ETL Technical Manual

***Version: 1.0*  
*Date: 03/25/2025*  
*Author: Girum Obse*  
*UW ETL Processing Final***

**Table of Contents**

1. Introduction
2. Project Overview
3. Environment Setup
4. Milestone 1: File-Based ETL
   * Data Review
   * Setting Up the Development Environment
   * Database Restoration
   * Data Transformation Documentation
   * SSIS Package Implementation
5. Milestone 2: Data Warehouse ETL
   * Creating the Data Warehouse
   * Reviewing the Database Design
   * SQL ETL and SSIS Implementation
6. Milestone 3: Non-SQL ETL
   * View Creation
   * Python-Based ETL
   * SSIS Package Integration
7. Milestone 4: Automation, Reports, and Documentation
   * Automation Strategies
   * Report Generation
   * Documentation Best Practices
8. Summery

# 1. Introduction

## 1.1 Purpose of the Document

## This document serves as a comprehensive technical guide for understanding and implementing ETL (Extract, Transform, Load) processes, of a small medical clinic data analytics project. The project involves automating the data ingestion, transformation, and loading processes from CSV files to a structured Data Warehouse. The ETL solution prioritizes scalability, efficiency, and maintainability using SQL Server, SSIS, and Python. This manual provides step-by-step instructions on ETL development, best practices, and troubleshooting techniques.

## Additionally, it outlines the four milestones of the project, covering file imports, data warehouse integration, non-SQL ETL processes, and automation. Security, compliance, and performance optimization strategies are also discussed to ensure smooth data flow between systems. Whether building an ETL pipeline for the first time or optimizing an existing one, this document serves as a valuable reference for designing scalable and efficient ETL solutions.

**Current Situation**

Currently, individual clinics send daily CSV files to the corporate office, where they are manually reviewed, cleaned, and added to one of two databases. This manual ETL process is time-consuming, prone to human error, and lacks scalability, especially as the number of clinics increases. The process delays data availability, affecting decision-making and efficiency.

**2. Project Overview**

The ETL project automates data movement from daily CSV uploads into structured databases, providing analytical insights. The existing system is entirely manual, and this solution introduces automation to streamline data integration.

* **Source Systems**: CSV files from clinics
* **Destination Systems**: Patients and DoctorsSchedules databases, DWClinicReportData (Data Warehouse), and Excel reports
* **ETL Tools**: SQL Server, SSIS, Python
* **Deployment**: Visual Studio-based structured solution

**2.1 Current Workflow over view**

**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**.

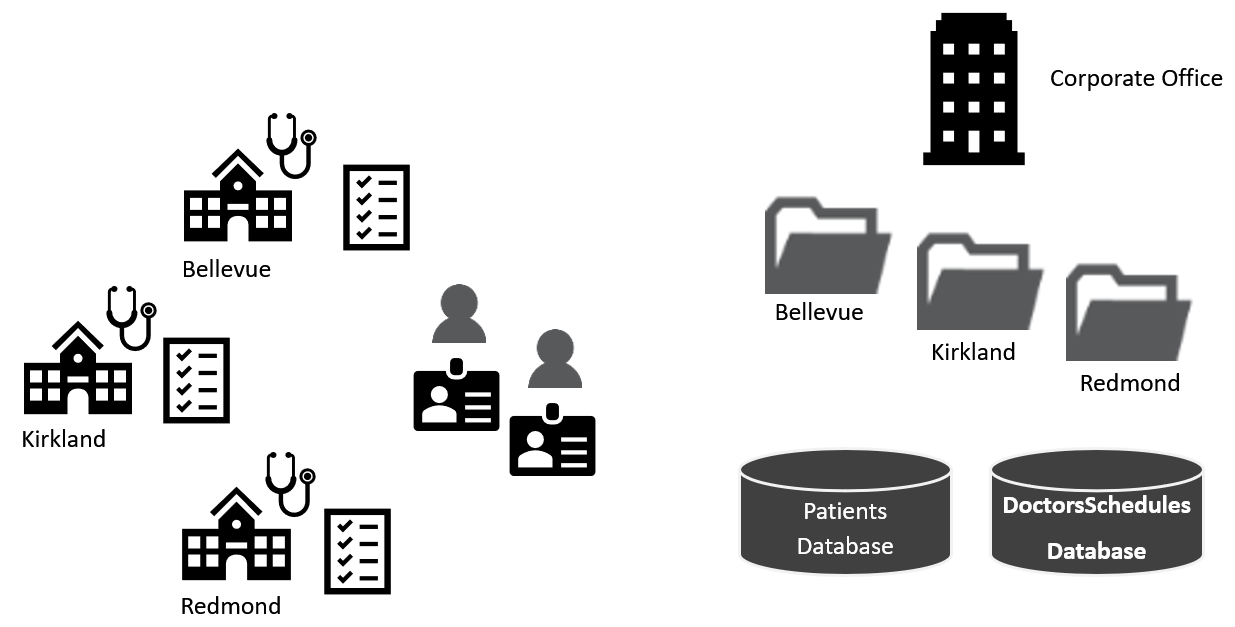


Figure 1. Topology of the current design.

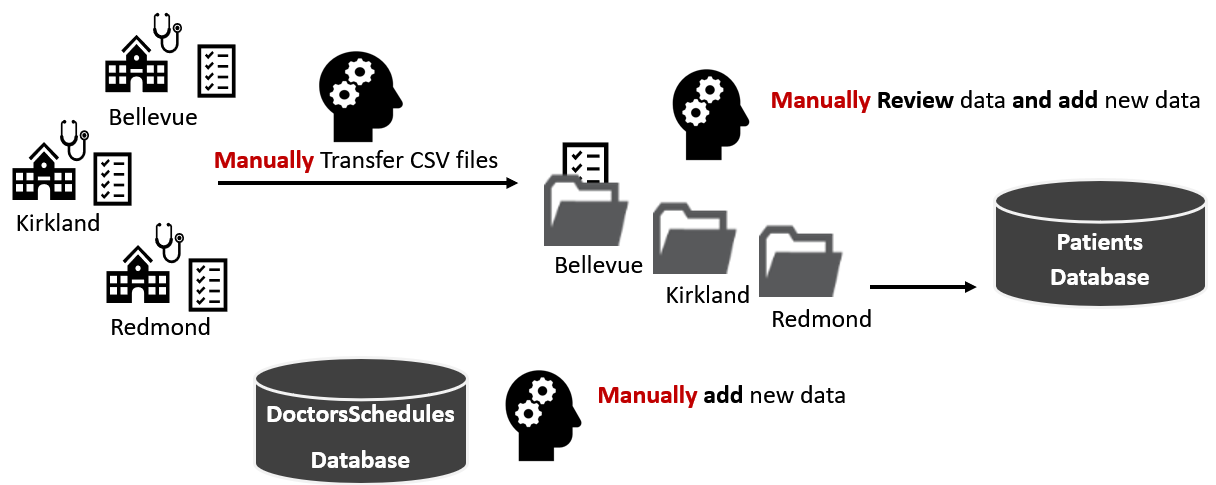


Figure 2. Visualization of the manual ETL file process

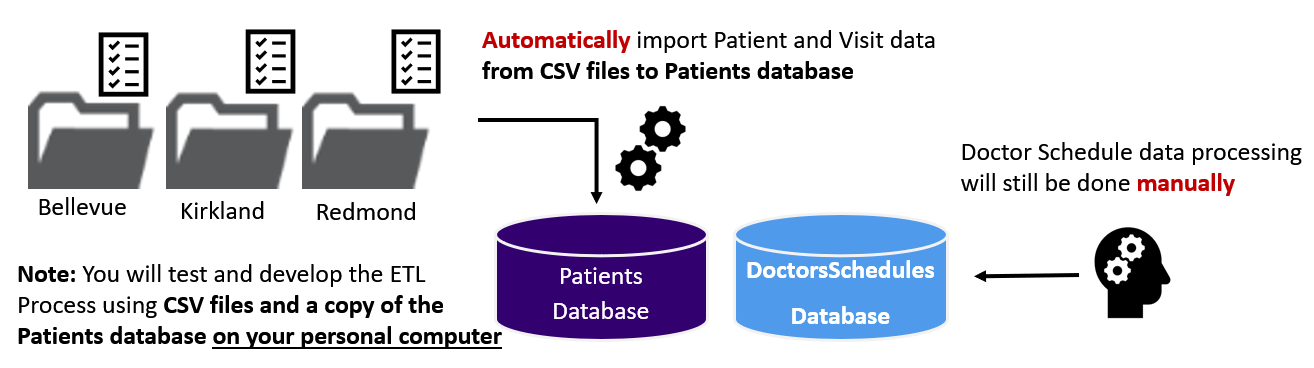


Figure 3. Visualization of the manual ETL file process

**2.2,Proposed ETL process**

A new ETL process will automate parts of this workflow, with potential for full automation in the future. Below, you'll find detailed descriptions of the entire process and steps, along with sample code snippets and screenshots for clarity.

2.3 Audience

This manual is intended for a wide range of professionals involved in data management and analytics, including:  
- ETL Developers – Those responsible for designing and implementing ETL pipelines.  
- Data Engineers – Professionals who manage data pipelines and ensure data flows correctly.  
- Database Administrators – Responsible for maintaining data integrity and optimizing database performance.  
- Business Intelligence Analysts – Those who analyze and report on business data.  
- IT Managers – Decision-makers overseeing data-related projects.  
This document assumes some familiarity with databases, SQL, and basic scripting but does not require advanced ETL experience.

# 2.4 ETL Overview

## 2.4.1 What is ETL?

ETL stands for Extract, Transform, and Load—a three-step process used for integrating data from multiple sources into a centralized data warehouse or analytics system. The Extract phase retrieves data from various sources such as databases, web APIs, and flat files. The Transform phase applies business rules to clean, standardize, and enrich the data. The Load phase ensures the data is stored in the target system, ready for reporting and analysis. ETL is widely used in data warehousing, analytics, and business intelligence, ensuring that organizations work with high-quality, consistent, and structured data.

## 2.4.1 Why ETL is Important?

ETL plays a crucial role in modern data-driven organizations by enabling them to consolidate data from multiple sources and prepare it for analysis. Without ETL, businesses would struggle with inconsistent, incomplete, and redundant data. ETL ensures that data is cleaned, standardized, and formatted properly before it reaches reporting tools. Additionally, ETL processes support historical data storage, allowing companies to perform trend analysis, forecasting, and strategic decision-making. ETL also enhances data governance and compliance, ensuring that sensitive information is processed securely according to industry regulations such as GDPR and HIPAA.

**3.Environment Setup**

Before beginning ETL development, ensure the following components are installed and configured:

* SQL Server with Management Studio
* Visual Studio with SSIS extensions
* Python 3.9 with required libraries
* Access to database backup files (.BAK) for restoration
* CSV files for testing in the C:\\_BISolutions\ClinicDailyData directory

**4. Milestone 1: File-Based ETL**

**Data Review**

* Assess CSV file structures for inconsistencies (e.g., column names, data types, missing values).

**Setting Up the Development Environment**

* Create a Visual Studio solution with organized folders.

**Database Restoration**

Execute the following SQL script to restore databases:

ALTER DATABASE [Patients] SET SINGLE\_USER WITH ROLLBACK IMMEDIATE;

RESTORE DATABASE [Patients]

FROM DISK = N'C:/\_BISolutions/Databases/Patients.bak'

WITH RECOVERY, REPLACE;

ALTER DATABASE [Patients] SET MULTI\_USER;

GO

**Data Transformation Documentation**

* Maintain an Excel metadata sheet tracking transformations.

**SSIS Package Implementation**

* Develop an SSIS package (ETLFilesToDatabases.dtsx) to automate CSV imports.

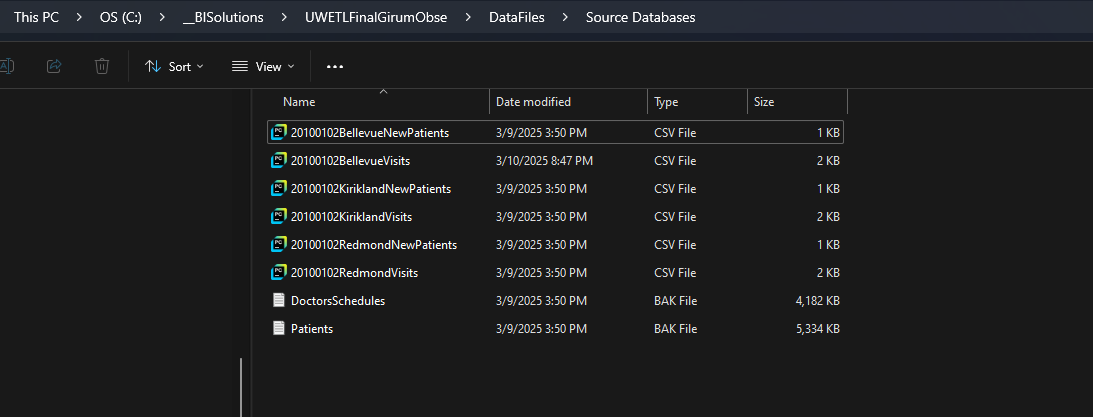
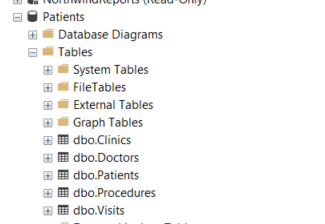


Figure 4 Excel source file for different clinics

A screenshot of a computer program

Description automatically generated

Figure5, sql code script showing restoring the patients and doctorschedule database



A screenshot of a computer

Description automatically generated

Fiqure6, showing doctorschedule and patients database tables

A screenshot of a computer program

Description automatically generated

Figure7: part of sql script showing how data loaded to databases

A screenshot of a computer

Description automatically generated

Figure 8, ssis package for dataloaded from excel source to visits table in patients database

A screenshot of a computer

Description automatically generated

Figure9, showing screenshot of scheduled job to load data to patients database tables

A screenshot of a computer

Description automatically generated

Figure 10, showing data in patients database tables

**5. Milestone 2: Data Warehouse ETL**

**Process Steps**

**Creating the Data Warehouse**

**- Run the provided SQL script (`\Scripts\CreateDWClinicReportData.sql`) to create the \*\*DWClinicReportData\*\* database.**

**A screenshot of a computer program

Description automatically generated**

**Figure11, sql code script of the datawarehouse**

**- Verify that all required tables, schemas, and constraints are properly created.**

**- Check for any errors or missing objects in the database after execution.**

**Reviewing the Database Design**

**- Identify the \*\*dimension tables\*\* (e.g., DimPatients, DimDoctors, DimShifts) and fact tables(e.g., FactDoctorShifts and FactVisits).**

**- Understand relationships between tables, primary and foreign keys.**

**A screenshot of a computer

Description automatically generated**

**SQL ETL and SSIS Implementation**

**- \*\*Create SQL Views and Stored Procedures\*\*:**

**- Develop \*\*views\*\* to extract necessary data from the \*\*DoctorsSchedules\*\* and \*\*Patients\*\* OLTP databases.**

**- Write \*\*stored procedures\*\* to transform and load the data into the \*\*DWClinicReportData\*\* warehouse.**

**- Implement \*\*data cleansing\*\* to remove duplicates, handle null values, and ensure consistency.**

**Figure12, showing ETL script of data load from OLTP database to the OLAP database example of Dimdoctor table**

**A screenshot of a computer program

Description automatically generated**

**- \*\*Develop an SSIS Package (DWClinicReportDataETL.dtsx)\*\*:**

**- Create an \*\*SSIS package\*\* to automate the ETL process.**

**- Configure \*\*Data Flow tasks\*\* to extract data from the source OLTP databases.**

**- Use \*\*Staging Tables\*\* for incremental loads and ensure proper error handling.**

**- Implement \*\*Logging and Notifications\*\* to track process execution and failures.**

**A screenshot of a computer

Description automatically generated**

**Figure13, showing ssis package of the datawarehouse**

**- Schedule the SSIS package for execution via \*\*SQL Server Agent\*\* or manual triggers.**

**A screenshot of a computer

Description automatically generated**

**Figure 14, showing the ssis scheduled job and job history.**

A screenshot of a computer

Description automatically generated

Figure15, demonstrating the data in DataClinicReport datawarehouse tables

A screenshot of a computer

Description automatically generated

Figure16, demonstrating the metadata worksheets

A screenshot of a computer

Description automatically generated

**6. Milestone 3: Non-SQL ETL**

**Process Steps**

**Creating SQL Views**

* **Develop two SQL views in the DWClinicReportData database:**
  + **One for doctor shifts data.**
  + **One for patient visits data.**
* **Ensure the views contain relevant columns for analytical reporting.**
* **Validate the views by querying sample data.**

**Sql code for the views..**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**Extract, Transform, and Load (ETL) using Python**

* **Develop a Python script to perform ETL operations:**
  + **Extract data from the SQL views.**
  + **Transform the data as needed (e.g., date formatting, column renaming).**
  + **Load the data into an Excel spreadsheet named ClinicReportsData\_.xlsx.**
* **Store the Excel file in the solution's Reports folder.**
* **Validate that the data is correctly loaded into the two worksheets (one per view).**

**Screernshot of the python script for the ETL for Clinic Reports**

**A screenshot of a computer program

Description automatically generated**

**SSIS Package Integration**

* **Develop an SSIS package (ETLClinicReportsDocumentData.dtsx):**
  + **Automate execution of the Python script.**
  + **Handle errors and log execution results.**
  + **Ensure proper folder and file management for storing output reports.**

**Updating ETL Process Metadata**

* **Update the Excel metadata spreadsheet:**
  + **Document source, destination, and transformation rules.**
  + **Ensure all tabs are updated as per the ETL workflow.**
* **Verify the metadata consistency with the implemented ETL process.**

**A screenshot of a computer

Description automatically generated**

**Ssis package the executes the python ETL script**

**Excel worksheet of the clinic reports**

**A screenshot of a computer

Description automatically generated**

**7. Milestone 4: Automation, Reports, and Documentation**

Process Steps

## Creating SQL Reporting Views

• Develop ETL report views to extract data from the ETL log and MSDB jobs tables.  
• Ensure the views include relevant columns for tracking ETL execution status, errors, and processing times.  
• Validate the views by running queries to confirm correct data extraction and filtering.  
• Optimize performance by adding appropriate indexing where necessary.

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

Figure20, sql script for the views vw\_ETL Log and vw\_JobHistory

## Creating ETL Dashboard Reports with SSRS

Create an ETL report that shows the contents of the ETL log and MSDB jobs table.  
 - The report should display execution details, success/failure statuses, error messages, and timestamps.  
 - Use appropriate filters and sorting options to ensure the report provides useful insights.  
 - Ensure that your Data Sources and DataSets are properly configured and named according to their respective database objects.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Figure22, screeshot of the SSRS report for MSDB jobs and ETL Log Report

## Summery

The ETL Technical Manual outlines the implementation of an automated ETL solution for a small medical clinic, replacing the current manual CSV-based process with a structured system using SQL Server, SSIS, and Python.

The project involves automating data ingestion, transformation, and loading into a Data Warehouse, improving efficiency and accuracy. Key milestones include file-based ETL for CSV imports, data warehouse ETL for structured reporting, Python-based ETL for Excel-based analytics, and automation using SSIS.

The solution also integrates SQL views, stored procedures, and SSRS dashboards to track ETL performance and errors. Designed for scalability and reliability, this manual serves as a guide for data engineers, ETL developers, and BI analysts involved in data integration projects.