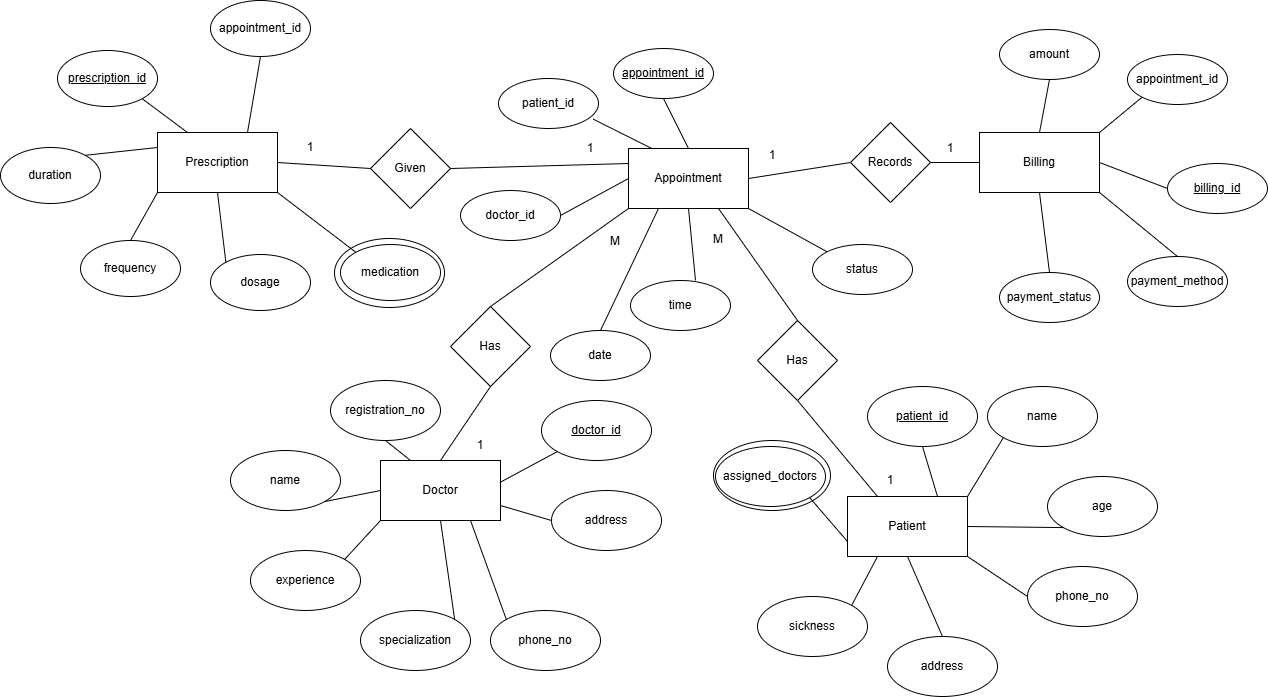
Data Warehousing and Business Intelligence

Assignment 01

1. 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.



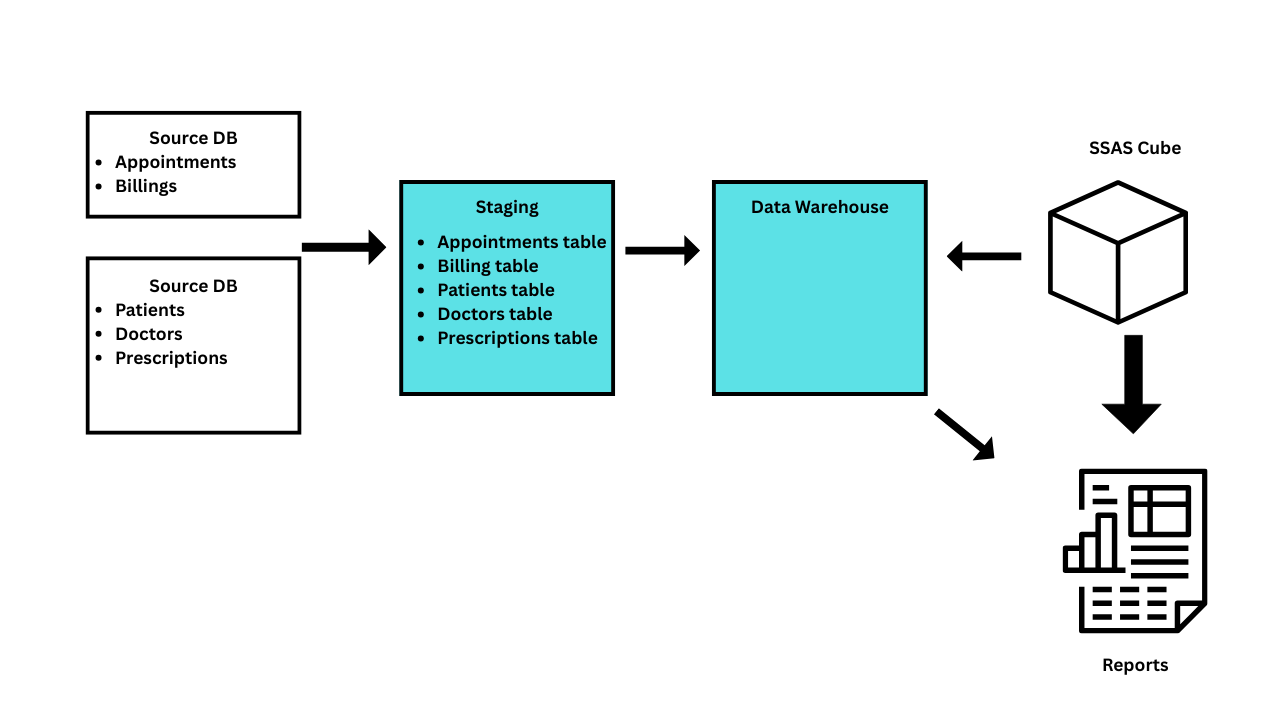
The patient’s sickness is recorded when he or she is registered then the system assigns a matching doctor related to that patient. Each scene happens in the appointment like prescriptions, patient and doctor details are recorded in the appointment entity.

A patient books an appointment where the appointment table records the doctor assigned along with the patient details. After the appointment it records the appointment data along with the prescriptions given. The billing entity records all the appointments and with their billing amounts.  
  
Here 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

1. Solution Architecture

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



Here the first two depicts the data sources available in different formats as data tables in SQL Database and other exported CSV files

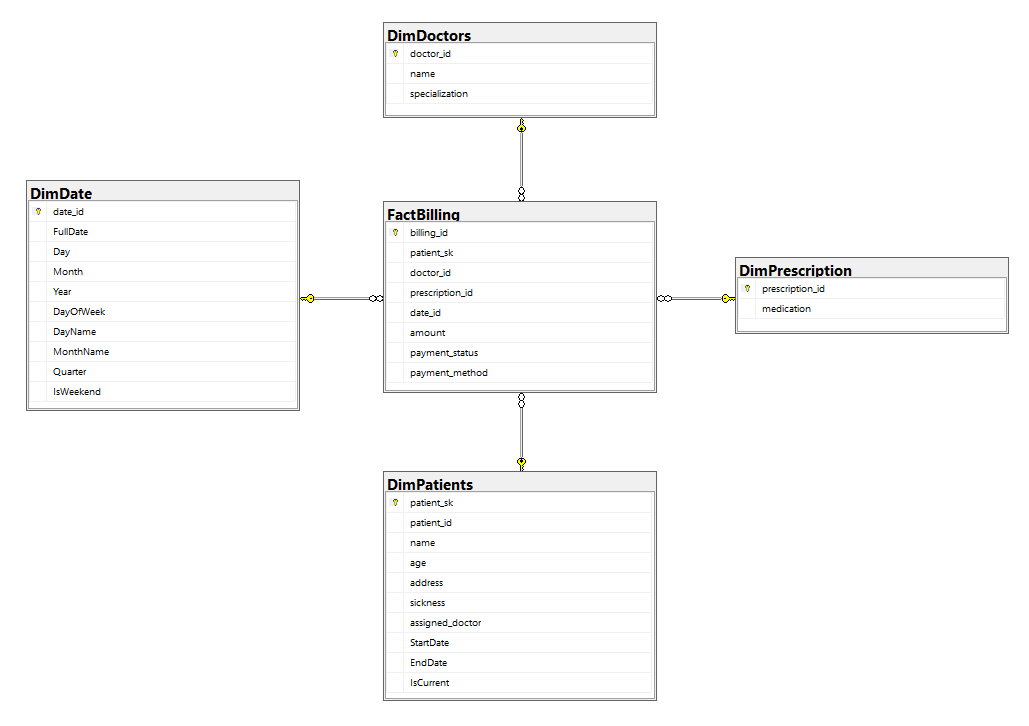
Then the data is loaded into the Staging Database where the raw data is cleansed and transformed before loading to the warehouse (Extract and Transform)

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

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

1. Data warehouse design and development

First, I created a DB named HealthCare\_DW and 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.



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: dimensions. First, I choose the data which will be transferred from staging to warehouse for further processes in each table for better business insight.

*-- 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

);

DimPatient consists of patients’ details which define the data in Fact table: Billing. Here I selected only patient\_id, name, age, address, sickness and assigned\_doctor as the forwarding attributes. 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 DimDoctor 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*

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)

);

FactBilling is the important Fact table which have foreign keys from patient, doctor, prescription and date tables. It holds the quantifiable values such as amount and measurable values such as payment methods.

*-- Fill 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);

DimDate table helps to store dates in a separate location so it will be easy to perform data-based analytics.

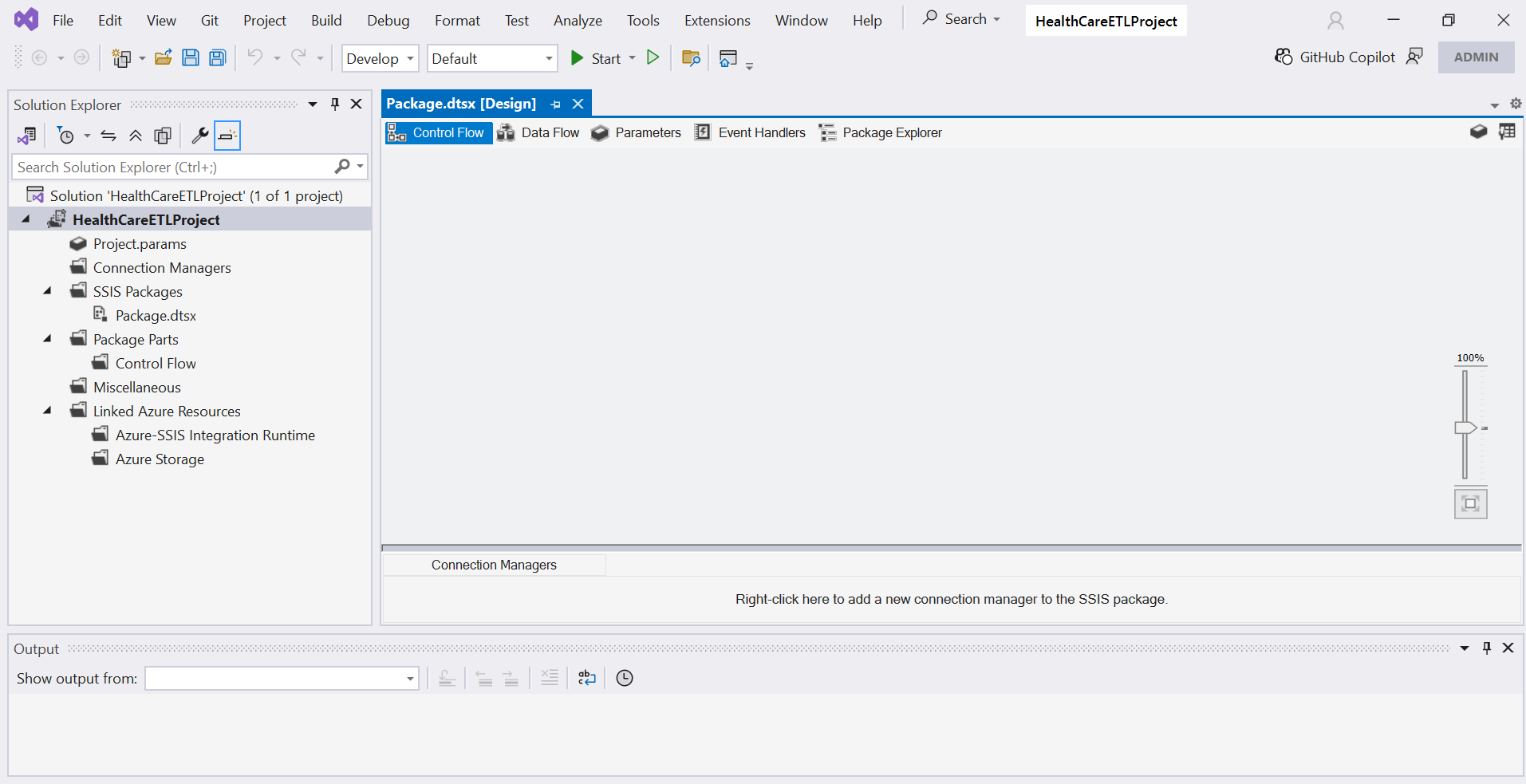
1. ETL development

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;  
  
  
1. Created a project name HealthCareETLProject



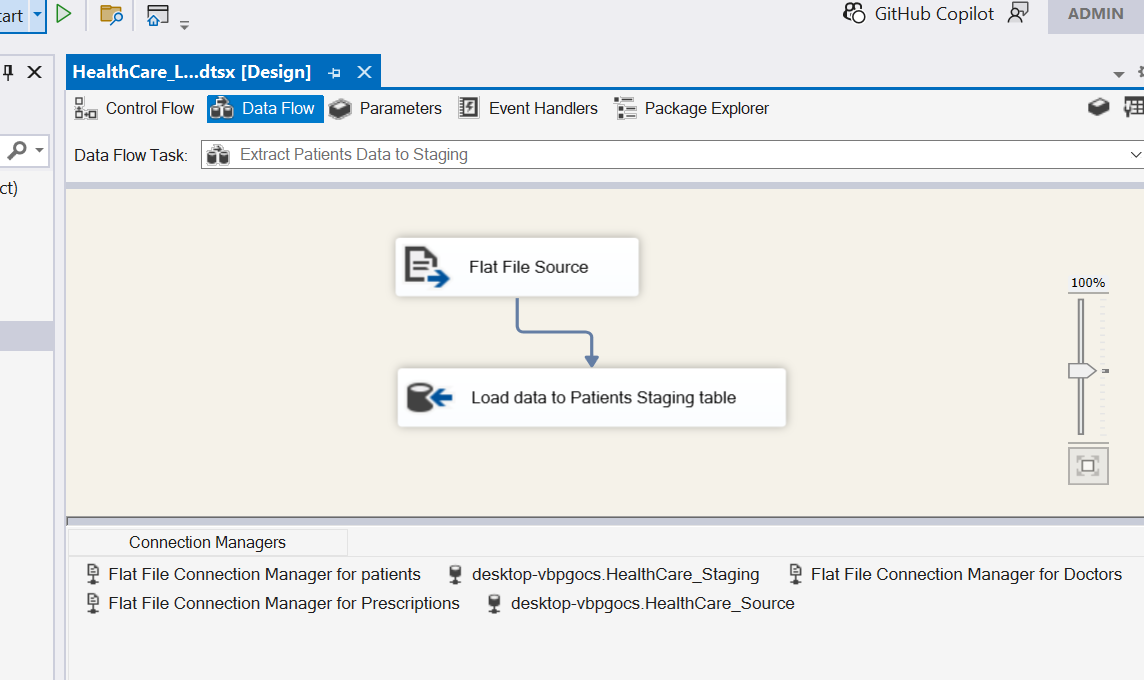
2. Data Extraction from sources

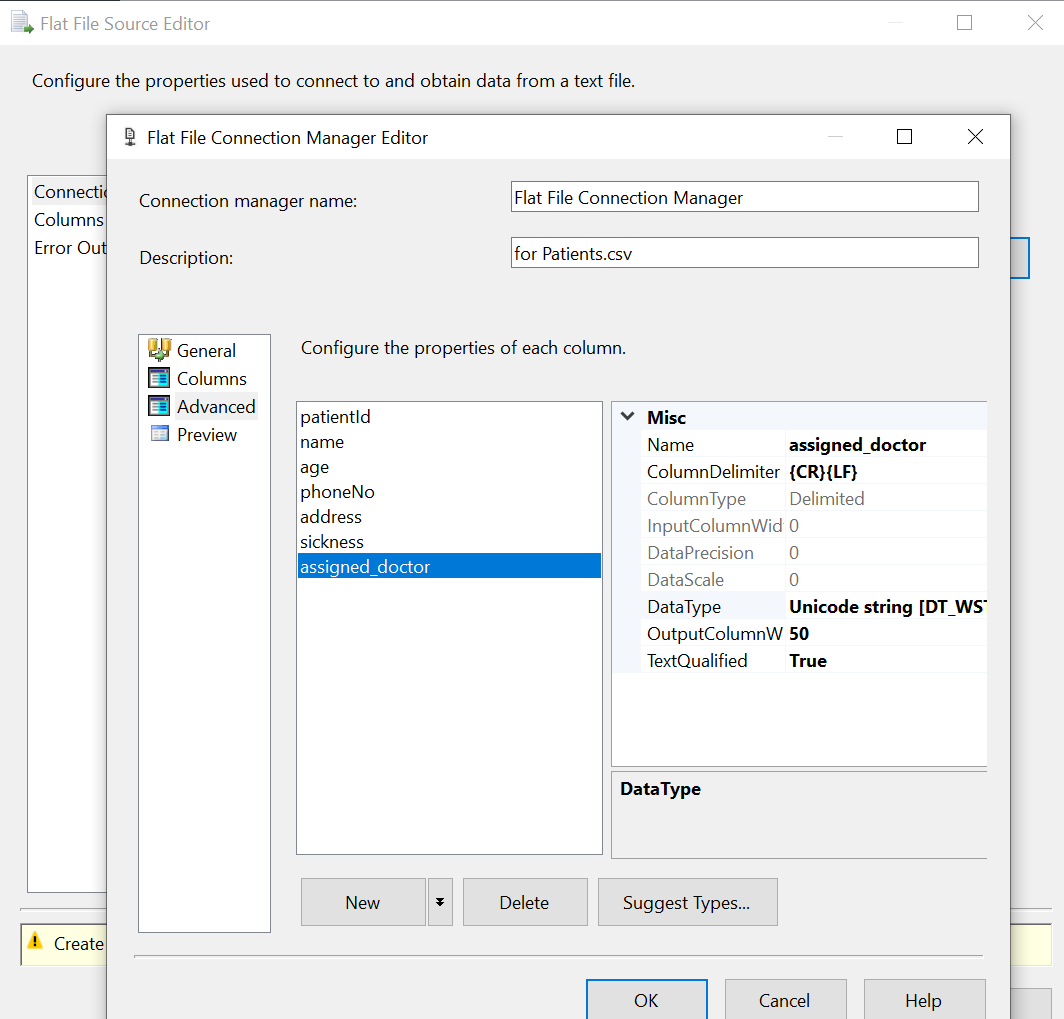
Considering the organization of data in this scenario, I follow the list as

* 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.





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

A screenshot of a computer

AI-generated content may be incorrect.  
  
After doing the things for patients and doctors, then carried out the process for appointments table which is an OLE DB source in HealthCare\_Source  
  
A screenshot of a computer

AI-generated content may be incorrect.

Then added the necessary configuration and executed the same source and destination concept along with PreExecute event handler for this too.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

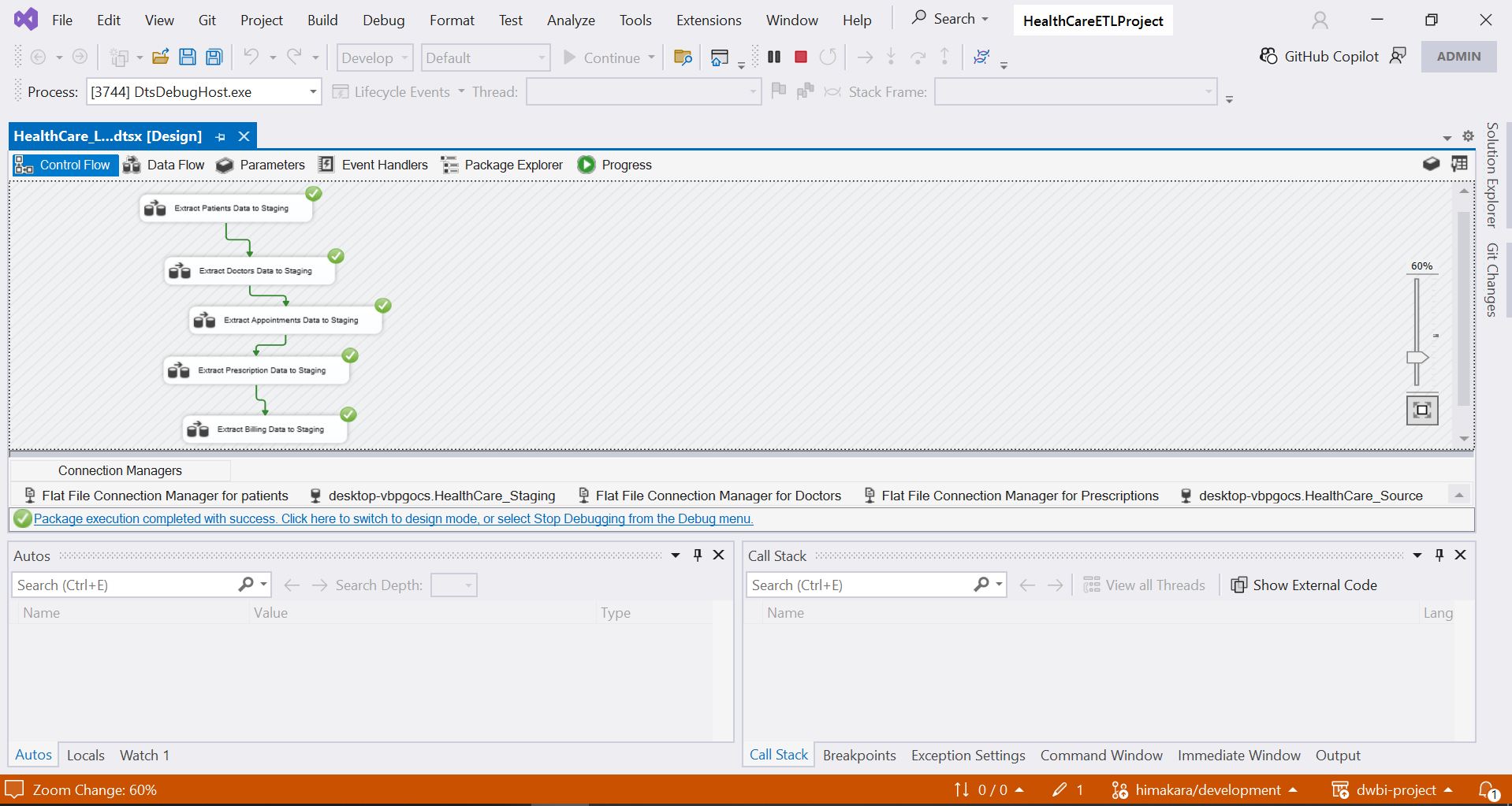
AI-generated content may be incorrect.  
  
A screenshot of a computer

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

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

AI-generated content may be incorrect.  
  
Likewise, the same procedure was carried out for Prescription CSV and Billing Table respectively.  
  
A diagram of a medical procedure

AI-generated content may be incorrect.

Executed the package after building the solution  
  
Now the Data can be observed in Staging tables.

Data Profiling part to understand what transformations should be performed on the data.

A screenshot of a computer

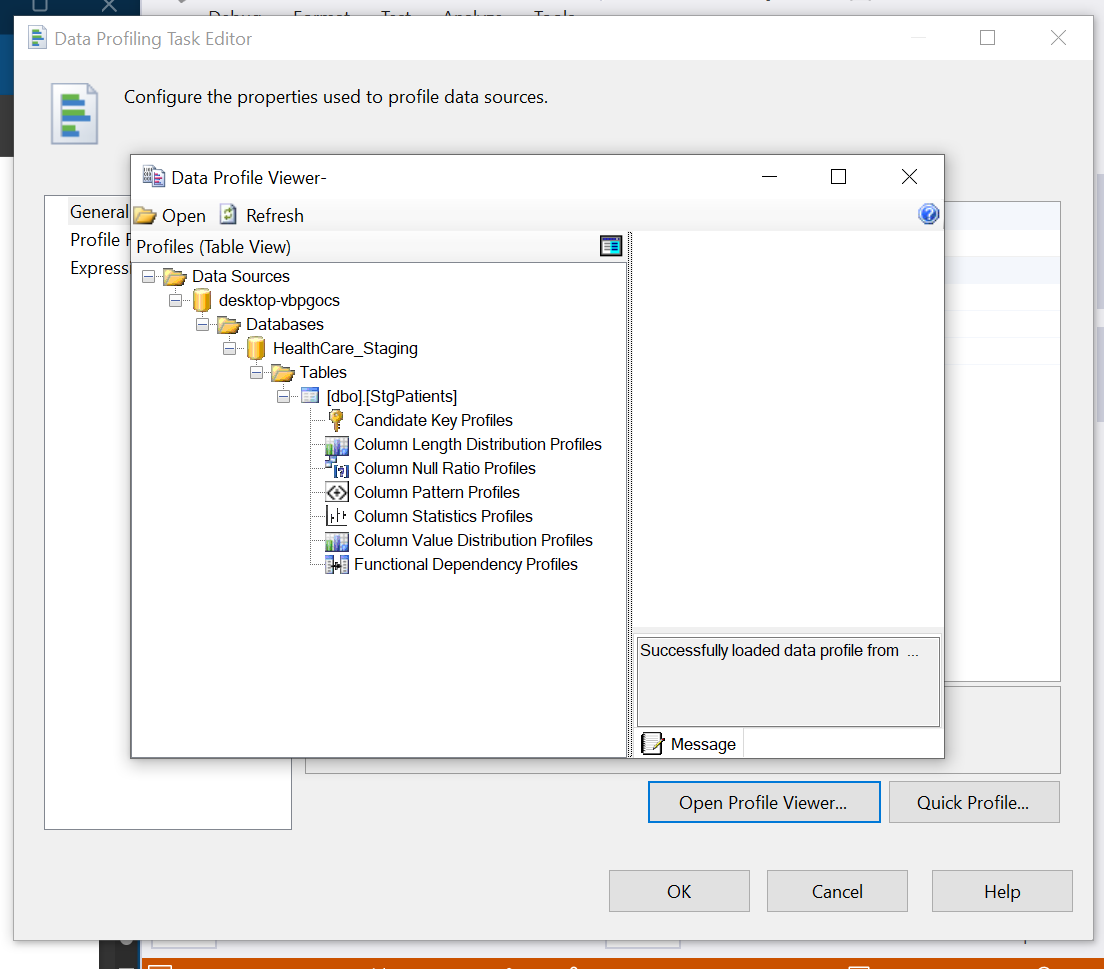
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



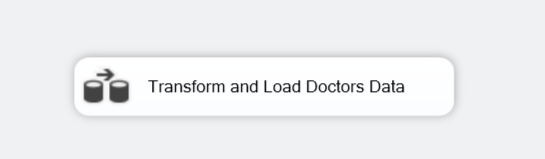
Profiled every table to understand the transformations

A screenshot of a computer

AI-generated content may be incorrect.

Some observable exported files appeared in the given location  
A screenshot of a computer screen

AI-generated content may be incorrect.

Then I created new SSIS package HealthCare\_Load\_DW.dtsx  
  
Here my main idea is to Load data from Staging Database to Data warehousing.  
First up, Transform and Load from StgDoctors to DimDoctors,  
  
I created a Data Flow task  
  
  
  
Inside it   
  
A screenshot of a computer screen

AI-generated content may be incorrect.  
I created OLEDB Source which points the table in Staging Database and OLEDB Destination which points table in Warehouse Database

1. Step 6

Here I 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;

Then I added a Derived column tool to modify the FactBilling and configured it like this  
  
A screenshot of a computer

AI-generated content may be incorrect.

Configured new changes in OLE DB Destination too  
  
A screenshot of a computer

AI-generated content may be incorrect.

Then I executed the dtsx  
Then I could see the Fact Table is populated correctly

Step 6  
  
First I created a table with txn\_id and txn\_completed\_time using a python script  
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)

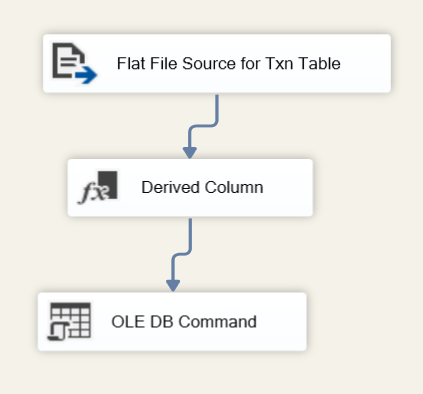
df.to\_csv("CSVs/txn.csv", index=False)

Then used for the step  
   
First I added an OLE DB flat source to retrieve the txn.CSV and a derived column of it to take a copy of txn\_complete\_time.  
Then in an OLE DB command task,  
 added this SQL line  
  
UPDATE dbo.FactBilling

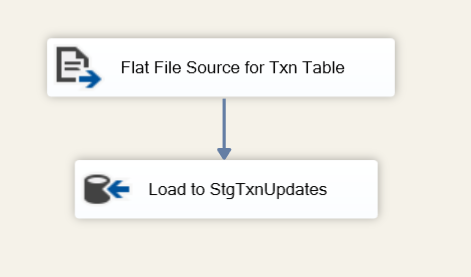
SET

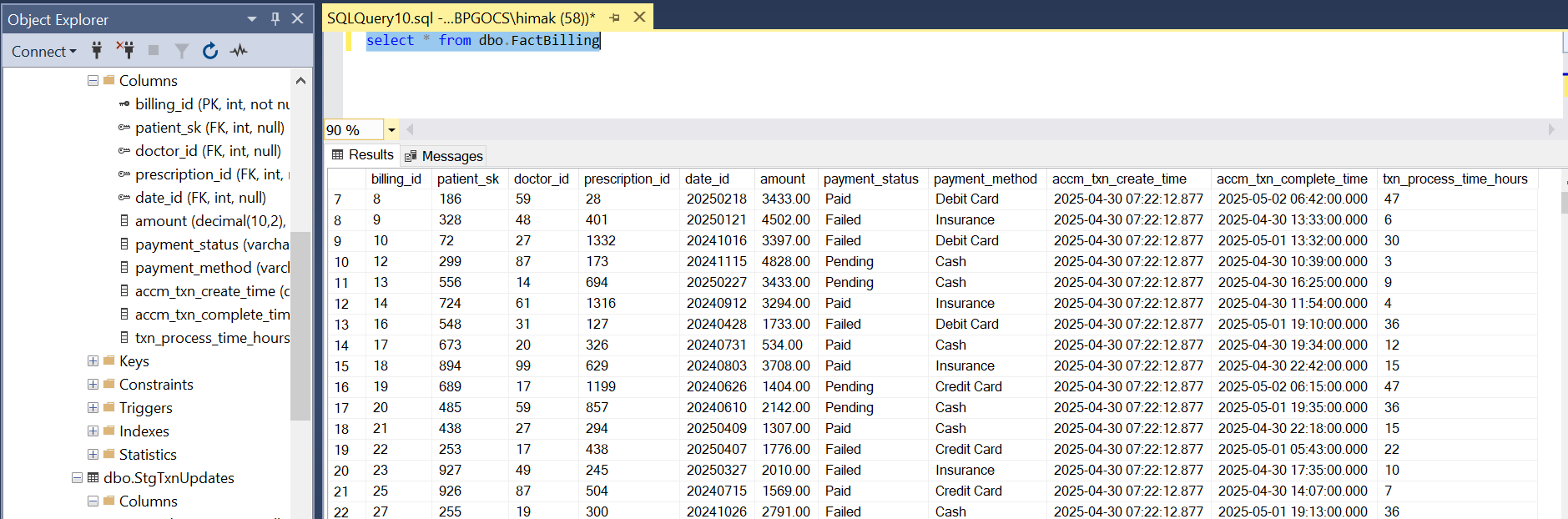
accm\_txn\_complete\_time = ?,

txn\_process\_time\_hours = DATEDIFF(hour, accm\_txn\_create\_time, ?)

WHERE billing\_id = ?;  
  
here for three parameters I passed accm\_txn\_complete\_time, accm\_txn\_complete\_time\_copy, txn\_id to correcty map and update the fact Billing table.  
  
The final Data Flow looks like this  
  


Then executed the new dtsx package.  
  
But I noticed some of the records haven’t updated correctly  
Then I added a different logic to update these flat data source to different table in Database and in Data flow , Event handling, PostExecute scenario update the FactBilling comparing the loaded table as needed.

It changed to this  
  
  
  
Here It loads the data to a new table and in a PreExecute command  
  
A white rectangular object with black text

AI-generated content may be incorrect.  
  
I added command to update the FactBilling table comparing txn\_id with billingId. Then the Data loaded to the FactBilling table correctly.  
  


1. Wdw
2. Wdwd
3. Wdwd
4. wd