Data Warehousing and Business Intelligence

Assignment 01

IT22550262



Table of Contents

[Dataset 2](#_Toc197026690)

[Solution Architecture 2](#_Toc197026691)

[Data warehouse design and development 3](#_Toc197026692)

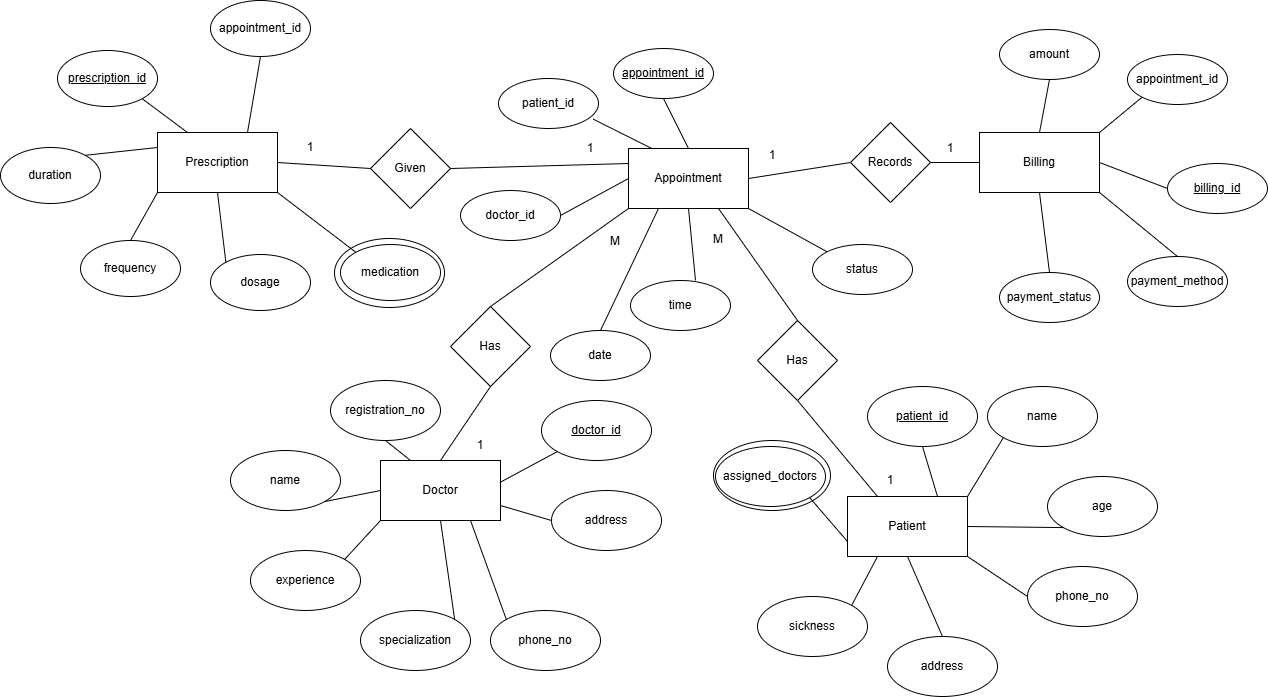
[ETL development 7](#_Toc197026693)

[Accumulating Fact Table 35](#_Toc197026694)

# 

# Dataset

This dataset is related to a hospital management system where they record the details of patients visited, Details of the staff (Doctors), Appointment details, Prescriptions given and the Transactional data like billing details. Since it has a good diversified meaningful amount of transactional data, I selected this as my project scenario. The below Entity-Relational diagram depicts how the data sources interact with each other.



(Entity relation diagram)

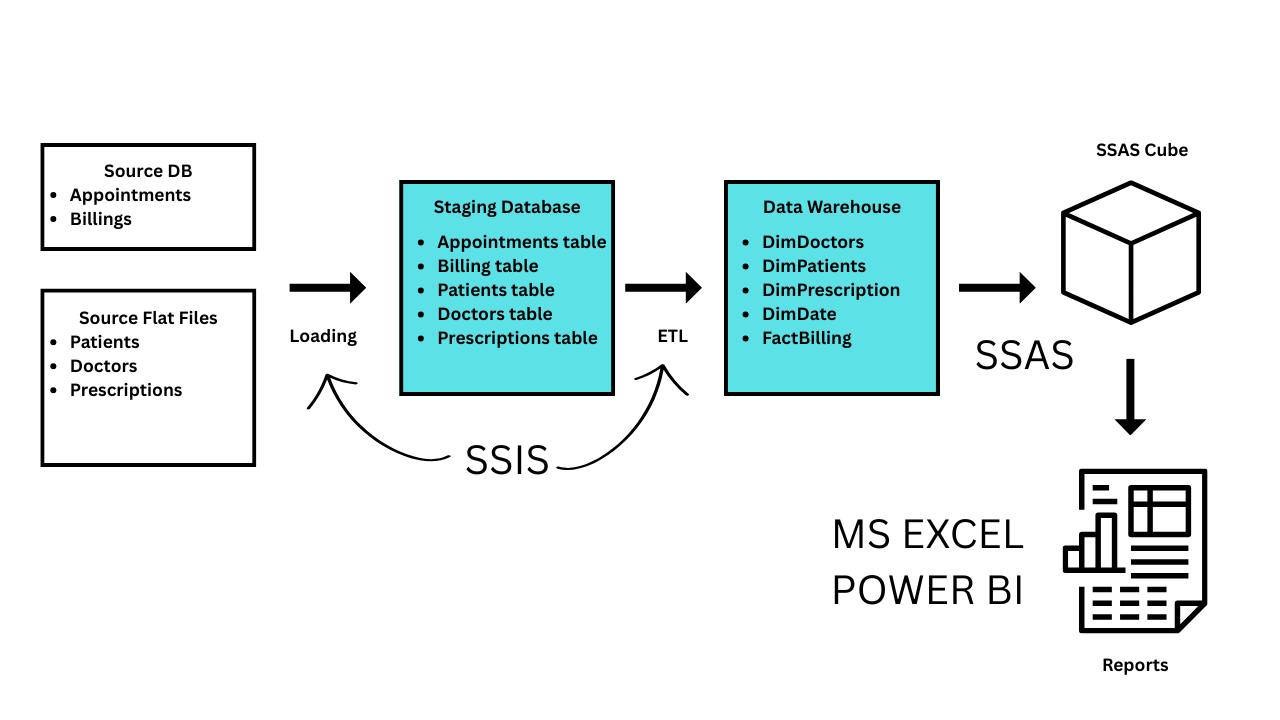
The patient’s details with sickness are recorded when he or she is registered. Then the system assigns a matching doctor related to that patient. Each detail connected with appointment like prescriptions given, patient and doctor details are recorded in the appointment entity. The billing entity records all the appointments with their billing amounts.

The data comes from two sources

* Flat files (CSVs): The Doctors, Patients and Prescriptions data come as CSV files. These are read by SSIS Flat File Source component
* SQL Server Database Tables: Appointments and Billing are preloaded and accessible through the Healthcare\_Source database. These are accessed by OLEDB source in SSIS

# Solution Architecture

The following image summarizes the architecture of the DW and BI solution



(Solution architecture)

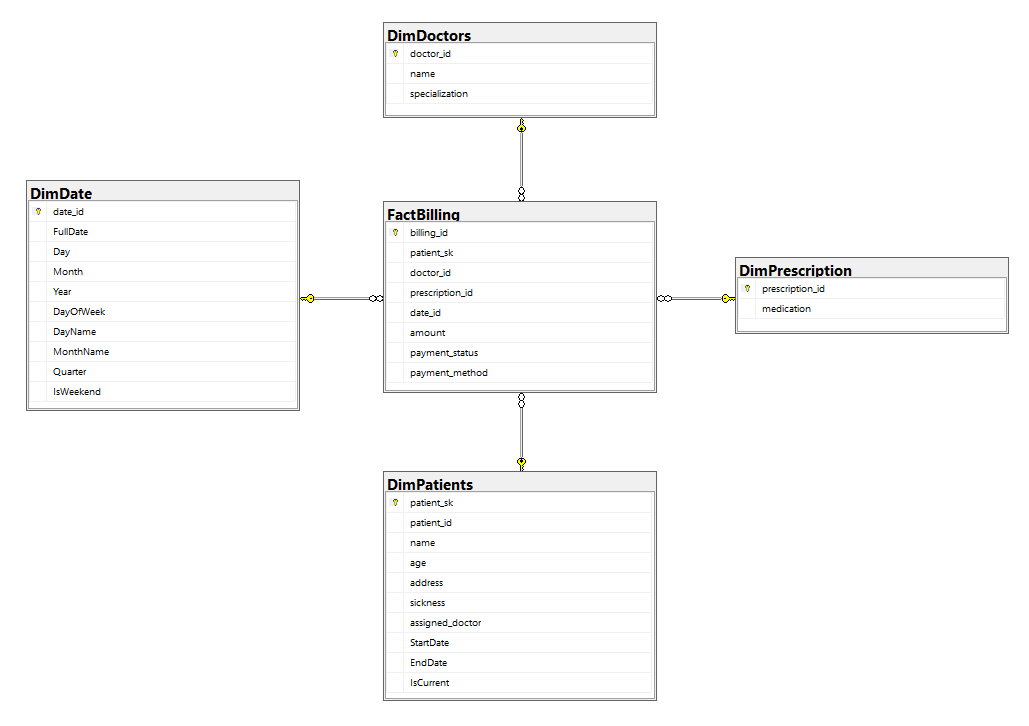
Here the data sources available in different formats as data tables in SQL Database and other exported CSV files. Then the data is loaded into the Staging Database where the raw data is cleansed and transformed before loading to the warehouse (Extract and Transform)

The warehouse layer consists of the clean data that can be used for decision making. It follows a star or snowflake scheme. Facilitates analytics.

A SSAS cube is created using data from warehouse DB which contain dimensions and measures. Reports layer is the user-friendly layer which finally outputs the findings in a readable manner using the cube created.

# Data warehouse design and development

First, I created a DB named HealthCare\_DW Microsoft SQL Server using SQL Server Management Studio. Then created the necessary fact and dimensions tables to proceed with the steps. The Image below shows how the fact and dimensions tables interact with each other.



(How fact and dimensions interact in a Star Scheme using Database diagrams)

This data warehouse is structured as a Star Schema. The below is the SQL queries I wrote to create these Fact and Dimensional tables. I selected the Billings as the fact tables because it has the measurable values which are defined by other tables.

*-- DimPatients*

CREATE TABLE *DimPatients* (

    patient\_sk INT IDENTITY(1,1) *PRIMARY KEY*,

    patient\_id INT,

    name VARCHAR(100),

    age INT,

    address VARCHAR(255),

    sickness VARCHAR(255),

    assigned\_doctor VARCHAR(100),

    StartDate DATE,

    EndDate DATE,

    IsCurrent BIT

);

DimPatients consists of patients’ details. I selected only patient\_id, name, age, address, sickness and assigned\_doctor as the forwarding attributes due irrelevancy of other attributes for important decision making. The SCD (slowly changing dimension) ~ address’s data will be handled by SSIS. It supports patient-based analysis.

*-- DimDoctors*

CREATE TABLE *DimDoctors* (

    doctor\_id INT *PRIMARY KEY*,

    name VARCHAR(100),

    specialization VARCHAR(100)

);

The DimDoctors table stores doctor\_id, name and specialization. This helps to doctor based analysis of the dataset.

*-- DimPrescription*

CREATE TABLE *DimPrescription* (

    prescription\_id INT *PRIMARY KEY*,

    medication VARCHAR(255)

);

The DimPrescription keeps track of the medication used in each patient appointment which will be recorded in billing table.

*-- DimDate*

WITH DateSequence AS (

    SELECT CAST('2020-01-01' AS DATE) AS DateValue

    UNION ALL

    SELECT DATEADD(DAY, 1, DateValue)

    FROM DateSequence

    WHERE DateValue < '2025-12-31'

)

INSERT INTO DimDate (date\_id, FullDate, Day, Month, Year, DayOfWeek, DayName, MonthName, Quarter, IsWeekend)

SELECT

    CONVERT(INT, FORMAT(DateValue, 'yyyyMMdd')) AS date\_id,

    DateValue AS FullDate,

    DAY(DateValue) AS Day,

    MONTH(DateValue) AS Month,

    YEAR(DateValue) AS Year,

    DATEPART(WEEKDAY, DateValue) AS DayOfWeek,

    DATENAME(WEEKDAY, DateValue) AS DayName,

    DATENAME(MONTH, DateValue) AS MonthName,

    DATEPART(QUARTER, DateValue) AS Quarter,

    CASE WHEN DATEPART(WEEKDAY, DateValue) IN (1, 7) THEN 1 ELSE 0 END AS IsWeekend

FROM DateSequence

OPTION (MAXRECURSION 32767);

CREATE TABLE *DimDate* (

    date\_id INT *PRIMARY KEY*,

    FullDate DATE,

    Day INT,

    Month INT,

    Year INT,

    DayOfWeek INT,

    DayName VARCHAR(10),

    MonthName VARCHAR(10),

    Quarter INT,

    IsWeekend BIT

);

DimDate table helps to categorize, filter the data on date basis

*-- FactBilling*

CREATE TABLE *FactBilling* (

    billing\_id INT *PRIMARY KEY*,

    patient\_sk INT,

    doctor\_id INT,

    prescription\_id INT,

    date\_id INT,

    amount DECIMAL(10, 2),

    payment\_status VARCHAR(50),

    payment\_method VARCHAR(50),

*FOREIGN KEY* (patient\_sk) *REFERENCES* DimPatients(patient\_sk),

*FOREIGN KEY* (doctor\_id) *REFERENCES* DimDoctors(doctor\_id),

*FOREIGN KEY* (prescription\_id) *REFERENCES* DimPrescription(prescription\_id),

*FOREIGN KEY* (date\_id) *REFERENCES* DimDate(date\_id)

);

The Fact table which has foreign keys from patient, doctor, prescription and date tables. It holds the quantifiable and measurable values such as amount.

# ETL development

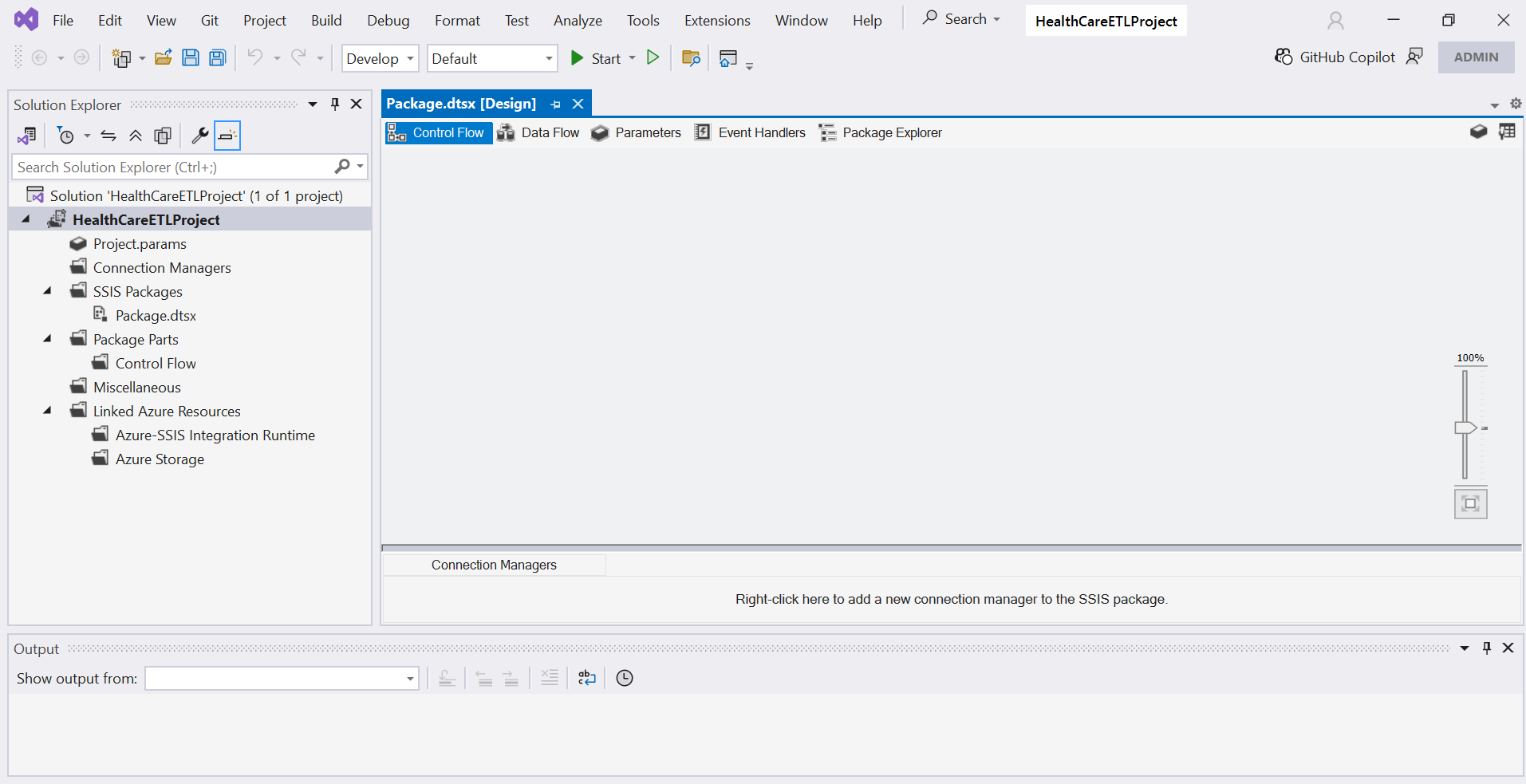
First as instructed in assignment I altered the FactBilling table with new columns.  
  
ALTER TABLE FactBilling

ADD accm\_txn\_create\_time DATETIME,

    accm\_txn\_complete\_time DATETIME,

    txn\_process\_time\_hours INT;

Created an Integration Services Project named HealthCareETLProject



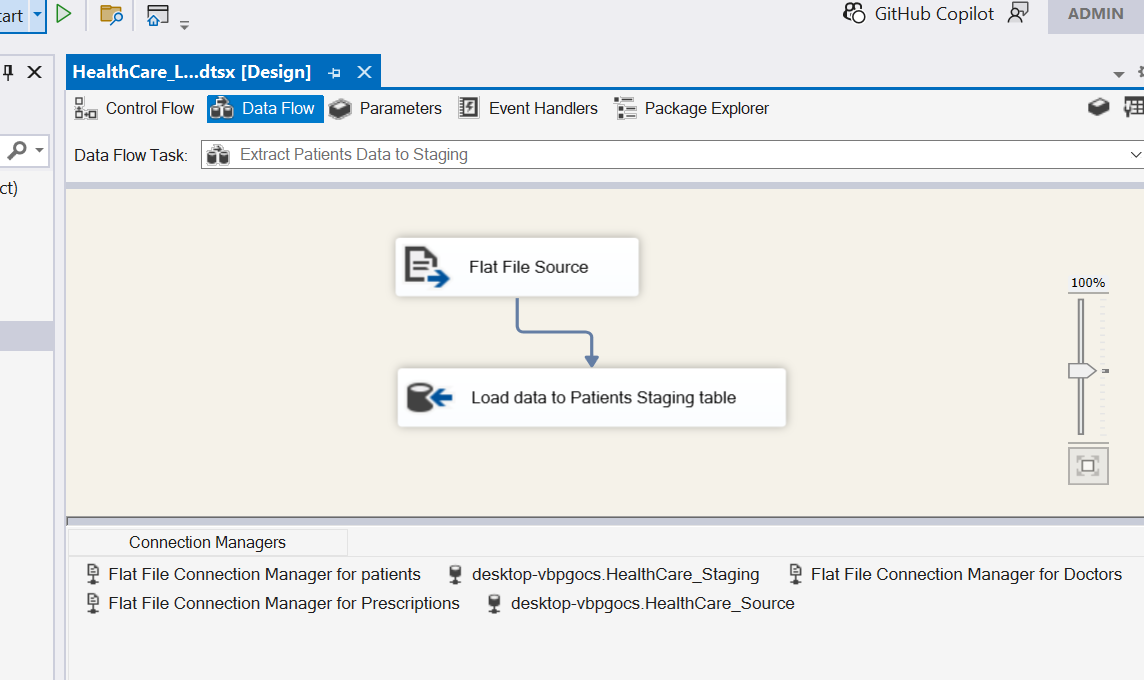
Data Extraction from sources

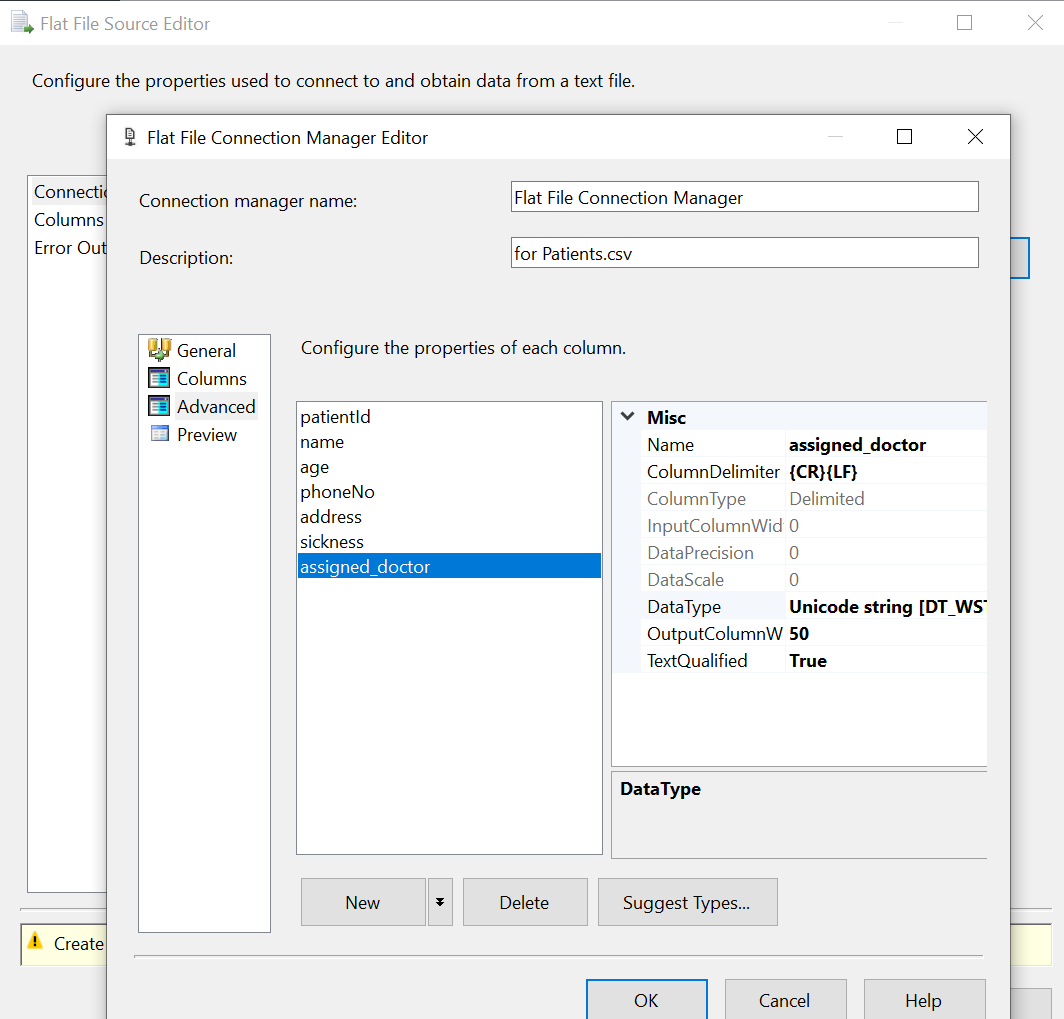
Considering the organization of data in this scenario, I followed the loading part as

* StgPatients
* StgDoctors
* StgAppointments
* StgPrescriptions
* StgBilling

The main reasons are the patients and doctors tables consist of information required to facilitate the appointment table and the prescription table require to reference the appointment table. Finally, the billing table which has referenced values from appointment table.

Then started the extraction process of data from patients.csv, doctors.csv flat files, added data flow tasks for these two and inside these configured the source and destinations as necessary. Used Flat File Source and OLE DB Destination SSIS tools.





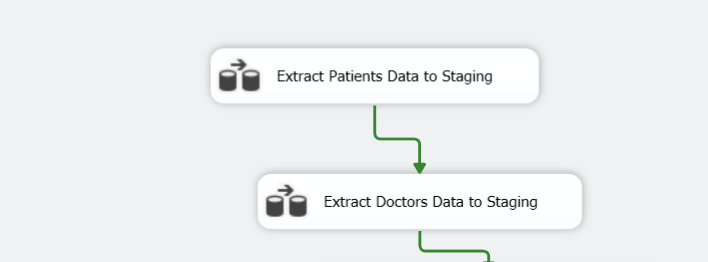
Created a connection manager which points to the CSV file in the known location and then changed data types as needed for strings and numeric values.

Also added a OnPreExecute event handler to truncate the table to avoid same data repeating.

A screenshot of a computer

AI-generated content may be incorrect.

For the CSV Flat Source File: Doctors same thing did with above tools along with OnPreExecute event handler.



Then carried out the process for appointments table which is an OLE DB source in HealthCare\_Source database. Used an OLE DB Source and OLE DB Destination from SSIS toolbox along with PreExecute event handler.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.  
(Configured the connection for the OLE DB Source)

Checked the Mappings section to confirm the columns

A screenshot of a computer

AI-generated content may be incorrect.  
  
A screenshot of a computer

AI-generated content may be incorrect.  
  
A screenshot of a computer

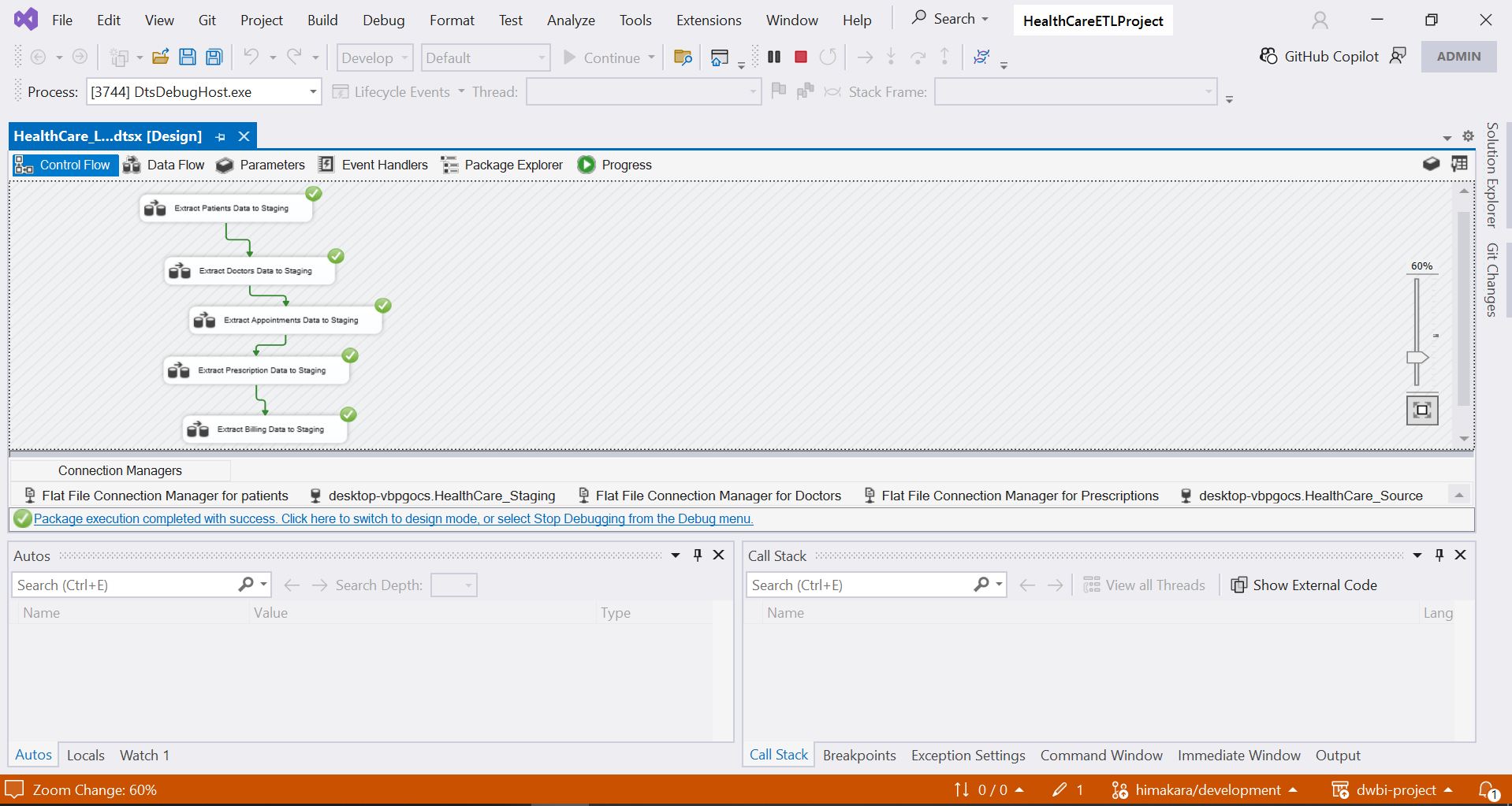
AI-generated content may be incorrect.

(The SQL Task to truncate before executing to avoid repetitions)  
  
Likewise, the same procedure was carried out for Prescription CSV as a Flat File Source and Billing Table as an OLE DB Source in HealthCare\_Source respectively.  
The Final Control Flow looked like this 👇  
  
A diagram of a medical procedure

AI-generated content may be incorrect.

(Control flow design for Source to Staging)

Then executed the HealthCare\_Load\_Staging package after building the solution



Then the required data from all the different sources could be observed in the Staging Database.

A Data Profiling part to understand what transformations should be performed on the data was carried out next to identify available data quality, hidden relationships and reduce data integration errors.

A screenshot of a computer

AI-generated content may be incorrect.

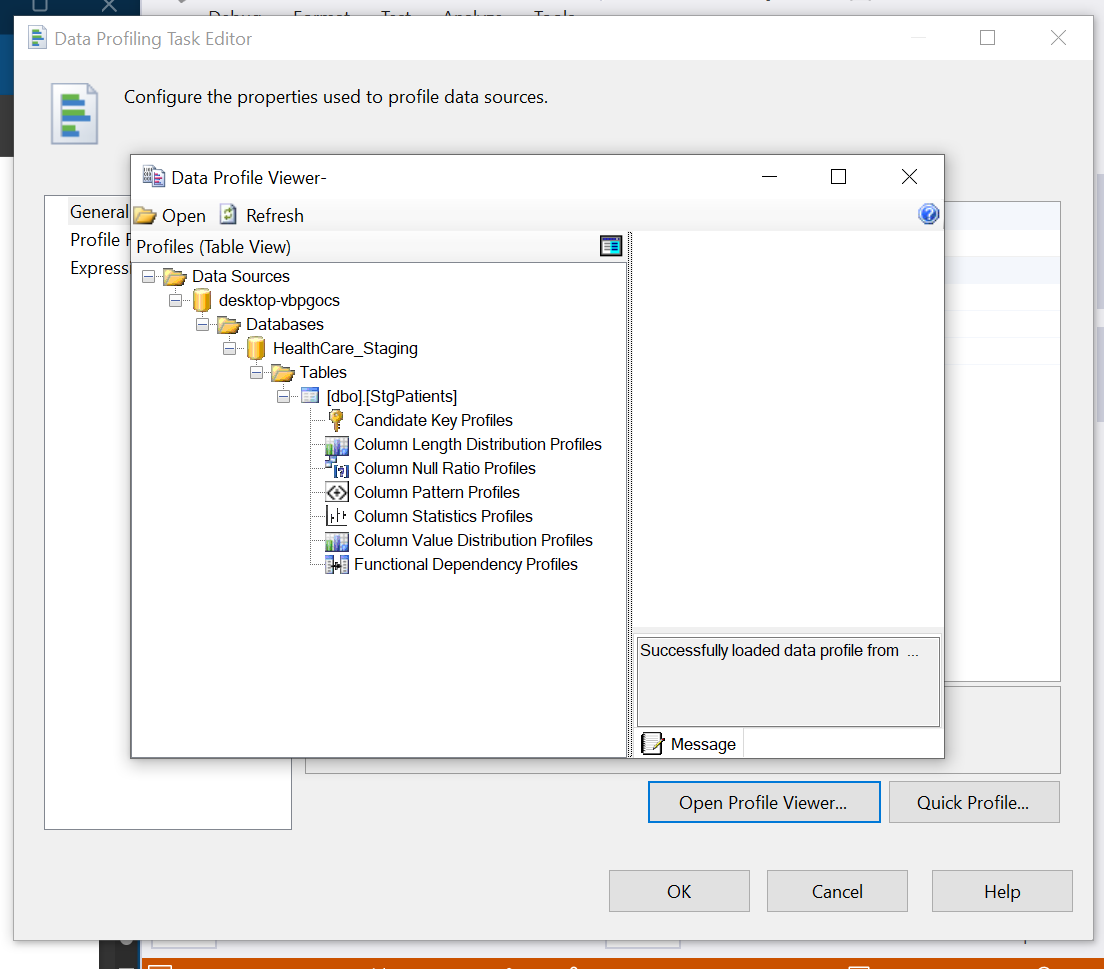
A screenshot of a computer

AI-generated content may be incorrect.

Here configured the usage file to “Create file” to create a new file and the location to be saved.

A screenshot of a computer

AI-generated content may be incorrect.



Here profiled every table to understand the transformations

A screenshot of a computer

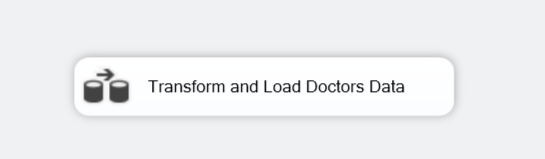
AI-generated content may be incorrect.

Then I could observe the exported files appeared in the given location

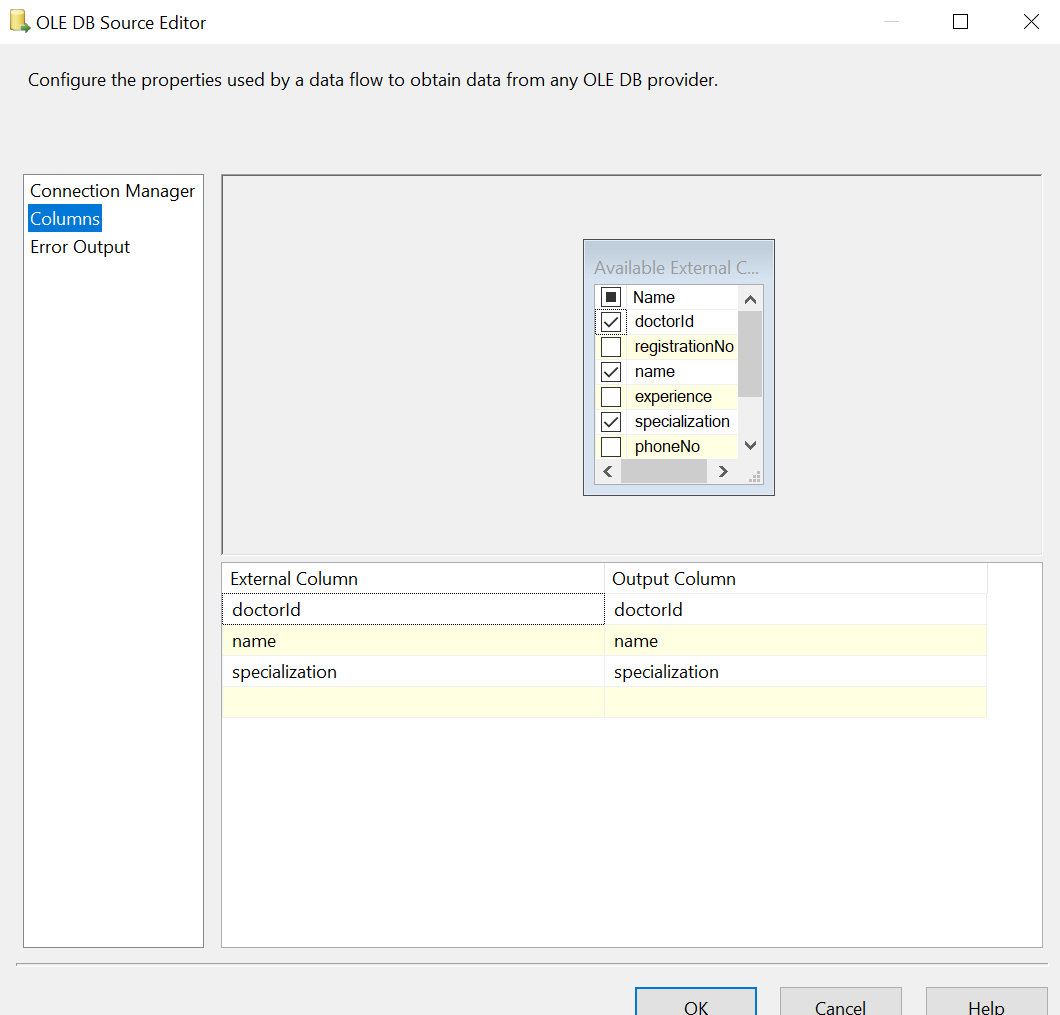
A screenshot of a computer screen

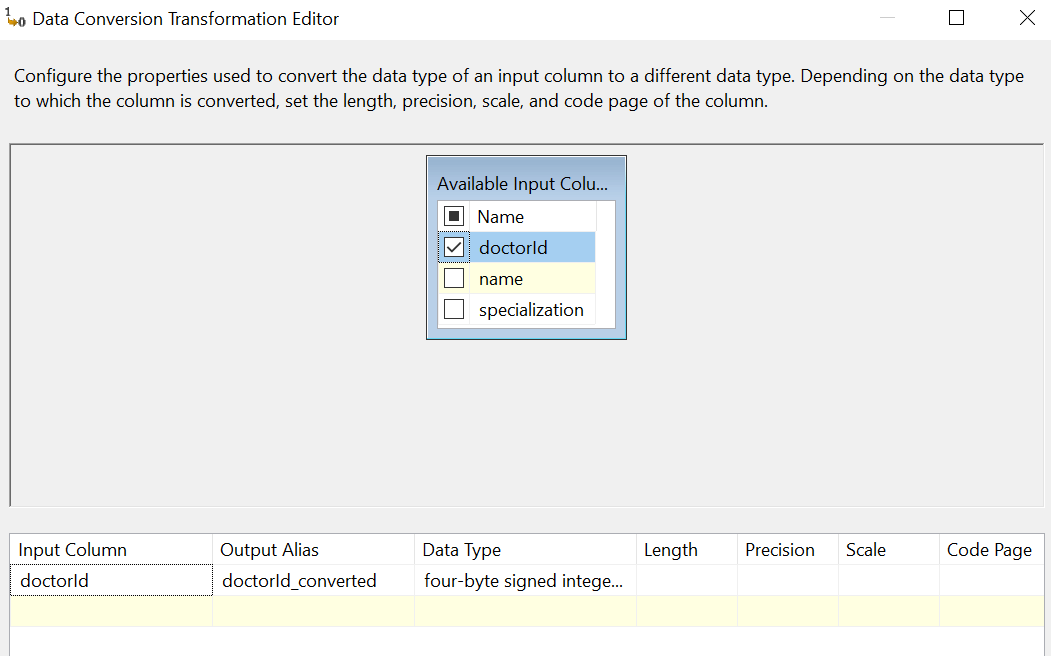
AI-generated content may be incorrect.

The SSIS Loading from Source to Staging has done. Then created new SSIS package named HealthCare\_Load\_DW to perform ETL tasks to populate data in HealthCare\_DW database which contains Fact and Dimension tables.  
  
Here my main idea is to Load data from Staging Database to Data warehouse.

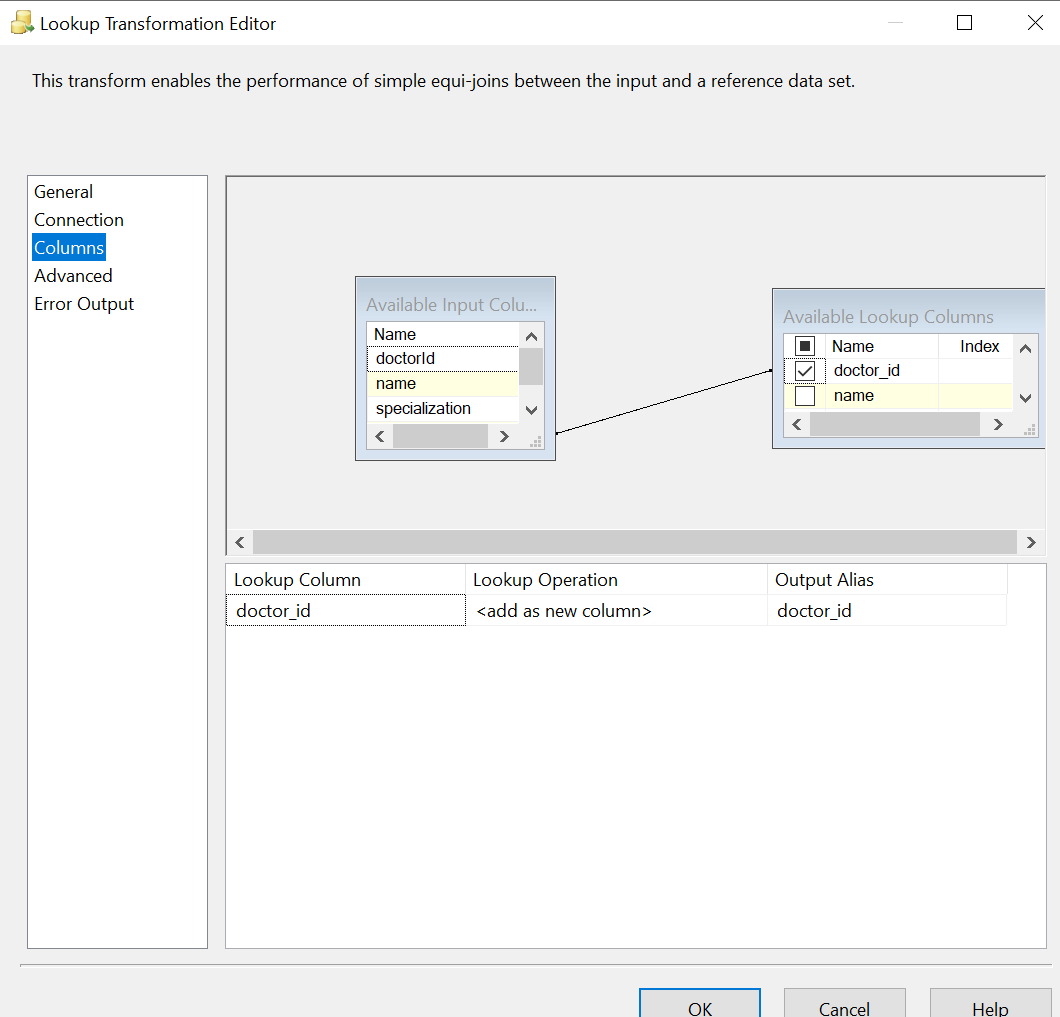
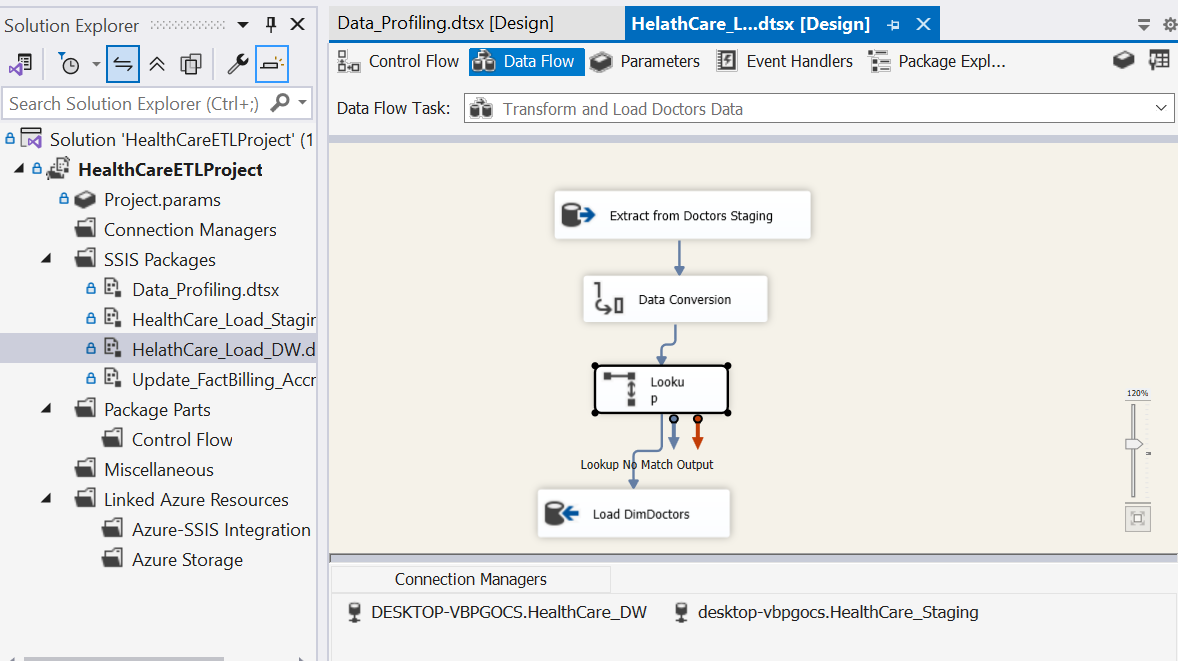
First up, Transform and Load from StgDoctors to DimDoctors, I created a Data Flow task  
  
  
  
Inside it I created OLEDB Source which points the table StgDoctors in Staging Database and OLEDB Destination which points table in Warehouse Database  
  
A screenshot of a computer screen

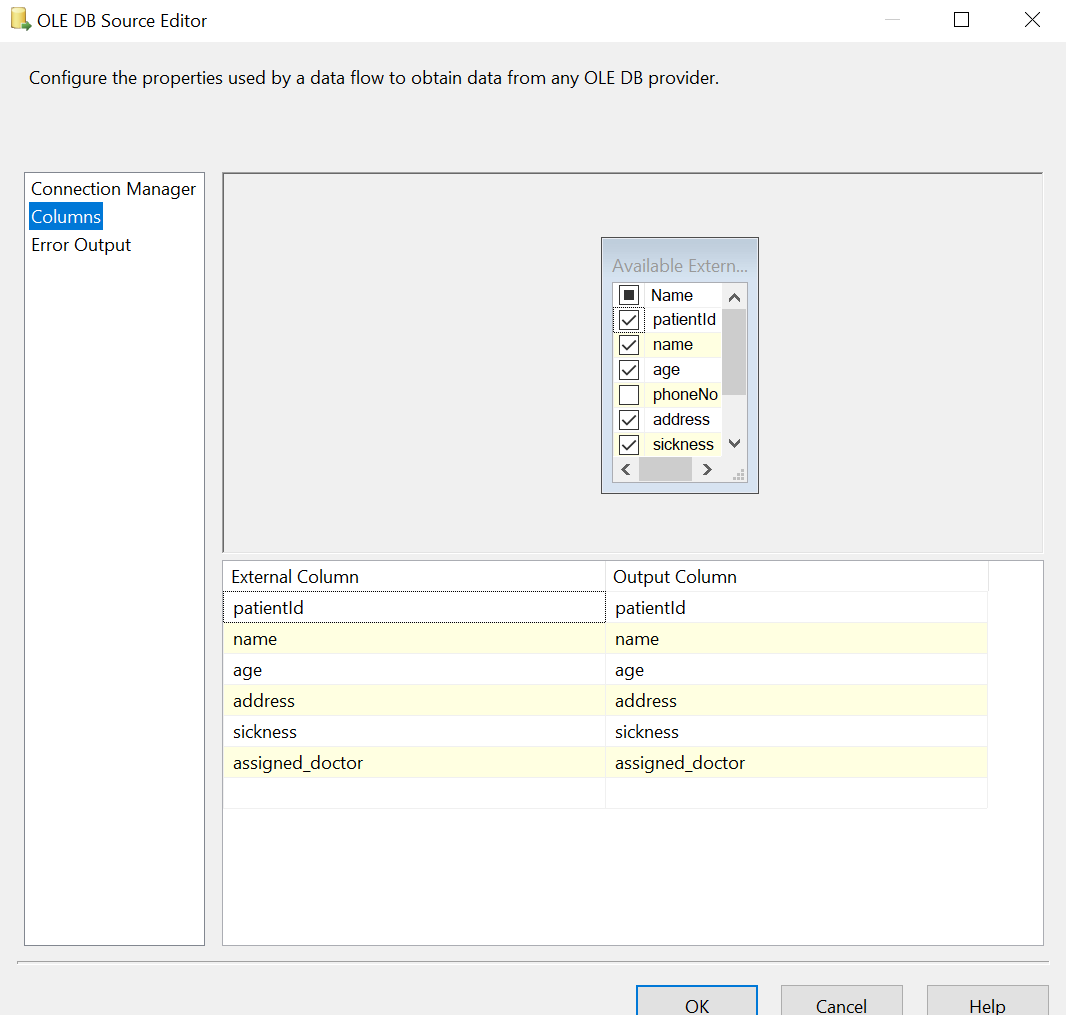
AI-generated content may be incorrect.

Inside OLE DB Source, added the connection, and selected most important attributes for the future analytical tasks  
  
  
Then added a Data conversion tool to convert doctorId to four-byte-signed-integer so there will not be mismatch between type in StgDoctors and DimDoctors.

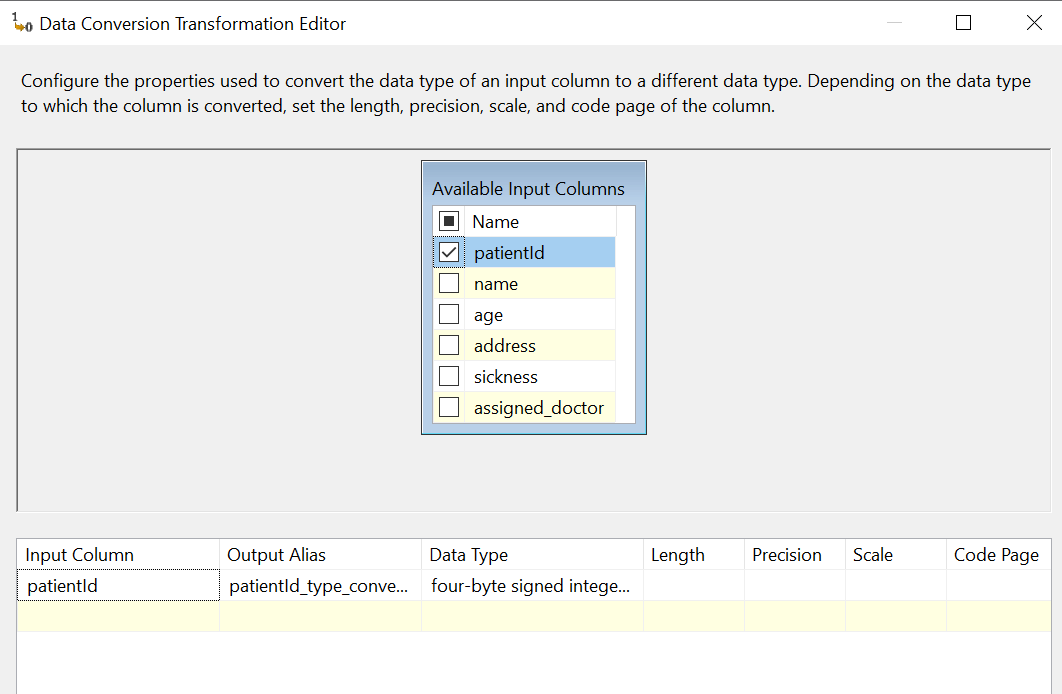


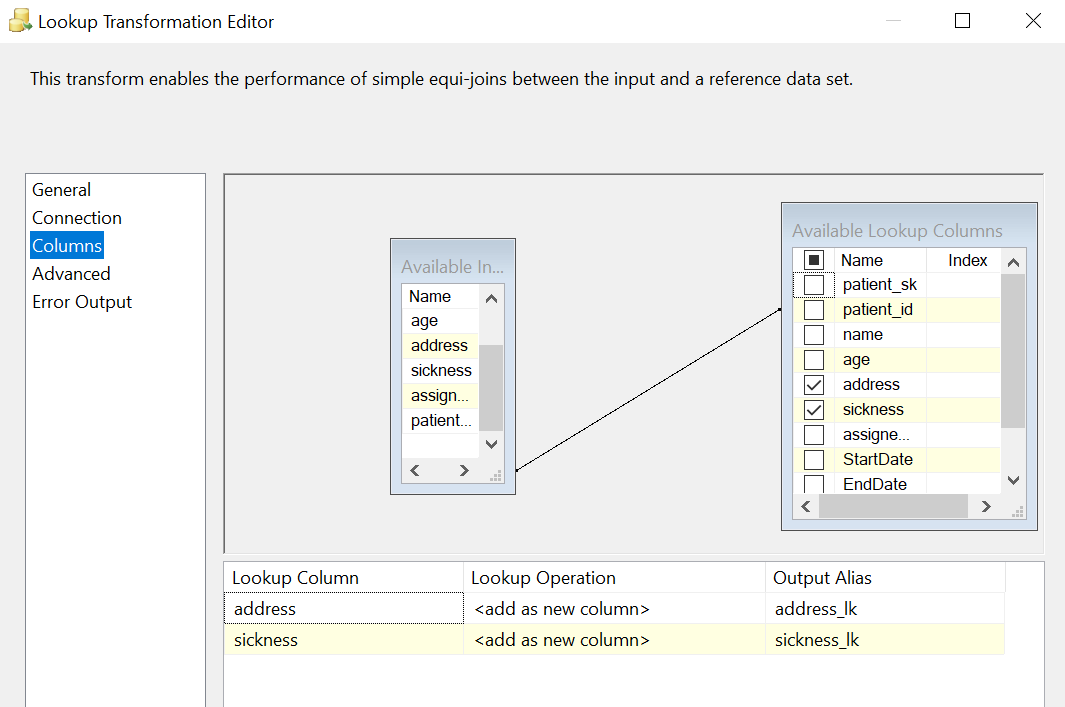
Then a Lookup tool to check any records already exist in DimDoctors.

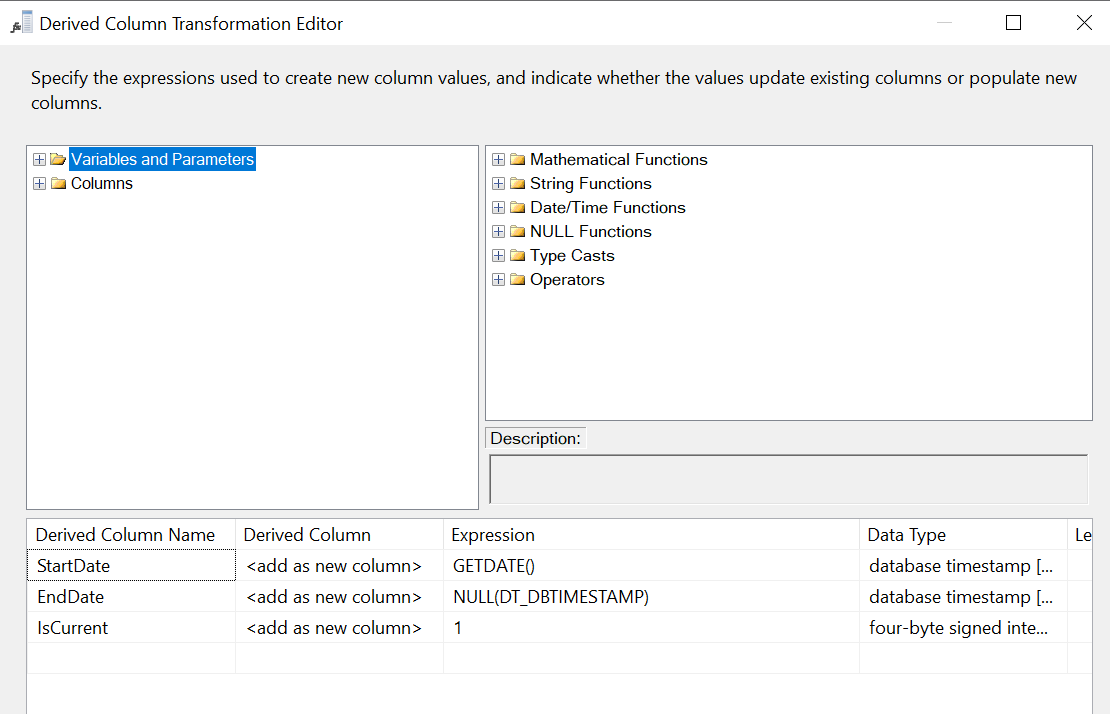
  
  
If no match was found then I loaded the data to the OLE DB Destination, DimDoctors. Here Lookup was used to avoid repetitions or conflicts when inserted. The final Data flow task looked like this 👇  
  
  
  
Then I performed ETL tasks on StgPatients. First I added an OLE DB connection and connected to the StgPatients in Stage database.

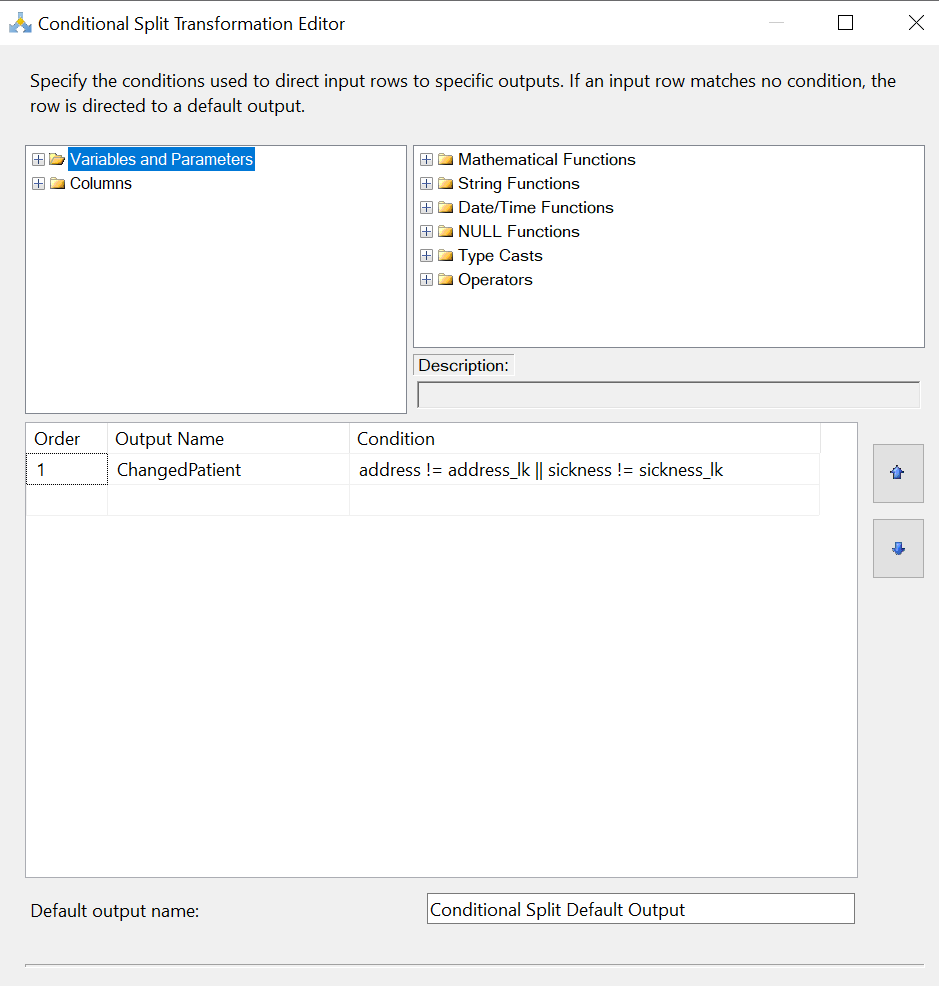
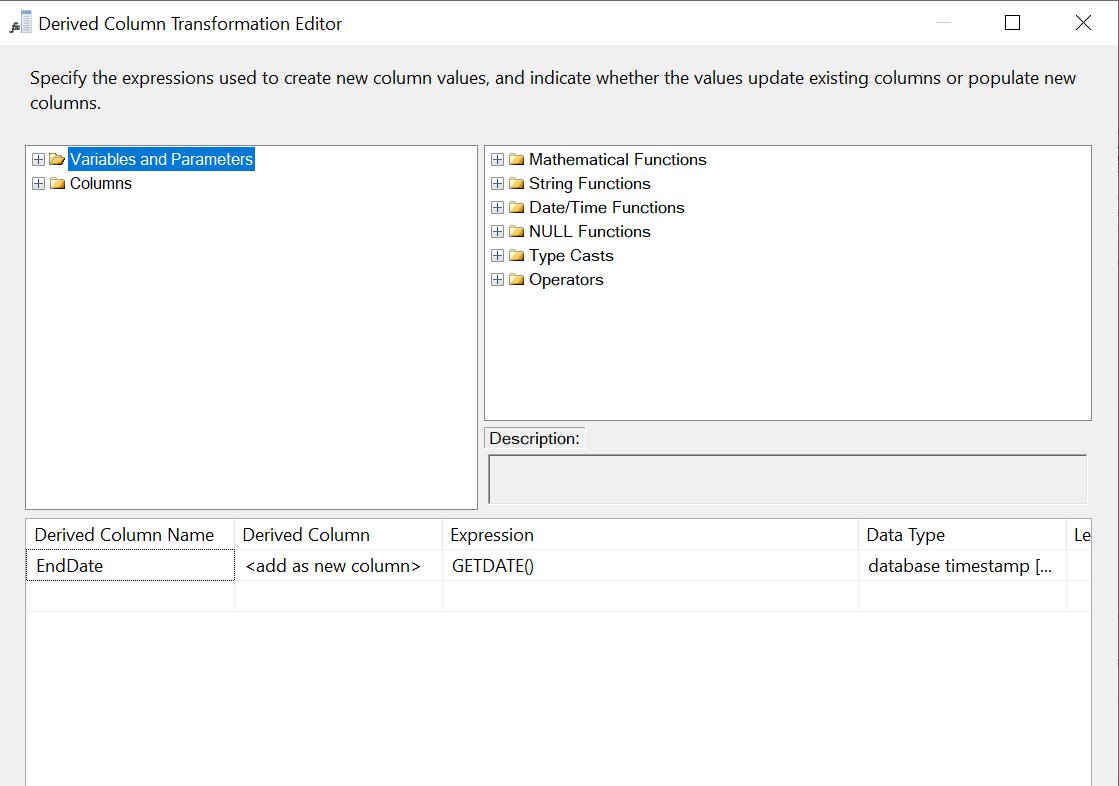
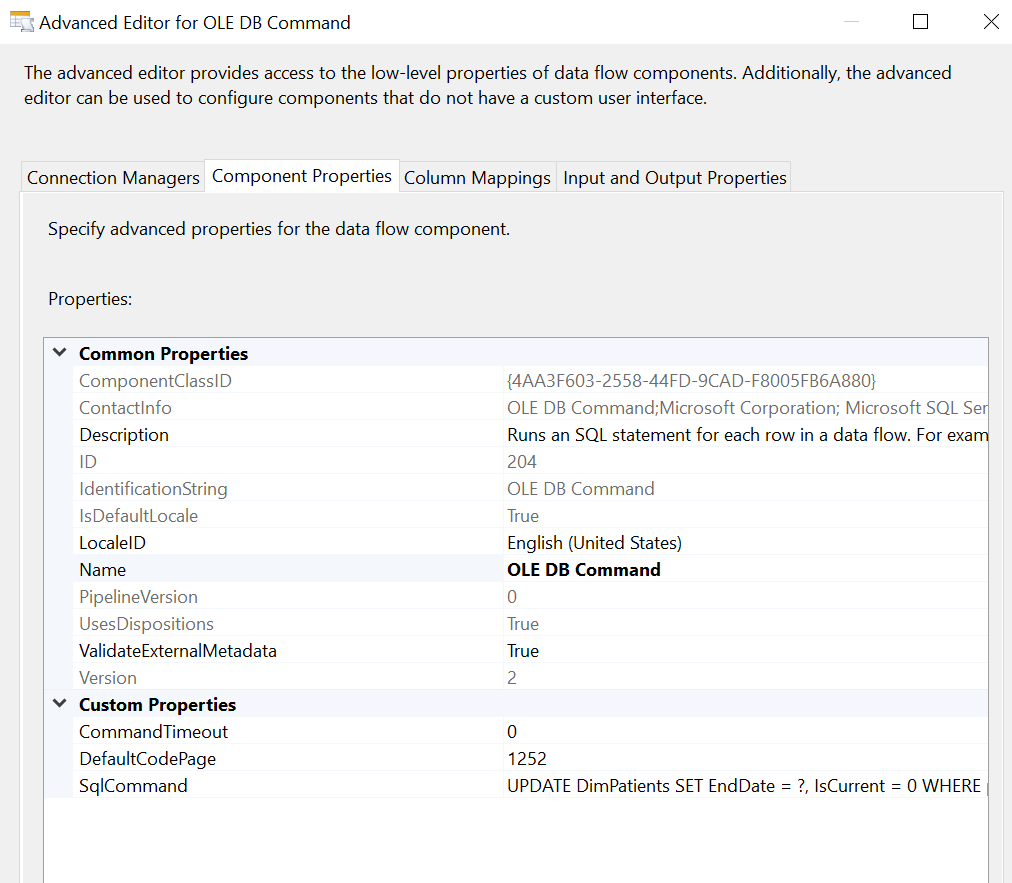
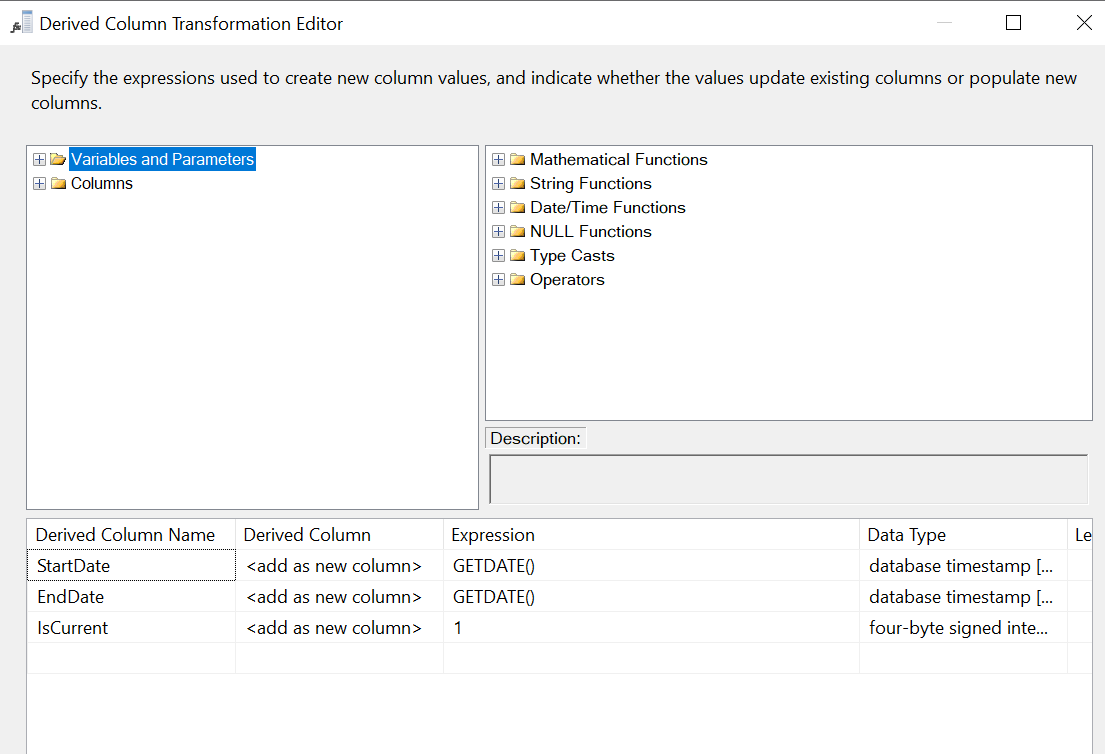
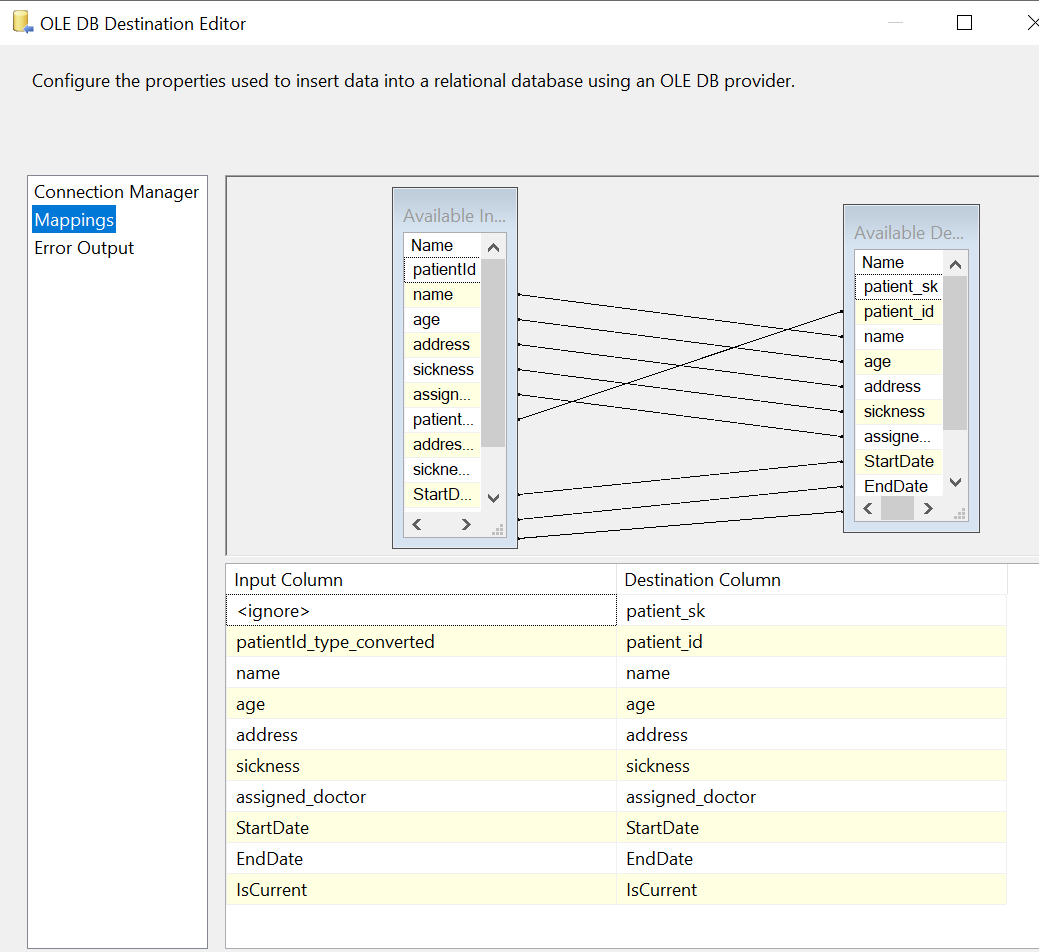
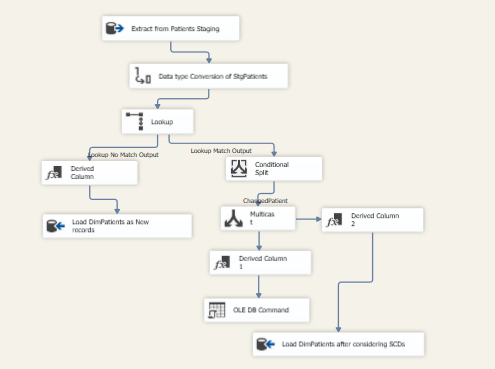
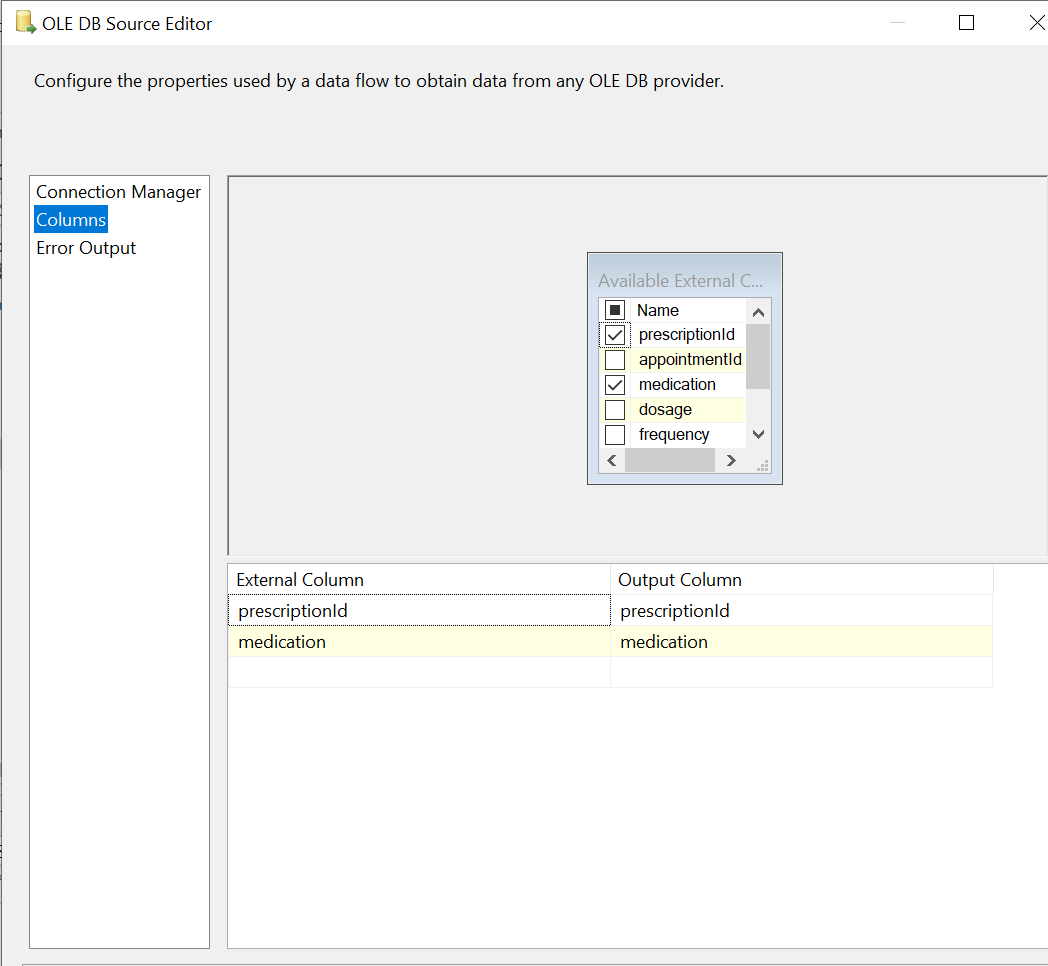
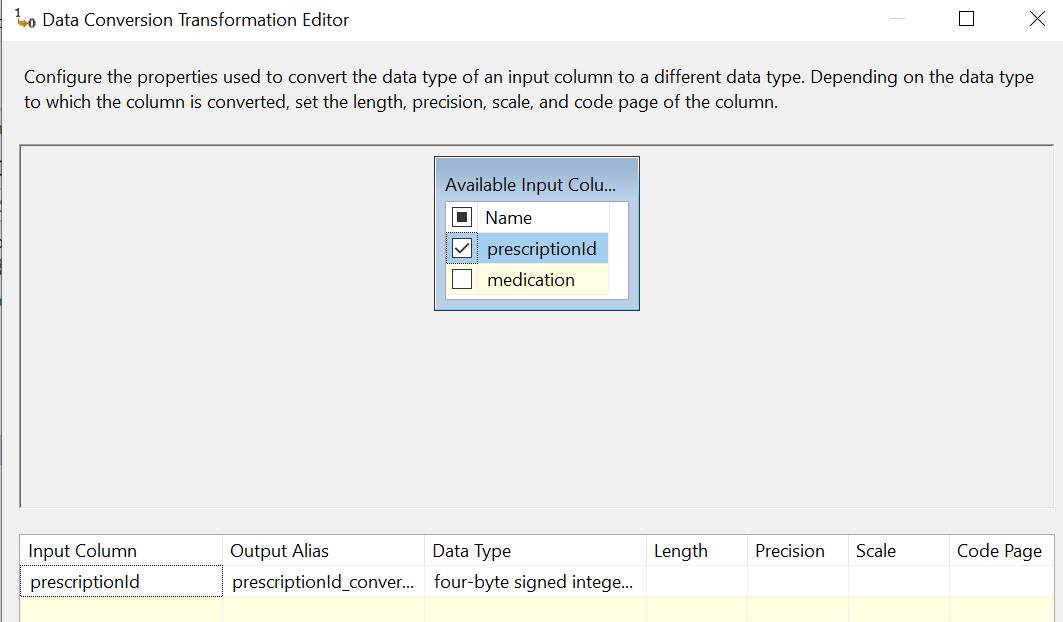
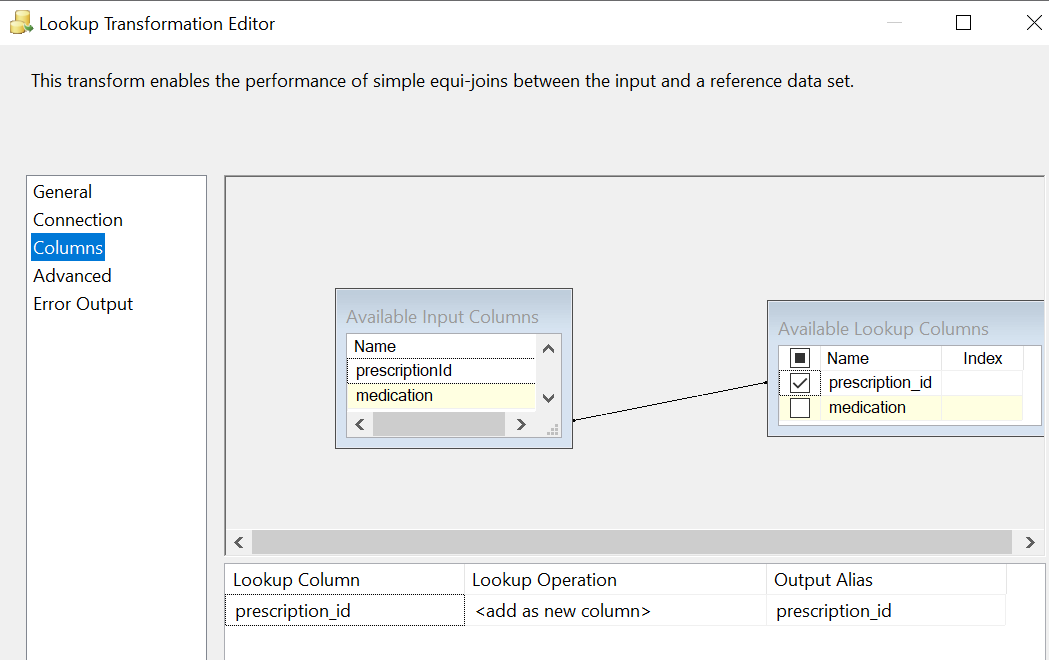
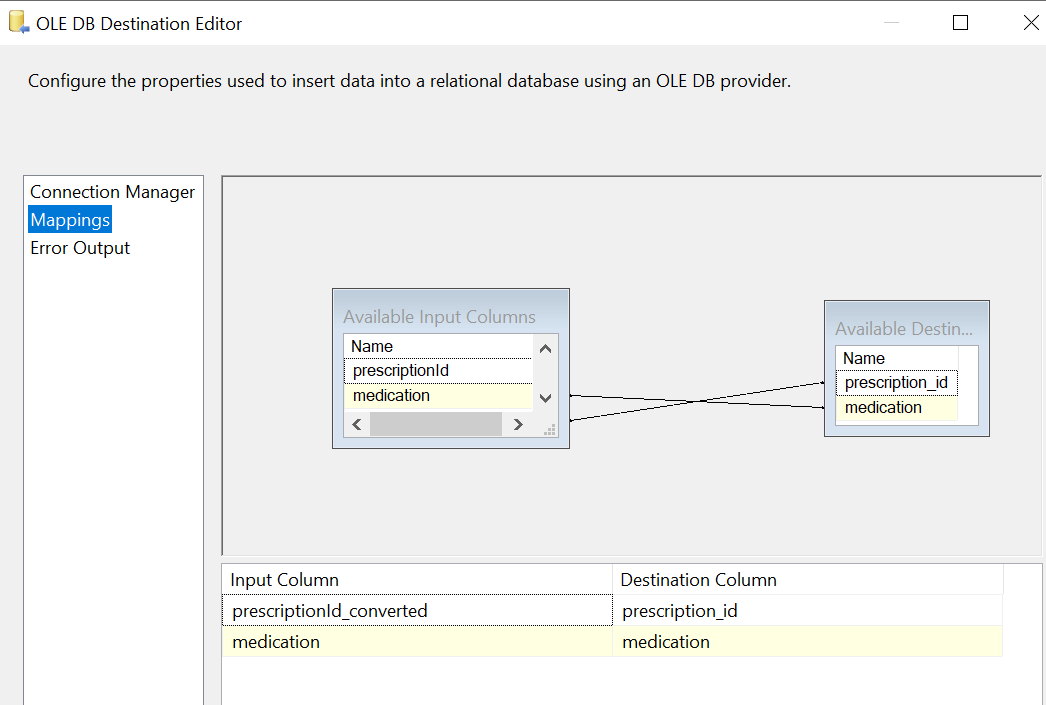
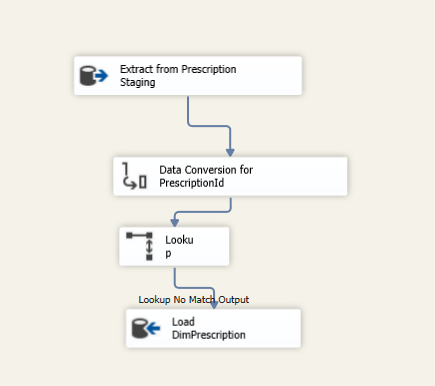
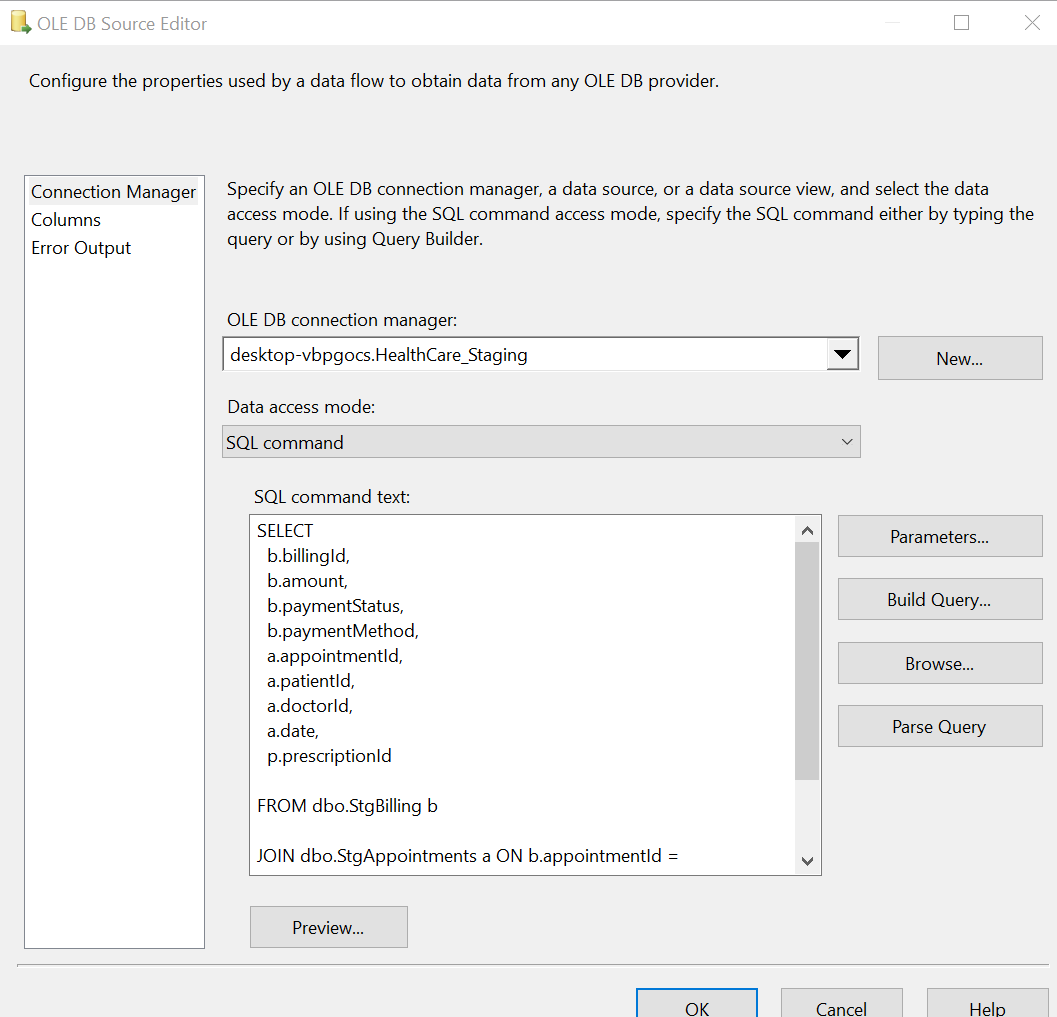
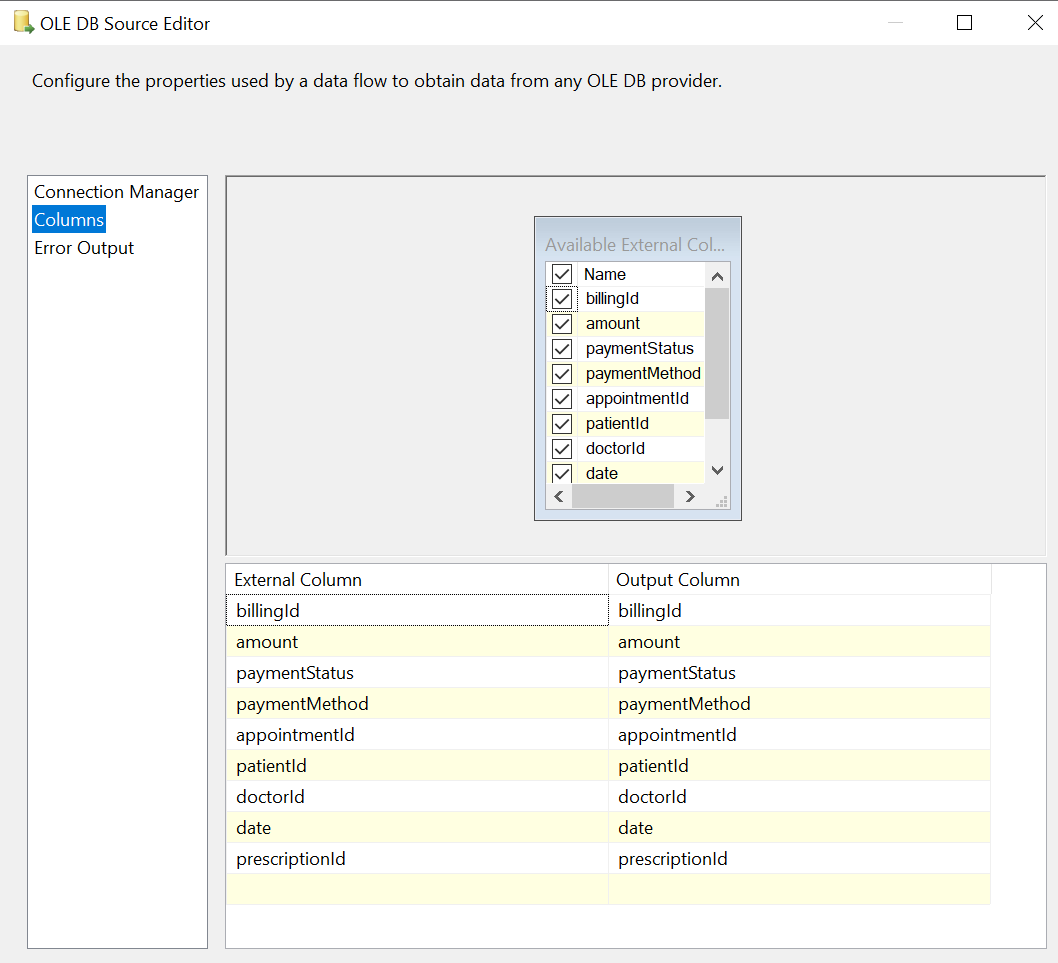


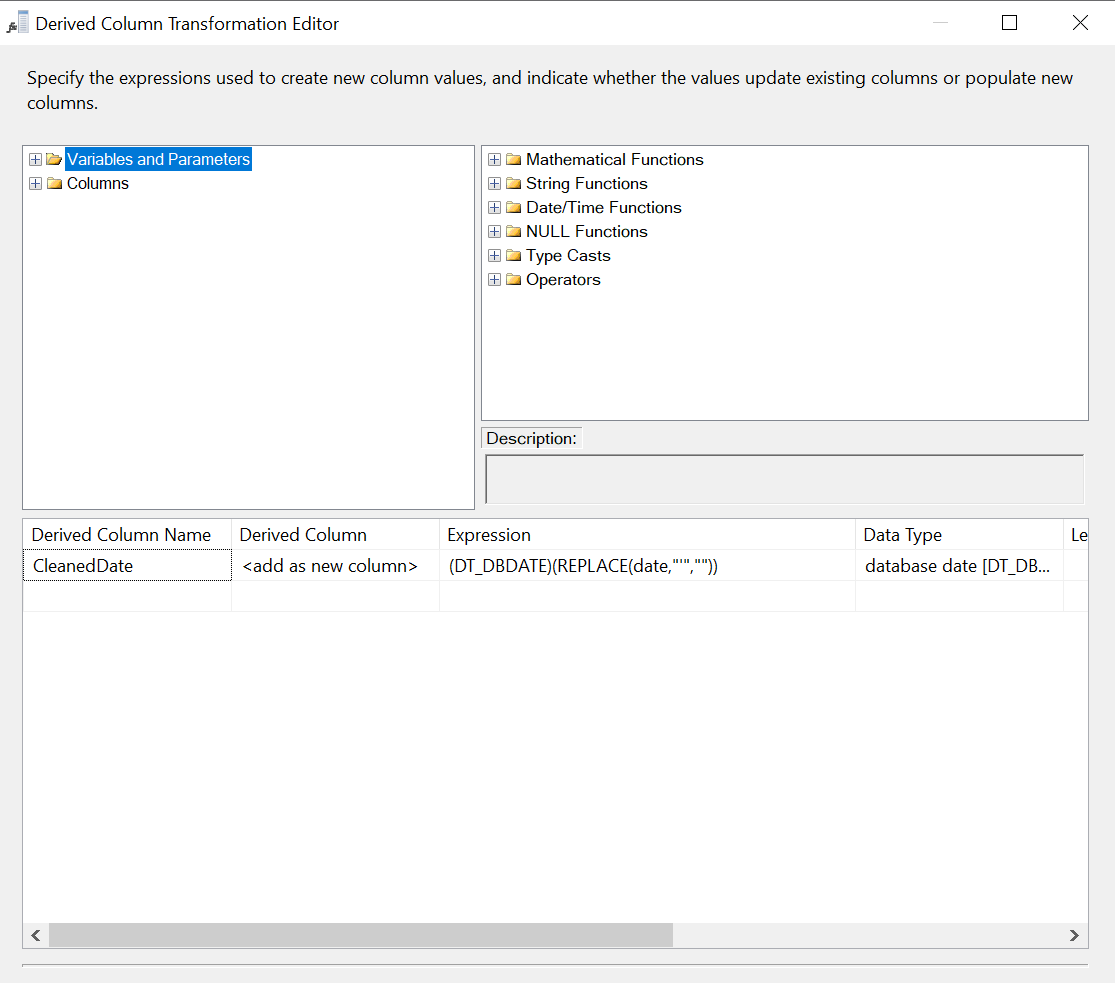
Here is also a Data conversion tool needed for the patientId

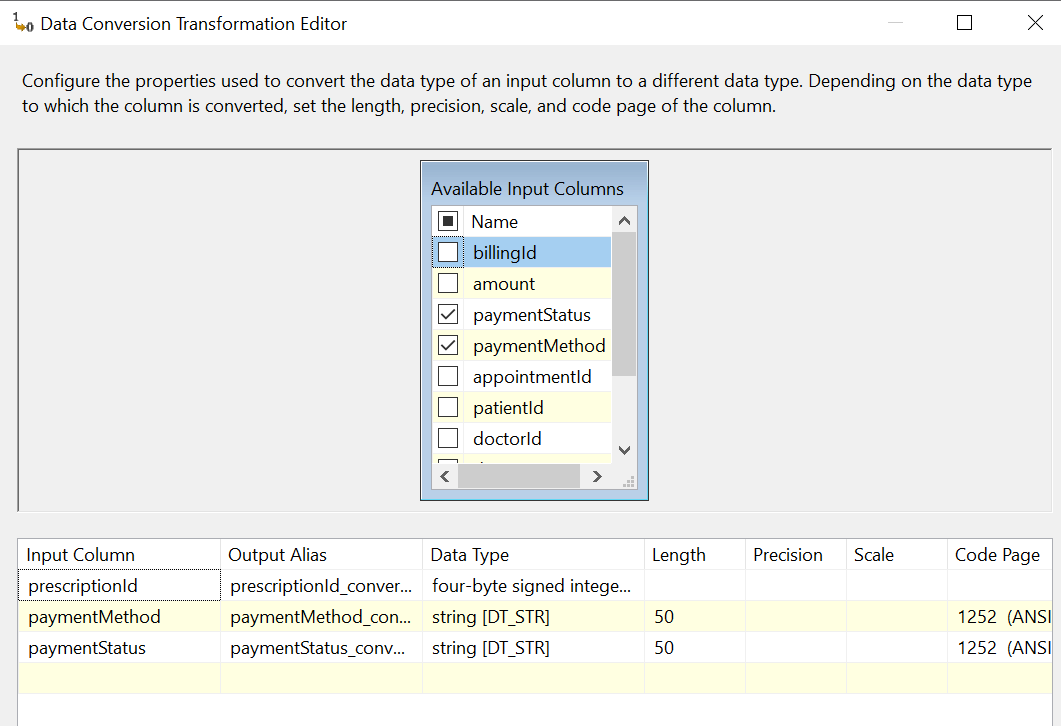
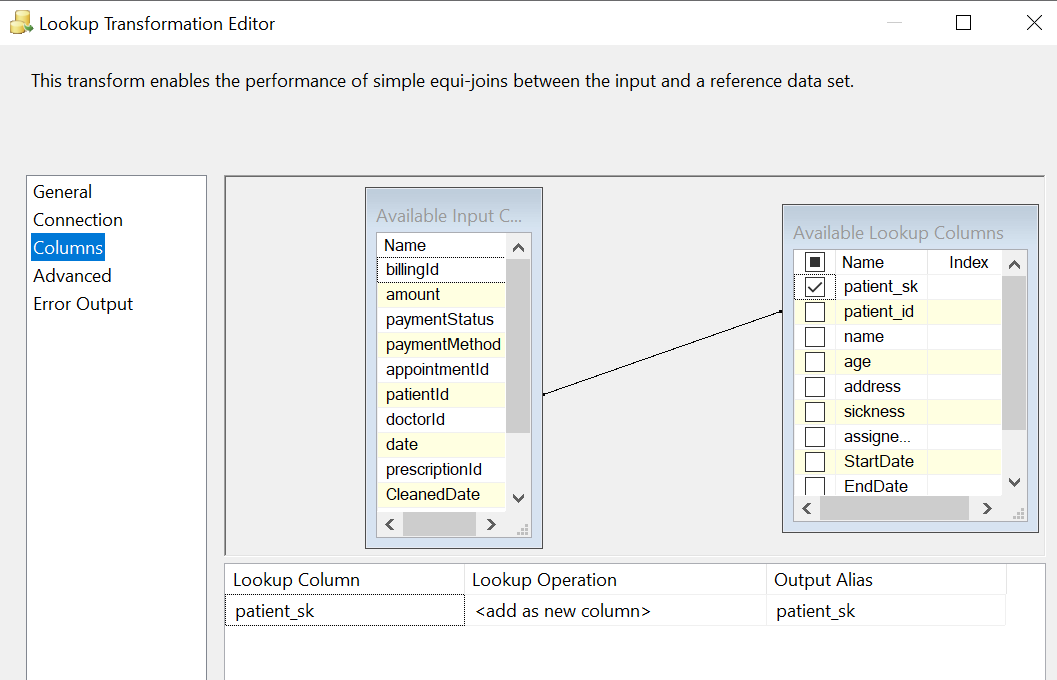
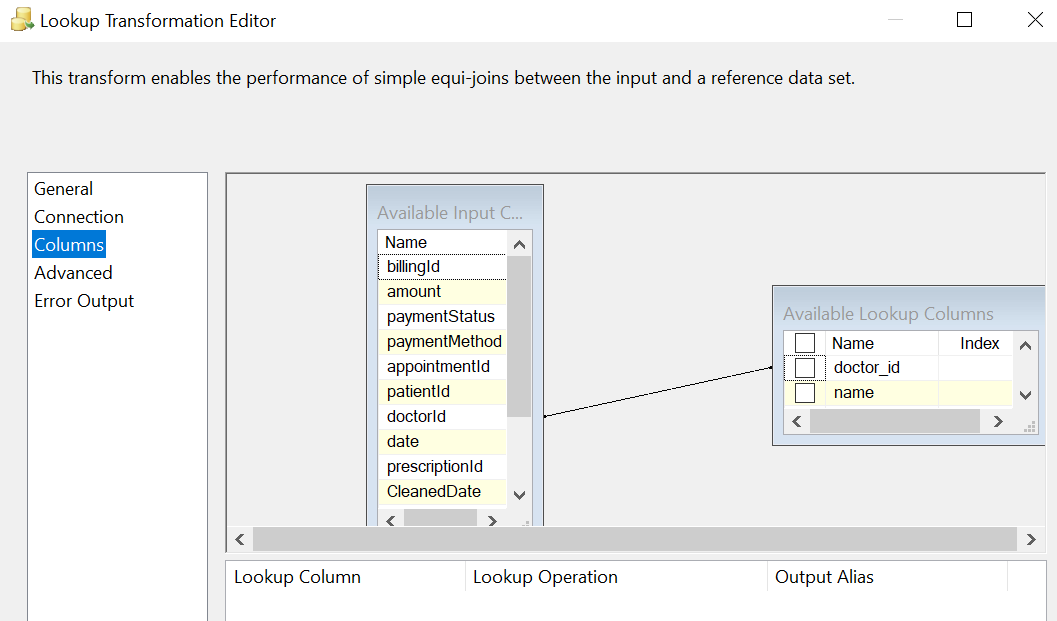
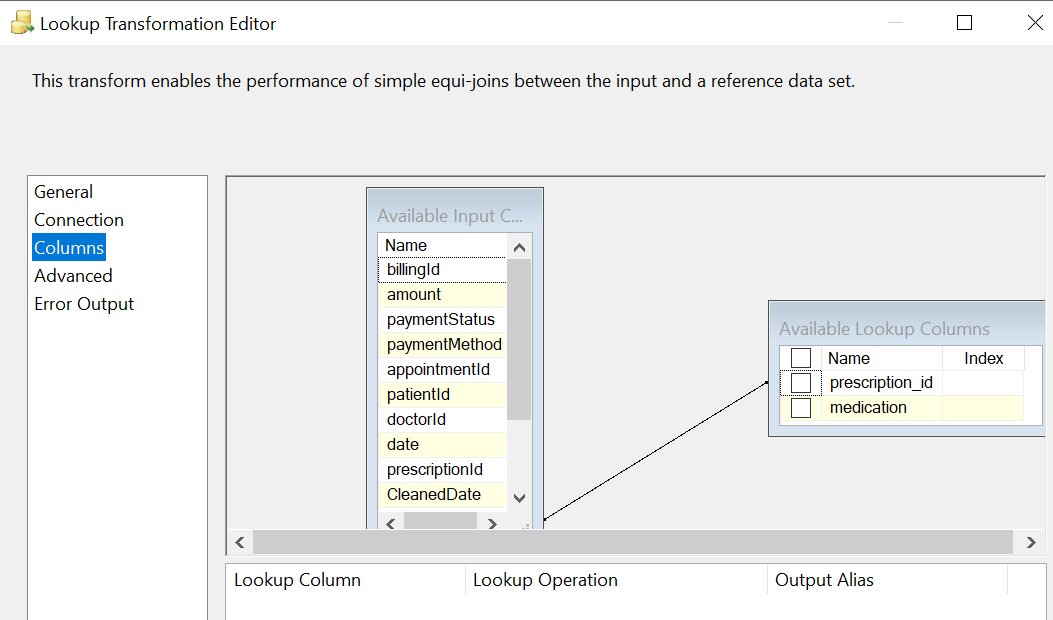
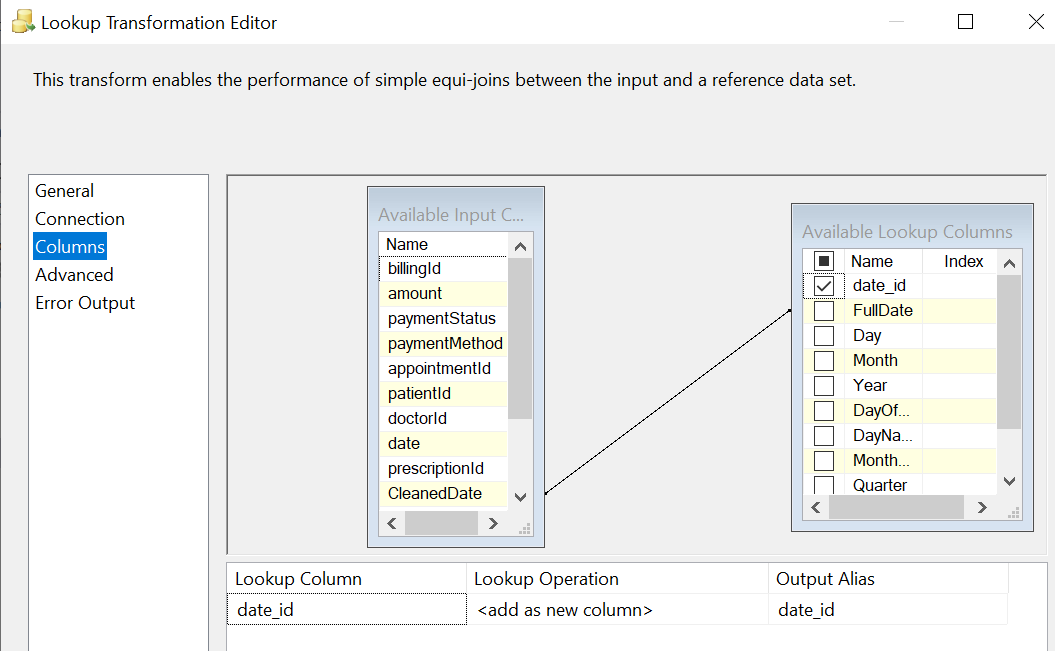
  
  
Since I’m going to prove Slowly Changing Variable operations (SCD) here, I added a Lookup tool to search for any matching records in DimPatients through patientId\_converted. Then returned values of address and sickness from that.

  
  
Here I concluded the logic that patient’s address or sickness can be changed over time, so there should be a proper way to identify their latest attribute values.  
  
From lookup, the no match output found directed to a Derived Column tool where I treat these records as fresh and add the SCD related columns to it.

  
  
Here StartDate set to take the current date via GETDATE() and EndDate set to NULL cause and due to this is the latest IsCurrent set to 1(Latest).

Then connected this to an OLE DB destination pointing DimPatients where we Extract, Transform and Load our new patient details.  
  
  
The secondary path of Lookup tool, A match found was directed to a Conditional Split tool where I checked whether the address or sickness has changed of the patient  
  
  
  
Then connected it to a multicast to perform necessary operations. First was to update the current record as an old one. Here I added a derived column tool to take the EndDate as GETDATE() to update the record.  
  
  
  
using that EndDate, I updated the current record as an old one (IsActive = 0). Used an OLE DB command tool for that.  
  
  
  
Used this SQL line in there  
  
*UPDATE DimPatients SET EndDate = ?, IsCurrent = 0 WHERE patient\_id = ? AND IsCurrent =1*  
  
Then the other part from Multicast component was sent to a derived column where we take new StartTime values for SCD and add a new updated record of the patient using OLE DB Destination.  
  
  
  
  
  
The Final data flow for DimPatients looked like this 👇  
  
  
Then I started working on DimPrescription. Here just like DimDoctors, only selected the columns which matters most.  
  
Used and OLE DB Source and a Destination along with a Data Conversion to convert prescriptionId and a Lookup tool to check record already exists.  
  
  
  
  
  
  
  
  
  
The Final Data Flow looked like this 👇  
  
  
  
Then I started working on Loading FactBilling table with the relevant data.  
  
Here first I added a OLE DB Source to JOIN Appointments table with Prescriptions table to feed the relevant data to the FactBilling.   
  
  
  
  
  
I had to convert the Date to a Database type, due to that added a Derived Column next.



Then added a Data Conversion to cast the prescriptionId, Payment status, Payment method to Integer and String types respectively  
  
  
  
Then added a Lookup for DimPatients to check the Appointment related Patient exists in there. And returned the surrogate key.  
  
  
  
Then added a Lookup for DimPatients to check whether the Doctor exists  
  
  
  
A Lookup for DimPrescription  
  
  
  
Then a Lookup in DimDate and return the date\_id which identifies the records in DimDate by matching CleanedDate with FullDate in DimDate.  
  
  
  
All the match found of these Lookup were connected if any not match found record found, It was neglected.  
  
Then as of Assignment guidelines added a Derived Column component to declare   
**accm\_txn\_create\_time, accm\_txn\_complete\_time, txn\_process\_time\_hours**

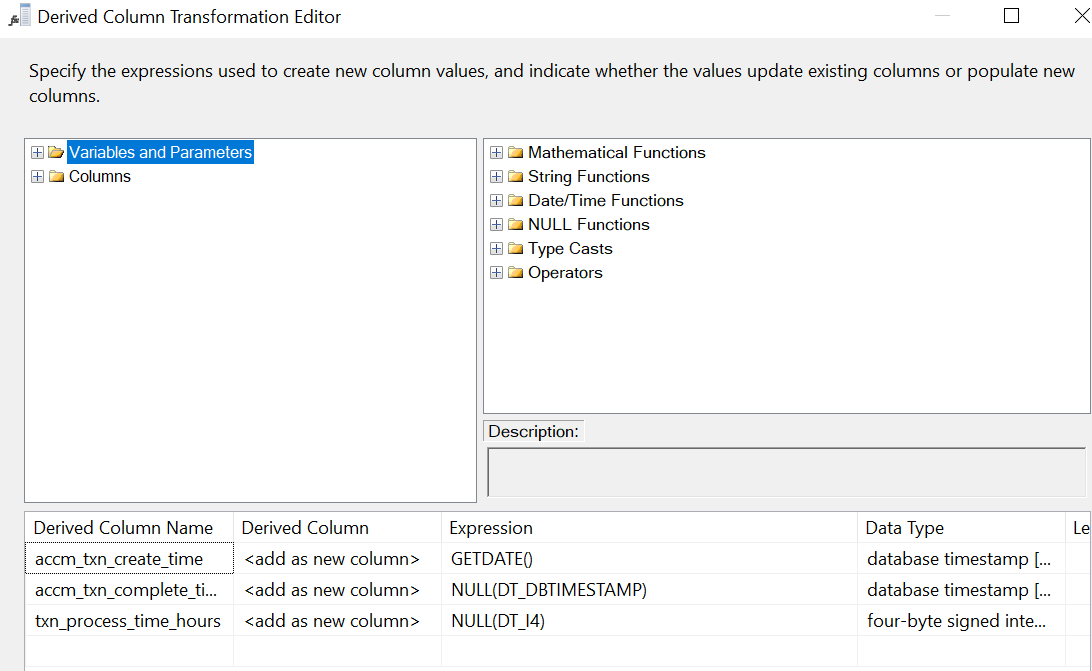
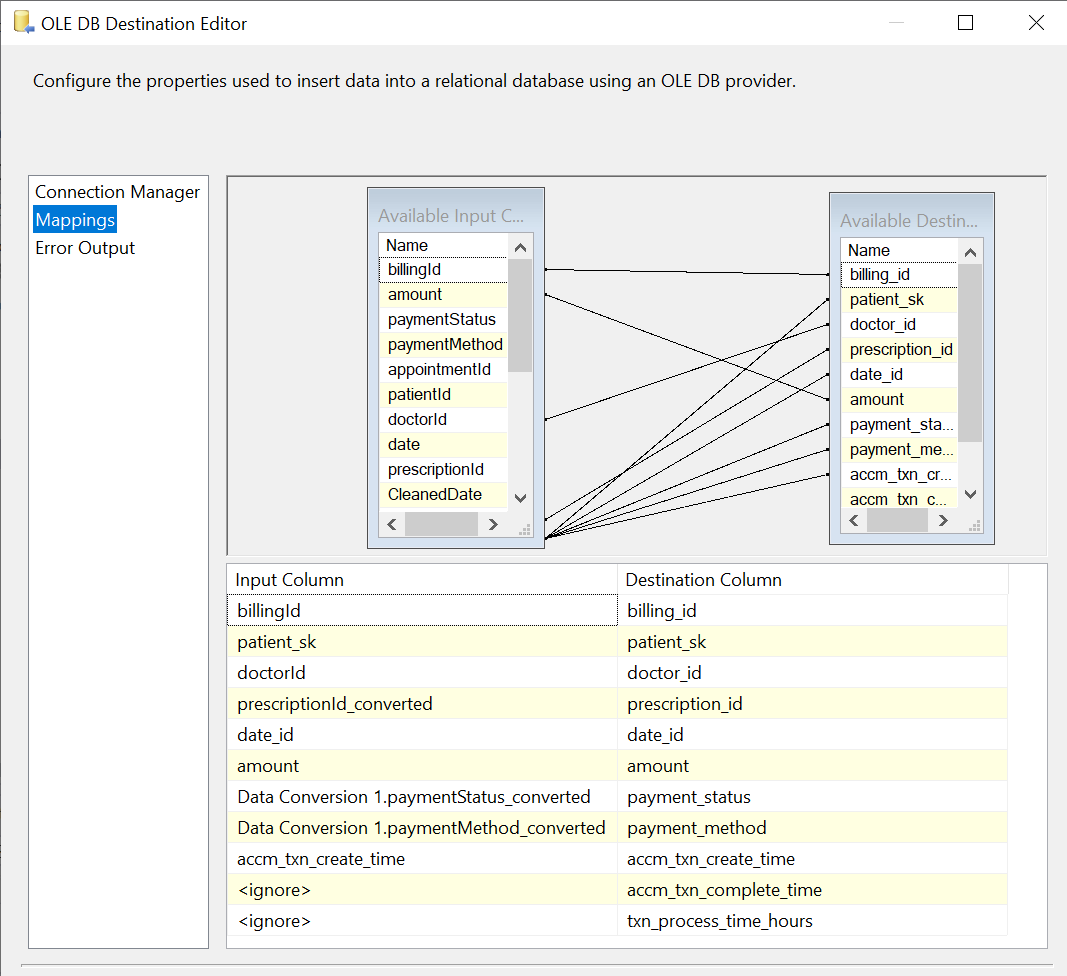
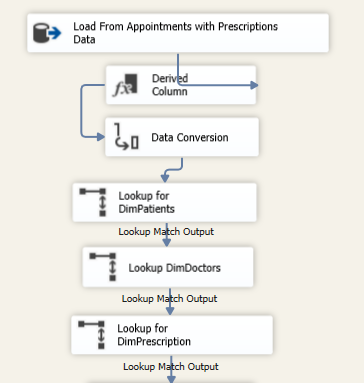
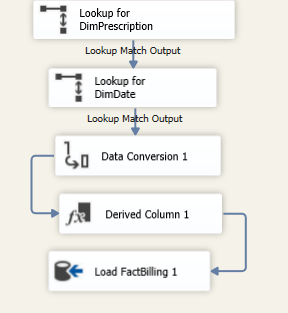
Here modified the FactBilling table as required in the assignment with the below SQL code  
  
*ALTER TABLE dbo.FactBilling*

*ADD*

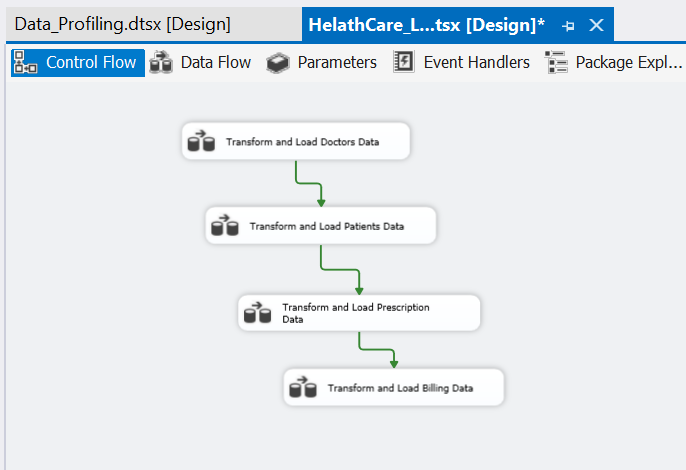
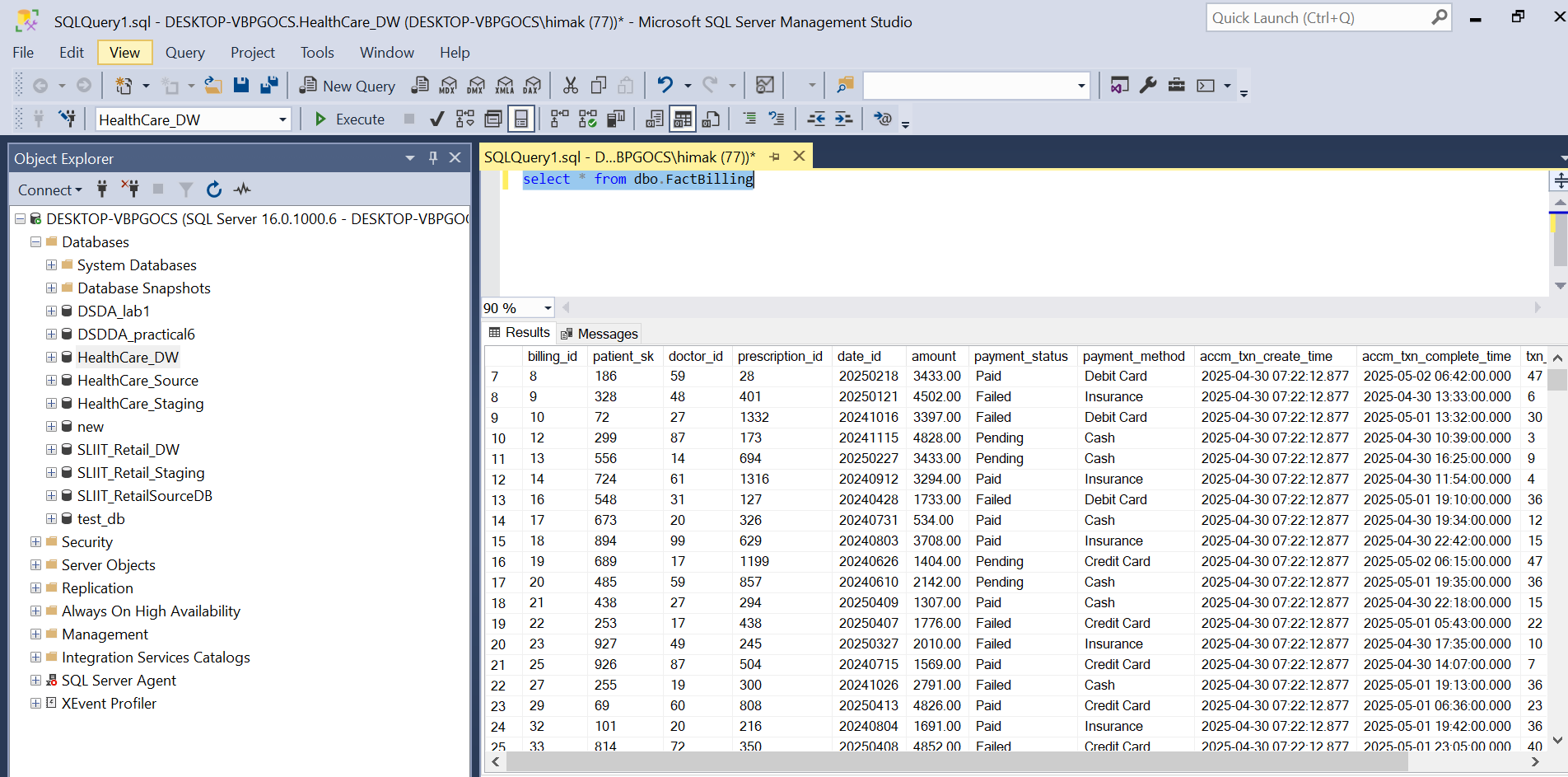
*accm\_txn\_create\_time DATETIME,*

*accm\_txn\_complete\_time DATETIME,*

*txn\_process\_time\_hours INT;*

  
  
Here the data of accm\_txn\_create\_time was filled as other were to be filled by a separate SSIS package.  
  
As the final component added a OLE DB Destination pointing FactBilling while matching the columns as needed.  
  
  
  
The Final Data Flow task looked like this 👇  
  
  
  


Final Control Flow view 👇

  
  
Then the package was executed and the results could be seen in the HealthCare\_DW Database tables.  
  


# Accumulating Fact Table

For the final part where we have update accm\_txn\_complete\_time and txn\_process\_time\_hours, First I created a table with txn\_id and txn\_completed\_time using a python script  
  
*#Table to create a sample table with txn\_id and accm\_tx\_complete\_time*

*import pandas as pd*

*from datetime import datetime, timedelta*

*import random*

*def generate\_txn\_table(n):*

*base\_time = datetime(2025, 4, 27, 8, 0, 0)*

*txn = []*

*for i in range(n):*

*random\_minutes = random.randint(1, 60 \* 48)*

*txn\_time = base\_time + timedelta(minutes=random\_minutes)*

*txn.append({*

*'txn\_id': i + 1001,*

*'accm\_txn\_complete\_time': txn\_time.strftime('%Y-%m-%d %H:%M:%S')*

*})*

*return txn*

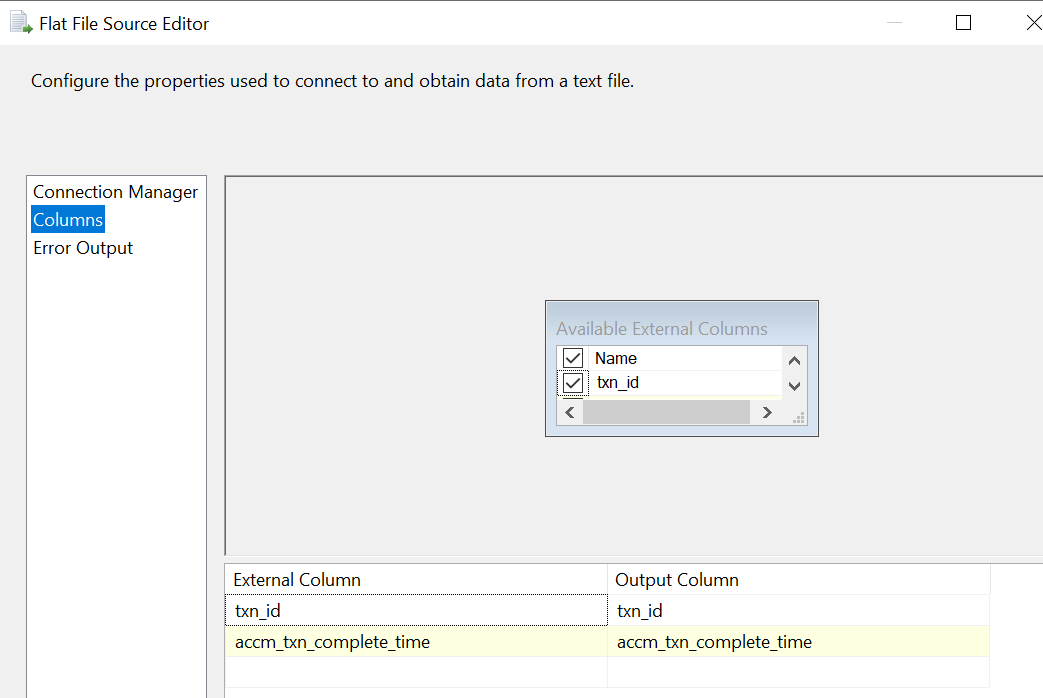
*txn\_data = generate\_txn\_table(1369)*

*# Save to CSV*

*df = pd.DataFrame(txn\_data)*

*df.to\_csv("CSVs/txn.csv", index=False)*

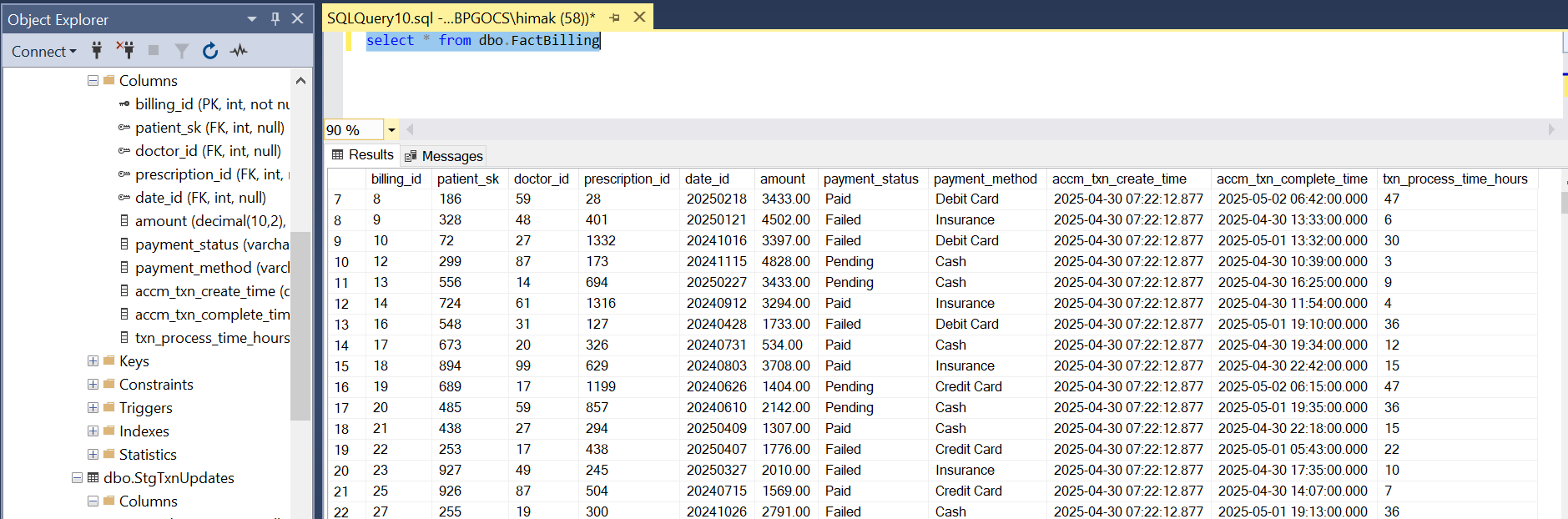
Then I created a new package for this step. Inside control flow added a new data flow and inside it, added a flat file source to take the above CSV generated as the input.

  
  
Then connected it to an OLE DB Destinationa to load this as a table in Data Warehouse DB.  
In Event Handler tab added a PostExecute SQL Command Task to update the FactBilling table as needed used below SQL statement in there.  
  
*UPDATE f*

*SET f.accm\_txn\_complete\_time = s.accm\_txn\_complete\_time, f.txn\_process\_time\_hours = DATEDIFF(HOUR, f.accm\_txn\_create\_time, s.accm\_txn\_complete\_time)*

*FROM dbo.FactBilling f INNER JOIN dbo.StgTxnUpdates s ON f.billing\_id = s.txn\_id;*

This compares the billing\_id with txn\_id and pass the CompleteTime and Calculated Time difference to the Fact Table. After Execution of this command a Fact Table with correct data could be seen



\*\*\* End OF Report \*\*\*