Data Warehousing and Business Intelligence

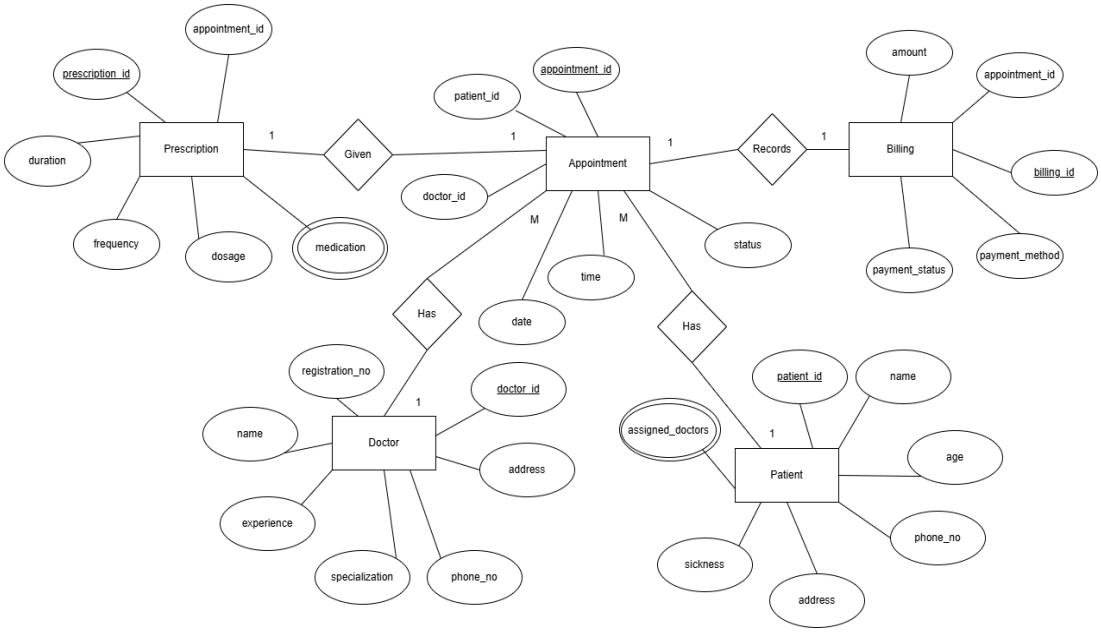
Assignment 01

IT22550262

Table of Contents

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)

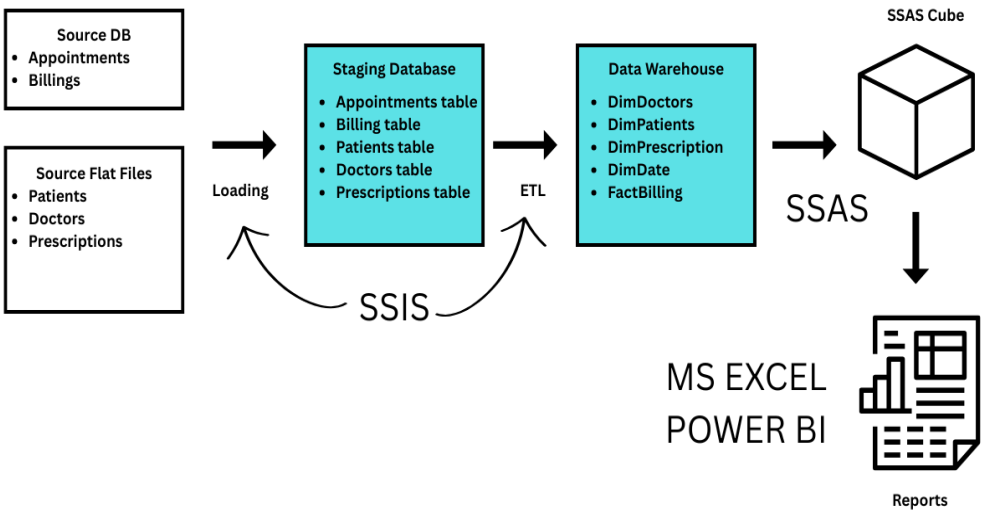
The patient's details with sickness are recorded when he or she is registered. Then the system assigns a matching doctor related to that patient. Each detail connected with appointment like prescriptions given, patient and doctor details are recorded in the appointment entity. The billing entity records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

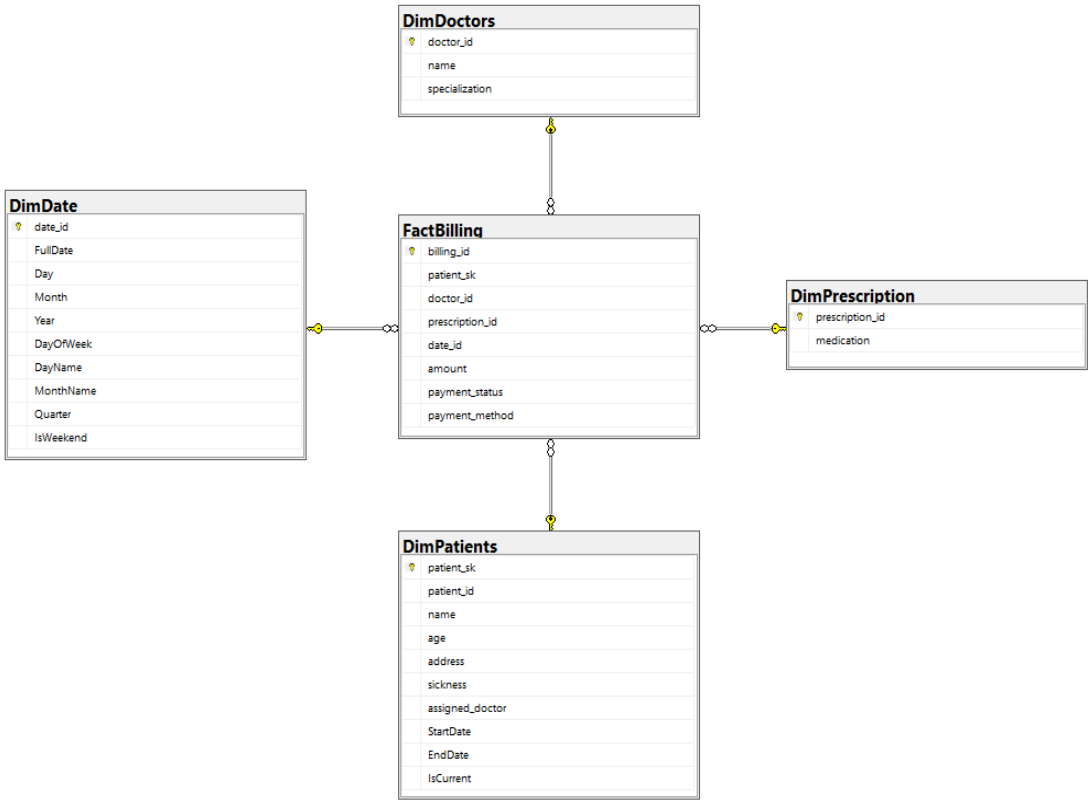


(Solution architecture)

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)
);
```

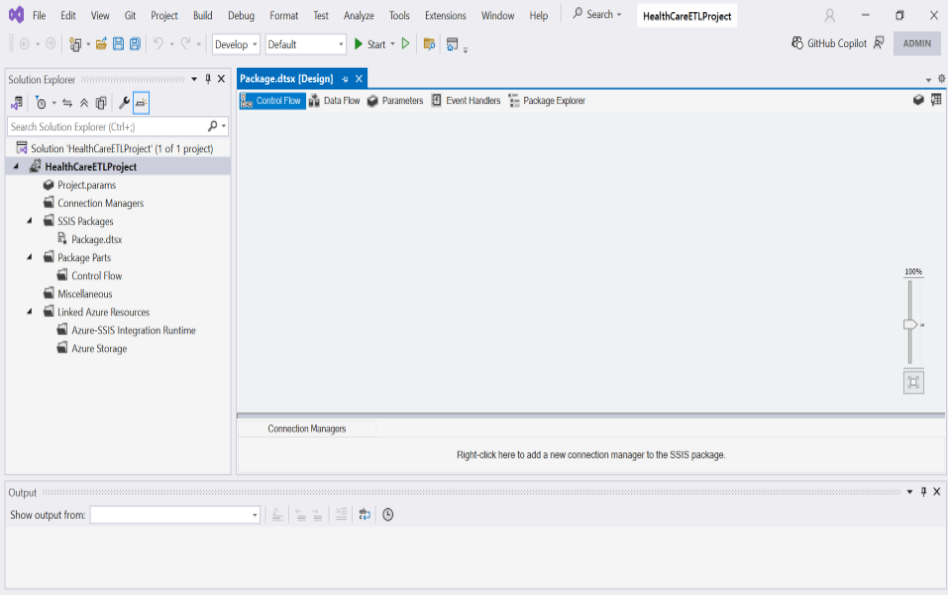
The Fact table which has foreign keys from patient, doctor, prescription and date tables. It holds the quantifiable and measurable values such as amount.

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



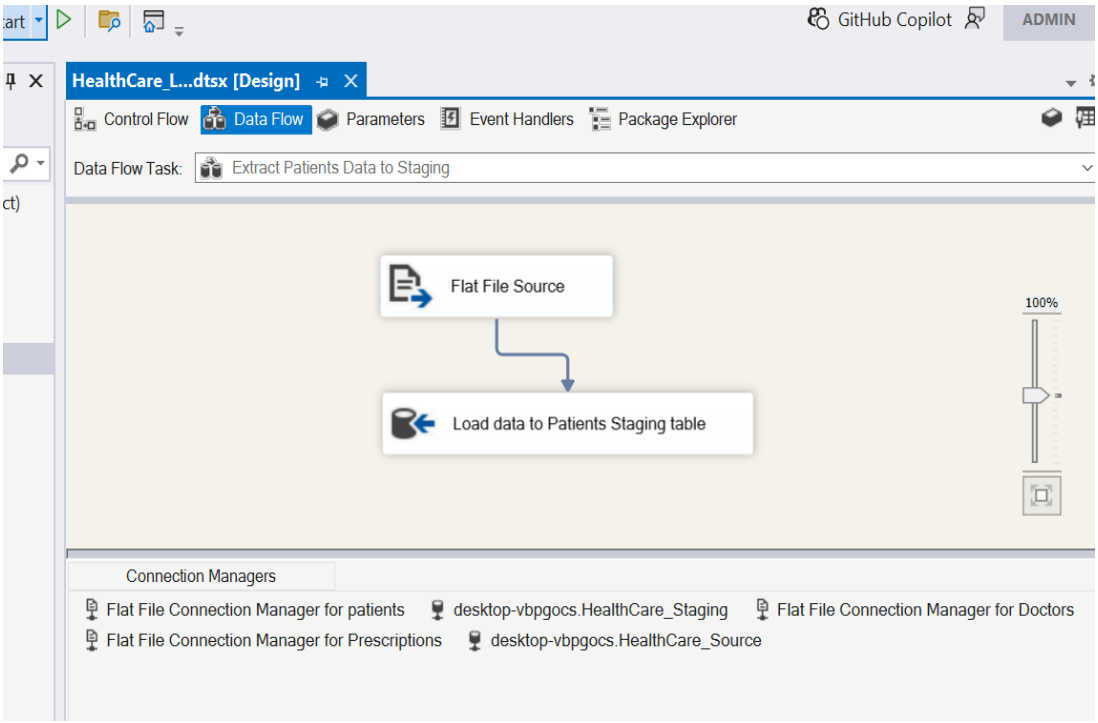
Data Extraction from sources

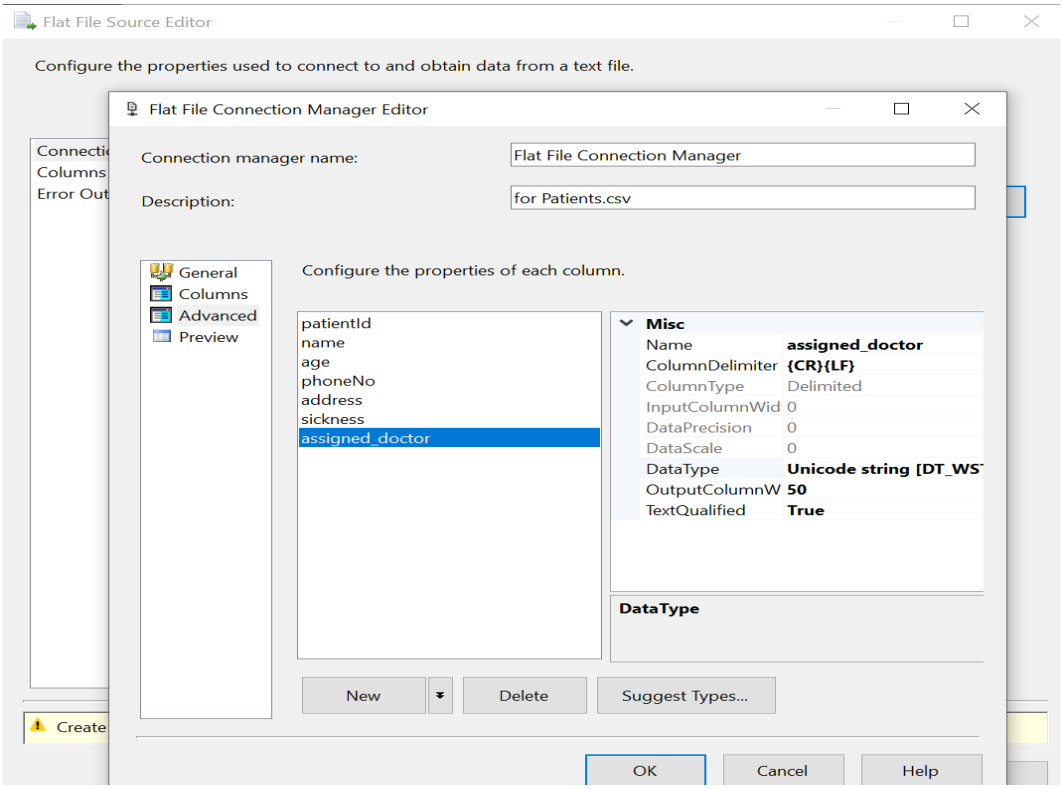
Considering the organization of data in this scenario, I followed the loading part as

- 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.

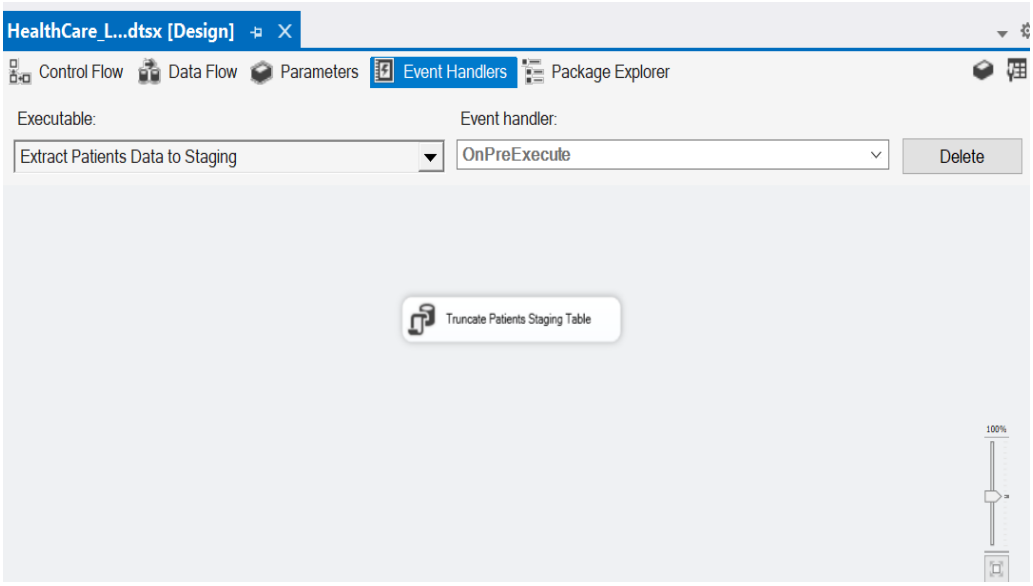
Then started the extraction process of data from patients.csv, doctors.csv flat files, added data flow tasks for these two and inside these configured the source and destinations as necessary. Used Flat File Source and OLE DB Destination SSIS tools.



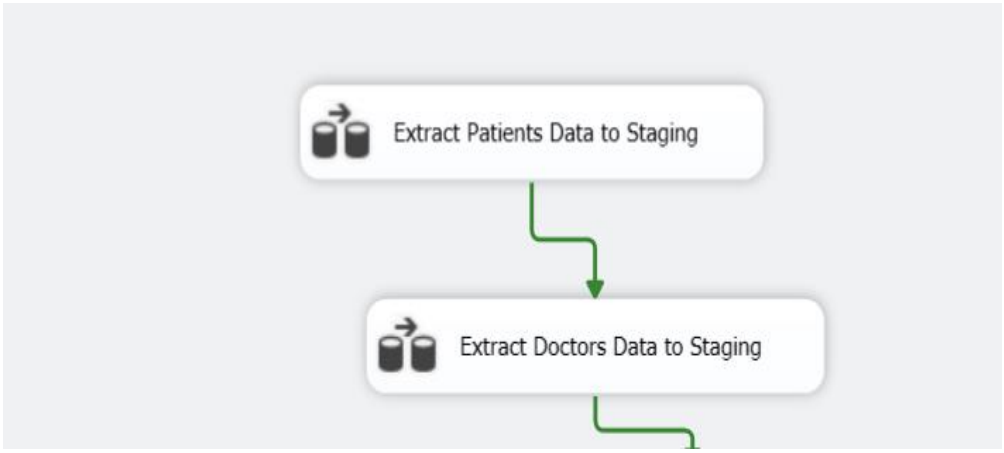


Created a connection manager which points to the CSV file in the known location and then changed data types as needed for strings and numeric values.

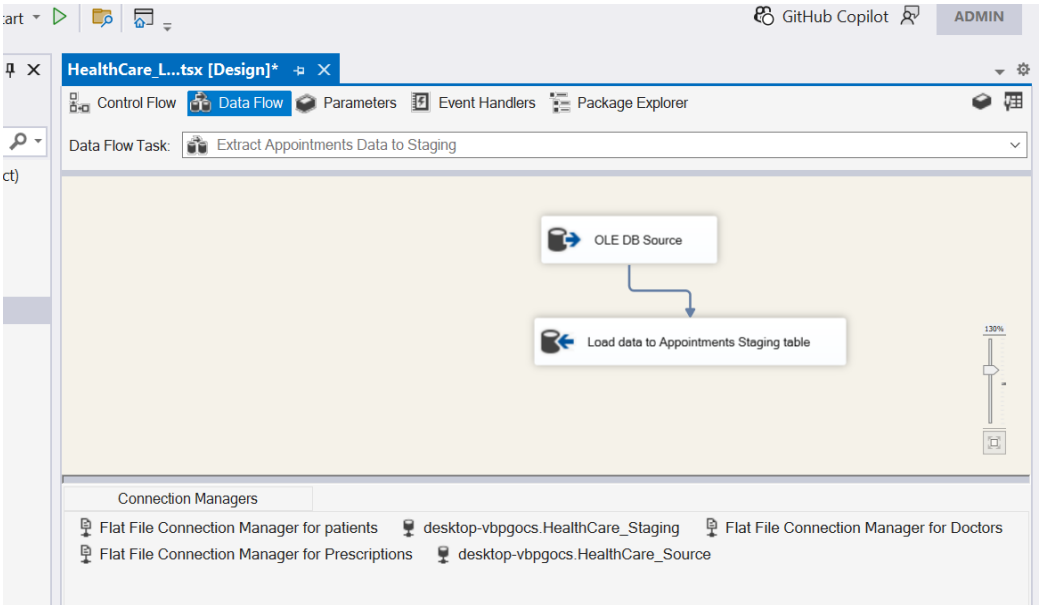
Also added a OnPreExecute event handler to truncate the table to avoid same data repeating.

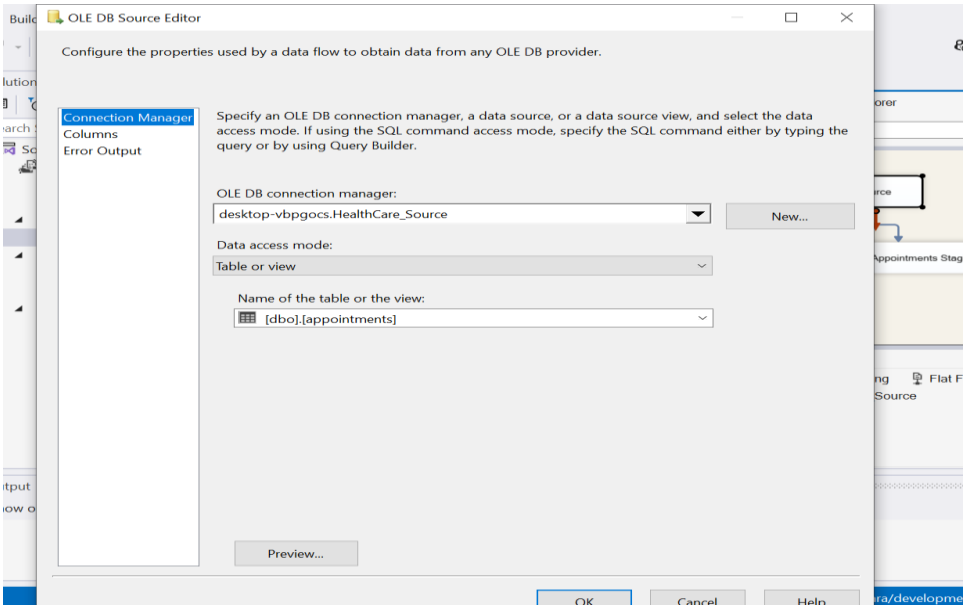


For the CSV Flat Source File: Doctors same thing did with above tools along with OnPreExecute event handler.



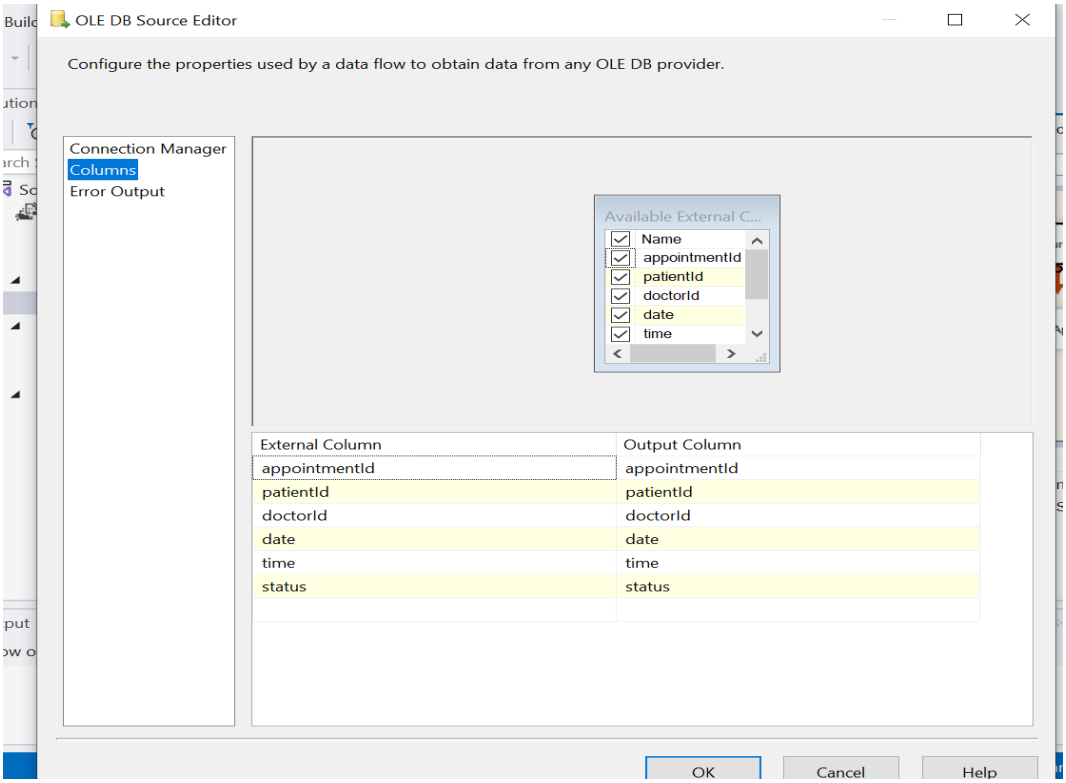
Then carried out the process for appointments table which is an OLE DB source in HealthCare_Source database. Used an OLE DB Source and OLE DB Destination from SSIS toolbox along with PreExecute event handler.

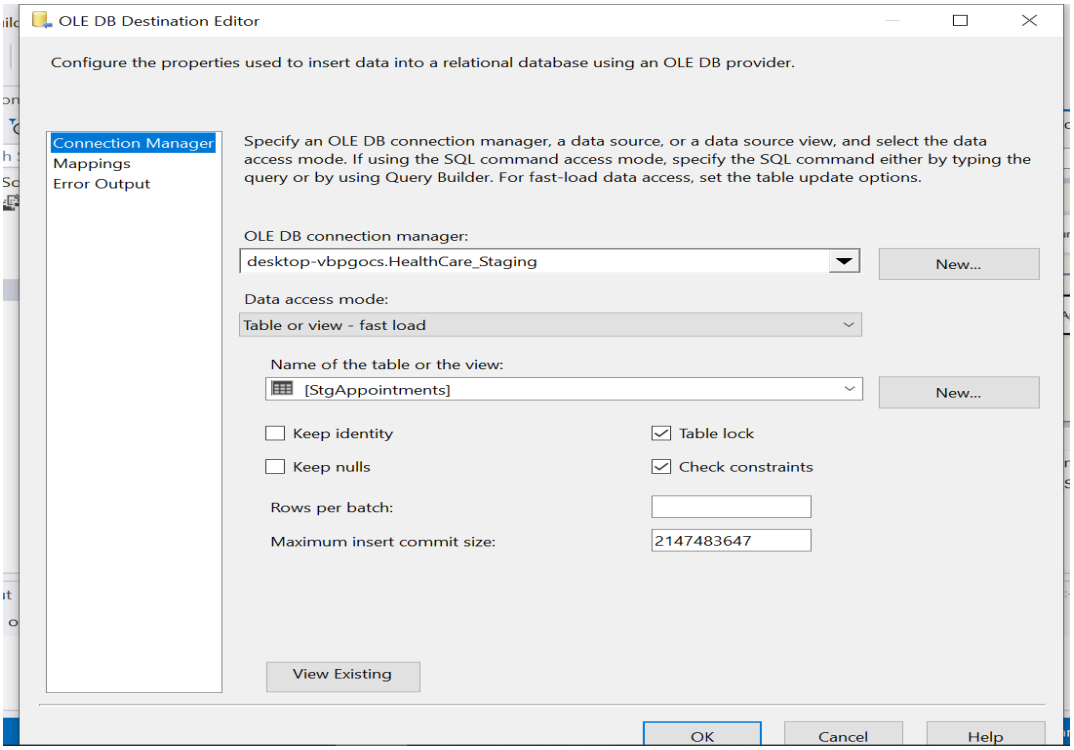


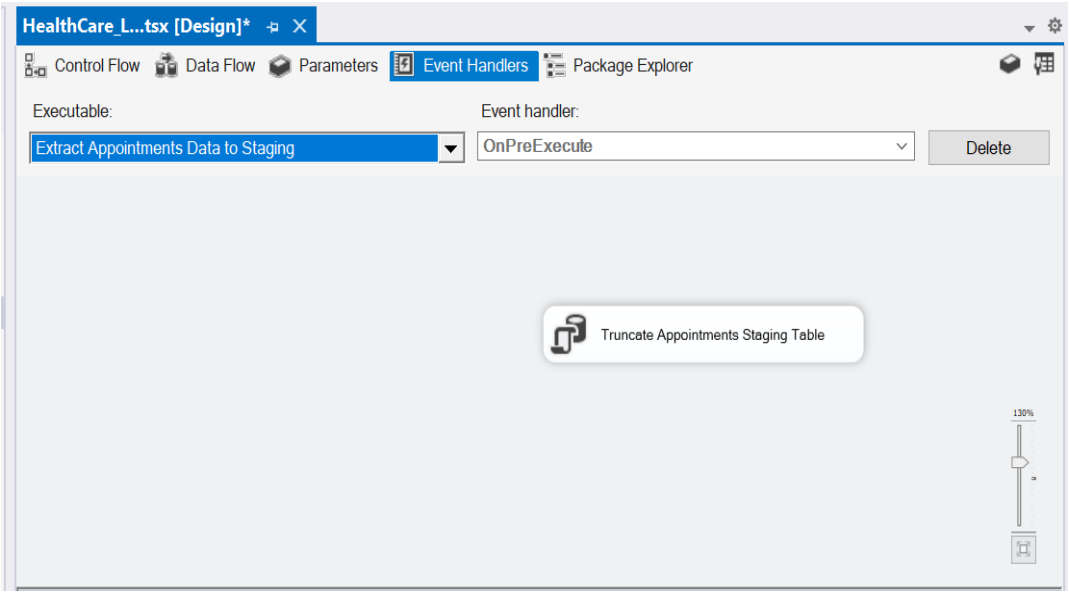


(Configured the connection for the OLE DB Source)

Checked the Mappings section to confirm the columns


OLE DB Source Editor
Configure the properties used by a data flow to obtain data from any OLE DB provider.
Connection Manager
Columns
Error Output
Available External C...
Name
appointmentId
patientId
doctorId
date
time
External Column
Output Column
appointmentId
appointmentId
patientId
patientId
doctorId
doctorId
date
date
time
time
status
status
OK
Cancel
Help

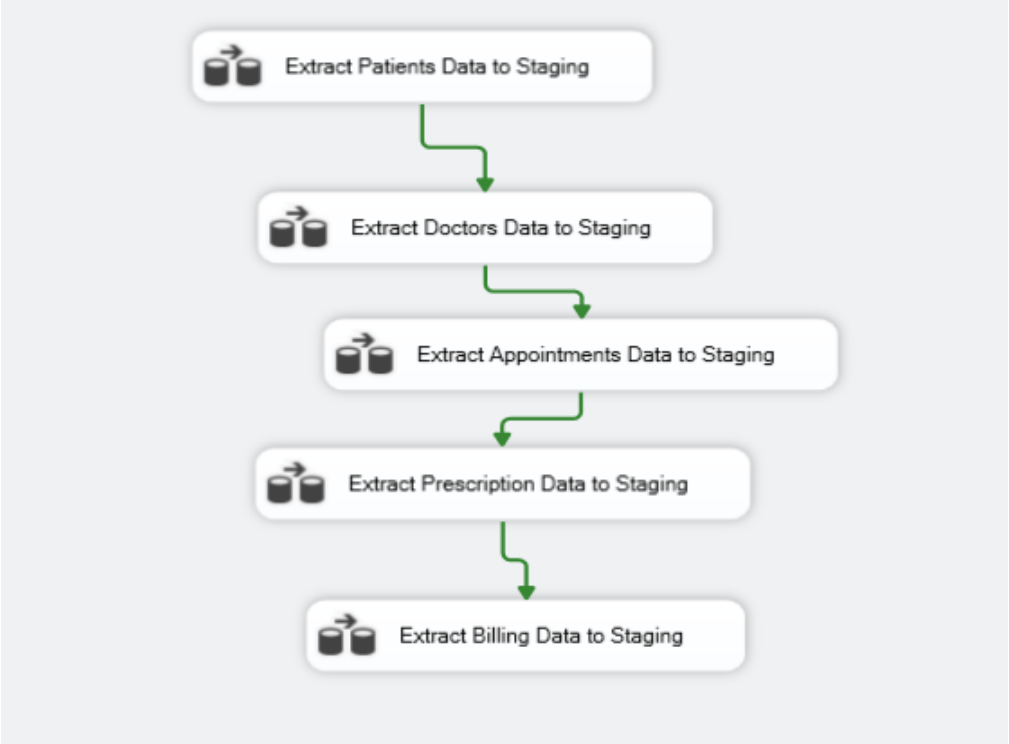

OLE DB Destination Editor
Configure the properties used to insert data into a relational database using an OLE DB provider.
Connection Manager
Mappings
Error Output
Specify an OLE DB connection manager, a data source, or a data source view, and select the data access mode. If using the SQL command access mode, specify the SQL command either by typing the query or by using Query Builder. For fast-load data access, set the table update options.
OLE DB connection manager:
desktop-vbpgocs.HealthCare_Staging
New...
Data access mode:
Table or view - fast load
Name of the table or the view:
[StgAppointments]
New...
Keep identity
Table lock
Keep nulls
Check constraints
Rows per batch:
Maximum insert commit size:
2147483647
View Existing
OK
Cancel
Help



(The SQL Task to truncate before executing to avoid repetitions)

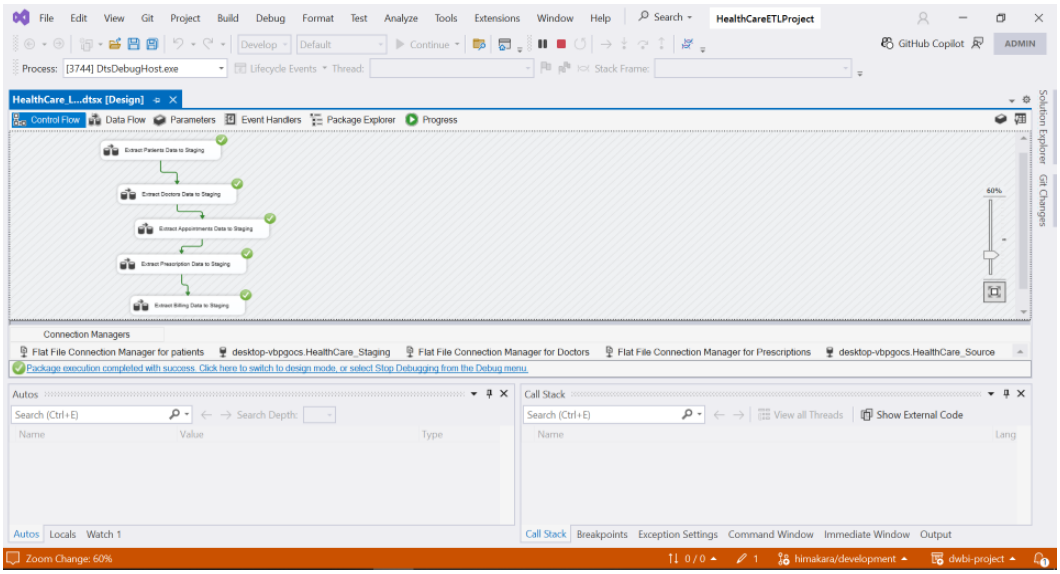
Likewise, the same procedure was carried out for Prescription CSV as a Flat File Source and Billing Table as an OLE DB Source in HealthCare_Source respectively.

The Final Control Flow looked like this 👇



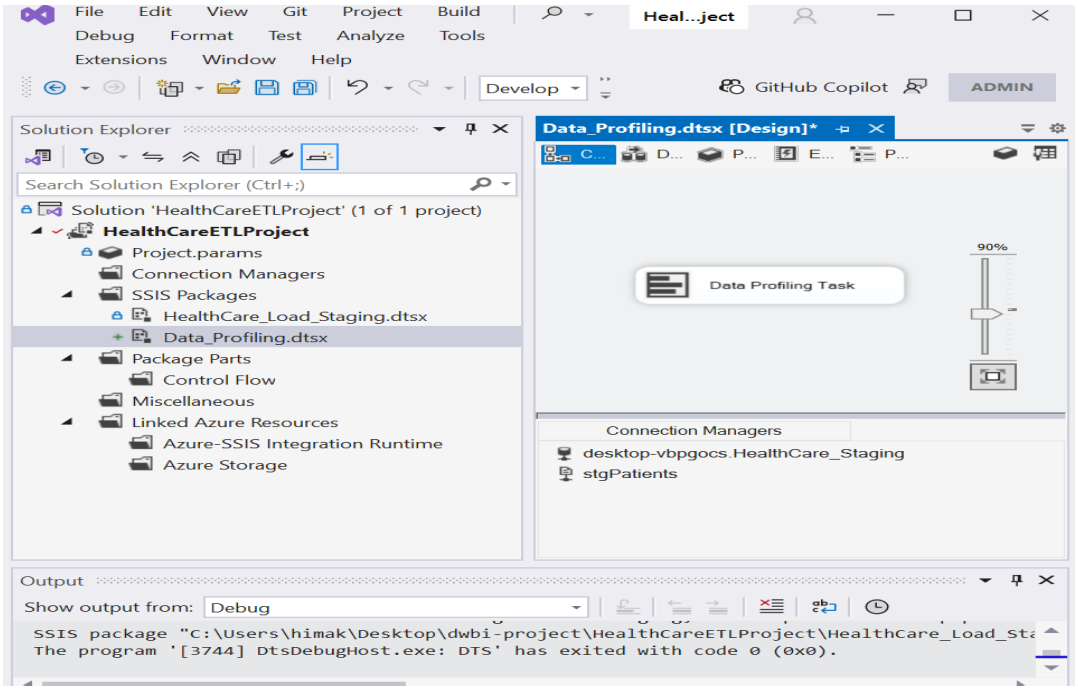
(Control flow design for Source to Staging)

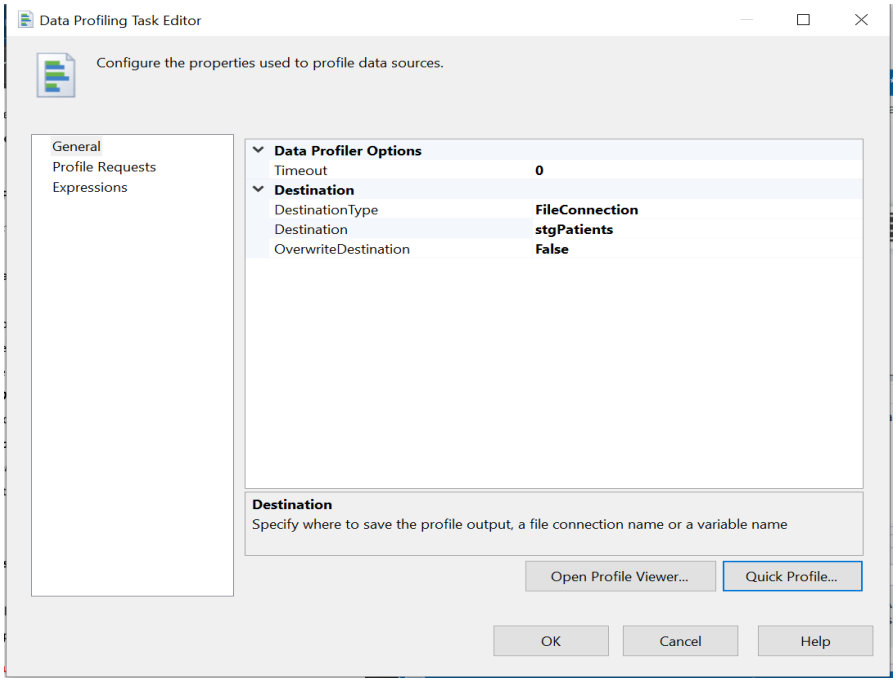
Then executed the HealthCare_Load_Staging package after building the solution



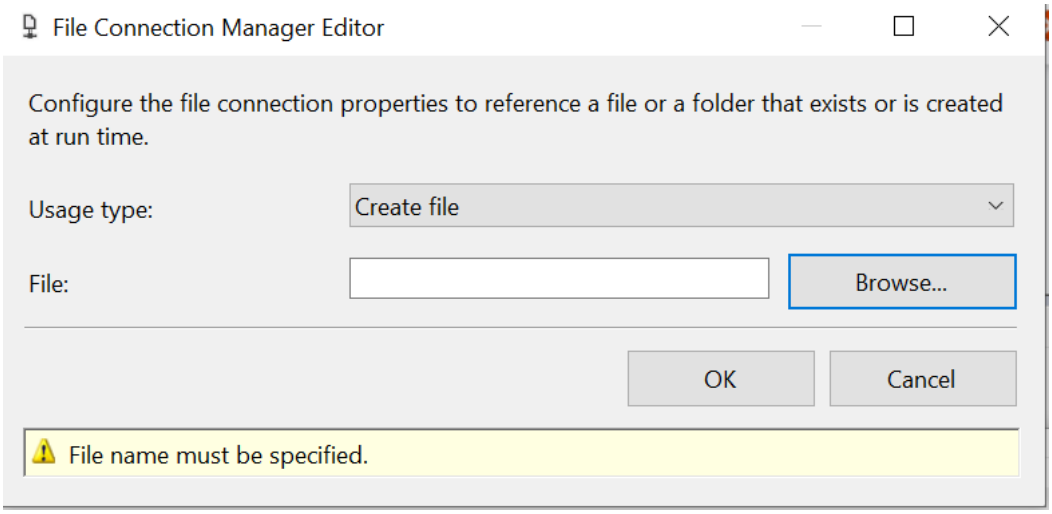
Then the required data from all the different sources could be observed in the Staging Database.

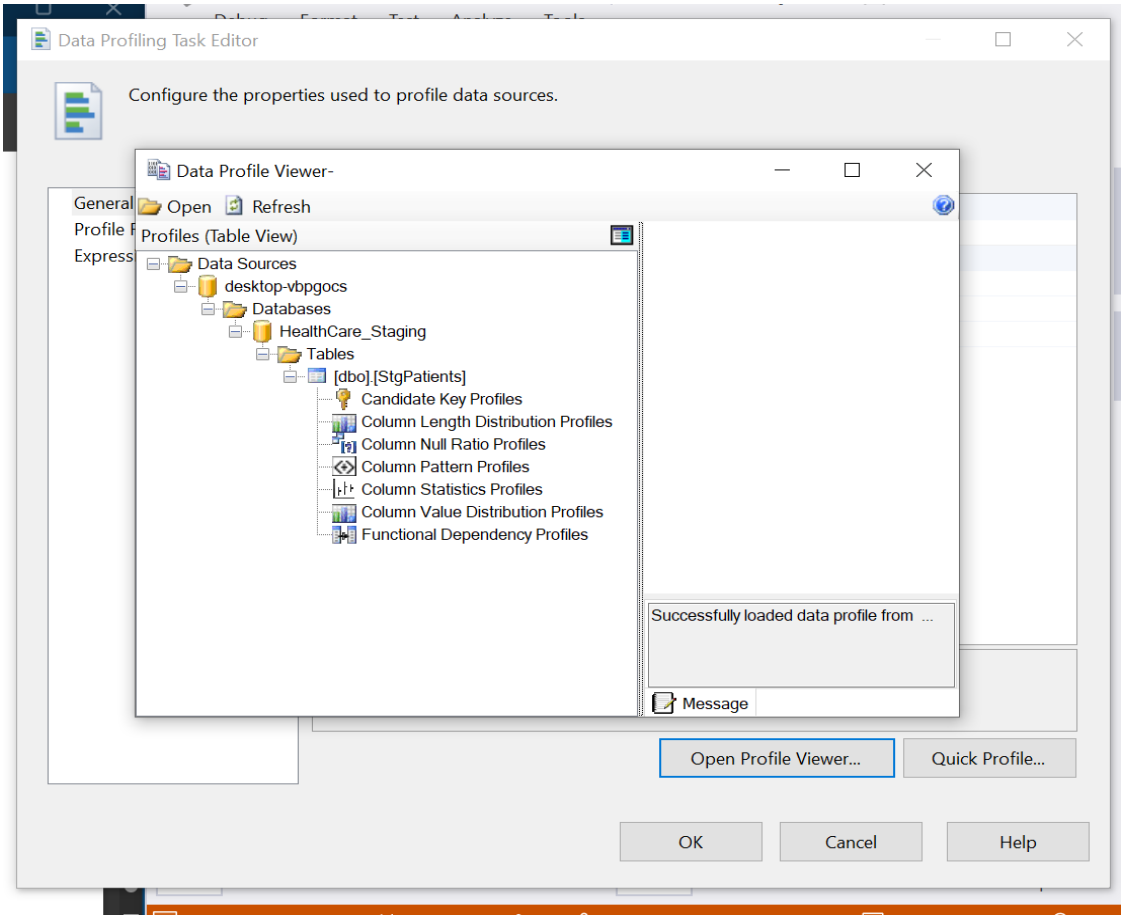
A Data Profiling part to understand what transformations should be performed on the data was carried out next to identify available data quality, hidden relationships and reduce data integration errors.



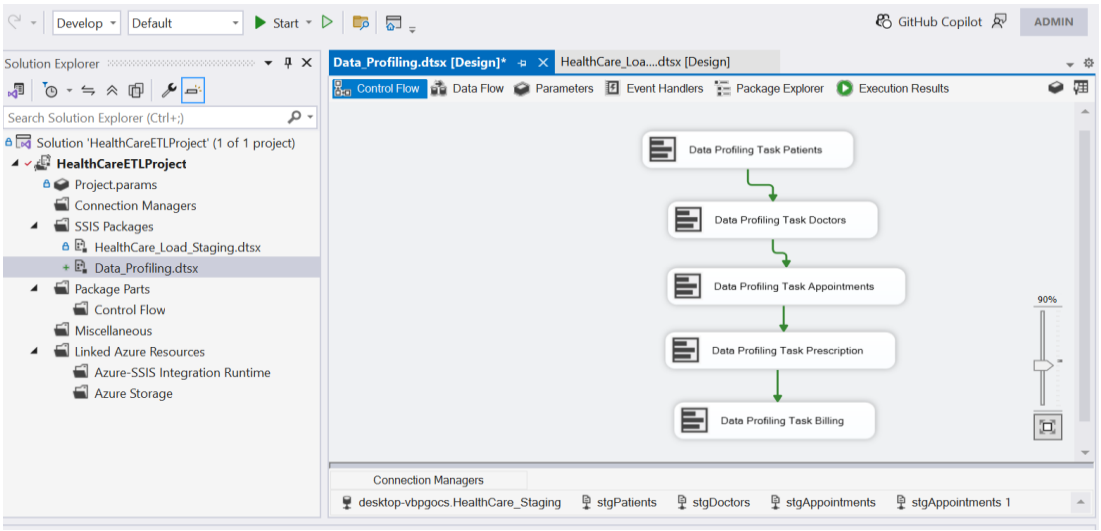


Here configured the usage file to “Create file” to create a new file and the location to be saved.





Here profiled every table to understand the transformations

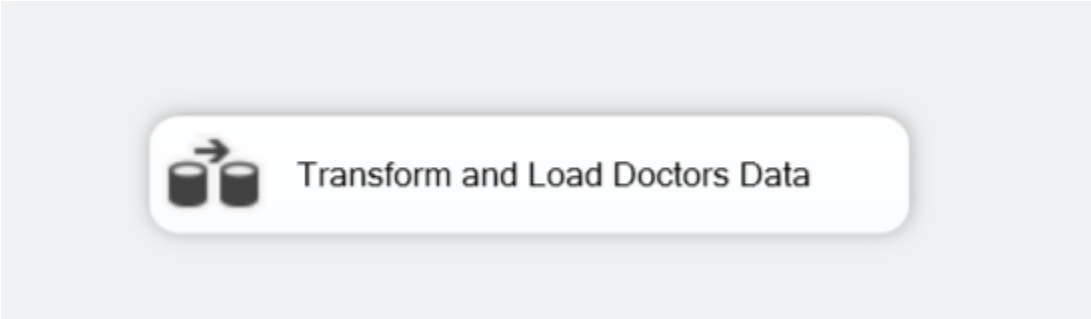


Then I could observe the exported files appeared in the given location

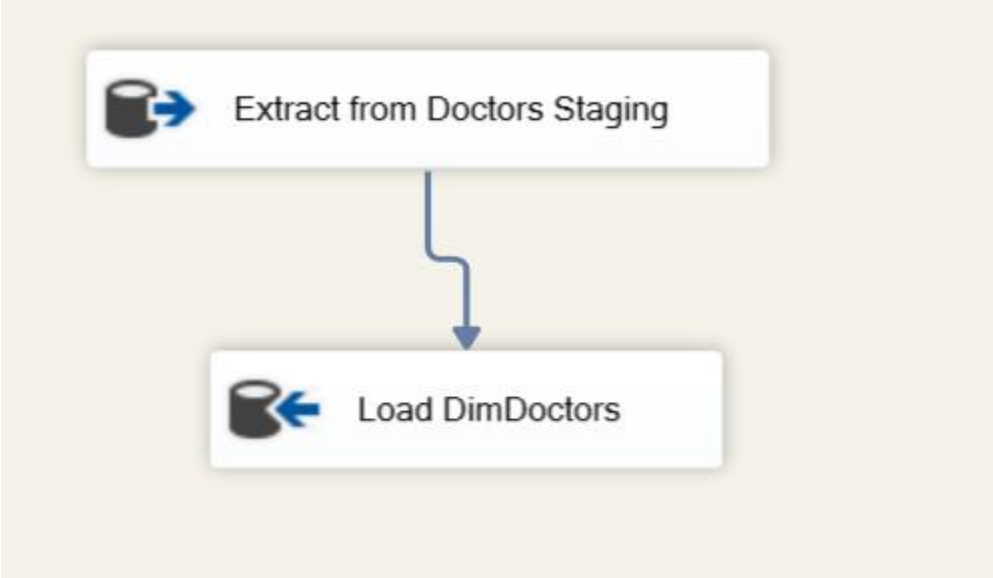
stgAppointments	4/21/2025 11:54 PM	File	207 KB
stgBilling	4/21/2025 11:54 PM	File	2 KB
stgDoctors	4/21/2025 11:54 PM	File	81 KB
stgPatients	4/21/2025 11:54 PM	File	62 KB
stgPrescription	4/21/2025 11:54 PM	File	37 KB

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.

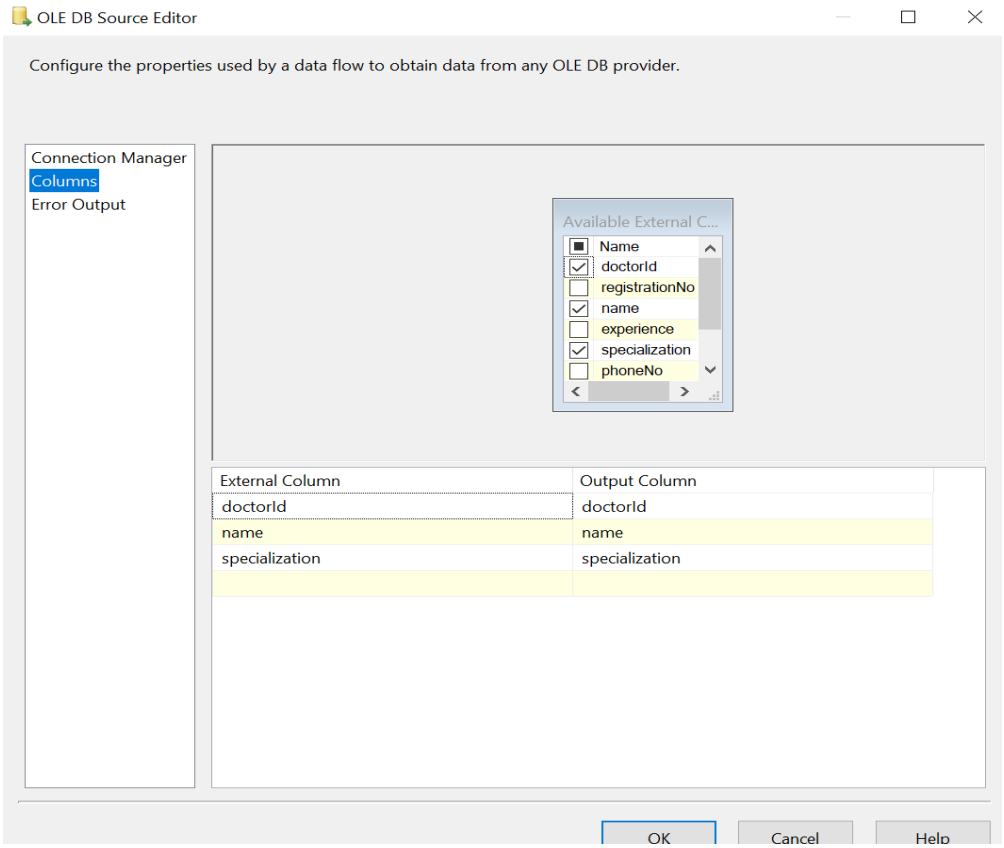
Here my main idea is to Load data from Staging Database to Data warehouse.
First up, Transform and Load from StgDoctors to DimDoctors, I created a Data Flow task



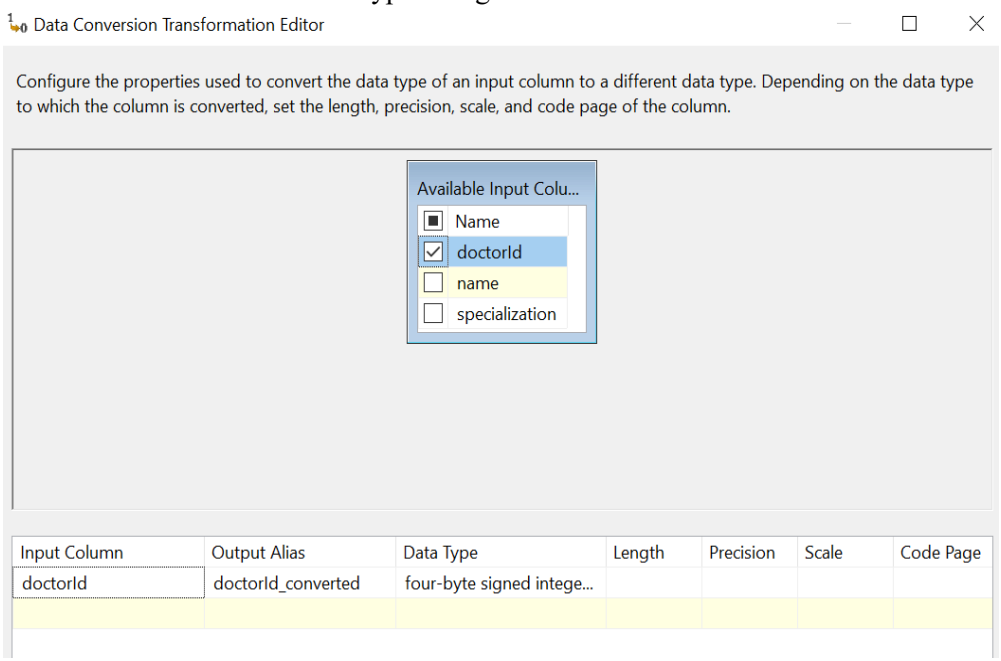
Inside it I created OLEDB Source which points the table StgDoctors in Staging Database and OLEDB Destination which points table in Warehouse Database



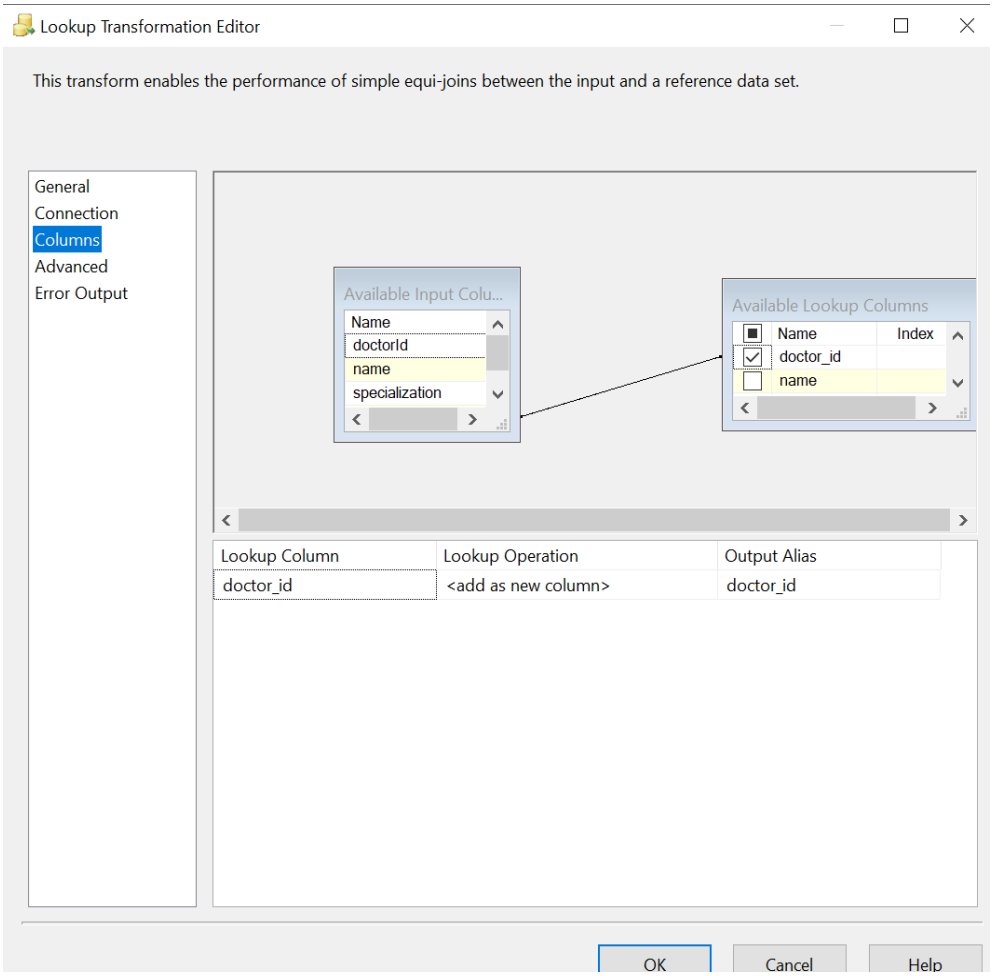
Inside OLE DB Source, added the connection, and selected most important attributes for the future analytical tasks



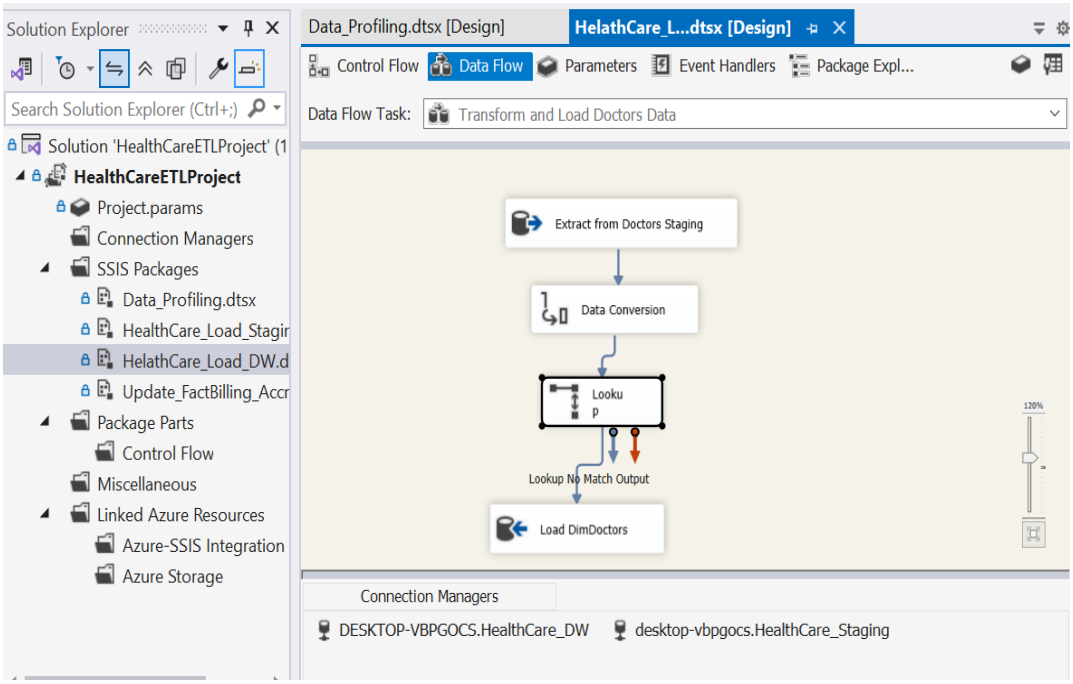
Then added a Data conversion tool to convert doctorId to four-byte-signed-integer so there will not be mismatch between type in StgDoctors and DimDoctors.



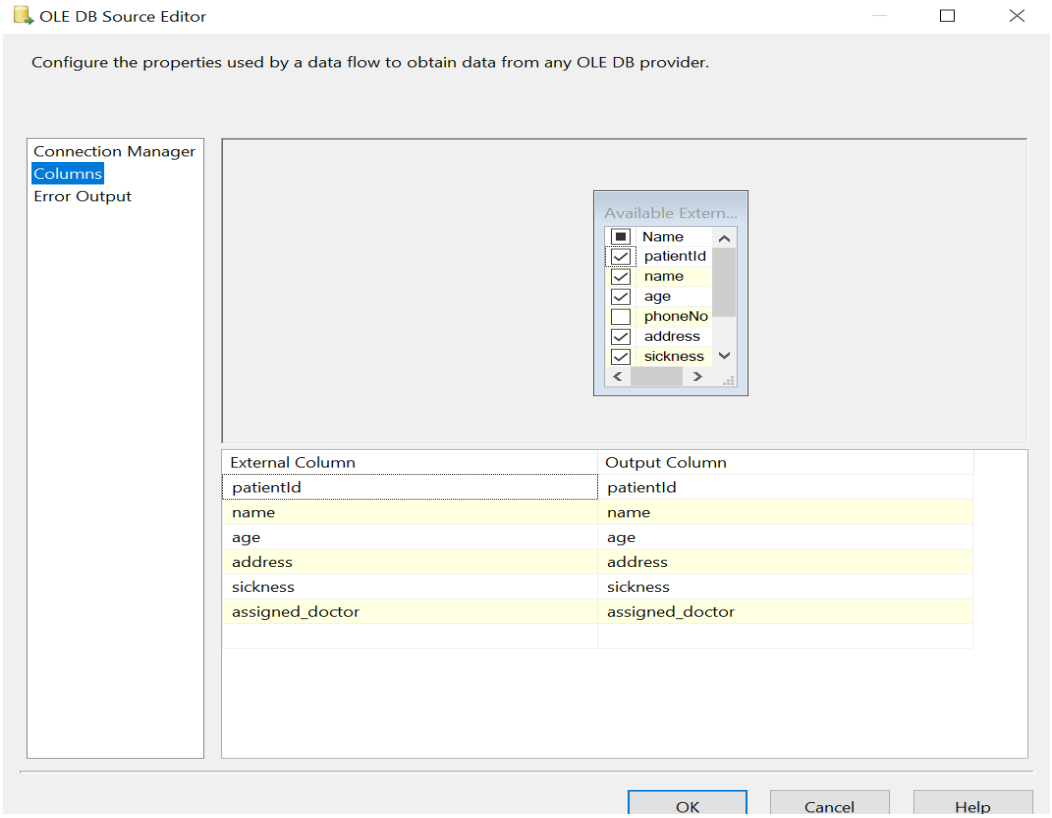
Then a Lookup tool to check any records already exist in DimDoctors.



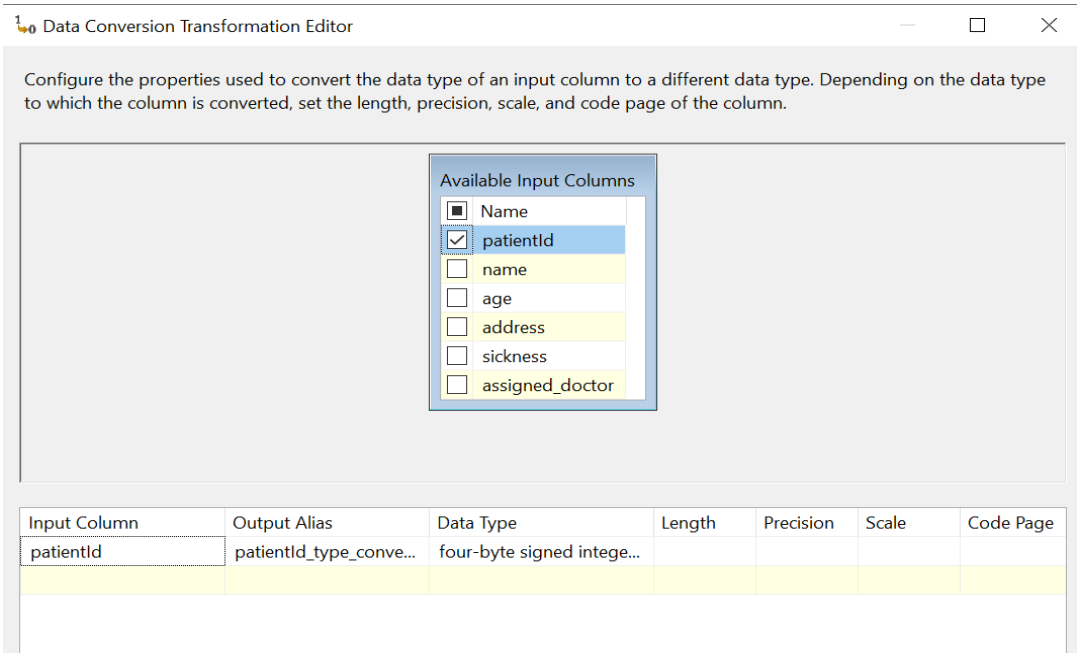
If no match was found then I loaded the data to the OLE DB Destination, DimDoctors. Here Lookup was used to avoid repetitions or conflicts when inserted. The final Data flow task looked like this 👇



Then I performed ETL tasks on StgPatients. First I added an OLE DB connection and connected to the StgPatients in Stage database.

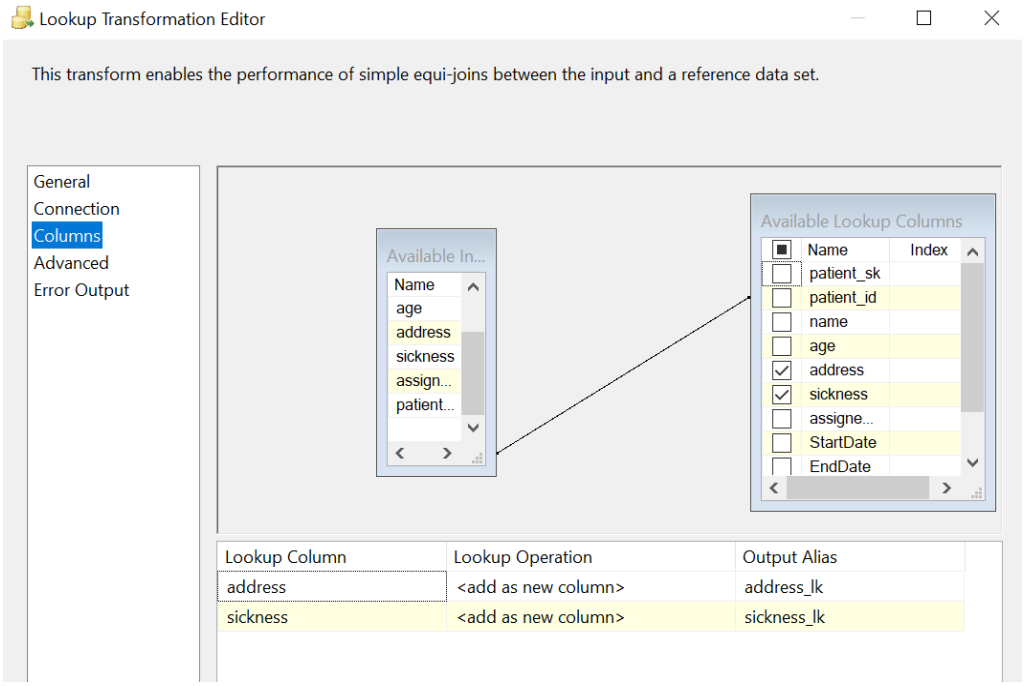


Here is also a Data conversion tool needed for the patientId



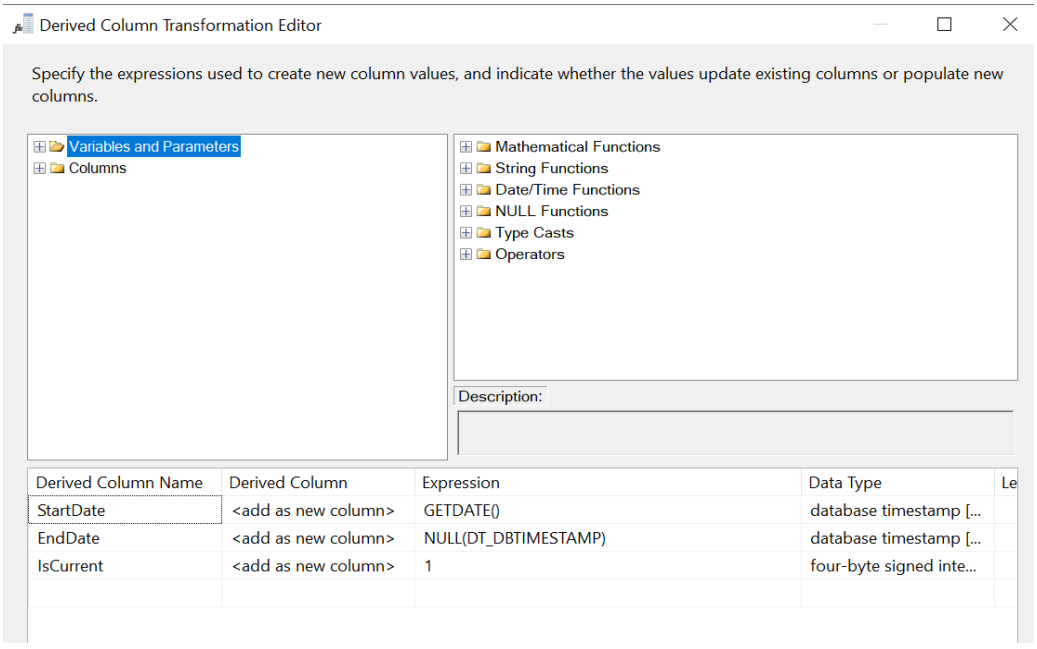
Since I'm going to prove Slowly Changing Variable operations (SCD) here, I added a Lookup

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.

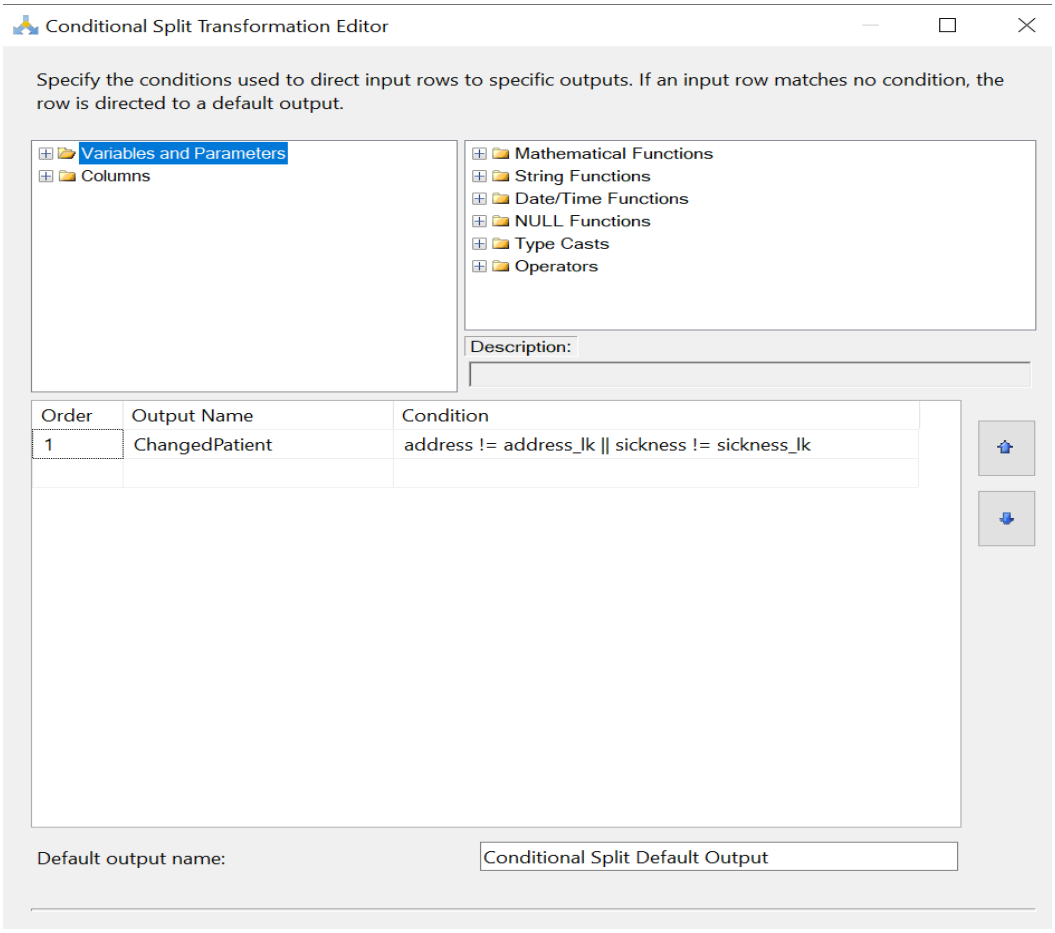
From lookup, the no match output found directed to a Derived Column tool where I treat these records as fresh and add the SCD related columns to it.



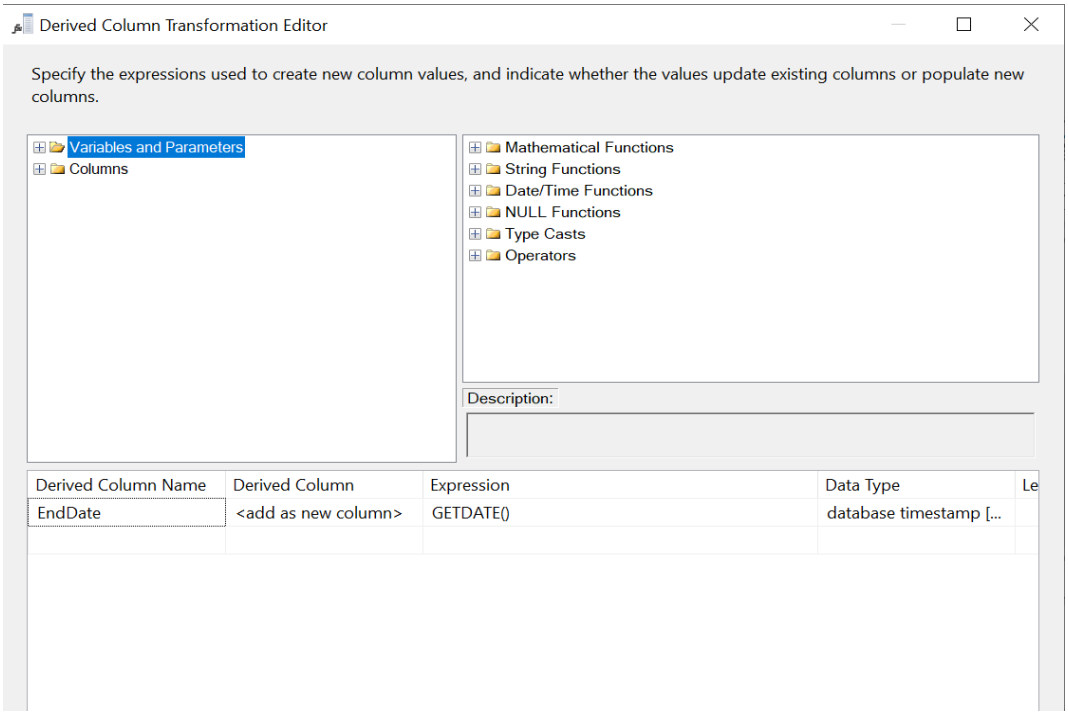
Here StartDate set to take the current date via GETDATE() and EndDate set to NULL cause and due to this is the latest IsCurrent set to 1(Latest).

Then connected this to an OLE DB destination pointing DimPatients where we Extract, Transform and Load our new patient details.

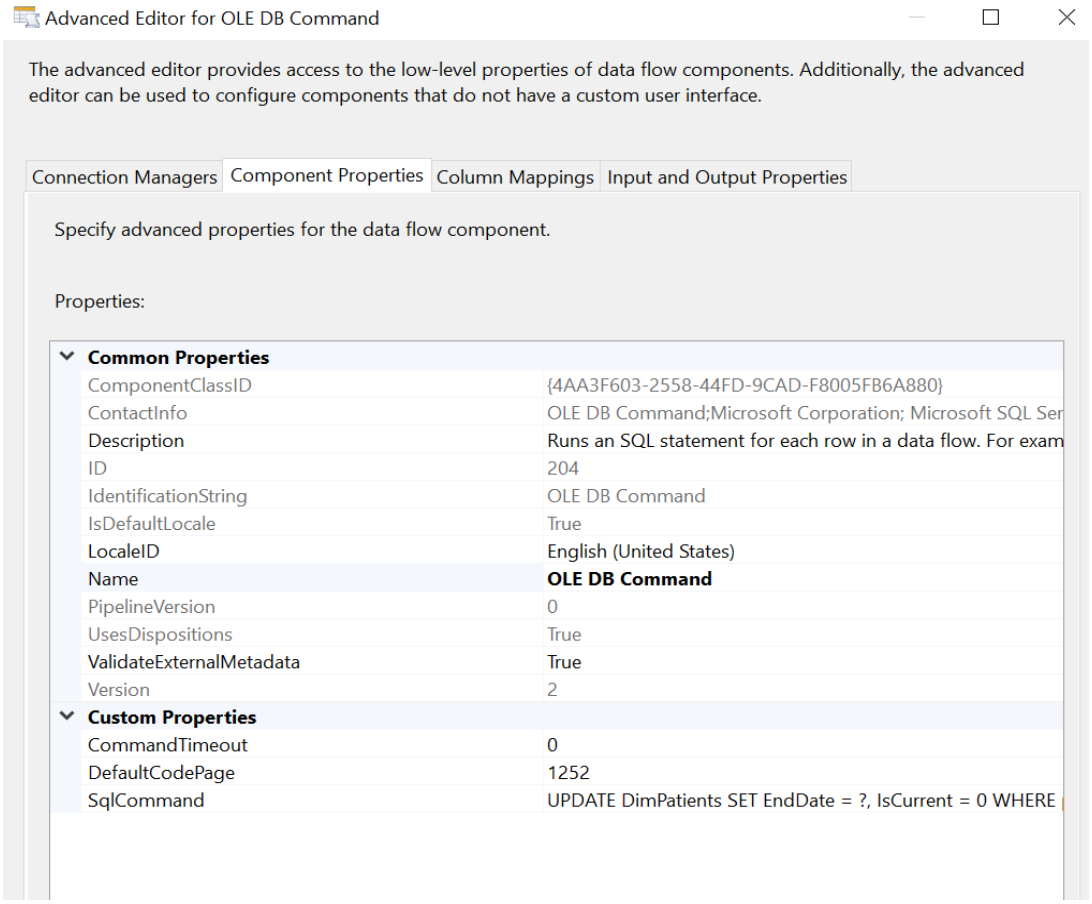
The secondary path of Lookup tool, A match found was directed to a Conditional Split tool where I checked whether the address or sickness has changed of the patient



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.



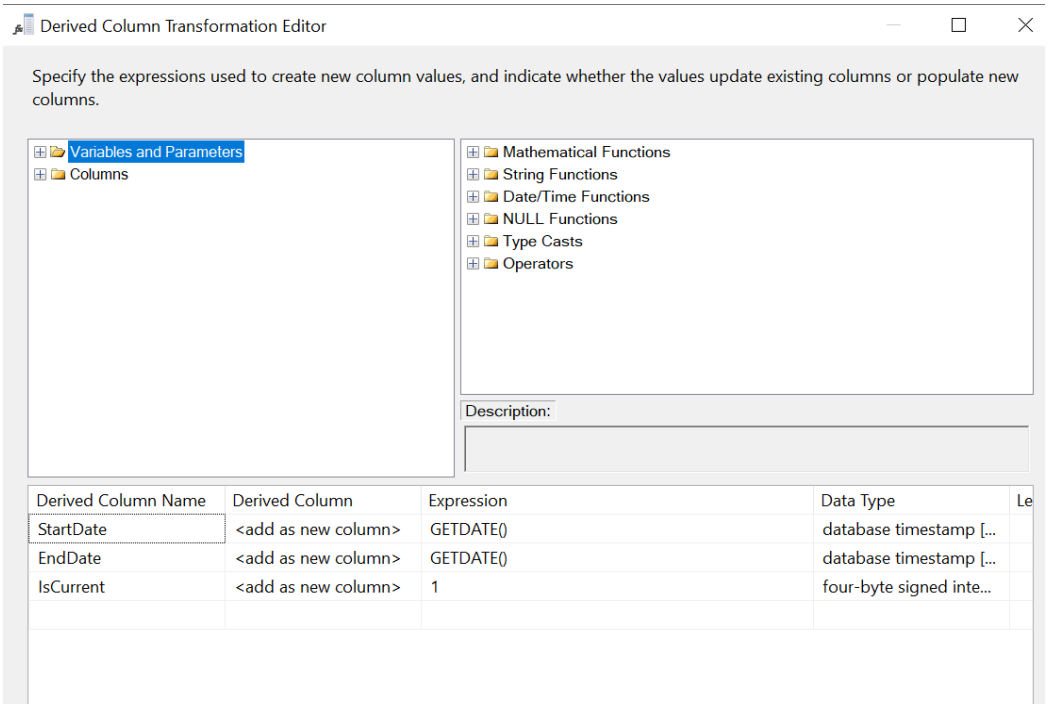
using that EndDate, I updated the current record as an old one (IsActive = 0). Used an OLE DB command tool for that.

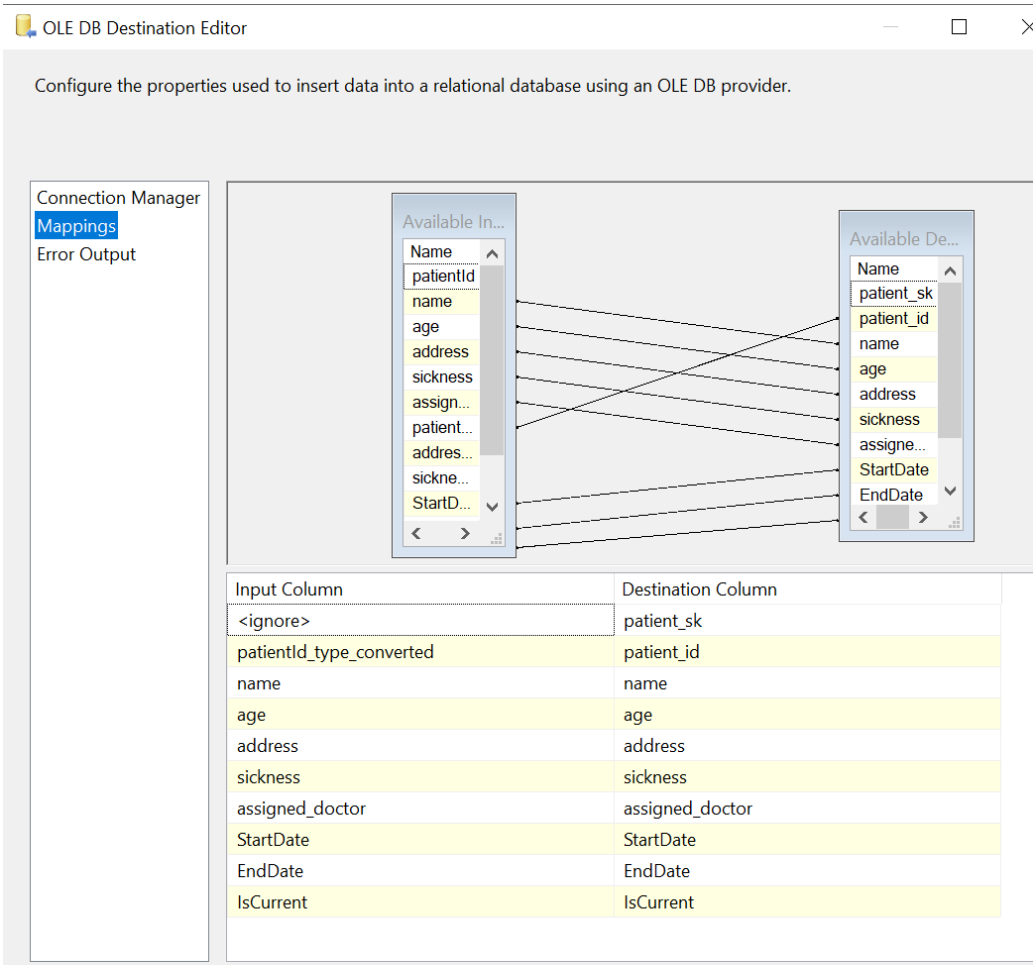


Used this SQL line in there

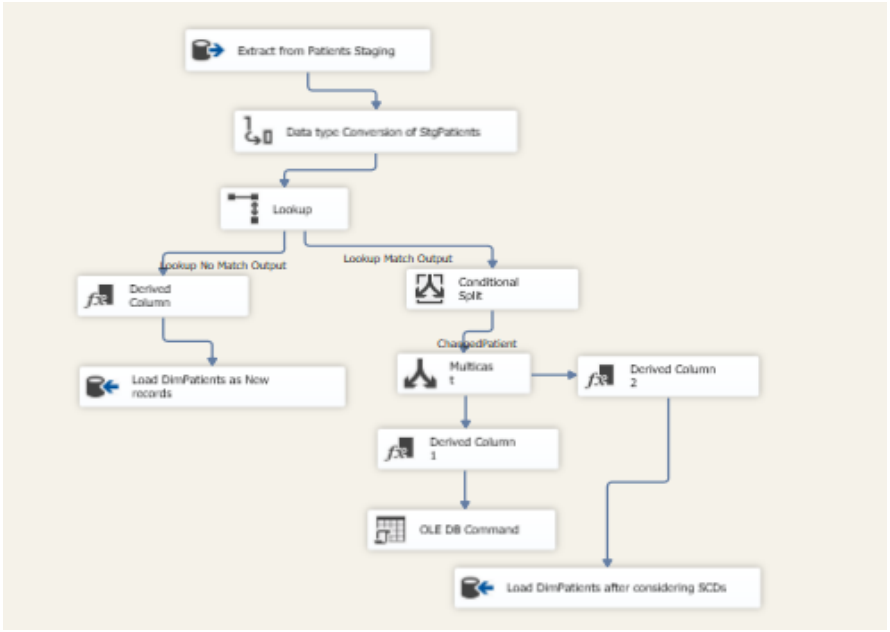
UPDATE DimPatients SET EndDate = ?, IsCurrent = 0 WHERE patient_id = ? AND IsCurrent =1

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.





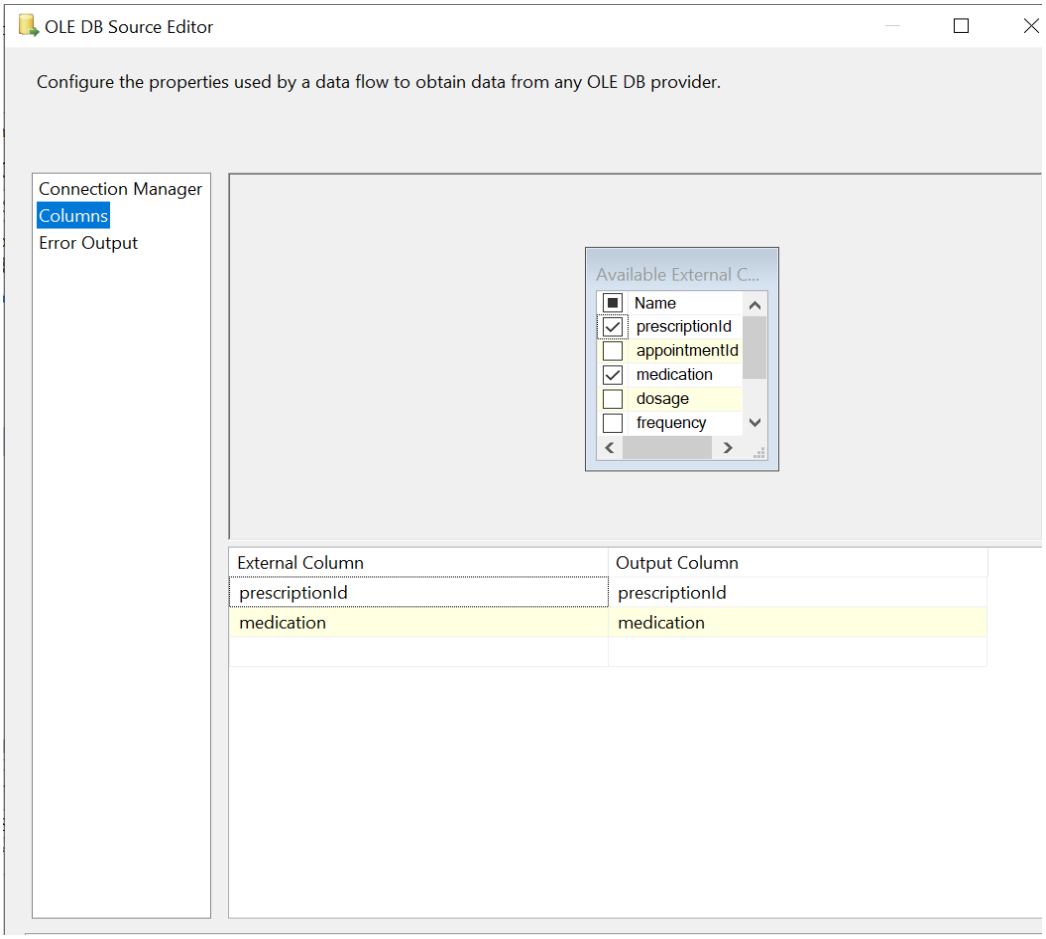
The Final data flow for DimPatients looked like this 👇

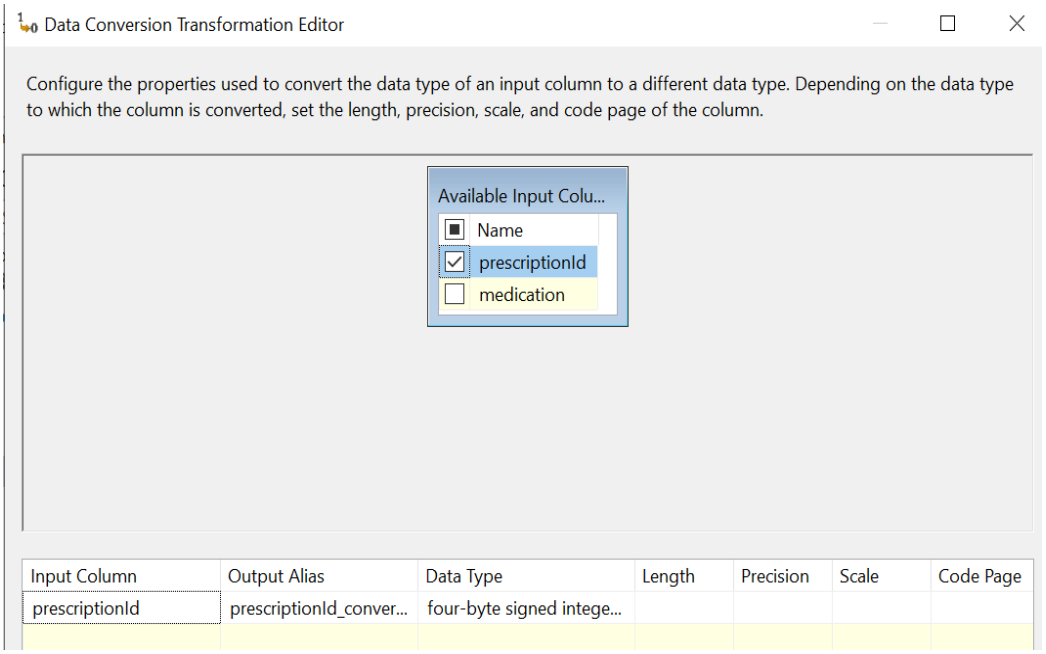


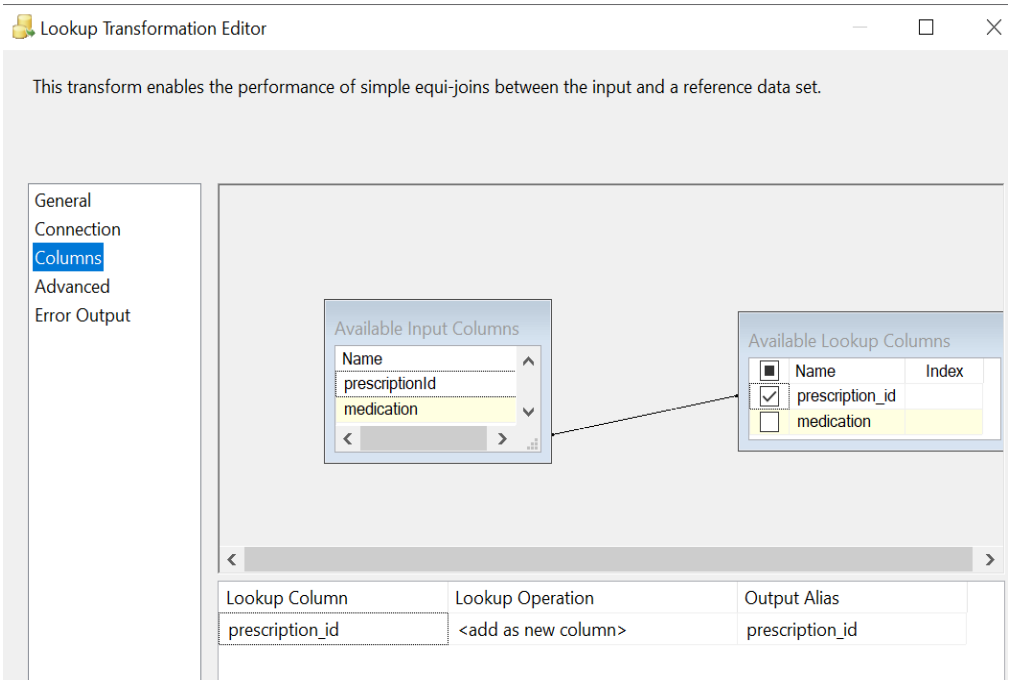
Then I started working on DimPrescription. Here just like DimDoctors, only selected the columns which matters most.

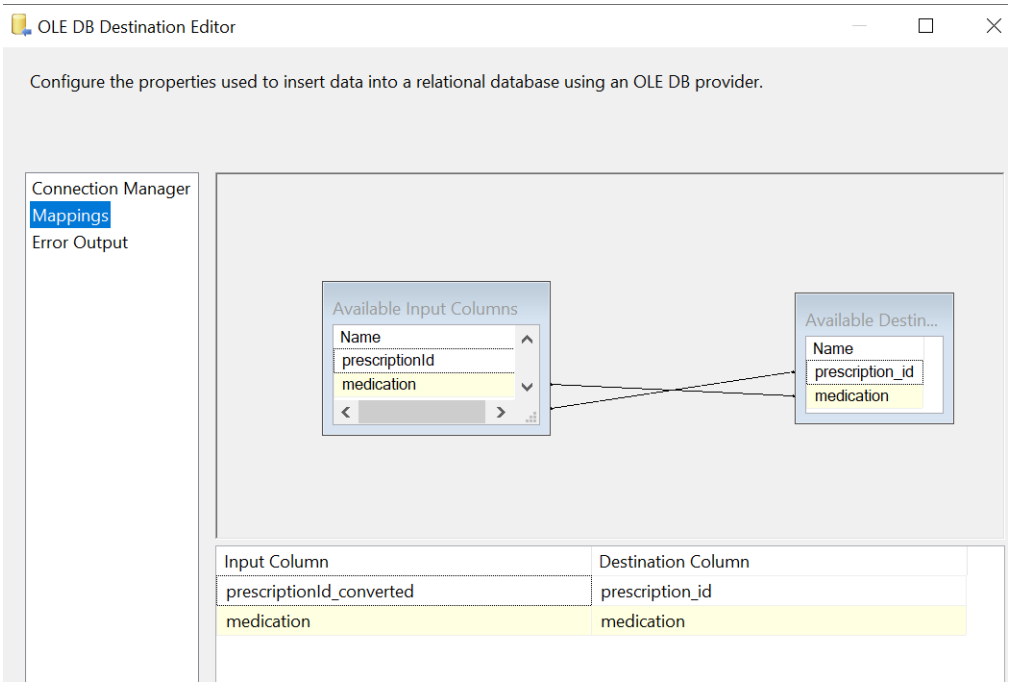
Used and OLE DB Source and a Destination along with a Data Conversion to convert

prescriptionId and a Lookup tool to check record already exists.

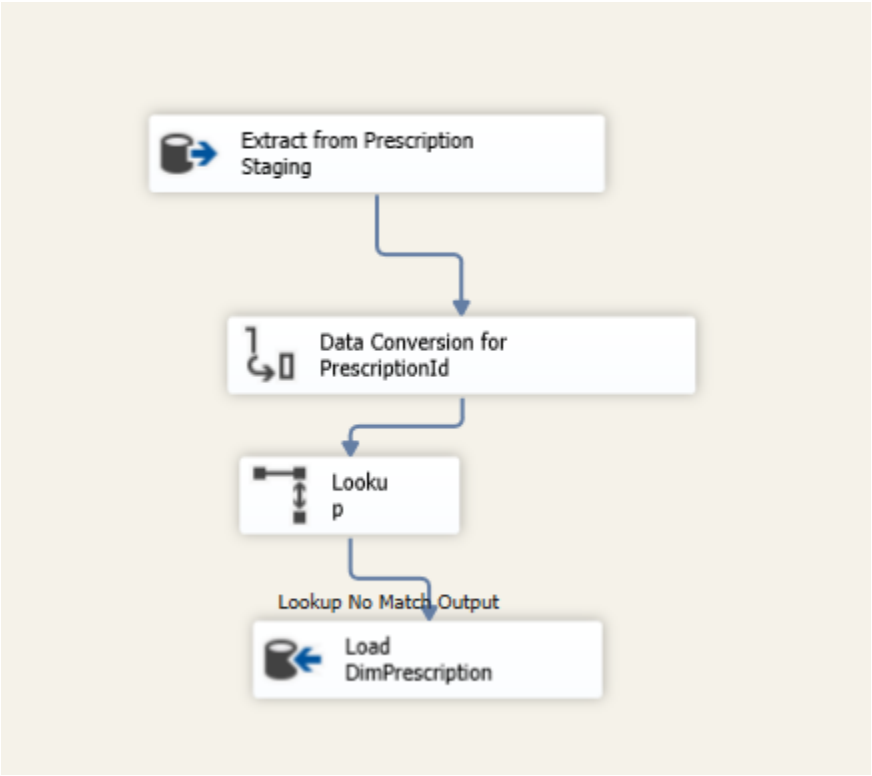






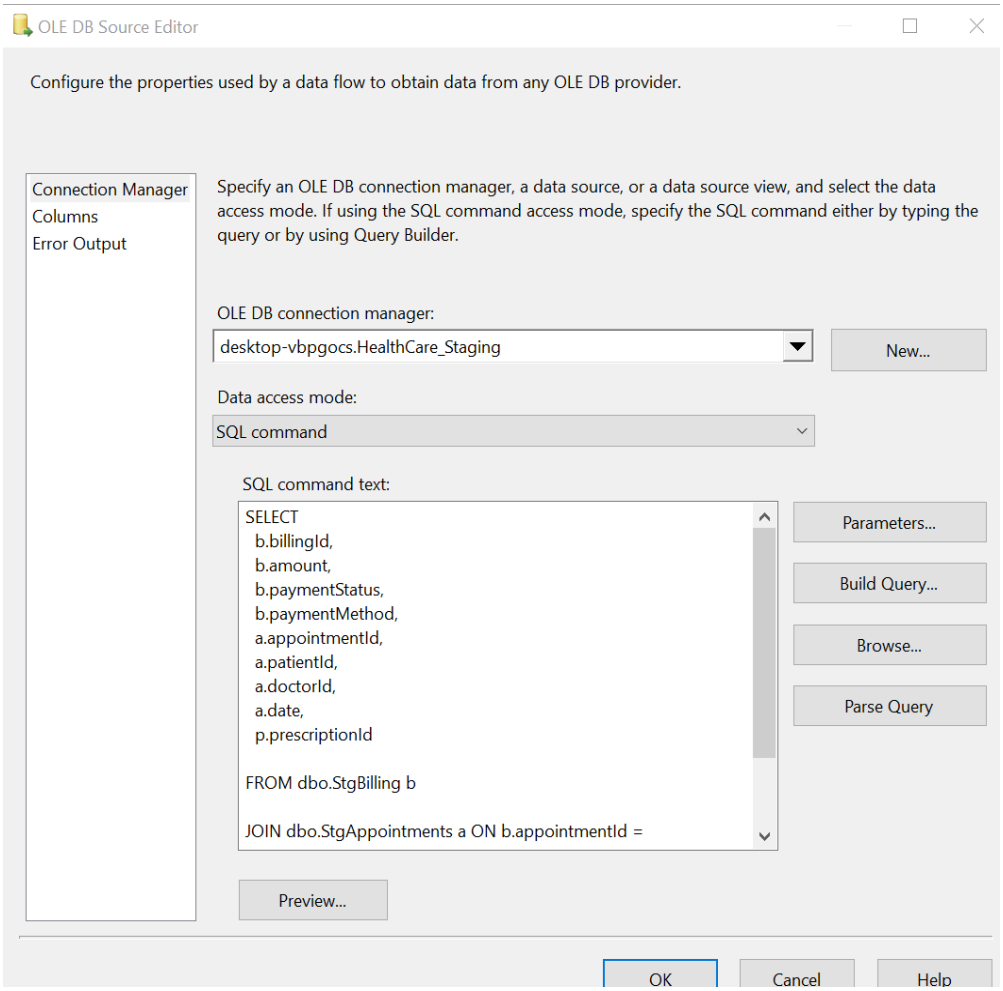


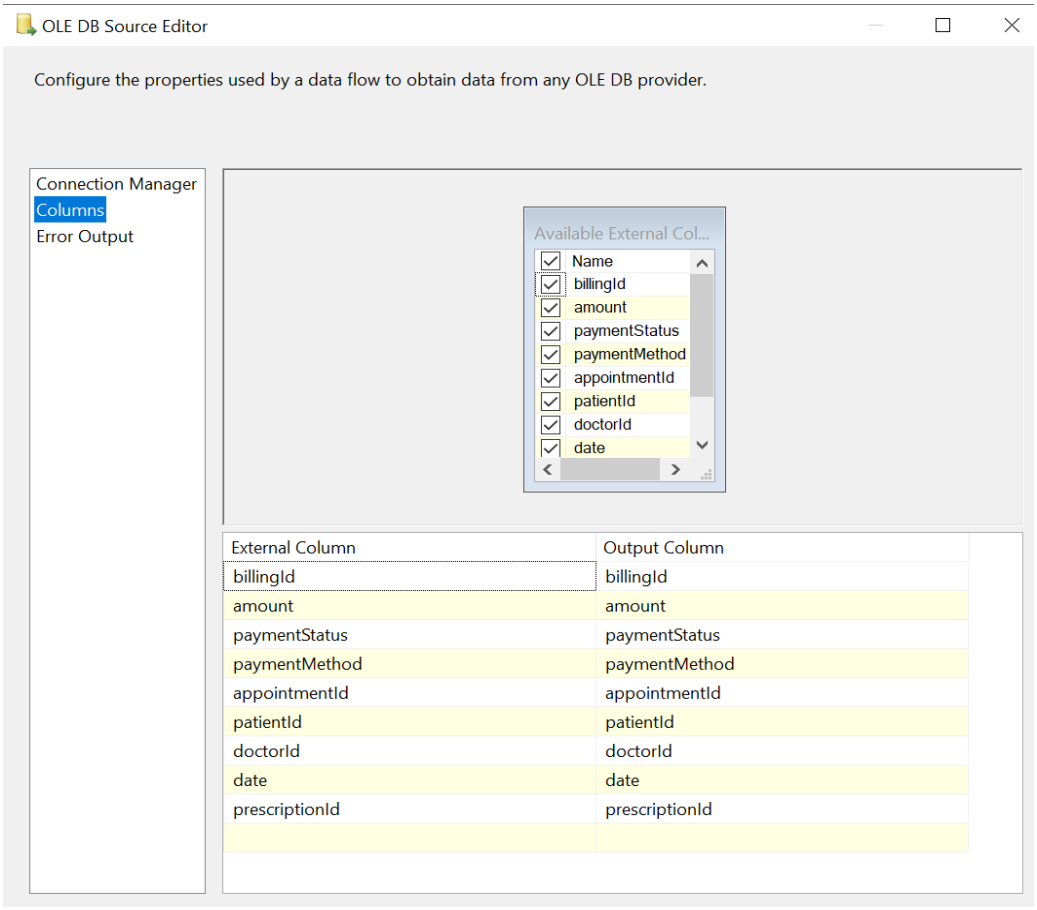
The Final Data Flow looked like this 👇



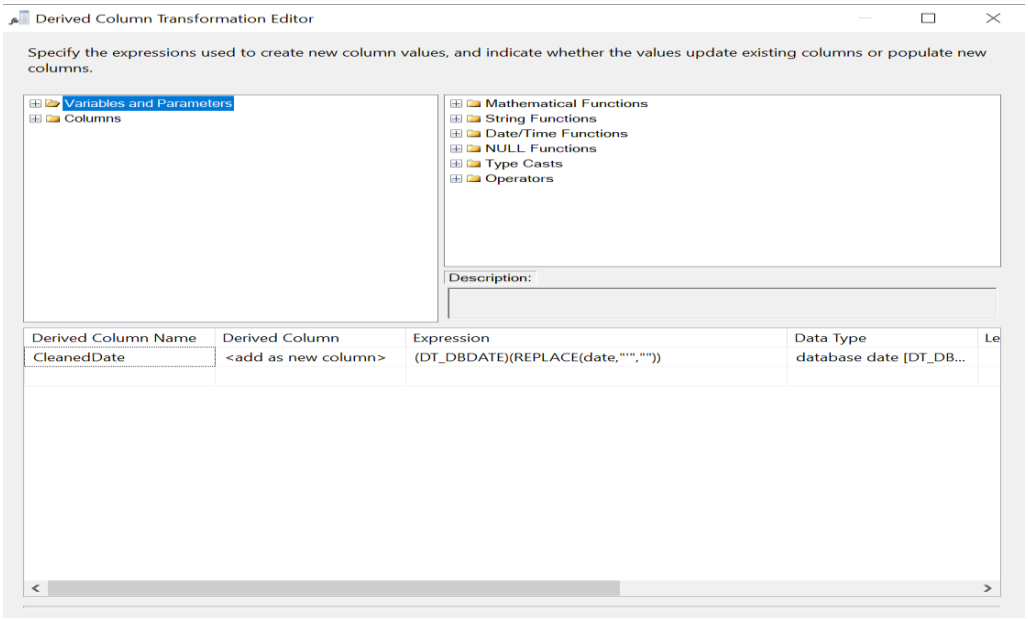
Then I started working on Loading FactBilling table with the relevant data.

Here first I added a OLE DB Source to JOIN Appointments table with Prescriptions table to feed the relevant data to the FactBilling.

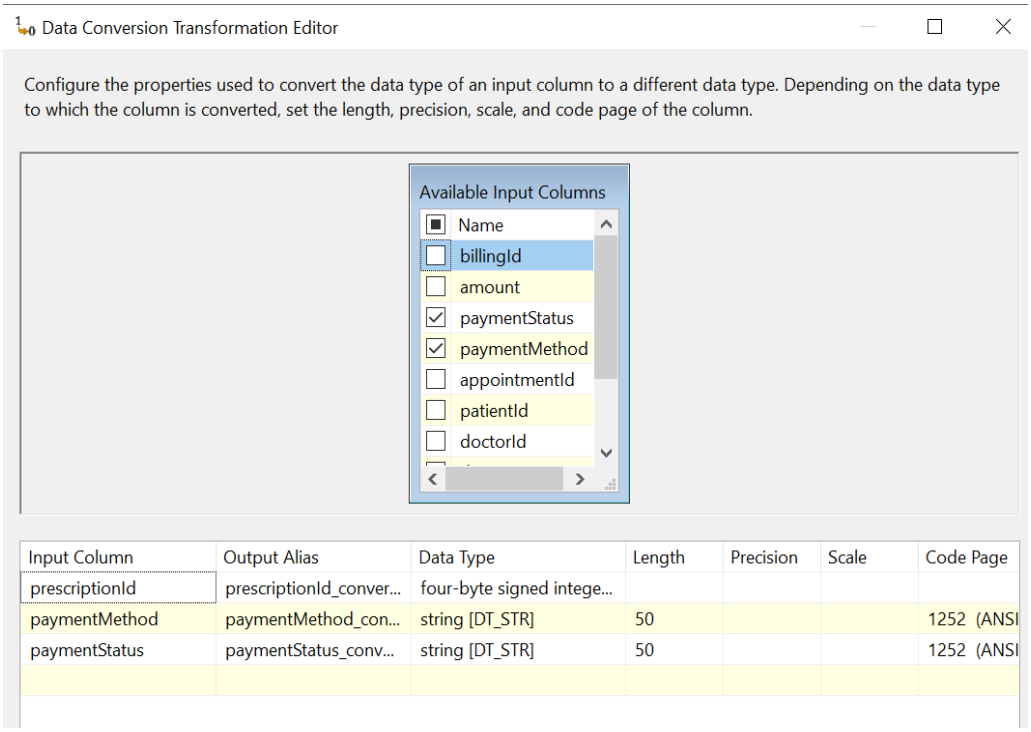

OLE DB Source Editor
Configure the properties used by a data flow to obtain data from any OLE DB provider.
Connection Manager
Columns
Error Output
Specify an OLE DB connection manager, a data source, or a data source view, and select the data access mode. If using the SQL command access mode, specify the SQL command either by typing the query or by using Query Builder.
OLE DB connection manager:
desktop-vbpgocs.HealthCare_Staging
New...
Data access mode:
SQL command
SQL command text:
SELECT
b.billingId,
b.amount,
b.paymentStatus,
b.paymentMethod,
a.appointmentId,
a.patientId,
a.doctorId,
a.date,
p.prescriptionId
FROM dbo.StgBilling b
JOIN dbo.StgAppointments a ON b.appointmentId =
Parameters...
Build Query...
Browse...
Parse Query
Preview...
OK
Cancel
Help



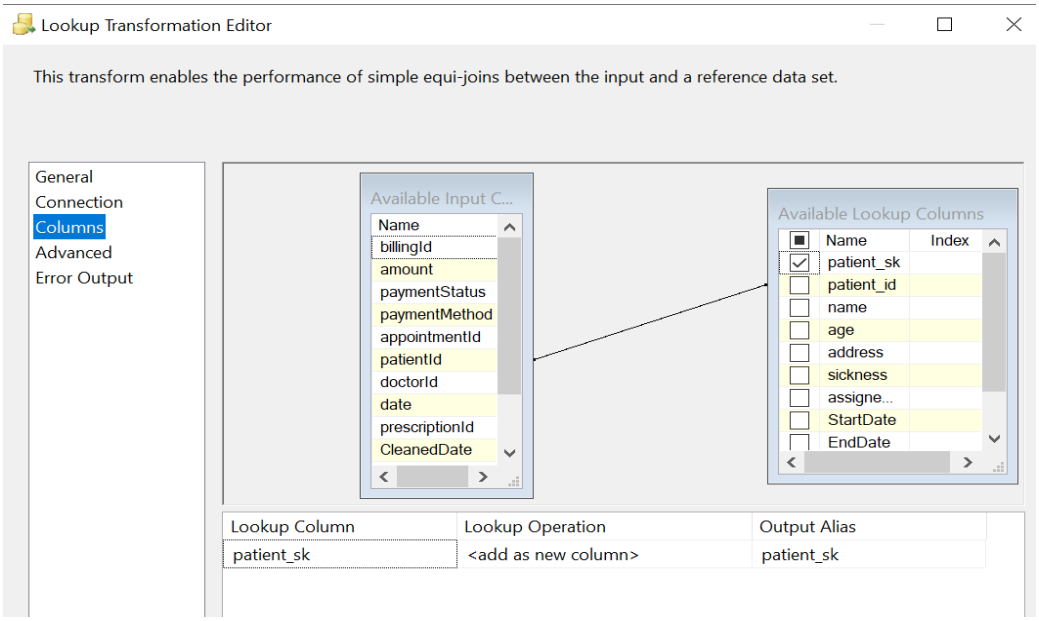
I had to convert the Date to a Database type, due to that added a Derived Column next.



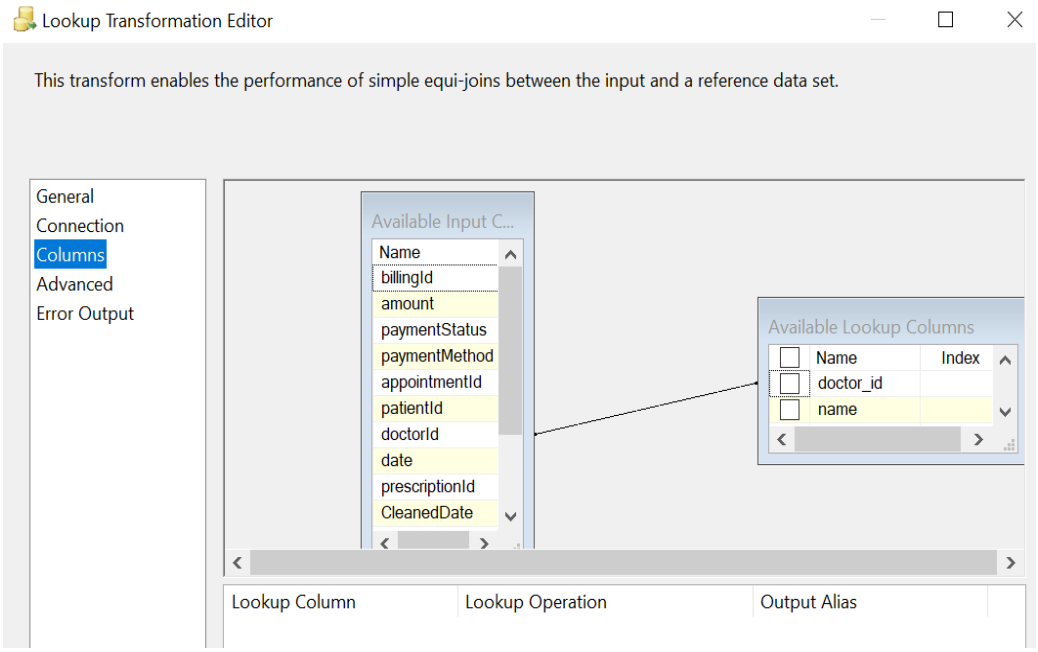
Then added a Data Conversion to cast the prescriptionId, Payment status, Payment method to Integer and String types respectively



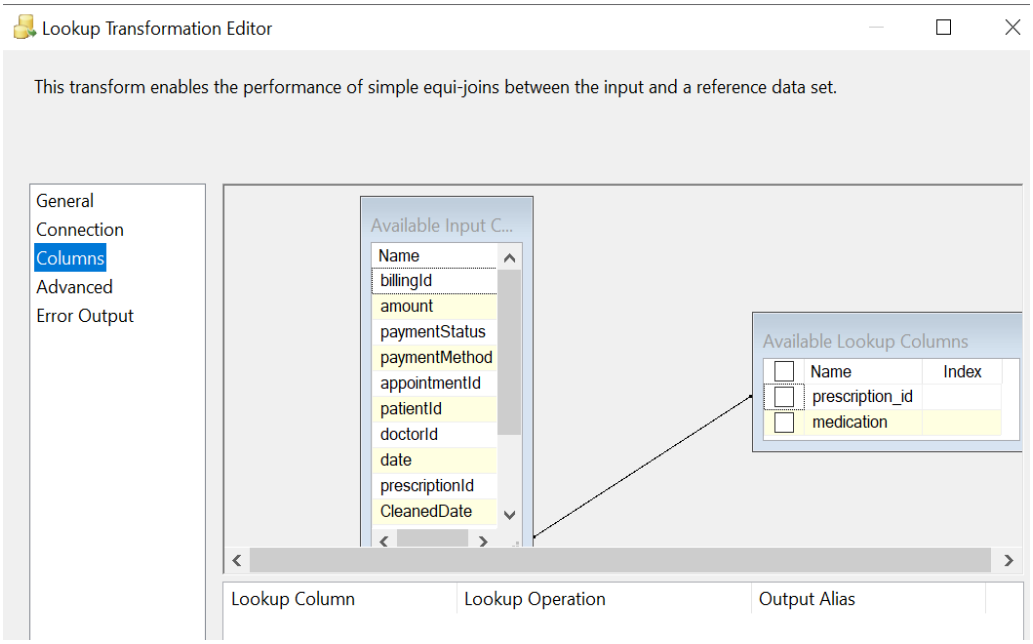
Then added a Lookup for DimPatients to check the Appointment related Patient exists in there. And returned the surrogate key.



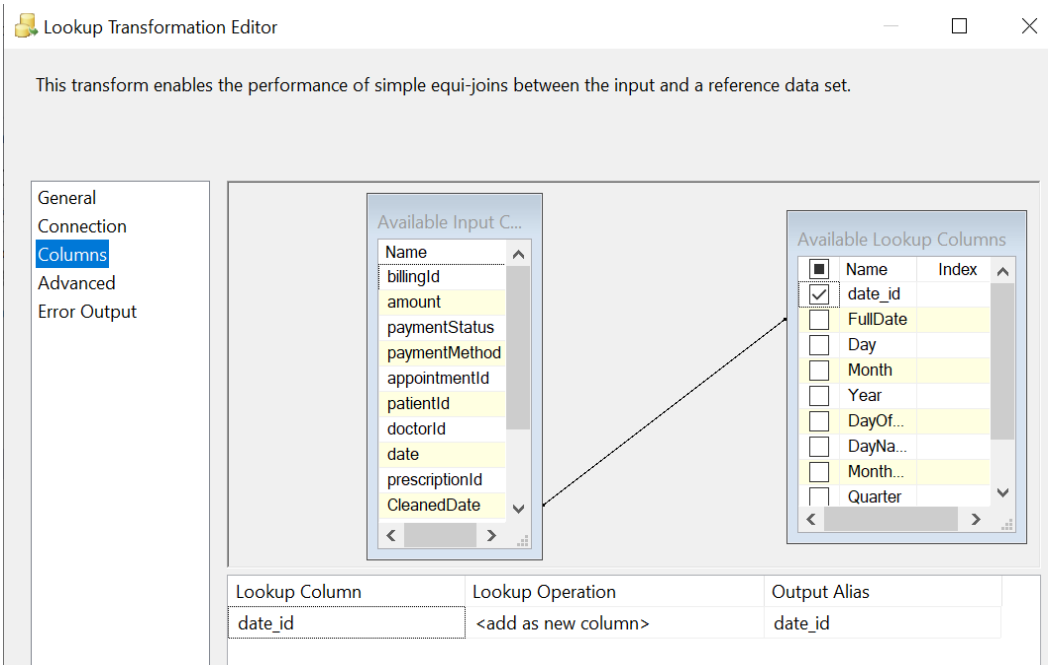
Then added a Lookup for DimPatients to check whether the Doctor exists



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

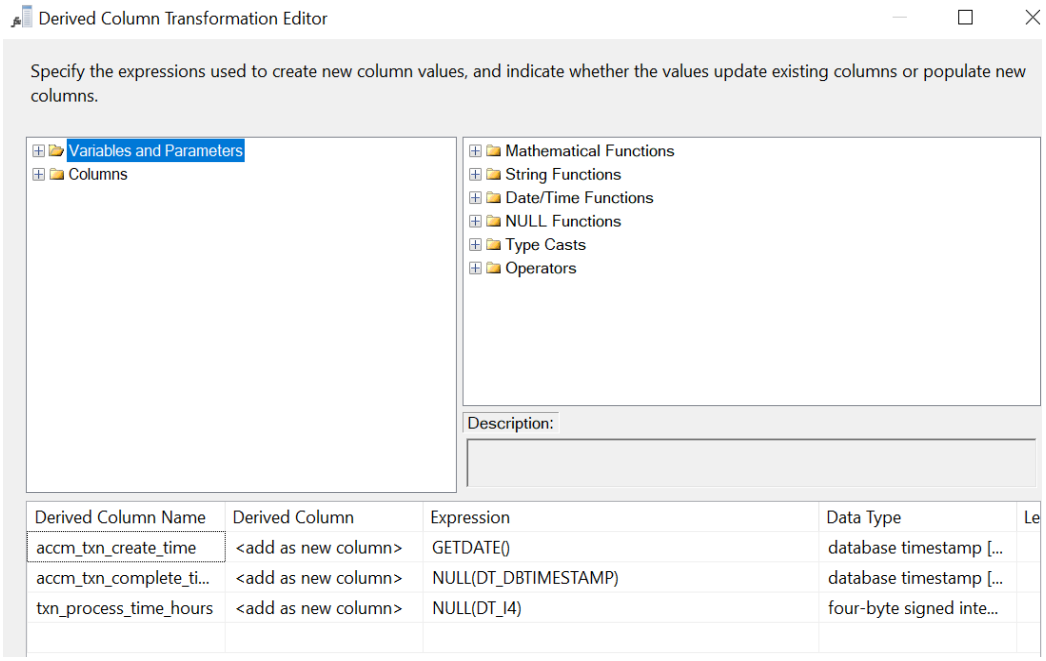
ALTER TABLE dbo.FactBilling

ADD

accm_txn_create_time DATETIME,

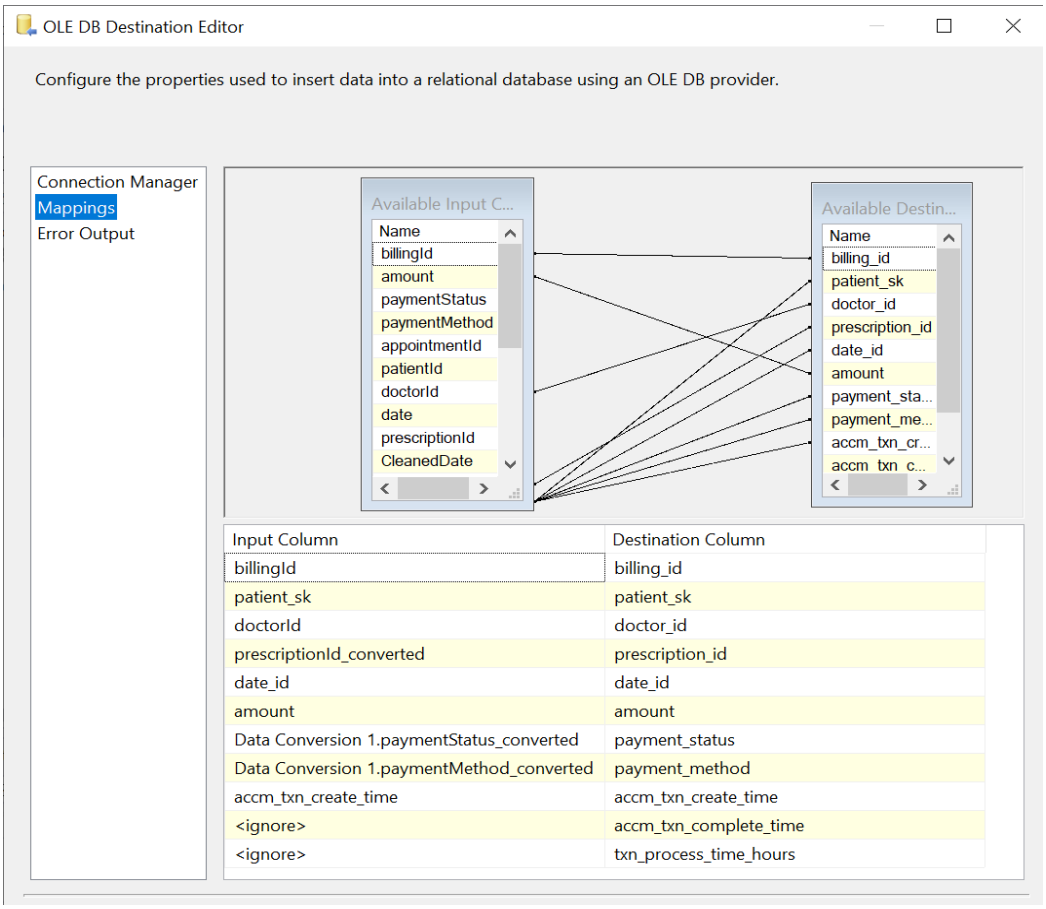
accm_txn_complete_time DATETIME,

txn_process_time_hours INT;

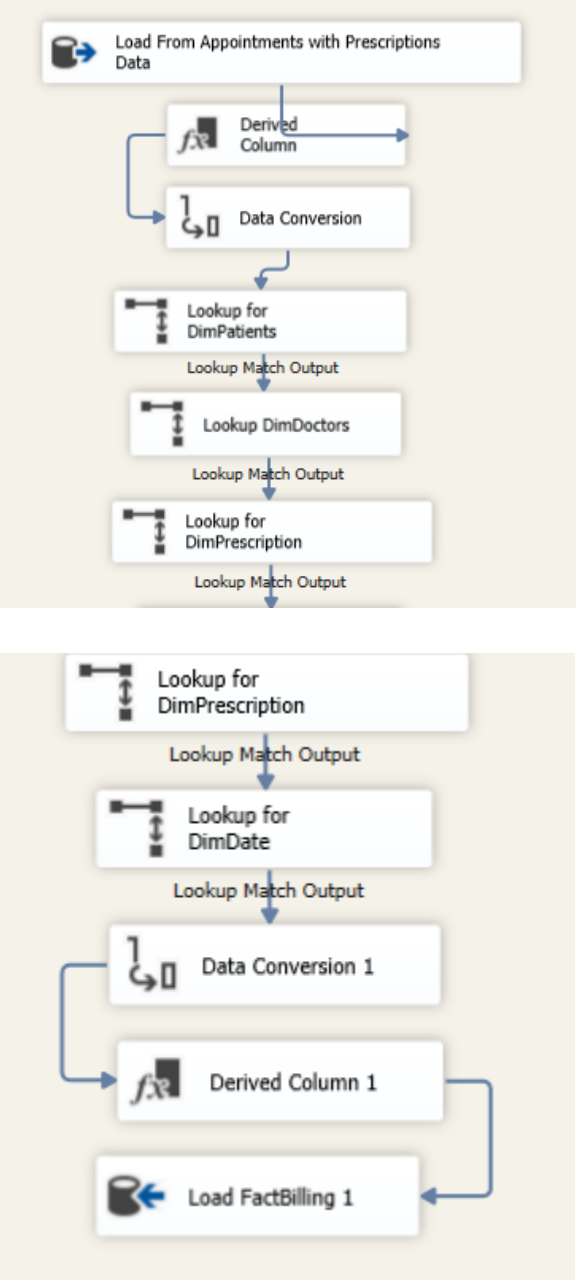


Here the data of accm_txn_create_time was filled as other were to be filled by a separate SSIS package.

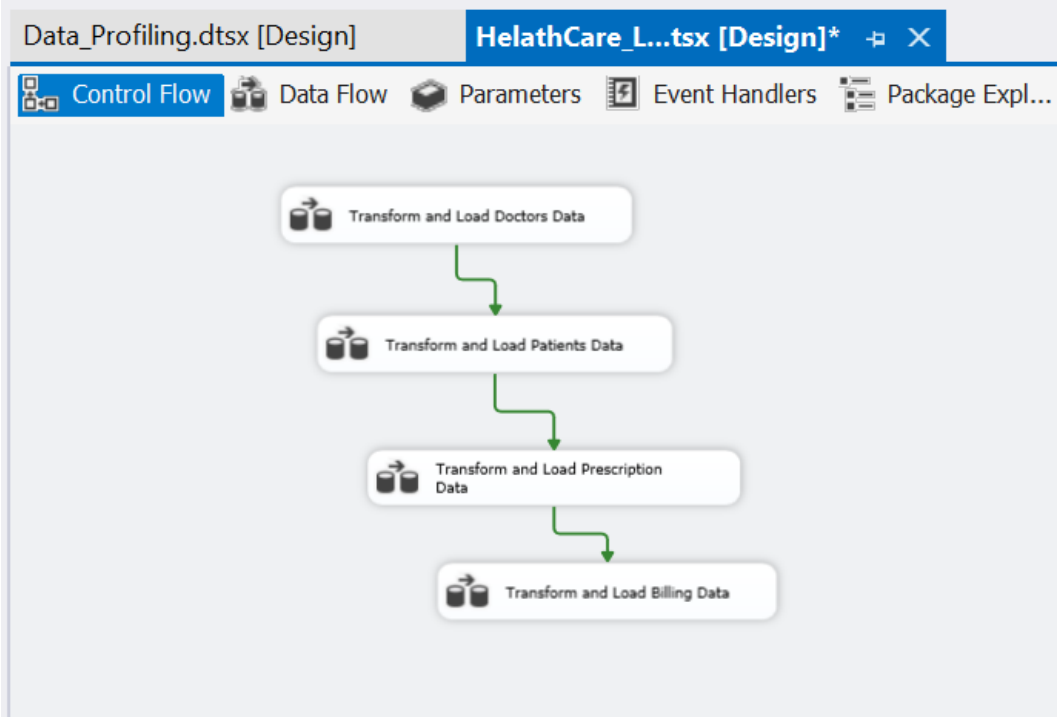
As the final component added a OLE DB Destination pointing FactBilling while matching the columns as needed.



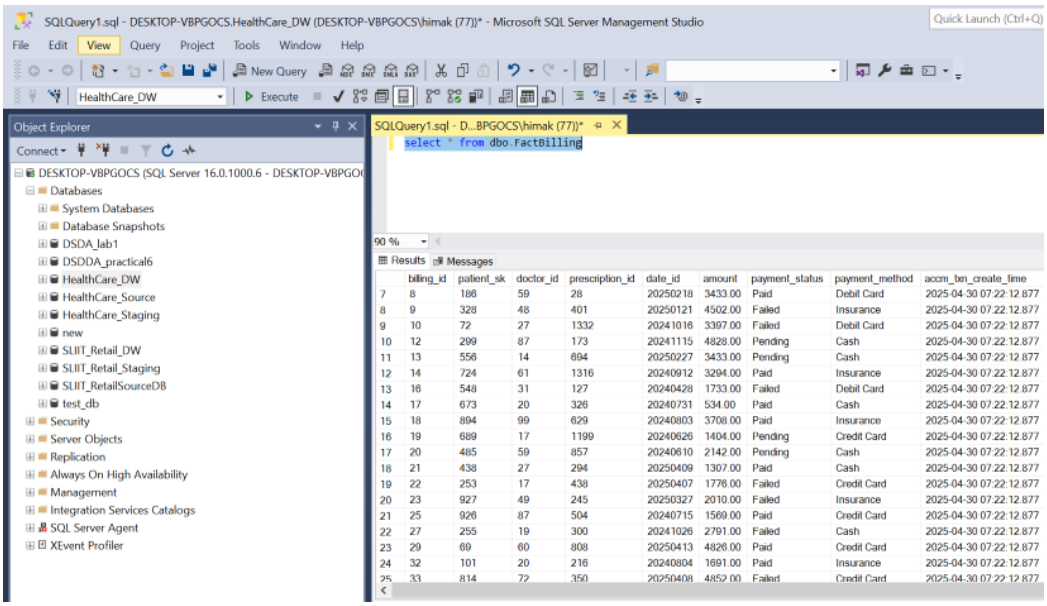
The Final Data Flow task looked like this 👇



Final Control Flow view 👇



Then the package was executed and the results could be seen in the HealthCare_DW Database tables.

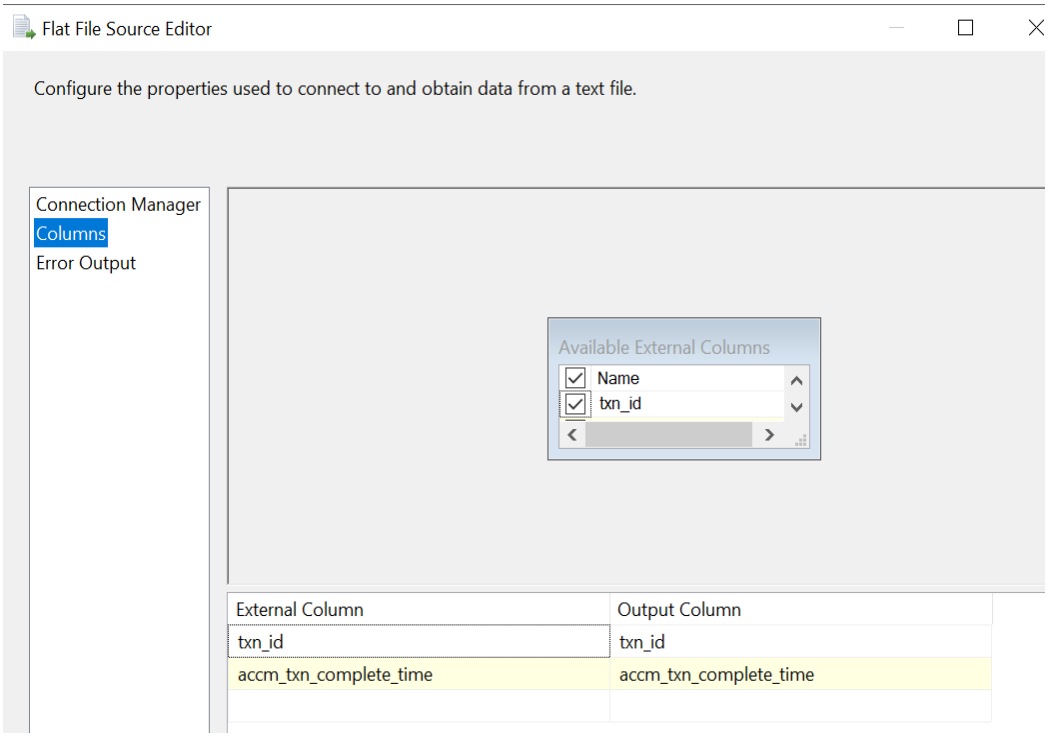


Accumulating Fact Table

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.



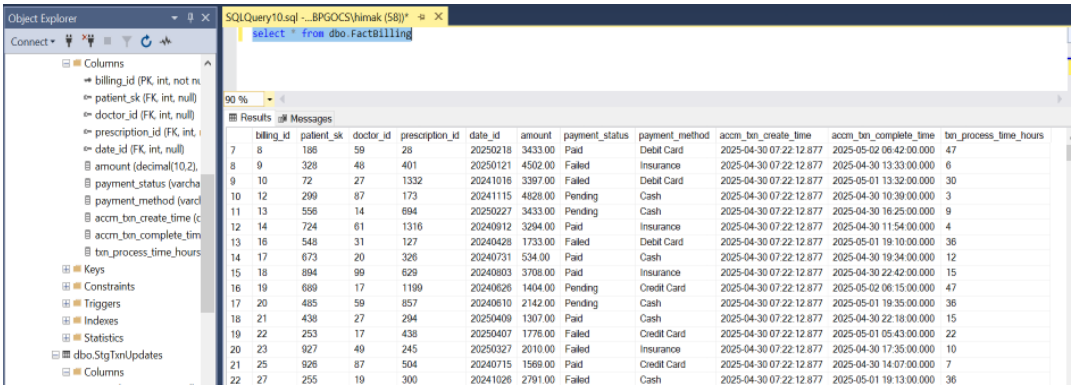
Then connected it to an OLE DB Destinationa to load this as a table in Data Warehouse DB. In Event Handler tab added a PostExecute SQL Command Task to update the FactBilling table as needed used below SQL statement in there.

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 ***