]
        ,[Email] = [Email]
        ,[Address] = [Address]
        ,[City] = [City]
        ,[State] = [State]
                ,[ZipCode] = [ZipCode]
    FROM [Staging].[dbo].[StagingNewPatient]
go
/* Testing Code:
 SELECT * FROM vETLNewPatient;
*/



go
Create or Alter Procedure pETLSyncPatients
/* Author: Yuanlong Zhang
** Desc: Inserts data into DimCustomers
** Change Log: When,Who,What
** 2022-01-24,Yuanlong Zhang, Created Sproc (MERGE).
*/
As
Begin
  Declare @RC int = 0;
        Begin Try
    -- ETL Processing Code --
    Merge Into Patients as t
     Using vETLNewPatient as s -- For Merge to work with SCD tables, I need to
insert a new row when the following is not true:
      On  t.FName = s.FName
      And t.LName = s.LName
      And t.Email = s.Email
     When Not Matched -- At least one column value does not match add a new
row:
      Then
       Insert (FName, LName, Email, Address, City, State, ZipCode)
        Values (s.FName
              ,s.LName
              ,s.Email
              ,s.Address
                          ,s.City
                          ,s.State
              ,s.ZipCode)
      When Matched
          AND (t.Address <> s.Address OR t.City <> s.City OR t.State <>
```

*Use MERGE statement for incremental loading into the database table*

```sql
s.State OR t.ZipCode <> s.ZipCode)
          -- If there is a row in the target (dim) table that is no longer in
the source table
       Then -- indicate that row is no longer current
        Update
         Set t.Address = s.Address
            ,t.City = s.City
                        ,t.State = s.State
                        ,t.ZipCode = s.ZipCode
    ;

    -- ETL Logging Code --
                Exec pInsETLLog
                @ETLAction = 'pETLSyncPatients'
               ,@ETLLogMessage = 'Patients synced';
                Set @RC = +1
        End Try

        -- Error Handling Code --
        Begin Catch
                Declare @ErrorMessage nvarchar(1000) = Error_Messag

    -- ETL Logging Code --
                Exec pInsETLLog
              @ETLAction = 'pETLSyncPatients'
             ,@ETLLogMessage = @ErrorMessage;
                Set @RC = -1
        End Catch
        Return @RC;
End
go
/* Testing Code:
 Declare @Status int;
 Exec @Status = pETLSyncDimCustomers;
 Print @Status;
*/
go

/*
go
Declare @Status int = 0;
Exec @Status = pETLSyncPatients;
Select [Object] = 'pETLSyncPatients', [Status] = @Status;
select * from dbo.ETLLog

select * from Patients.dbo.Patients
--WHERE fname = 'Michael'
*/
```

*Logging the ETL processes into the ETLLog table.*

*You can always uncomment these code for debugging and testing purposes.*

### 6.2.2. *DWETLObjects.sql (Milestone 2)*

**Note:**

- The SQL Script is very long. Thus, we only pick up one of these tables as an example.

```sql
--*************************************************************************--
-- Title: DWFinal
-- Author: Yuanlong Zhang
-- Desc: This file tests your knowledge on how to create an Incremental ETL
process with SQL code
-- Change Log: When,Who,What
-- 2021-01-17,RRoot,Created File
-- 2022-01-24,Yuanlong Zhang, Completed File
-- 2022-03-07,Yuanlong Zhang, Updated the file for Final Projects

```

```
--**************************************************************************--
IF NOT EXISTS (SELECT 1 from sys.servers where name =
'continuumsql.westus2.cloudapp.azure.com')
BEGIN
  EXEC sp_addlinkedserver @server = 'continuumsql.westus2.cloudapp.azure.com'
  EXEC sp_addlinkedsrvlogin 'continuumsql.westus2.cloudapp.azure.com'
                           ,'false'
                           ,NULL
                           ,'BICert'
                           ,'BICert'
END

IF NOT EXISTS (SELECT 1 from sys.servers where name = 'is-
root01.ischool.uw.edu\BI')
BEGIN
  EXEC sp_addlinkedserver @server = 'is-root01.ischool.uw.edu\BI'
  EXEC sp_addlinkedsrvlogin 'is-root01.ischool.uw.edu\BI'
                           ,'false'
                           ,NULL
                           ,'BICert'
                           ,'BICert'
END


USE [DWClinicReportDataYuanlongZhang];
go
SET NoCount ON;

--  Setup Logging Objects ----------------------------------------------------

                      Script for ETLLog table is omitted

--********************************************************************--
-- A) Drop the FOREIGN KEY CONSTRAINTS and Clear the tables
 -- NOT NEEDED FOR INCREMENTAL LOADING:
--********************************************************************--


--********************************************************************--
-- B) Synchronize the Tables
--********************************************************************--
                      Script for DimDate table is omitted


/****** [dbo].[DimDoctors] ******/
go
Create or Alter View vETLDimDoctors
/* Author: Yuanlong Zhang
** Desc: Extracts and transforms data for DimDoctors
** Change Log: When,Who,What
** 2022-01-24,Yuanlong Zhang,Created Sproc (MERGE).
** 2022-03-07,Yuanlong Zhang,Created Sproc (MERGE) for final project.
*/
As
  Select
             [DoctorID] = d.DoctorID
            ,[DoctorFullName] = d.FirstName + ' ' + d.LastName
            ,[DoctorEmailAddress] = d.EmailAddress
            ,[DoctorCity] = TRIM(d.City)
            ,[DoctorState] = CASE WHEN LEN(TRIM(d.State)) > 2 THEN '(ERROR) ' +
TRIM(d.State) ELSE TRIM(d.State) END
            ,[DoctorZip] = d.Zip
  From
[continuumsql.westus2.cloudapp.azure.com].[DoctorsSchedules].[dbo].[Doctors] d
  UNION
  Select -1, 'No Doctor', 'N/A', 'N/A', 'N/A', 000000
go
/* Testing Code:
 Select * From vETLDimDoctors;
*/
```

*Define external server connections*

*Created view for ETL purposes.*

*To avoid violation of none null value in doctor, union a "no doctor" column.*

```

go
Create or Alter Procedure pETLSyncDimDoctors
/* Author: Yuanlong Zhang
** Desc: Updates data in DimDoctors using the vETLDimDoctors view
** Change Log: When,Who,What
** 2022-01-24,Yuanlong Zhang, Created Sproc (MERGE).
** 2022-03-07,Yuanlong Zhang, Created Sproc (MERGE) for the final project.
*/
AS
Begin
        Declare @RC int = 0;
  Begin Try
    -- ETL Processing Code --
    Merge Into DimDoctors as t
     Using vETLDimDoctors as s -- For Merge to work with SCD tables, I need to
insert a new row when the following is not true:
      On  t.DoctorID = s.DoctorID
     When Not Matched -- At least one column value does not match add a new
row:
      Then
       Insert (DoctorID, DoctorFullName,
DoctorEmailAddress,DoctorCity,DoctorState, DoctorZip)
        Values (s.DoctorID
                    ,s.DoctorFullName
             ,s.DoctorEmailAddress
             ,s.DoctorCity
             ,s.DoctorState
                    ,s.DoctorZip)
      When Matched
          AND t.DoctorEmailAddress <> s.DoctorEmailAddress
          OR t.DoctorCity <> s.DoctorCity
          OR t.DoctorState <> s.DoctorState
          OR t.DoctorZip <> s.DoctorZip-- If there is a row in the target
(dim) table that is no longer in the source table
       Then -- indicate that row is no longer current
        Update
         Set t.DoctorEmailAddress = s.DoctorEmailAddress
            ,t.DoctorCity = s.DoctorCity
                   ,t.DoctorState = s.DoctorState
                   ,t.DoctorZip = s.DoctorZip
          When Not Matched by Source
          Then
           Update
            Set t.DoctorFullName =
iif(patindex('%(Deleted)%',[DoctorFullName]) > 0, [DoctorFullName],
[DoctorFullName] + ' (Deleted)')
    ;

    -- ETL Logging Code --
                Exec pInsETLLog
                @ETLAction = 'pETLSyncDimDoctors'
               ,@ETLLogMessage = 'DimDoctors synced';
                Set @RC = +1
        End Try
        Begin Catch
                Declare @ErrorMessage nvarchar(1000) = Error_Message();
                Exec pInsETLLog
              @ETLAction = 'pETLSyncDimDoctors'
             ,@ETLLogMessage = @ErrorMessage;
                Set @RC = -1
        End Catch
        Return @RC;
End
go
/* Testing Code:
 Declare @Status int;
 Exec @Status = pETLSyncDimDoctors;
 Print @Status;
 Select * From DimDoctors
```

*MERGE Statement for incremental loading. We use doctor ID to identify the unique record*

*To avoid violate the FK constraints, we add "(deleted)" when we detected the column had been deleted*

```
*/


                      Script for other table is omitted


/****** [dbo].[DimPatients] ******/
go
Create or Alter View vETLDimPatients
/* Author: Yuanlong Zhang
** Desc: Extracts and transforms data for DimPatients - SCD-2
** Change Log: When,Who,What
** 2022-01-25,Yuanlong Zhang,Created Sproc (MERGE).
** 2022-03-07,Yuanlong Zhang,Created Sproc (MERGE) for final project.
*/
As
  Select [PatientID] = p.ID
        ,[PatientFullName] = Cast(p.FName+ ' ' + p.LName as nvarchar(100))
        ,[PatientCity] = p.City
        ,[PatientState] = p.State
        ,[PatientZipCode] = p.ZipCode
    FROM [is-root01.ischool.uw.edu\BI].[Patients].[dbo].[Patients] p
go
/* Testing Code:
 Select * From vETLDimPatients;
*/

go
Create or Alter Procedure pETLSyncDimPatients
/* Author: Yuanlong Zhang
** Desc: Inserts data into DimPatients  - Type 2 SCD
** Change Log: When,Who,What
** 2022-01-24,Yuanlong Zhang, Created Sproc (MERGE).
** 2022-03-07,Yuanlong Zhang, Created Sproc (MERGE) for the final project.
*/
As
Begin
  Declare @RC int = 0;
        Begin Try
    -- ETL Processing Code --
    Merge Into DimPatients as t
     Using vETLDimPatients as s -- For Merge to work with SCD tables, I need
to insert a new row when the following is not true:
      On  t.PatientID = s.PatientID
      And t.PatientFullName = s.PatientFullName
      And t.PatientCity = s.PatientCity
      And t.PatientState = s.PatientState
          And t.PatientZipCode = s.PatientZipCode
     When Not Matched -- At least one column value does not match add a new
row:
      Then
       Insert (PatientID, PatientFullName, PatientCity, PatientState,
PatientZipCode,
               StartDate, EndDate, IsCurrent)
        Values (s.PatientID
               ,s.PatientFullName
               ,s.PatientCity
               ,s.PatientState
                           ,s.PatientZipCode
               ,Cast(Convert(nvarchar(100), GetDate(), 112) as date) -- Smart
Key can be joined to the DimDate
               ,Null
               ,1)
      When Not Matched By Source -- If there is a row in the target (dim)
table that is no longer in the source table
       Then -- indicate that row is no longer current
        Update
         Set t.EndDate = Cast(Convert(nvarchar(100), GetDate(), 112) as date)
-- Smart Key can be joined to the DimDate
            ,t.IsCurrent = 0
```

*Patient table use SCD Type-2.
We added 3 columns for StartDate, EndDate and IsCurrent.
The date will get system time when executed.*

*When deletion, we update the EndDate and marked iscurrent as 0.*

```
    ;

    -- ETL Logging Code --
                Exec pInsETLLog
                @ETLAction = 'pETLSyncDimPatients'
               ,@ETLLogMessage = 'DimPatients synced';
                Set @RC = +1
        End Try

        -- Error Handling Code --
        Begin Catch
                Declare @ErrorMessage nvarchar(1000) = Error_Message();

    -- ETL Logging Code --
                Exec pInsETLLog
              @ETLAction = 'pETLSyncDimPatients'
             ,@ETLLogMessage = @ErrorMessage;
                Set @RC = -1
        End Catch
        Return @RC;
End
go
/* Testing Code:
 Declare @Status int;
 Exec @Status = pETLSyncDimPatients;
 Print @Status;
 select * from dimpatients
*/
go

                    Script for other table is omitted
```

### 6.2.3. *DWMongoDBView.sql (Milestone 3)*

The two views have a similar structure. Thus, we use vRptDoctorShifts as an example:

```
--*************************************************************************--
-- Title: DWFinal-DWMongoDBView
-- Author: Yuanlong Zhang
-- Desc: This file used to create the view to feed MongoDB
-- Change Log: When,Who,What
-- 2021-01-17,RRoot,Created File
-- 2022-01-24,Yuanlong Zhang, Completed File
-- 2022-03-10,Yuanlong Zhang, Updated the file for Final Projects

--**************************************************************************--

USE DWClinicReportDataYuanlongZhang;
GO

IF (Object_ID('vRptDoctorShifts') is not null) Drop View vRptDoctorShifts;
GO
Create View vRptDoctorShifts
AS
Select
[ShiftDate] = Cast(Cast([FullDate] as date) as varchar(100))
,[ClinicName] = dc.ClinicName
,[ClinicCity] = dc.ClinicCity
,[ClinicState] = dc.ClinicState
,[ShiftID] = ds.ShiftID
,[ShiftStart] = ds.ShiftStart
,[ShiftEnd] = ds.ShiftEnd
,[DoctorFullName] = ddo.DoctorFullName
,[HoursWorked] = fds.HoursWorked
FROM
dbo.FactDoctorShifts fds
JOIN dbo.DimDates dd
```

*To avoid dirty read, drop the view every time.*

*Convert all the data into string to avoid data type issue*

```sql
ON fds.ShiftDateKey = dd.DateKey
JOIN dbo.DimClinics dc
ON fds.ClinicKey = dc.ClinicID
JOIN dbo.DimShifts ds
ON fds.ShiftKey = ds.ShiftID
JOIN dbo.DimDoctors ddo
ON fds.DoctorKey = ddo.DoctorKey

```

### 6.2.4. *ETLJob.sql (Milestone 4)*

```sql
--*************************************************************************--
-- Title: Create the DW ETL Job
-- Desc: This file will drop and create a SQL Agent Job
-- Change Log: When,Who,What
-- 2020-01-01,RRoot,Created File
-- 2022-03-14,Yuanlong Zhang, Updated file for Final Project
--*************************************************************************--


USE [master]
GO
Begin Try
-- Access to the Server
CREATE LOGIN [DESKTOP-AO8N3T1\i_ecn]
 FROM WINDOWS
  WITH DEFAULT_DATABASE=[master], DEFAULT_LANGUAGE=[us_english]
```

*Create Login for local account. You have to update the account name on your computer.*

```sql
ALTER SERVER ROLE [sysadmin] ADD MEMBER [DESKTOP-AO8N3T1\i_ecn]
End Try
Begin Catch
        Print Error_Message()
End Catch
GO

-- Abstaction layer to the Login
Begin Try
CREATE CREDENTIAL [CredentialForETLAutomations]
 WITH IDENTITY = N'DESKTOP-AO8N3T1\i_ecn'
End Try
Begin Catch
        Print Error_Message()
End Catch
GO
```

*Create a credential for he identity we set up before.*

```sql
-- Connection to an account with enough permission
-- using the abstraction layer credentials
Begin Try
        EXEC msdb.dbo.sp_add_proxy
         @proxy_name=N'SSIS Proxy'
        ,@credential_name=N'CredentialForETLAutomations'
        ,@enabled=1

        -- Map to SSIS subsystems
        EXEC msdb.dbo.sp_grant_proxy_to_subsystem
         @proxy_name=N'SSIS Proxy'
        ,@subsystem_id=11  -- SSIS Package
End Try
Begin Catch
        Print Error_Message()
End Catch
GO
```

*Create a SSIS Proxy for SQL Agent Jobs; Use Credential which we created before*

```sql

USE [msdb]
GO
BEGIN TRY
  IF Exists (Select * from SysJobs Where Name = 'ETLDWClinicReportData')
```

```
    Begin
      Exec sp_delete_job @job_name = ETLDWClinicReportData
    End

  /****** Object:  Job [DWClinicReportDataYuanlongZhang]    Script Date: 
8/21/2021 3:46:14 PM ******/
  BEGIN TRANSACTION
  DECLARE @ReturnCode INT
  SELECT @ReturnCode = 0
  /****** Object:  JobCategory [[Uncategorized (Local)]]    Script Date: 
8/21/2021 3:46:14 PM ******/
  IF NOT EXISTS (SELECT name FROM msdb.dbo.syscategories WHERE 
name=N'[Uncategorized (Local)]' AND category_class=1)
  BEGIN
  EXEC @ReturnCode = msdb.dbo.sp_add_category @class=N'JOB', @type=N'LOCAL', 
@name=N'[Uncategorized (Local)]'
  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

  END

  DECLARE @jobId BINARY(16)
  EXEC @ReturnCode =  msdb.dbo.sp_add_job @job_name=N'ETLDWClinicReportData', 
                  @enabled=1, 
                  @notify_level_eventlog=0, 
                  @notify_level_email=0, 
                  @notify_level_netsend=0, 
                  @notify_level_page=0, 
                  @delete_level=0, 
                  @description=N'Performs ETL tasks for 
ETLDWClinicReportData', 
                  @category_name=N'[Uncategorized (Local)]', 
                  @owner_login_name=N'sa', @job_id = @jobId OUTPUT
  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
  /****** Object:  Step [Run DWIndependentBookSellersETLpackage.dtsx]    
Script Date: 8/21/2021 3:46:14 PM ******/
  EXEC @ReturnCode = msdb.dbo.sp_add_jobstep @job_id=@jobId, @step_name=N'Run 
ETLJob.dtsx', 
                  @step_id=1, 
                  @cmdexec_success_code=0, 
                  @on_success_action=1, 
                  @on_success_step_id=0, 
                  @on_fail_action=2, 
                  @on_fail_step_id=0, 
                  @retry_attempts=0, 
                  @retry_interval=0, 
                  @os_run_priority=0, @subsystem=N'SSIS', 
                  @command=N'/FILE 
"C:\_BISolutions\ETLFinal_YuanlongZhang\ETLPackages\ETLJob.dtsx" 
/CHECKPOINTING OFF /REPORTING E', 
                  @database_name=N'master', 
                  @flags=0, 
                  @proxy_name=N'SSIS Proxy'
  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
  EXEC @ReturnCode = msdb.dbo.sp_update_job @job_id = @jobId, @start_step_id = 
1
  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
  EXEC @ReturnCode = msdb.dbo.sp_add_jobschedule @job_id=@jobId, 
@name=N'EachNight', 
                  @enabled=1, 
                  @freq_type=4, 
                  @freq_interval=1, 
                  @freq_subday_type=1, 
                  @freq_subday_interval=0, 
                  @freq_relative_interval=0, 
                  @freq_recurrence_factor=0, 
                  @active_start_date=20210821, 
                  @active_end_date=99991231, 
                  @active_start_time=10000, 
                  @active_end_time=235959, 
                  @schedule_uid=N'ac7412ed-e42f-46a0-a8bb-d16ccf0310fb'
```

*Deleted SQL Agent Job if existed*

*Created a new job which execute the ETLJob.dtsx*

*Scheduled the job to run each night.*

```
  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
  EXEC @ReturnCode = msdb.dbo.sp_add_jobserver @job_id = @jobId, @server_name
= N'(local)'
  IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
  COMMIT TRANSACTION
  GOTO EndSave
  QuitWithRollback:
      IF (@@TRANCOUNT > 0) ROLLBACK TRANSACTION
  EndSave:

END TRY
BEGIN CATCH
  IF (@@TRANCOUNT > 0) ROLLBACK TRANSACTION
  Print Error_Message()
END CATCH

GO

```

## 6.2.5. *ETLViews.sql (Milestone 4)*

```
--*************************************************************************--
-- Title: Testing the Reporting Views
-- Author: RRoot, Yuanlong Zhang
-- Desc: This file creates several ETL views used in Admin reports
-- Change Log: When,Who,What
-- 2018-02-07,RRoot,Created File
-- 2022-03-15, Yuanlong Zhang, Updated for the final project
--**************************************************************************--
Use DWClinicReportDataYuanlongZhang;


--Create views for SSIS Job
Create or Alter View vDWClinicReportDataYuanlongZhangETLJobHistory
As
Select Top 100000
 [JobName] = j.name
,[StepName] = h.step_name
,[RunDateTime] = msdb.dbo.agent_datetime(run_date, run_time)
,[RunDurationSeconds] = h.run_duration
,[RunStatus] = iif(h.run_status = 1, 'Success', 'Failure')
From msdb.dbo.sysjobs as j
  Inner Join msdb.dbo.sysjobhistory as h
    ON j.job_id = h.job_id
--Where j.enabled = 1 And j.name = 'ETLDWClinicReportData'
Order by JobName, RunDateTime desc;


--Create a view for row count reports
Create or Alter View DWClinicReportDataRowCounts
As
With [RowCounts] -- Using a CTE to access the Top Command for the Order By
statement in the view
As(
Select [SortCol] = 1, [TableName] = 'DimDates', [CurrentNumberOfRows] =
Count(*) From [DimDates]
Union
Select [SortCol] = 2, [TableName] = 'DimClinics', [CurrentNumberOfRows] =
Count(*) From [DimClinics]
Union
Select [SortCol] = 3, [TableName] = 'DimDoctors', [CurrentNumberOfRows] =
Count(*) From [DimDoctors]
Union
Select [SortCol] = 4, [TableName] = 'DimPatients', [CurrentNumberOfRows] =
Count(*) From [DimPatients]
Union
Select [SortCol] = 5, [TableName] = 'DimProcedures', [CurrentNumberOfRows] =
Count(*) From [DimProcedures]
Union
```

*Created a SQL Agent Job log view from MSDB.*

*Convert the running status from 1,0 to success and failure.*

*Create rowcount table by union all the individual row count stats*

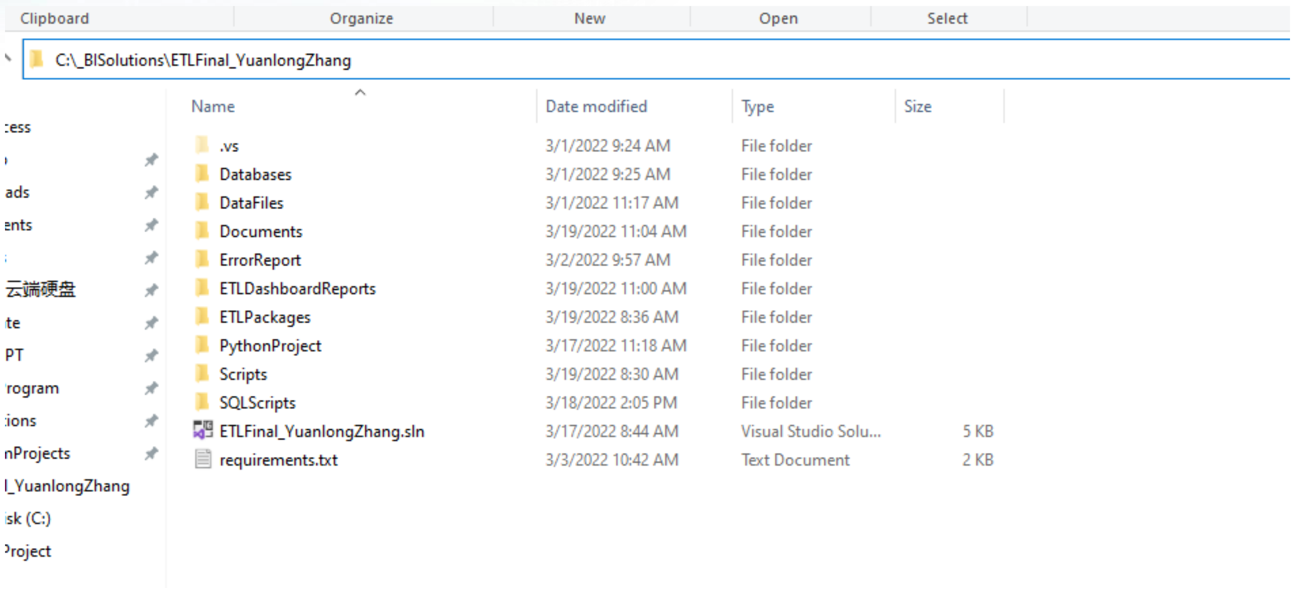
```sql
Select [SortCol] = 6, [TableName] = 'DimShifts', [CurrentNumberOfRows] =
Count(*) From [DimShifts]
Union
Select [SortCol] = 7, [TableName] = 'FactDoctorShifts', [CurrentNumberOfRows]
= Count(*) From [FactDoctorShifts]
Union
Select [SortCol] = 8, [TableName] = 'FactVisits', [CurrentNumberOfRows] =
Count(*) From [FactVisits]
Union
Select [SortCol] = 9, [TableName] = 'ETLLog', [CurrentNumberOfRows] = Count(*)
From [ETLLog]
)
Select Top 100000 [SortCol],[TableName],[CurrentNumberOfRows]
  From [RowCounts]
  Order By [SortCol] asc; -- Use a sort column, so it does not sort by table
name.
go

```

*Sorted by SortCol rather than by table name.*

## 7. Summary

This manual provides an overview of our ETL process.

You must carefully follow **section 2 Checklist** and ensure the zip has been **unzipped into C:_BISolutions\ETLFinal_YuanlongZhang** before starting.


Clipboard | Organize | New | Open | Select

C:_BISolutions\ETLFinal_YuanlongZhang

| Name | Date modified | Type | Size |
|---|---|---|---|
| .vs | 3/1/2022 9:24 AM | File folder | |
| Databases | 3/1/2022 9:25 AM | File folder | |
| DataFiles | 3/1/2022 11:17 AM | File folder | |
| Documents | 3/19/2022 11:04 AM | File folder | |
| ErrorReport | 3/2/2022 9:57 AM | File folder | |
| ETLDashboardReports | 3/19/2022 11:00 AM | File folder | |
| ETLPackages | 3/19/2022 8:36 AM | File folder | |
| PythonProject | 3/17/2022 11:18 AM | File folder | |
| Scripts | 3/19/2022 8:30 AM | File folder | |
| SQLScripts | 3/18/2022 2:05 PM | File folder | |
| ETLFinal_YuanlongZhang.sln | 3/17/2022 8:44 AM | Visual Studio Solu... | 5 KB |
| requirements.txt | 3/3/2022 10:42 AM | Text Document | 2 KB |

*Figure 57 ETL Solution Folder*

You can search the keywords to locate the topic you are interested in quickly.

For more information, don't hesitate to contact Yuanlong Zhang via email: yuanlong@uw.edu.

■