

MILESTONE-7

DONE BY

**ARAVIND SWAMY
AND UMA
MAHESHWARI
DEIVASIGAMANI**

Table of Contents:

- 1. KPIs and
Dashboards**
- 2. Complete
Documentation
of both projects**
- 3. Presentation PPT**

Dashboards & KPIs:

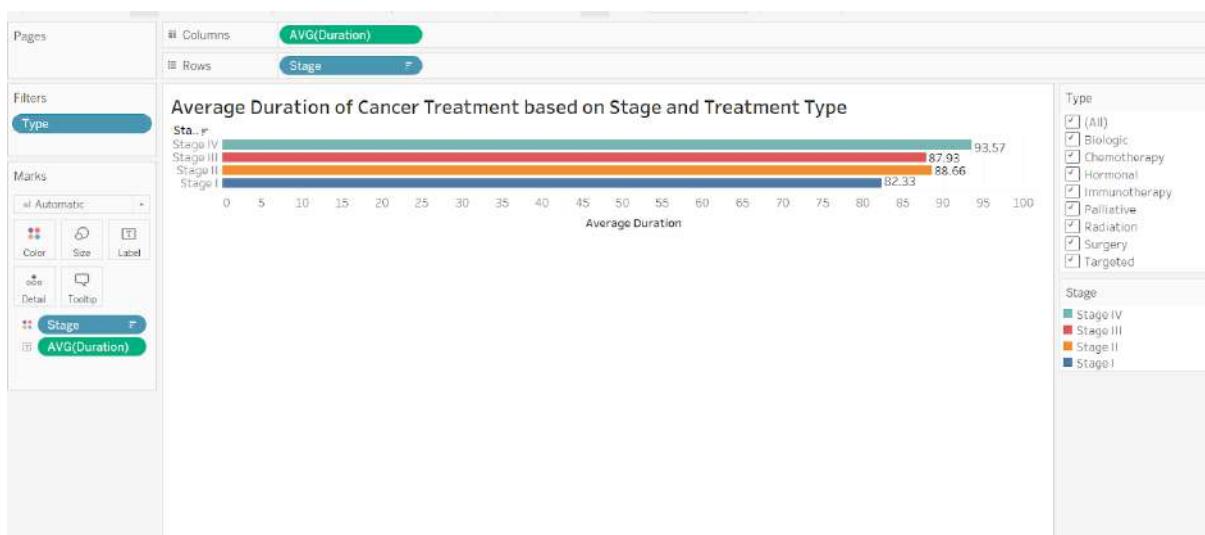
We have chosen the same project idea for both on-prem and cloud projects. In the on-prem project we have 3 fact tables : Treatment Fact Table, Supplies Fact Table and Reviews Fact Table.

In the cloud project we removed the supplies part of it. And we have only two fact tables in it: Treatment Fact Table and Reviews Fact Table.

We have **3 dashboards** for this project, one each for each of the fact table.i.e., one for treatment KPIs, one for reviews KPIs and one for supplies KPIs. In addition to KPIs mentioned in OLAP operations we have also done some extra KPIs

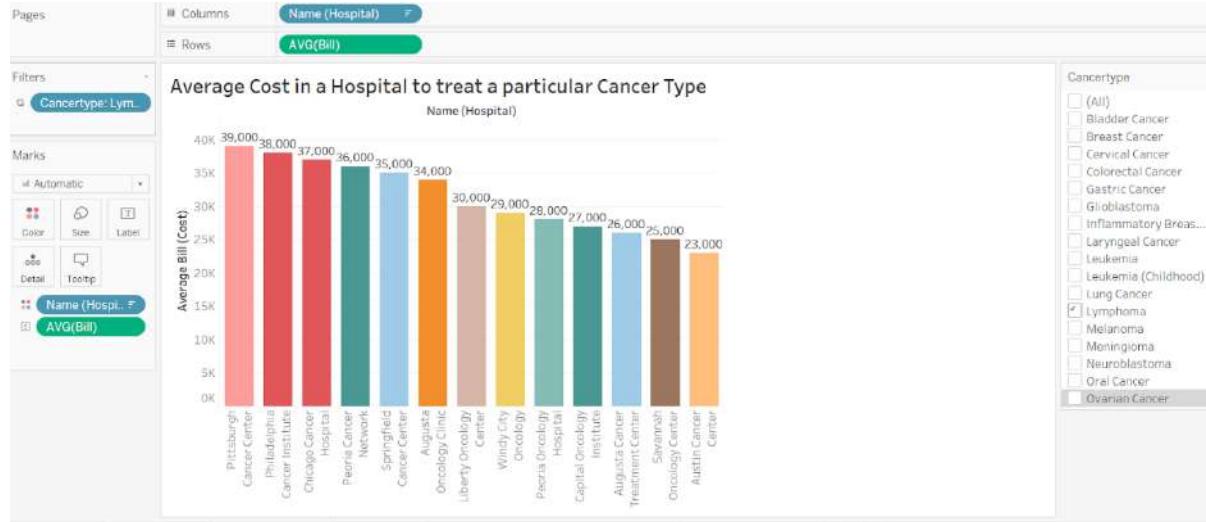
Treatment KPIs & Dashboard:

1. 1.Average Duration of Cancer Treatment based on Stage and Treatment Type:



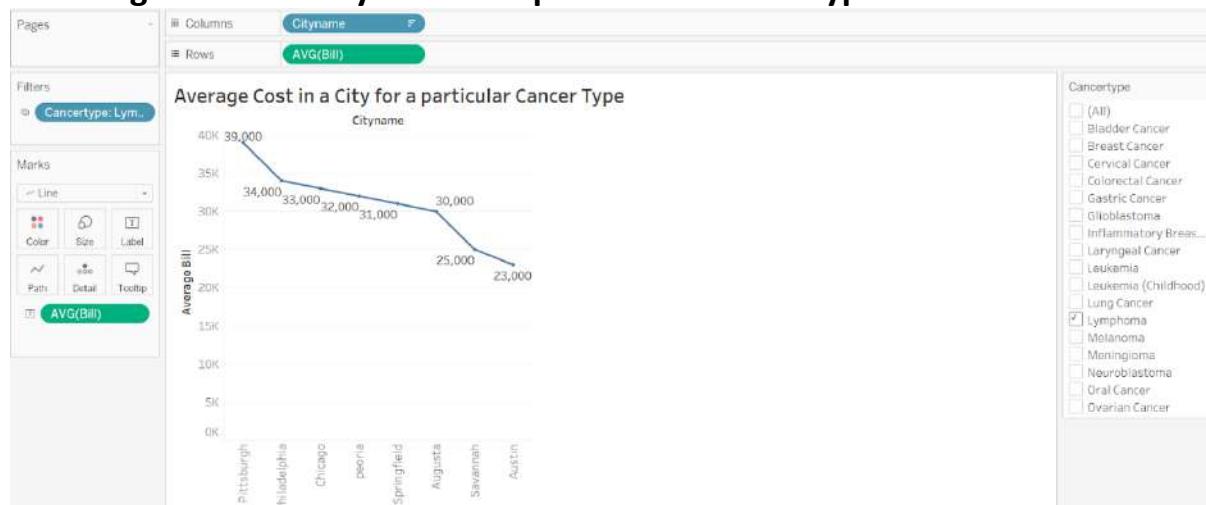
The above graph shows the average duration to treat each stage of cancer using a particular treatment type.(This follows from the OLAP operation mentioned: Treatment duration based on cancer stage (measure involved:Duration))

2. Average Cost in a Hospital to treat a particular Cancer Type



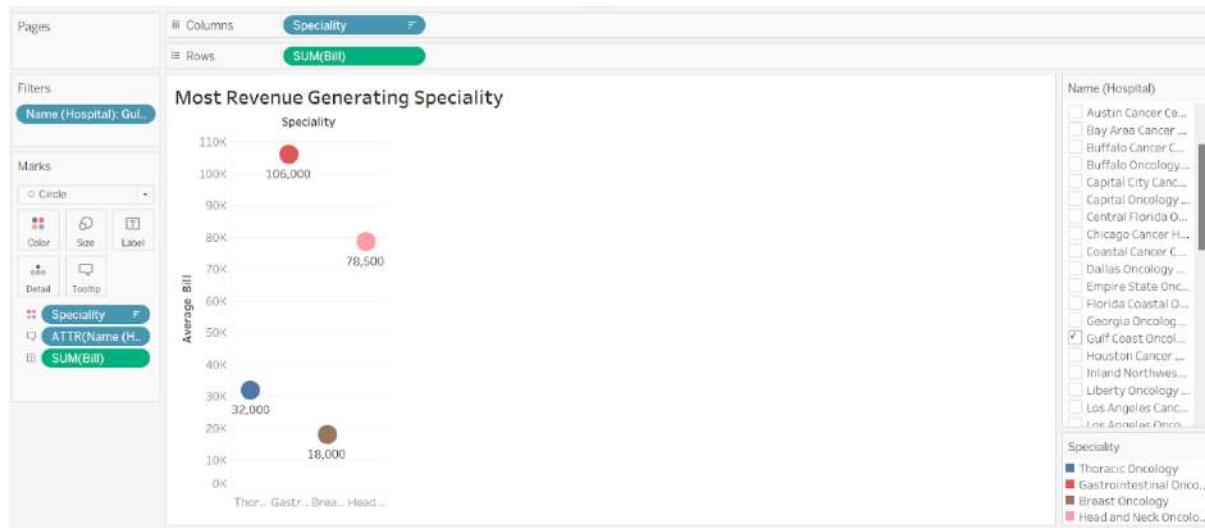
The above graph shows the average cost to treat a particular cancer in a hospital. This could help the patients to take an informed decision and choose a hospital. (This follows from the OLAP operation mentioned: Average cost in hospital to treat a particular cancer type (measure involved:Bill))

3. Average Cost in a city to treat a particular Cancer Type



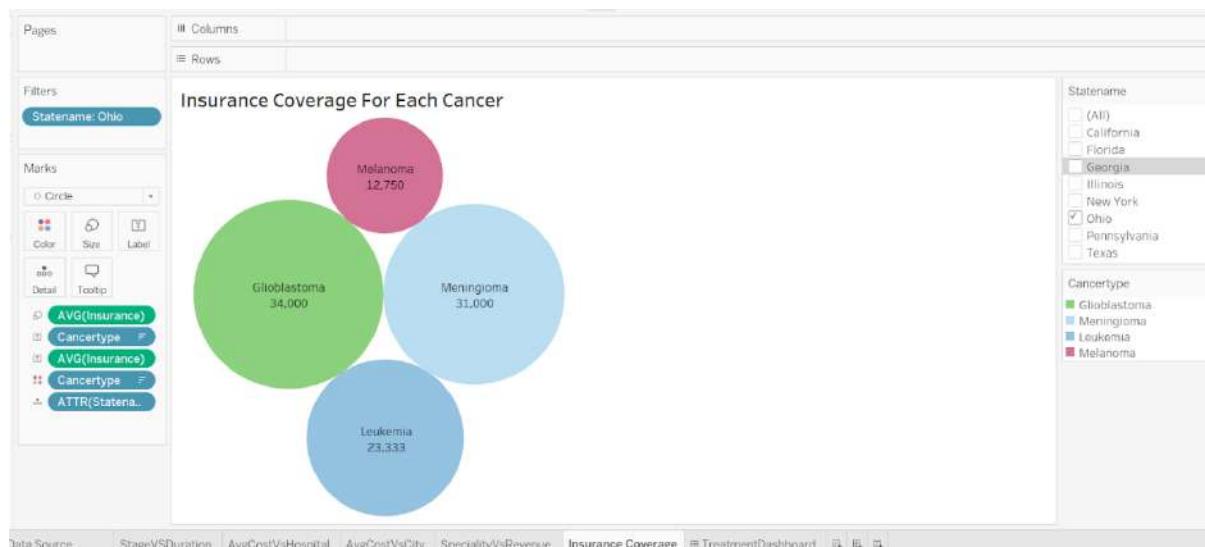
The above graph shows the average cost to treat a particular cancer in a city. This could help the patients to take an informed decision and choose a particular city. (This follows from the OLAP operation mentioned: Average cost in city to treat a particular cancer type(measure involved:Bill))

4. Most Revenue Generating Speciality



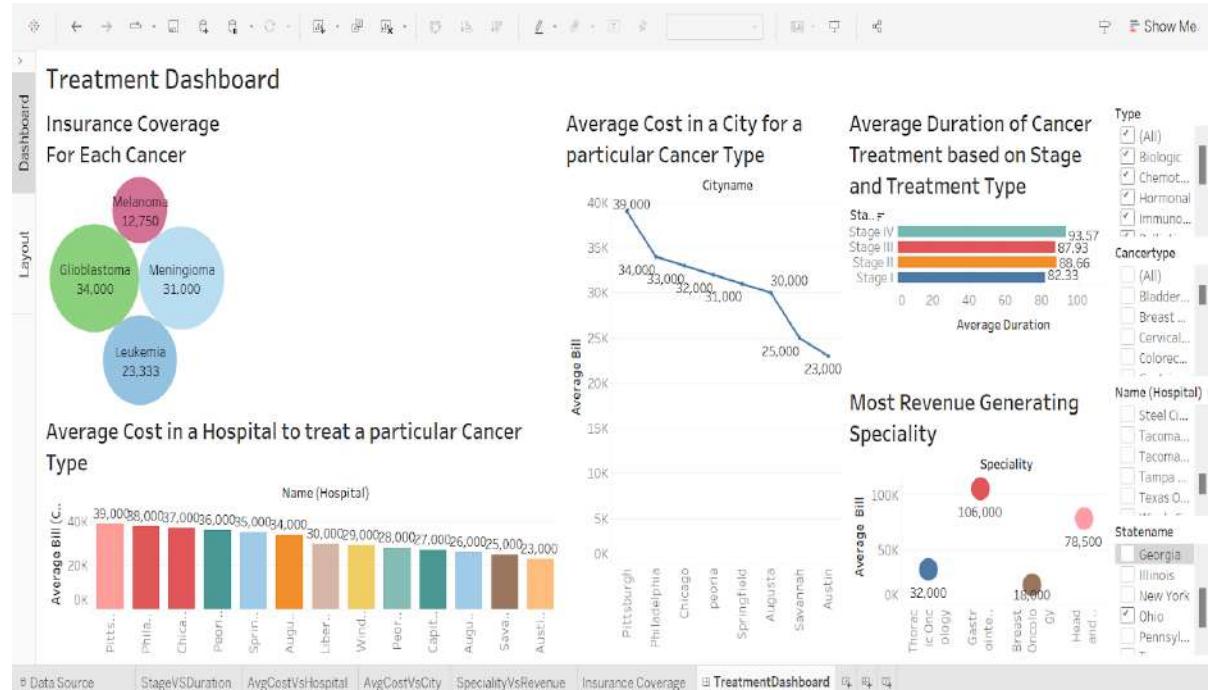
The above graph shows which speciality generates the most revenue for a chosen hospital.(This follows from the OLAP operation mentioned: Which specialty of doctors generates the most revenue(measure involved: Bill))

5. Insurance Coverage For a Cancer Type in a chosen state:



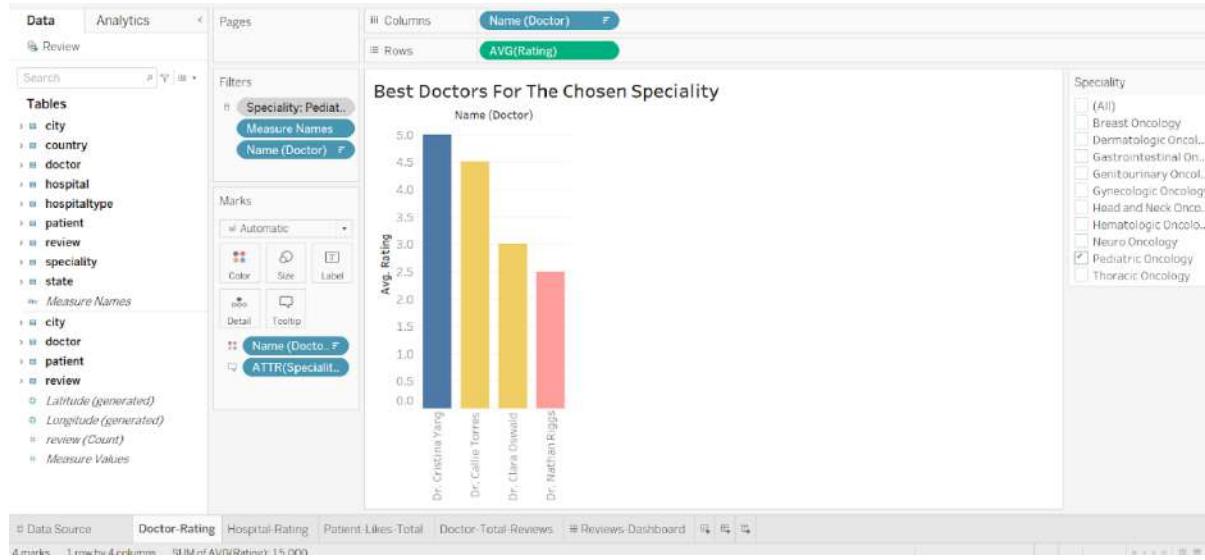
The above graph shows the average insurance coverage for a cancer type in a chosen state. Using this information the patients can choose a state to get treated. (This follows from the OLAP operation mentioned: Sort by highest average insurance coverage based on cancer type(measure involved:insurance))

Treatment Dashboard:



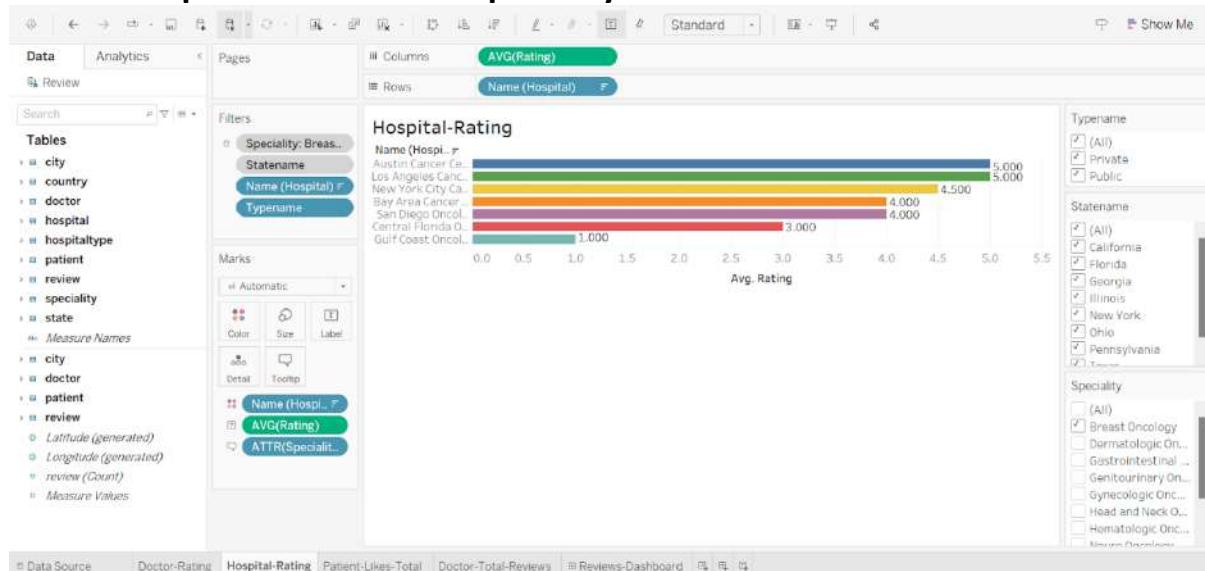
Reviews KPI & Dashboard: (FROM CLOUD)

1.Best Doctors for a chosen speciality based on ratings



The above graph shows the best top doctors available for a chosen speciality based on this info the patients can take a decision. (This follows from the OLAP operation: Sort doctors by highest rating based on speciality (measure involved:rating))

2. Best Hospitals for a chosen speciality:



The above graph shows which hospital is best for treating cancers of a particular speciality based on rating and using the filters we can see the information for each state and hospital type.

3. Details of the patients with most liked reviews for each speciality



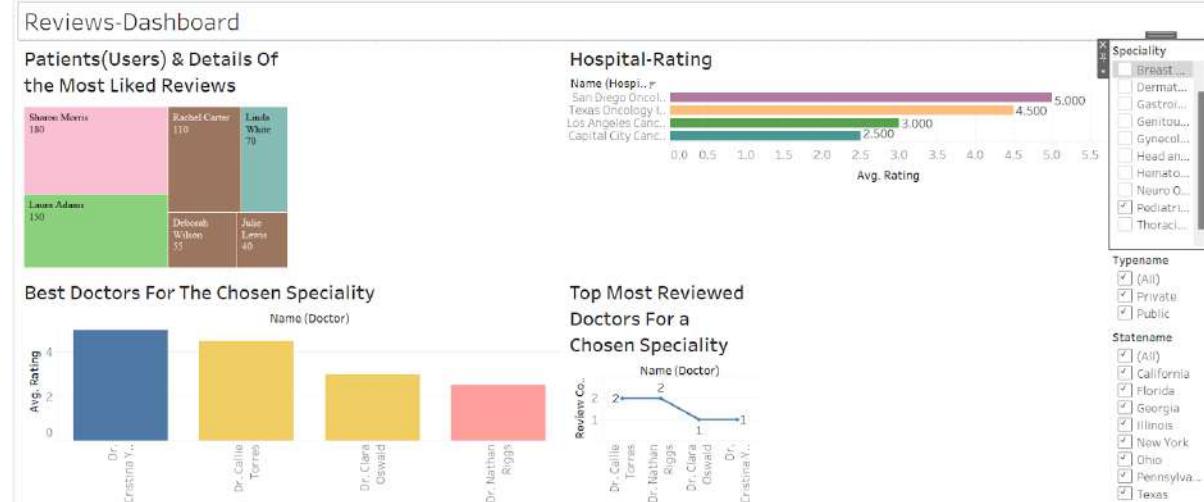
The above graph shows the most liked reviews for a chosen specialty. New users can sort by this and gain valuable information about the reviews. (This follows from the olap operation: Sort patients by the number of likes on their comments (measure involved: review helpfulness score/likes))

4. Most Reviewed Doctors in a chosen speciality (Total Number of reviews for a doctor):



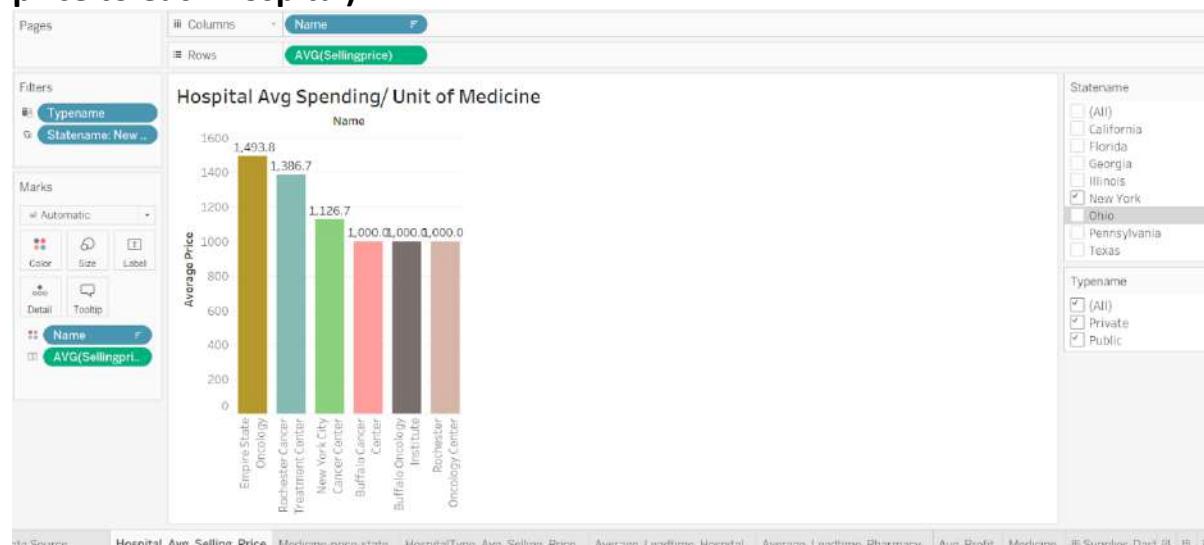
The above graph shows the most reviewed doctors in each speciality. This information can serve as valuable information for patients trying to choose a doctor.(This follows from the OLAP operation: Find the total number of reviews for a specific doctor)

Reviews Dashboard:



Supplies KPIs & Dashboard:

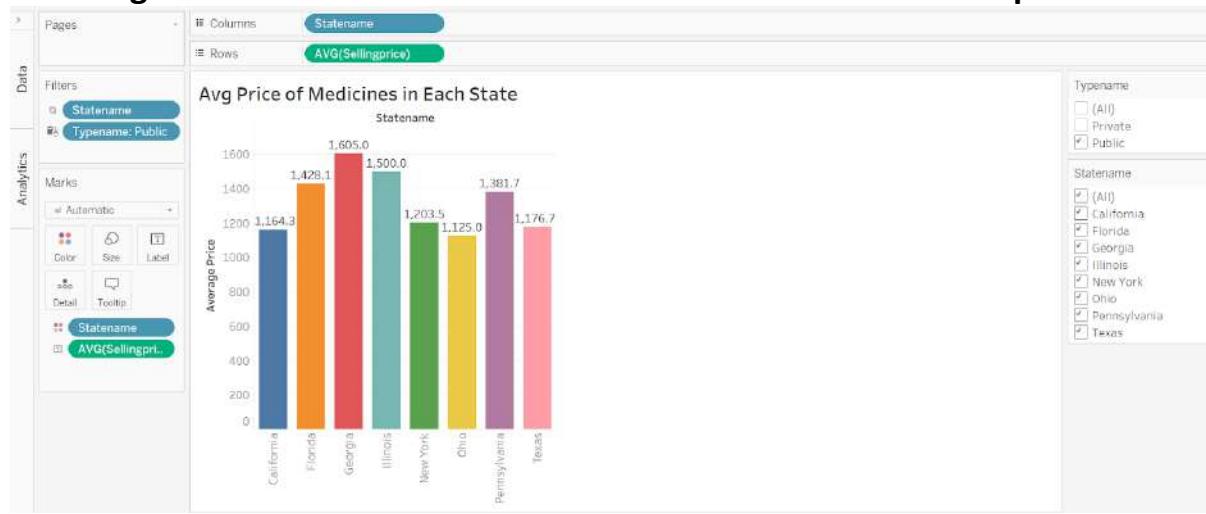
1. Average Spending for Medicine in Each Hospital in a chosen state (Avg Selling price to each hospital):



This graph gives the information on how much a specific hospital spends on medicines (follows from OLAP operation mentioned in milestone 3: The average

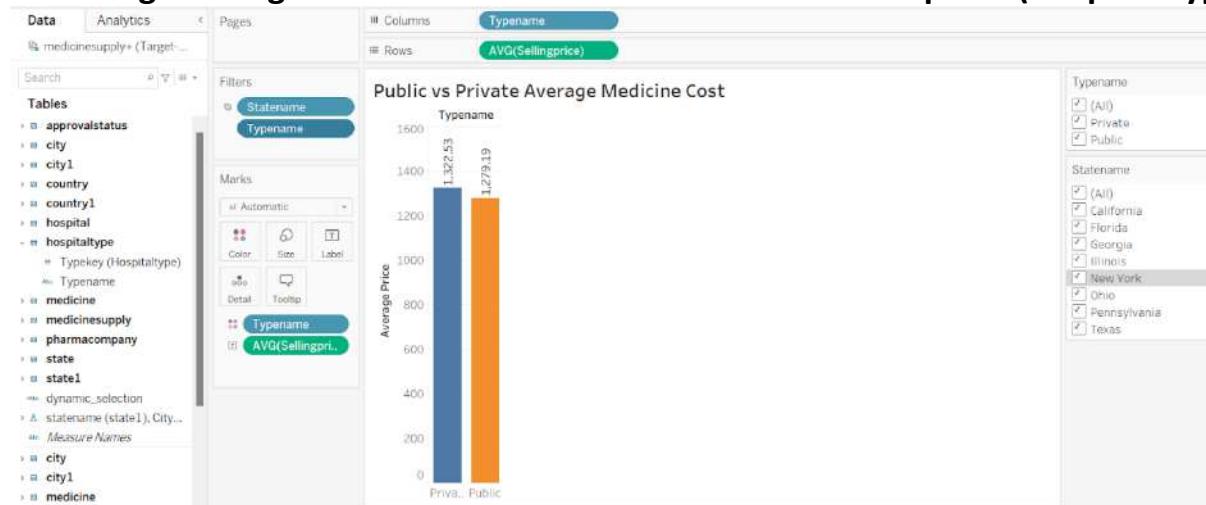
selling price of a medicine supplied to a particular hospital(measure involved: selling price))

2. Average Price of Medicine in Each State in Public & Private Hospitals:



This Graph gives more insights since we can choose the hospital type.

3. Average Selling Price of Medicine in Public & Private Hospitals (Hospital Type):



This graph gives information on how much the selling price for each hospital type (This follows from the OLAP operation mentioned: Average Selling Price of Medicines for a Specific Hospital Type(measure involved: selling price))

4. LeadTime for Hospitals in a chosen state:



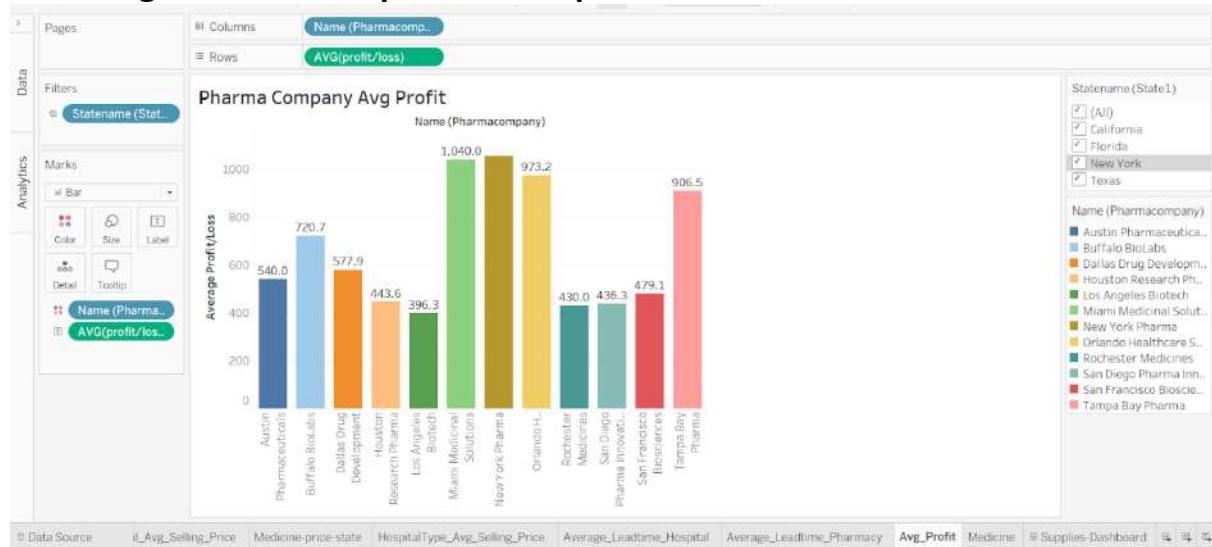
This graph gives the information on the average time it takes for a medicine to be delivered to an hospital

5. Average Lead Time for pharmacies to deliver their medicines:



This graph gives an information on the average time a pharma company takes to deliver its medicines.

6. Average Profit of the pharma companies:



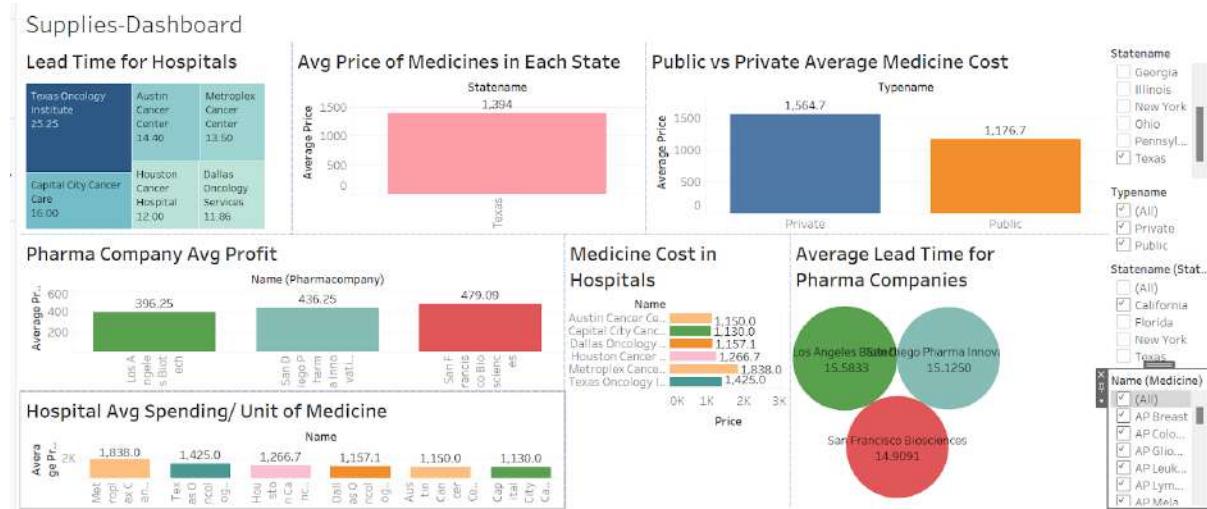
This gives the average profit a pharma company makes with each transaction

7. Avg Cost of a chosen medicine in each hospital where its available:



This gives the varying cost of the same medicine in different hospital.

Supplies Dashboard:



Complete Documentation of both projects

INTRODUCTION :

The healthcare sector, particularly in cancer treatment, faces numerous challenges in transparency, treatment effectiveness, and decision-making for patients. These challenges highlight the importance of providing accurate and accessible information to cancer patients, especially previvors, to make well-informed decisions. The CanCure initiative is a project designed to address these challenges by developing a comprehensive data warehouse system for cancer treatment. This system integrates static reference data (such as hospital details, doctor specialties, and patient demographics) and transactional data (including treatment records, patient reviews, and medicine logistics). We have implemented the ETL process on-prem using Talend and by leveraging AWS tools like Glue, S3, and Athena, this project establishes an effective data pipeline for processing, transforming, and analyzing this information using cloud. The ultimate goal is to enable data-driven decisions that improve treatment success rates, enhance patient satisfaction, and optimize hospital operations.

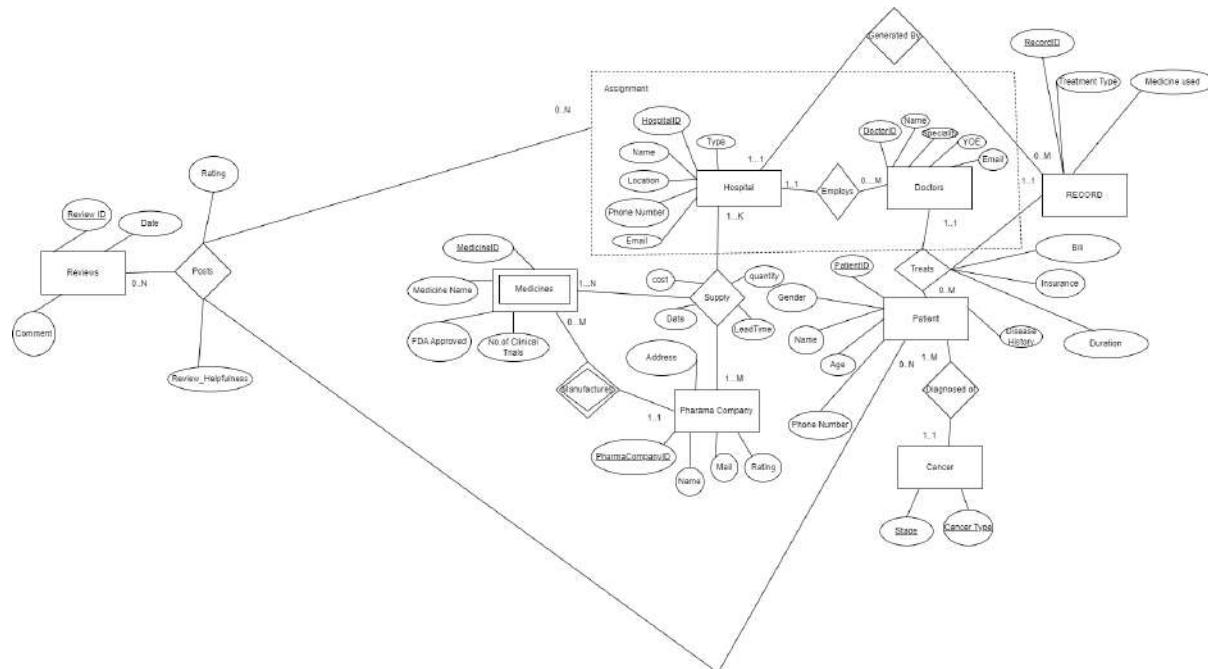
Project 1: On-Prem

Problem Definition:

The healthcare sector, particularly in cancer treatment, faces significant challenges related to transparency, treatment effectiveness, especially when it comes to patient's decision making . The need for knowledge, awareness on various improvements in treatments is largely unknown for a cancer previvor .CanCure is a government-funded initiative designed to regulate and monitor cancer institute hospitals operations while empowering citizens to make well-informed decisions regarding their cancer treatment.This project aims to develop a comprehensive database system for the Cancer Institute, capturing detailed static reference data (including but not limited to hospital details, doctor specialties, and patient demographics, and Pharma company details , medicine details, and cancer details) and transactional data (including but not limited to medical history, treatment records, patient reviews, medicine supply logistics and so on.). By establishing this system, we will enable thorough analysis of treatment success patterns, identification of influential factors, and evaluation of their effects on patients and hospital performance.

Operational Business Context: In the context of cancer treatment, the accuracy of diagnosis, treatment efficacy, and hospital performance significantly affect patient survival and overall experience and patients often lack the information needed to make fully informed decisions. CanCure seeks to bridge this gap by providing data on hospital and doctor performance, treatment success rates, and the effectiveness of specific drugs and treatments by various doctors. Initially, we plan to assess and analyze success rate based on hospital type (public or private), hospital ratings, doctor performance and years of experience, patient reviews and the medicine efficacy to identify areas for improvement. This analysis will empower the stakeholders to make data-driven decisions that enhance cancer care, improve patient satisfaction, and streamline hospital operations.

ERD:



Relational Model of relational database:

Primary Key-Underlined

Foreign key- Italics,#

Hospital (HospitalID, Name, Location, Ph-No, Email)

Primary Key: HospitalID

Foreign Keys: None

Doctor (DoctorID, Name, Specialty, YOE, Email, #*HospitalID*)

Primary Key: DoctorID

Foreign Key: HospitalID (references Hospital.HospitalID)

Patient (PatientID, Gender, Name, Age, ph.No, Symptoms, Diagnosis date,

Diseasehistory,#*CancerStage*, #*CancerType*)

Primary Key: PatientID

Foreign Keys:CancerStage and CancerType together (reference Cancer.Stage and Cancer.CancerType)

Cancer (Stage, CancerType)

Primary Key: Composite key of (Stage, CancerType)

Foreign Keys: None

Record (RecordID, TreatmentType, Status, StartDay, EndDay, Bill, #*HospitalID*)

Primary Key: RecordID

Foreign Key:HospitalID (references Hospital.HospitalID)

Treats (#*PatientID*, #*DoctorID*, #*RecordID*,#*medicineused*)

Primary Key: Composite key of (PatientID, DoctorID, RecordID, MedicineUsed)

Foreign Keys:

- PatientID (references Patient.PatientID)
- DoctorID (references Doctor.DoctorID)
- RecordID (references Record.RecordID)
- MedicineUsed (references Medicines.MedicineID)

Pharma-Company (CompanyID, Name, Mail, Address, Rating)

Primary Key: CompanyID

Foreign Keys: None

Medicine (*MedicineID, Name, FDA, No. of Clinical Trials, #CompanyID*)

Primary Key: MedicineID

Foreign Key: CompanyID (references PharmaCompany.CompanyID)

Supplies (*#HospitalID, #MedicineID, #PharmaID, Cost, Quantity, Unit, Date*)

Primary Key: Composite key of (HospitalID, MedicineID, PharmaID)

Foreign Keys:

- HospitalID (references Hospital.HospitalID)
- MedicineID (references Medicines.MedicineID)
- PharmaID (references PharmaCompany.CompanyID)

Assignment (*#HospitalID, #DoctorID*)

Primary Key: Composite key of (HospitalID, DoctorID)

Foreign Keys:

- HospitalID (references Hospital.HospitalID)
- DoctorID (references Doctor.DoctorID)

Posts (*#ReviewID, #PatientID, #HospitalID, #DoctorID*)

Primary Key: Composite key of (ReviewID, PatientID, HospitalID, DoctorID)

Foreign Keys:

- ReviewID (references Reviews.ReviewID)
- PatientID (references Patient.PatientID)
- HospitalID (references Hospital.HospitalID)
- DoctorID (references Doctor.DoctorID)

Reviews (*ReviewID, Date, Rating, Comments*)

Primary Key: ReviewID

Foreign Keys: None

SQL Implementation of the Relational Model:

```
CREATE DATABASE Cancure;
```

```
CREATE TABLE Hospital (
    HospitalID VARCHAR(10) PRIMARY KEY,
    Name VARCHAR(255),
    City VARCHAR(255),
    State VARCHAR(50),
    PhoneNo VARCHAR(15),
    Email VARCHAR(255)
);
```

```
CREATE TABLE Doctor (
    DoctorID VARCHAR(10) PRIMARY KEY,
    Name VARCHAR(255),
    Speciality VARCHAR(255),
    YOE INT,
    Email VARCHAR(255),
    HospitalID VARCHAR(10) REFERENCES Hospital(HospitalID)
);
```

```
CREATE TABLE Cancer (
    Stage VARCHAR(255),
    CancerType VARCHAR(255),
    PRIMARY KEY (Stage, CancerType)
);
```

```
CREATE TABLE Patient (
    PatientID VARCHAR(10) PRIMARY KEY,
    Gender VARCHAR(10),
    Name VARCHAR(255) NOT NULL,
    Age INT,
    PhoneNo VARCHAR(15),
    Symptoms TEXT,
    DiagnosisDate DATE,
    CancerStage VARCHAR(255),
    CancerType VARCHAR(255),
    FOREIGN KEY (CancerStage, CancerType) REFERENCES Cancer(Stage, CancerType)
);
```

```
CREATE TABLE Record (
    RecordID VARCHAR(10) PRIMARY KEY,
    TreatmentType VARCHAR(255),
    Status VARCHAR(255),
    StartDay DATE,
    EndDay DATE,
    Bill NUMERIC(10, 2),
    HospitalID VARCHAR(10) REFERENCES Hospital(HospitalID)
);
```

```
CREATE TABLE PharmaCompany (
    CompanyID VARCHAR(10) PRIMARY KEY,
    Name VARCHAR(255),
    Mail VARCHAR(255),
    City VARCHAR(100),
    State VARCHAR(100),
    Rating INT CHECK (Rating >= 1 AND Rating <= 5)
);
```

```
CREATE TABLE Medicines (
    MedicineID VARCHAR(10) PRIMARY KEY,
    Name VARCHAR(255),
    FDA BOOLEAN,
    NoOfClinicalTrials INT,
    Cost NUMERIC(10, 2),
    PharmaID VARCHAR(10) REFERENCES PharmaCompany(CompanyID)
);
```

```
CREATE TABLE Treats (
    PatientID VARCHAR(10) REFERENCES Patient(PatientID),
    DoctorID VARCHAR(10) REFERENCES Doctor(DoctorID),
    RecordID VARCHAR(10) REFERENCES Record(RecordID),
    MedicineUsed VARCHAR(10) REFERENCES Medicines(MedicineID),
    PRIMARY KEY (PatientID, DoctorID, RecordID, MedicineUsed)
);
```

```
CREATE TABLE Supplies (
    HospitalID VARCHAR(10) REFERENCES Hospital(HospitalID),
    MedicineID VARCHAR(10) REFERENCES Medicines(MedicineID),
```

```
PharmalD VARCHAR(10) REFERENCES PharmaCompany(CompanyID),
CostPerUnit NUMERIC(10, 2),
Quantity INT,
SupplyDate DATE,
PRIMARY KEY (HospitalID, MedicineID, PharmalD)
);
```

```
CREATE TABLE Assignment (
    HospitalID VARCHAR(10) REFERENCES Hospital(HospitalID),
    DoctorID VARCHAR(10) REFERENCES Doctor(DoctorID),
    PRIMARY KEY (HospitalID, DoctorID)
);
```

```
CREATE TABLE Reviews (
    ReviewID VARCHAR(10) PRIMARY KEY,
    Date DATE,
    Rating INT,
    Comments TEXT
);
```

```
CREATE TABLE Posts (
    ReviewID VARCHAR(10) REFERENCES Reviews(ReviewID),
    PatientID VARCHAR(10) REFERENCES Patient(PatientID),
    HospitalID VARCHAR(10) REFERENCES Hospital(HospitalID),
    DoctorID VARCHAR(10) REFERENCES Doctor(DoctorID),
    PRIMARY KEY (ReviewID, PatientID, HospitalID, DoctorID)
);
```

Data Population Methodology:

We generated our data using various LLM in batches and filled the static data like Hospital, Doctor, Cancer, PharmaCompany and Medicines first and then using that static data we generated the data for our transactional data like Record, Treats, Supplies.

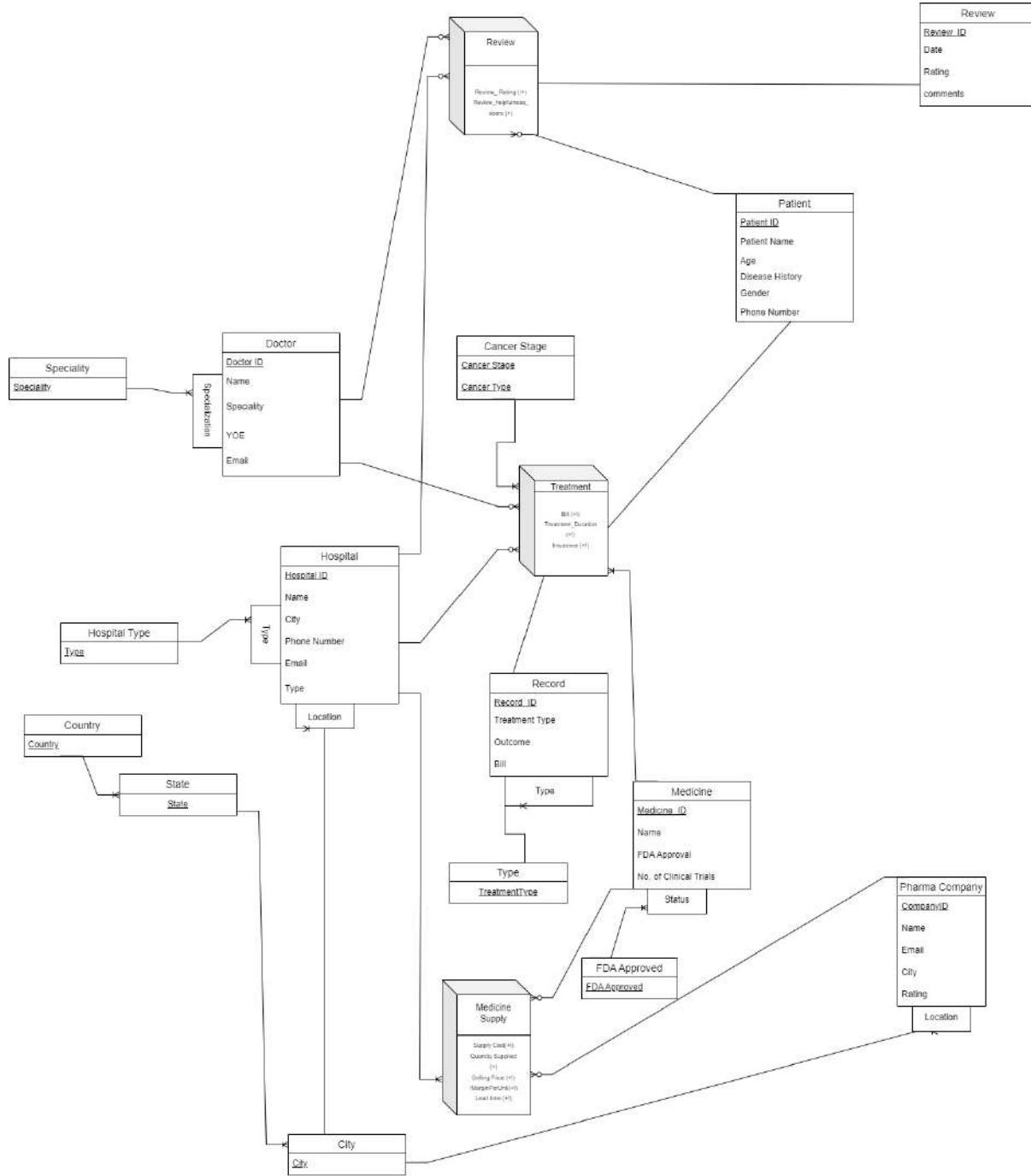
And with this generated data we populated the data into our tables in our database name Cancure.

The database is set up in thedbeaver and it is named Cancure and the database totally has 12 tables.

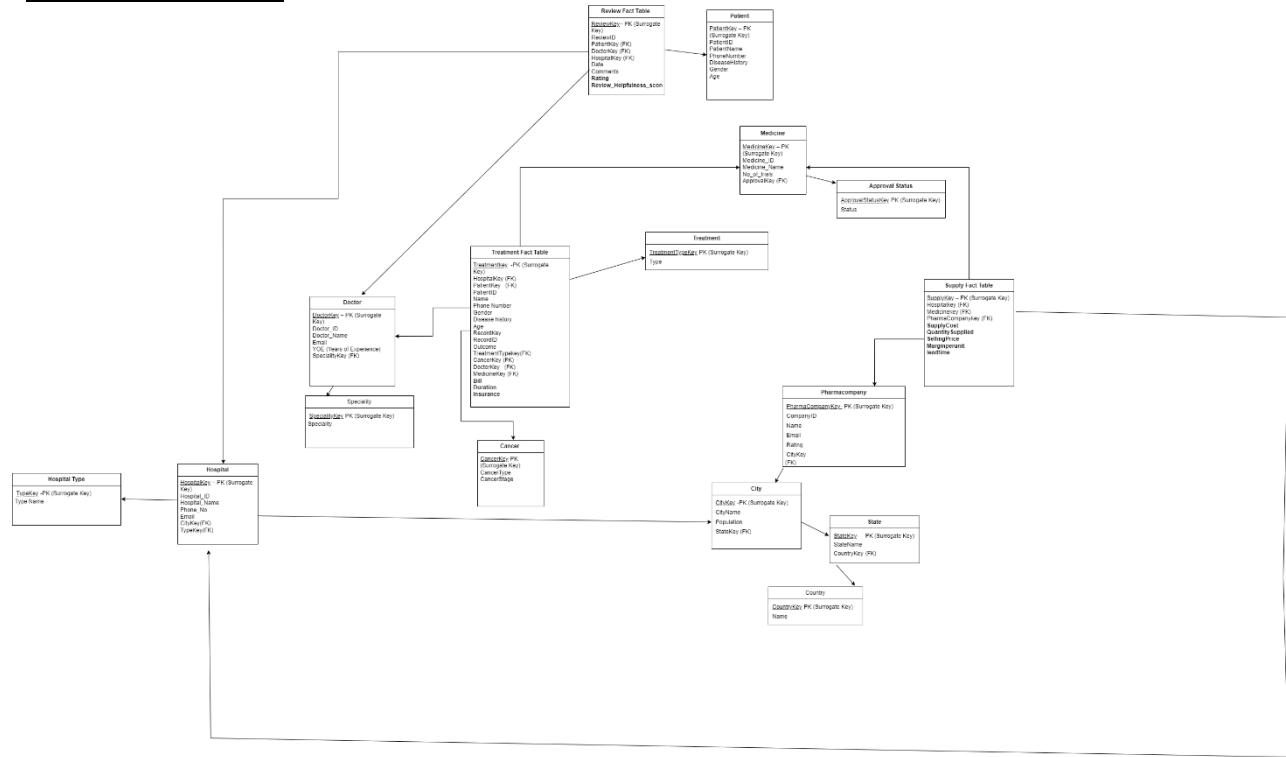
The screenshot shows thedbeaver interface with the database 'Cancure-Final-db' selected. Under the 'Schemas' node, the 'public' schema is expanded, revealing the 'Tables' node. The 'Tables' node is highlighted with a blue selection bar. Below it, a list of 12 tables is shown, each with a small icon and its size in kilobytes (K). The tables are: assignment (24K), cancer (32K), doctor (64K), hospital (32K), medicines (64K), patient (88K), pharmaccompany (32K), posts (72K), record (72K), reviews (72K), supplies (64K), and treats (80K). There are also three collapsed categories: Foreign Tables, Views, and Materialized Views.

>	assignment	24K
>	cancer	32K
>	doctor	64K
>	hospital	32K
>	medicines	64K
>	patient	88K
>	pharmaccompany	32K
>	posts	72K
>	record	72K
>	reviews	72K
>	supplies	64K
>	treats	80K
>	Foreign Tables	
>	Views	
>	Materialized Views	
-		

Conceptual Model:



Logical Model:



DB Implementation:

Hospital(HospitalKey,HospitalID,Name,Phone Number, Email, #citykey,#Typekey)

Primary Key: HospitalKey

Foreign Keys:

- #citykey references City.CityKey
- #Typekey references HospitalType.TypeKey
-

City(CityKey, CityName,Population, #StateKey)

Primary Key: CityKey

Foreign Keys: #StateKey references State.StateKey

State(StateKey, StateName,#CountryKey)

Primary Key: StateKey

Foreign Keys: #CountryKey references Country.CountryKey

Country(CountryKey, Name)

Primary Key: CountryKey

HospitalType(TypeKey, TypeName)

Primary Key: TypeKey

Speciality(SpecialityKey, Speciality)

Primary Key: SpecialityKey

Doctor(DoctorKey, DoctorID, Name, Email, YOE, #SpecialityKey)

Primary Key: DoctorKey

Foreign Keys: #SpecialityKey references Speciality.SpecialityKey

Patient(PatientKey, PatientID, Name, Ph.NO, Gender, DiseaseHistory, Age)

Primary Key: PatientKey

PharmaCompany (PharmaCompanyKey, CompanyID, Name, Email, Rating, #CityKey)

Primary Key: PharmaCompanyKey

Foreign Keys: #CityKey references City.CityKey

ApprovalStatus(ApprovalStatusKey, Status)

Primary Key: ApprovalStatusKey

Medicine(MedicineKey, MedicineID, Name, No.of Trials, #ApprovalStatuskey)

Primary Key: MedicineKey

Foreign Keys: #ApprovalStatuskey references ApprovalStatus.ApprovalStatusKey

TreatmentType(TypeKey, Type)

Primary Key: TypeKey

CancerType(CancerKey, CancerType, Stage)

Primary Key: CancerKey

Fact Tables:

Review(Review key, #Patientkey, #DoctorKey, #Hospitalkey, ReviewID, Date, Comments, Rating, review_helpfulness_score)

Primary Key: Reviewkey

Foreign Keys:

- Patientkey references Patient.PatientKey
- DoctorKey references Doctor.DoctorKey
- Hospitalkey references Hospital.HospitalKey

Treatment(TreatmentKey, #Hospitalkey, #PatientKey, PatientID, Name, Ph.NO, Gender, DiseaseHistory, AgeGroupKey, RecordKey, RecordID, Outcome, TreatmentTypeKey, #Cancerkey, #DoctorKey, Medicinekey, Bill, Duration, insurance, dosage)

Primary Key: TreatmentKey

Foreign key:

- Cancerkey references CancerType.Cancerkey
- DoctorKey references Doctor.DoctorKey
- PatientKey references Patient.PatientKey
- HospitalKey references Hospital.HospitalKey

MedicineSupply(SupplyKey,#Hospitalkey,#Medicinekey,#PharmaCompanykey,SupplyCost,Quantity,SellingPrice,MarginPerUnit, lead_time)

Primary Key: SupplyKey

Foreign Keys:

- Hospitalkey references Hospital.Hospitalkey
- Medicinekey references Medicine.Medicinekey
- PharmaCompanykey references PharmaCompany.Companykey

SQL Implementation of Warehouse Design:

```
CREATE TABLE Country (
    CountryKey INTEGER PRIMARY KEY,
    Name VARCHAR(255) NOT NULL
);

CREATE TABLE State (
    StateKey INTEGER PRIMARY KEY,
    StateName VARCHAR(255) NOT NULL,
    CountryKey INTEGER NOT NULL,
    FOREIGN KEY (CountryKey) REFERENCES Country(CountryKey)
);

CREATE TABLE City (
    CityKey INTEGER PRIMARY KEY,
    CityName VARCHAR(255) NOT NULL,
    Population INTEGER,
    StateKey INTEGER NOT NULL,
    FOREIGN KEY (StateKey) REFERENCES State(StateKey)
);

CREATE TABLE HospitalType (
    TypeKey INTEGER PRIMARY KEY,
    TypeName VARCHAR(255) NOT NULL
);

CREATE TABLE Hospital (
    HospitalKey INTEGER PRIMARY KEY,
    HospitalID VARCHAR(255) NOT NULL,
    Name VARCHAR(255) NOT NULL,
    Phone_Number VARCHAR(255),
    Email VARCHAR(255),
    CityKey INTEGER NOT NULL,
    TypeKey INTEGER NOT NULL,
    FOREIGN KEY (CityKey) REFERENCES City(CityKey),
    FOREIGN KEY (TypeKey) REFERENCES HospitalType(TypeKey)
```

```
);  
CREATE TABLE PharmaCompany (  
    PharmaCompanyKey INTEGER PRIMARY KEY,  
    CompanyID VARCHAR(255) NOT NULL,  
    Name VARCHAR(255) NOT NULL,  
    Email VARCHAR(255),  
    Rating DECIMAL(3,2),  
    CityKey INTEGER NOT NULL,  
    FOREIGN KEY (CityKey) REFERENCES City(CityKey)  
);  
CREATE TABLE CancerType (  
    CancerKey INT PRIMARY KEY,  
    CancerType VARCHAR(100) NOT NULL,  
    Stage VARCHAR(50) NOT NULL  
);  
CREATE TABLE Speciality (  
    SpecialityKey INT PRIMARY KEY,  
    Speciality VARCHAR(100) NOT NULL  
);  
CREATE TABLE Doctor (  
    DoctorKey INT PRIMARY KEY,  
    DoctorID VARCHAR(50) NOT NULL,  
    Name VARCHAR(100) NOT NULL,  
    Email VARCHAR(100) NOT NULL,  
    YOE INT NOT NULL, -- Years of Experience  
    SpecialityKey INT,  
    FOREIGN KEY (SpecialityKey) REFERENCES Speciality(SpecialityKey)  
);  
CREATE TABLE ApprovalStatus (  
    ApprovalStatusKey INT PRIMARY KEY,  
    Status VARCHAR(100) NOT NULL  
);  
CREATE TABLE TreatmentType (  
    TypeKey INT PRIMARY KEY,  
    Type VARCHAR(50) NOT NULL  
);
```

```

CREATE TABLE Medicine (
    MedicineKey INT PRIMARY KEY,
    MedicineID VARCHAR(50) NOT NULL,
    Name VARCHAR(100) NOT NULL,
    No_of_Trials INT,
    ApprovalStatusKey INT,
    FOREIGN KEY (ApprovalStatusKey) REFERENCES
    ApprovalStatus(ApprovalStatusKey)
);

CREATE TABLE Patient (
    PatientKey INT PRIMARY KEY,
    PatientID VARCHAR(50) NOT NULL,
    Name VARCHAR(100) NOT NULL,
    PhNo VARCHAR(20),
    Gender VARCHAR(10),
    DiseaseHistory TEXT,
    Age INT
);

CREATE TABLE Record (
    RecordKey INT PRIMARY KEY,
    RecordID VARCHAR(50) NOT NULL,
    Outcome VARCHAR(10),
    TreatmentTypeKey INT,
    FOREIGN KEY (TreatmentTypeKey) REFERENCES TreatmentType(TypeKey)
);


```

--Fact Tables

```

CREATE TABLE Review (
    ReviewKey INT PRIMARY KEY,
    PatientKey INT,
    DoctorKey INT,
    HospitalKey INT,
    ReviewID VARCHAR(50),
    Date DATE,
    Comments TEXT,
    Rating INT CHECK (Rating BETWEEN 1 AND 5),
    ReviewHelpfulnessScore INT,
    FOREIGN KEY (PatientKey) REFERENCES Patient(PatientKey),
    FOREIGN KEY (DoctorKey) REFERENCES Doctor(DoctorKey),

```

```

    FOREIGN KEY (HospitalKey) REFERENCES Hospital(HospitalKey)
);
CREATE TABLE MedicineSupply (
    SupplyKey INT PRIMARY KEY,
    HospitalKey INT,
    MedicineKey INT,
    PharmaCompanyKey INT,
    SupplyCost DECIMAL(10, 2),
    Quantity INT,
    SellingPrice DECIMAL(10, 2),
    MarginPerUnit DECIMAL(10, 2),
    LeadTime INT,
    FOREIGN KEY (HospitalKey) REFERENCES Hospital(HospitalKey),
    FOREIGN KEY (MedicineKey) REFERENCES Medicine(MedicineKey),
    FOREIGN KEY (PharmaCompanyKey) REFERENCES
    PharmaCompany(PharmaCompanyKey)
);
CREATE TABLE TreatmentFact (
    TreatmentKey INT PRIMARY KEY,
    HospitalKey INT,
    PatientKey INT,
    PatientID VARCHAR(50),
    Name VARCHAR(100),
    PhNo VARCHAR(15),
    Gender VARCHAR(10),
    DiseaseHistory TEXT,
    Age INT,
    RecordKey INT,
    RecordID VARCHAR(50),
    Outcome VARCHAR(10),
    TreatmentTypeKey INT,
    CancerKey INT,
    DoctorKey INT,
    MedicineKey INT,
    Bill DECIMAL(10, 2),
    Duration INT,
    Insurance DECIMAL(10, 2),
    FOREIGN KEY (HospitalKey) REFERENCES Hospital(HospitalKey),
    FOREIGN KEY (PatientKey) REFERENCES Patient(PatientKey),
    FOREIGN KEY (RecordKey) REFERENCES Record(RecordKey),
    FOREIGN KEY (TreatmentTypeKey) REFERENCES TreatmentType(TypeKey),

```

```
FOREIGN KEY (DoctorKey) REFERENCES Doctor(DoctorKey),
FOREIGN KEY (MedicineKey) REFERENCES Medicine(MedicineKey)
);
```

Primary Events:

Taking in patient records and Hospitals ordering medicines from pharma companies and patients posting their reviews would be the primary events for the project because these are the critical first steps in ensuring that the system has accurate and comprehensive data on patient transactions, treatments, supply interactions and billing, payments, and review ratings which will drive the subsequent analysis and reporting processes

OLAP Operations:

Treatment duration based on cancer stage

ROLLUP(Treatment, Cancer Stage, AVG(Duration))

Average cost in hospital to treat a particular cancer type

ROLLUP(Treatment, Hospital→Hospital, Cancer Type, AVG(Bill))

Average cost in a city to treat a particular cancer type

ROLLUP(Treatment, Hospital→City, Cancer Type, AVG(Cost))

Which specialty of doctors generates the most revenue

ROLLUP(Treatment, Doctor→Specialty, SUM(Bill))

Sort by highest average insurance coverage based on cancer type

SORT(Treatment, Cancertype, (AVG(Insurance) DESC))

The average selling price of a medicine supplied to a particular hospital

ROLLUP(MedicineSupply, Hospital, Medicine, AVG(Selling Price))

Average Selling Price of Medicines for a Specific Hospital Type

RolledUpCube<–ROLLUP(MedicineSupply, Hospital—>Type, Medicine, AVG(Selling Price))

SLICE(RolledUpCube, Type = 'SpecificType')

Find the total number of reviews for a specific doctor

SlicedReviewCube<—SLICE(Review, Doctor = 'SpecificDoctorID')
ROLLUP(Review, COUNT(ReviewID))

Sort doctors by highest rating based on specialty

RolledUpReview<—ROLLUP(Review, Doctor→Specialty, AVG(Rating))
SORT(RolledUpReview, Specialty, (AVG(Rating) DESC))

Sort patients by the number of likes on their comments

RolledUpReviews<—ROLLUP(Review, Patient, SUM(ReviewHelpfulnessScore))
SORT(RolledUpReview, Patient, (SUM(ReviewHelpfulnessScore) DESC))

Talend Implementation:

Data Flow and Control Flow

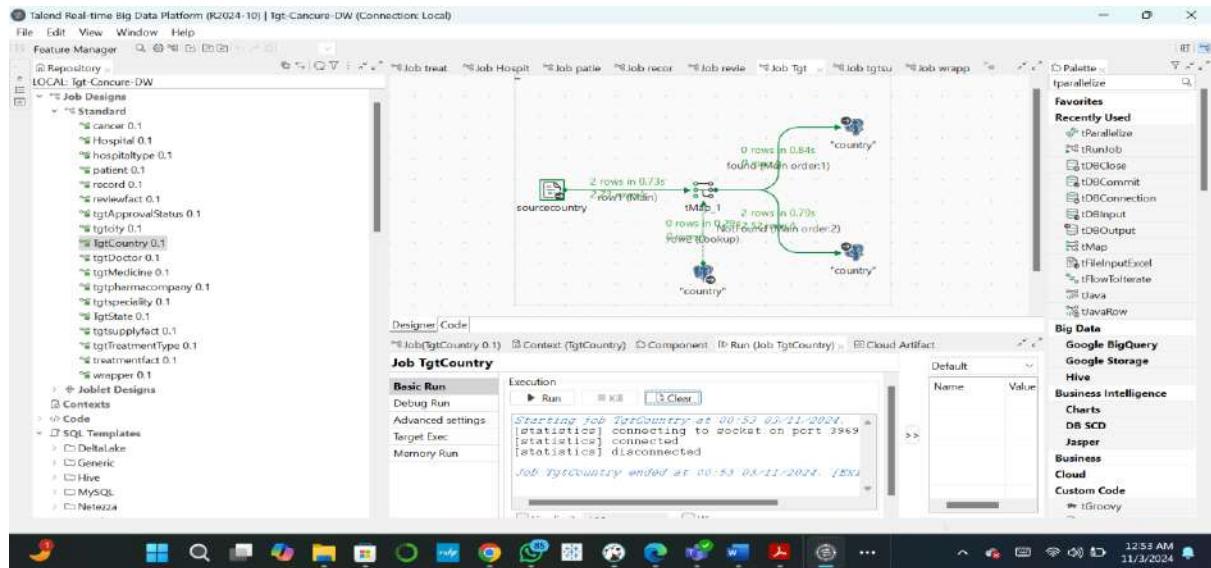
Insert flow:

The Following Diagram shows the insert flow execution and proof of execution for each table in the data warehouse, and the screenshots below also show surrogate key has been implemented for all tables.

Not Found-Insert

Found-Update

1. Country:



DBeaver 24.2.0 - <postgres> Tgt-Concure-create

```

ALTER SEQUENCE hospitaltypekey_seq RESTART WITH 1;
select * from hospitaltypekey_seq;

delete from city;
ALTER SEQUENCE city_citykey_seq RESTART WITH 1;
select * from city;

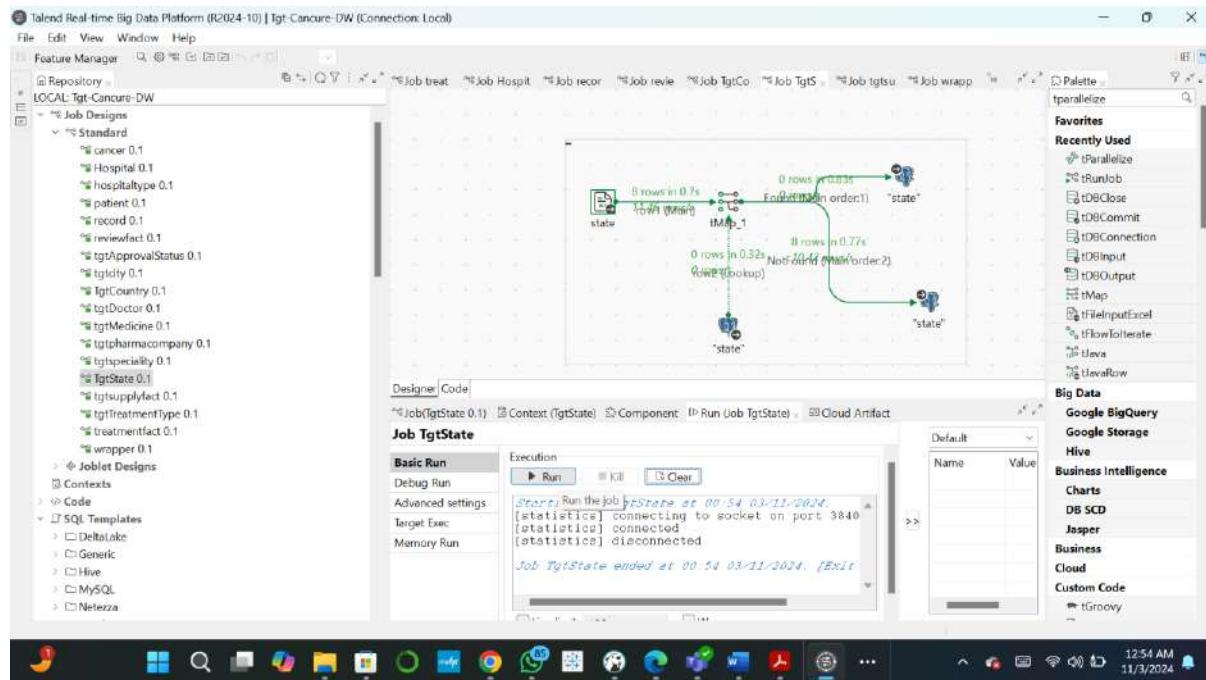
delete from state;
ALTER SEQUENCE state_statekey_seq RESTART WITH 1;
select * from state;

delete from country;
ALTER SEQUENCE country_countrykey_seq RESTART WITH 1;
select * from country;

```

countrykey	name
1	United States
2	India

2. State:



DBeaver 24.2.0 - <postgres>: Tgt-Cancure create

```

ALTER SEQUENCE hospitaltype_typekey_seq RESTART WITH 1;
select * from hospitaltype_h ;
delete from city;
ALTER SEQUENCE city_citykey_seq RESTART WITH 1;
select * from city;

delete from state;
ALTER SEQUENCE state_statekey_seq RESTART WITH 1;
select * from state;

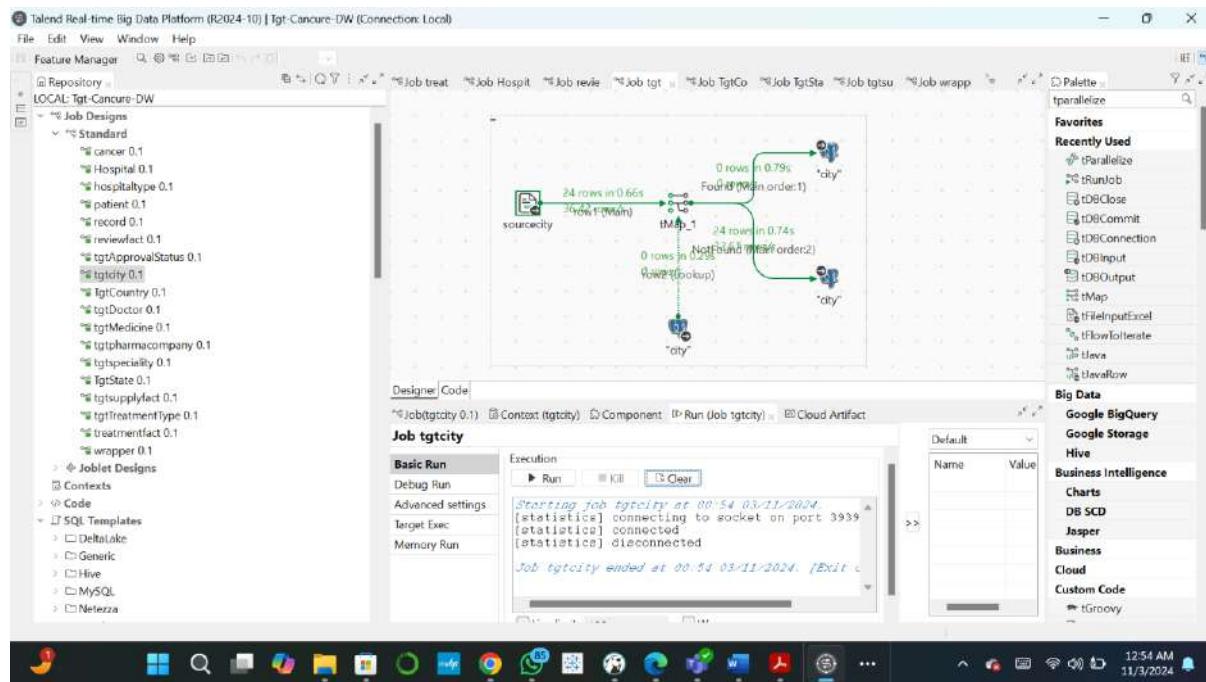
from country;
ALTER SEQUENCE country_countrykey_seq RESTART WITH 1;
select * from country;

```

state

statekey	statename	countrykey
1	California	1
2	Texas	1
3	Florida	1
4	New York	1
5	Illinois	1
6	Pennsylvania	1
7	Ohio	1
8	Georgia	1

3. City:



DBeaver 24.2.0 - <postgres> Tgt-Cancure-create

```

ALTER SEQUENCE hospitaltype_typekey_seq RESTART WITH 1;
select * from hospitaltype_n;

delete from city;
ALTER SEQUENCE city_citykey_seq RESTART WITH 1;
select * from city;

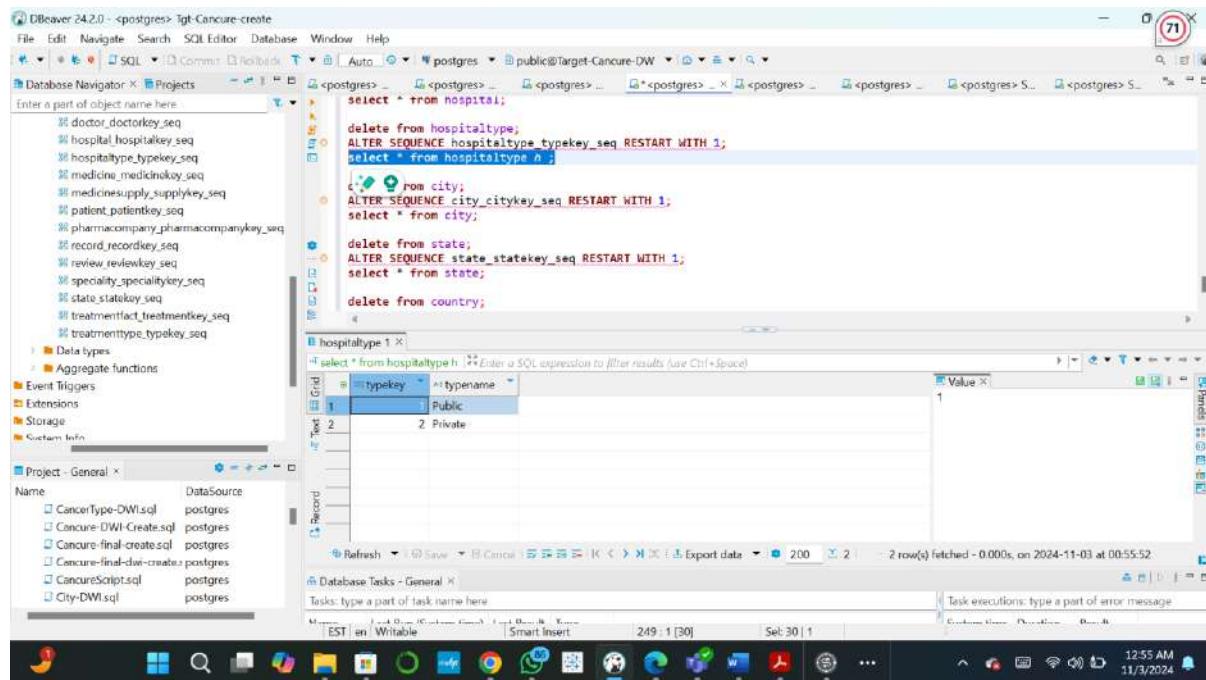
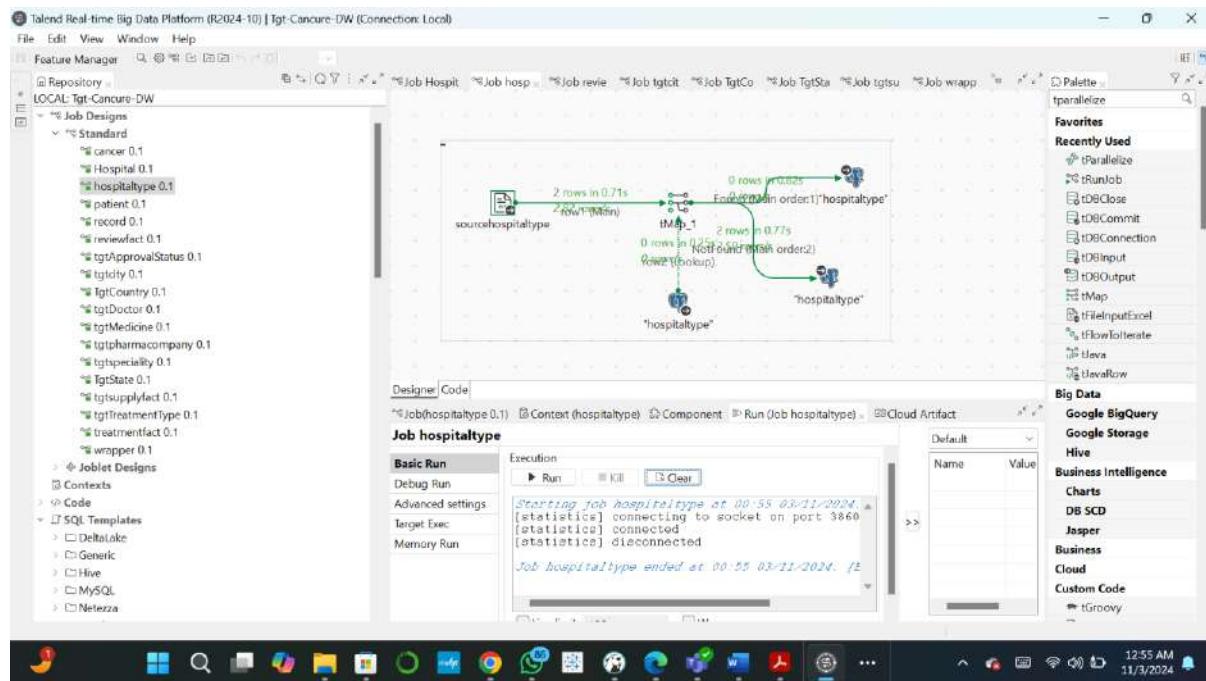
CREATE TABLE state;
ALTER SEQUENCE state_statekey_seq RESTART WITH 1;
select * from state;

delete from country;
ALTER SEQUENCE country_countrykey_seq RESTART WITH 1;
select * from country;

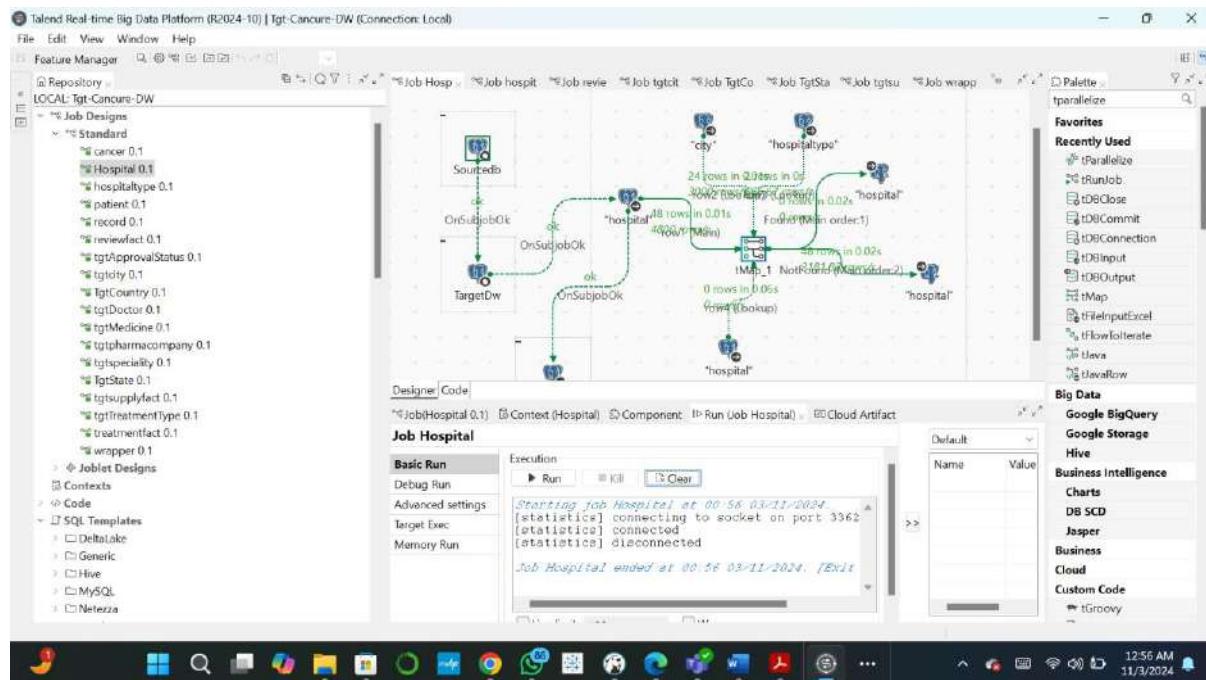
```

citykey	cityname	population	statekey
1	Los Angeles	380,000	1
2	San Francisco	880,000	1
3	New York	1,420,000	1
4	Houston	2,300,000	2
5	Austin	980,000	2
6	Dallas	1,340,000	2
7	Miami	550,000	3
8	Orlando	290,000	3

4. Hospital Type:



5. Hospital:



The screenshot shows the DBeaver 24.2.0 interface. The top navigation bar includes File, Edit, Navigate, Search, SQL Editor, Database, Window, Help. The main area shows a database session for "postgres" connected to "public@Target-Cancure-DW". The left sidebar includes Database Navigator, Projects, and a search bar. The central pane displays a SQL editor with the following code:

```

delete from hospital;
ALTER SEQUENCE hospital_hospitalkey_seq RESTART WITH 1;
select * from hospital;

-- From hospitaltype;
ALTER SEQUENCE hospitaltype_typekey_seq RESTART WITH 1;
select * from hospitaltype n ;

-- From city;
ALTER SEQUENCE city_citykey_seq RESTART WITH 1;
select * from city;

-- From state;
ALTER SEQUENCE state_statekey_seq RESTART WITH 1;
select * from state;

-- From treatmentfact_treatmentkey_seq;
ALTER SEQUENCE treatmentfact_treatmentkey_seq RESTART WITH 1;
select * from treatmentfact_treatmentkey_seq;

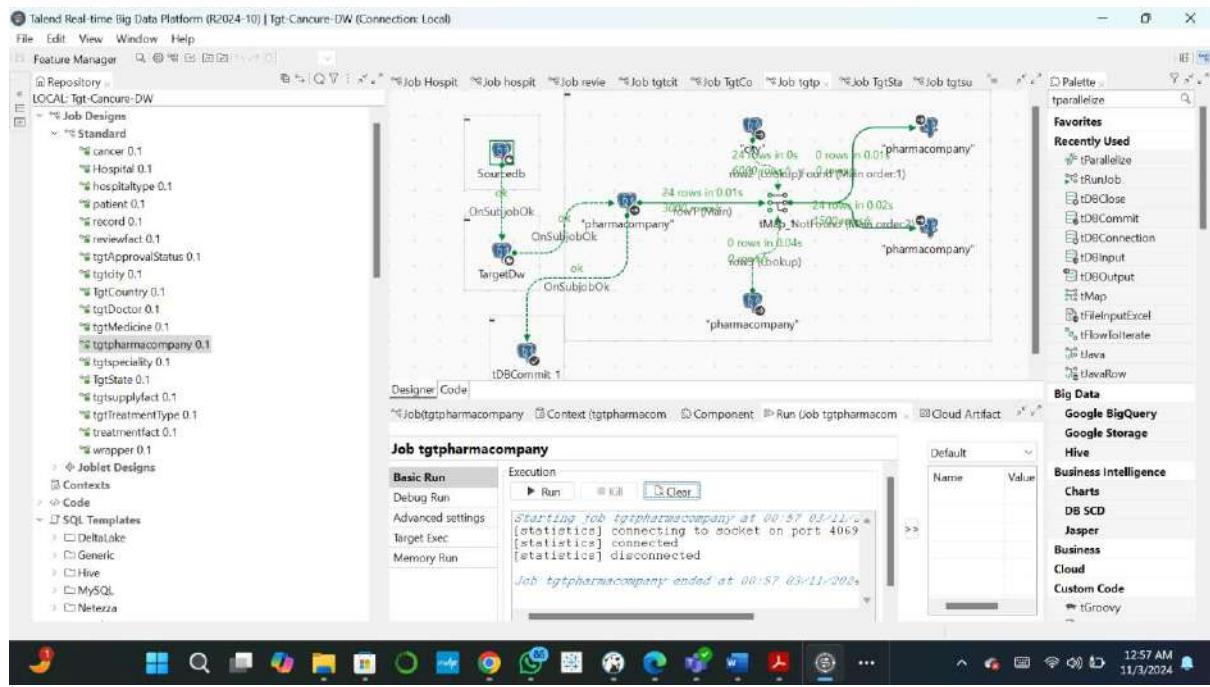
```

Below the SQL editor is a data grid titled "hospital 1" showing a list of hospitals. The columns are hospitalkey, hospitalid, name, phone_number, and oldemail. The data is as follows:

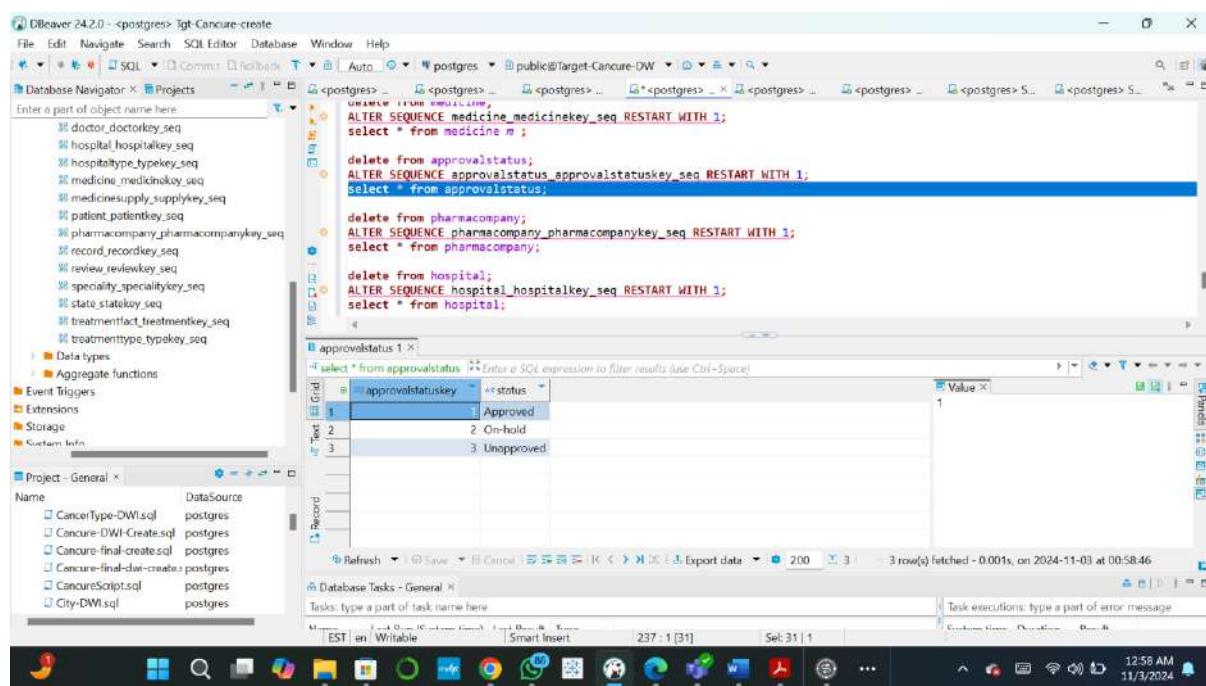
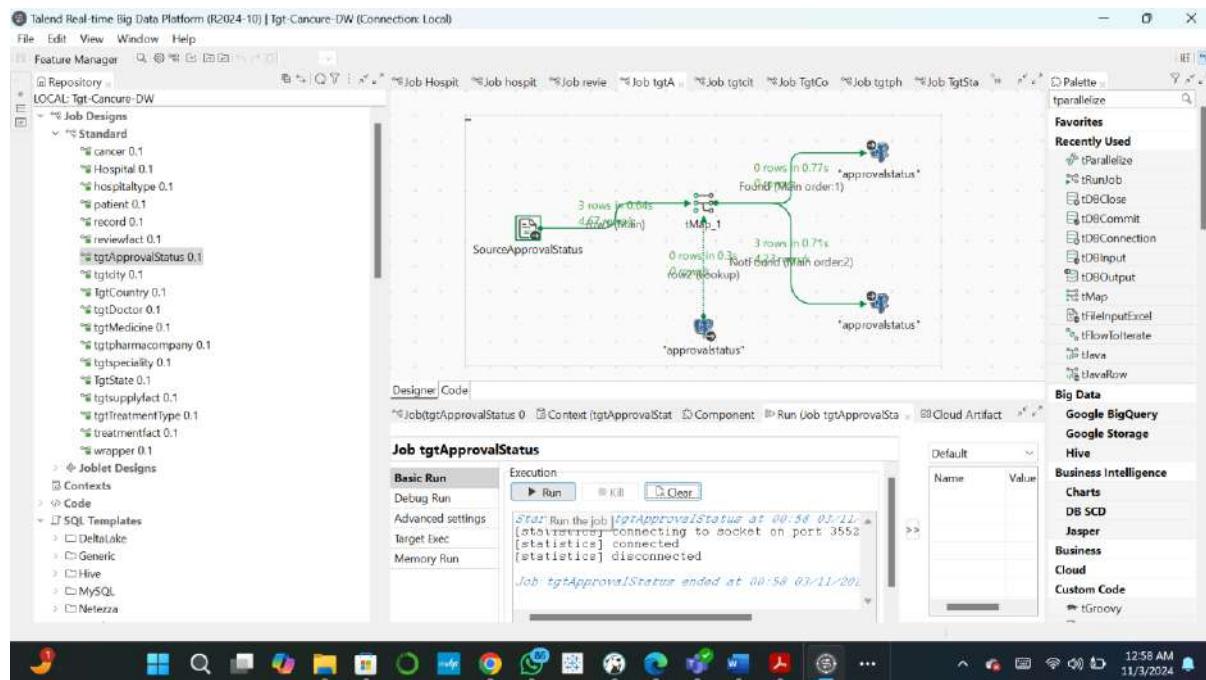
hospitalkey	hospitalid	name	phone_number	oldemail
1	HOSP001	Los Angeles Cancer Center	123-456-7890	contact@lacc.com
2	HOSP002	Los Angeles Oncology Institute	123-456-7891	info@laoi.com
3	HOSP003	San Francisco Cancer Treatment Cen	123-456-7892	contact@sftcc.co
4	HOSP004	Bay Area Cancer Hospital	123-456-7893	info@bach.com
5	HOSP005	San Diego Oncology Center	123-456-7894	contact@sdonc
6	HOSP006	Coastal Cancer Care	123-456-7895	info@ccc.com
7	HOSP007	Houston Cancer Hospital	123-456-7896	contact@hch.co

The bottom status bar shows the date and time as 11/3/2024 and 12:57 AM.

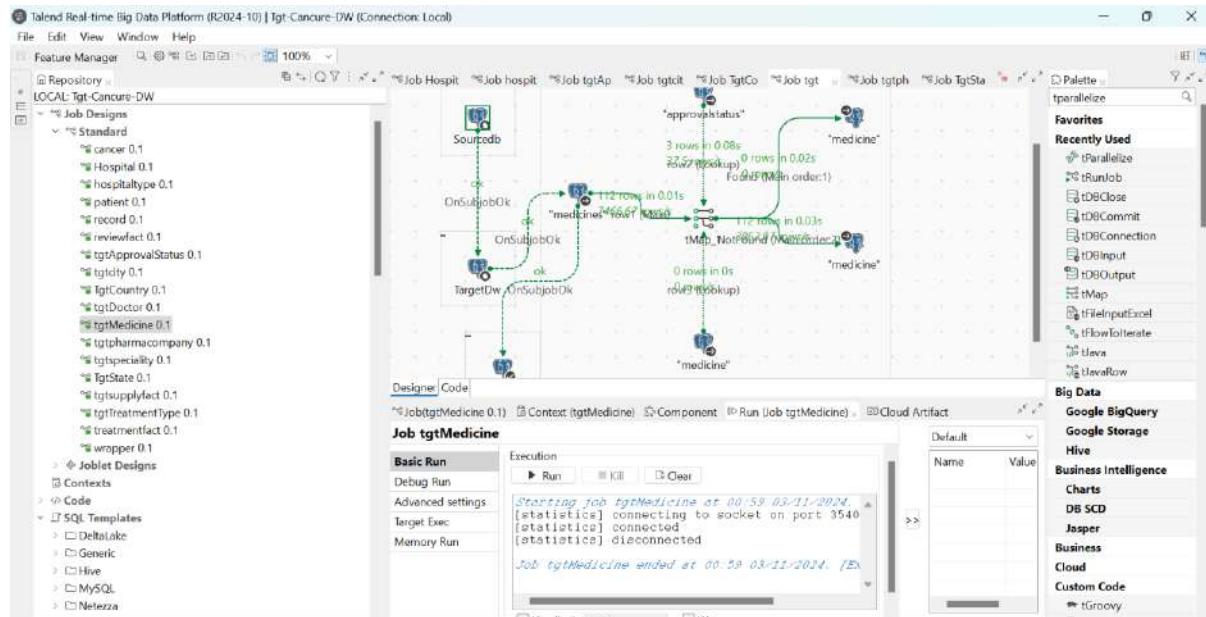
6. Pharma company:



7.ApprovalStatus:



8. Medicine:



DBeaver 24.2.0 - <postgres> Tgt-Cancure-create

File Edit Navigate Search SQL Editor Database Window Help

Database Navigator X Projects

Enter a part of object name here

Project - General X

Name Data Source

- CancerType-DW.sql postgres
- Concure-DWI-Create.sql postgres
- Concure-final-create.sql postgres
- Concure-final-dw-create.sql postgres
- ConcureScript.sql postgres
- City-DWI.sql postgres

DataSource

12:59 AM 11/3/2024

SQL Editor

```

SELECT * FROM patient p;

DELETE FROM medicine;
ALTER SEQUENCE medicine_medicinekey_seq RESTART WITH 1;
SELECT * FROM medicine m;

-- From approvalstatus;
ALTER SEQUENCE approvalstatus_approvalstatuskey_seq RESTART WITH 1;
SELECT * FROM approvalstatus;

DELETE FROM pharmaccompany;
ALTER SEQUENCE pharmaccompany_pharmaccompanykey_seq RESTART WITH 1;
SELECT * FROM pharmaccompany;
  
```

medicine 1 X

Grid

medicinekey	medicineid	name	no_of_trials	approvalstatuskey
1	MED1	OncBroast	5	1
2	MED2	OncOvarian	3	2
3	MED3	OncLeukem	4	1
4	MED4	OncGlloma	6	1
5	MED5	OncColorec	3	2
6	MED6	OncProstati	5	1
7	MED7	OncMelano	4	3
8	MED8	OncNeurob	6	1

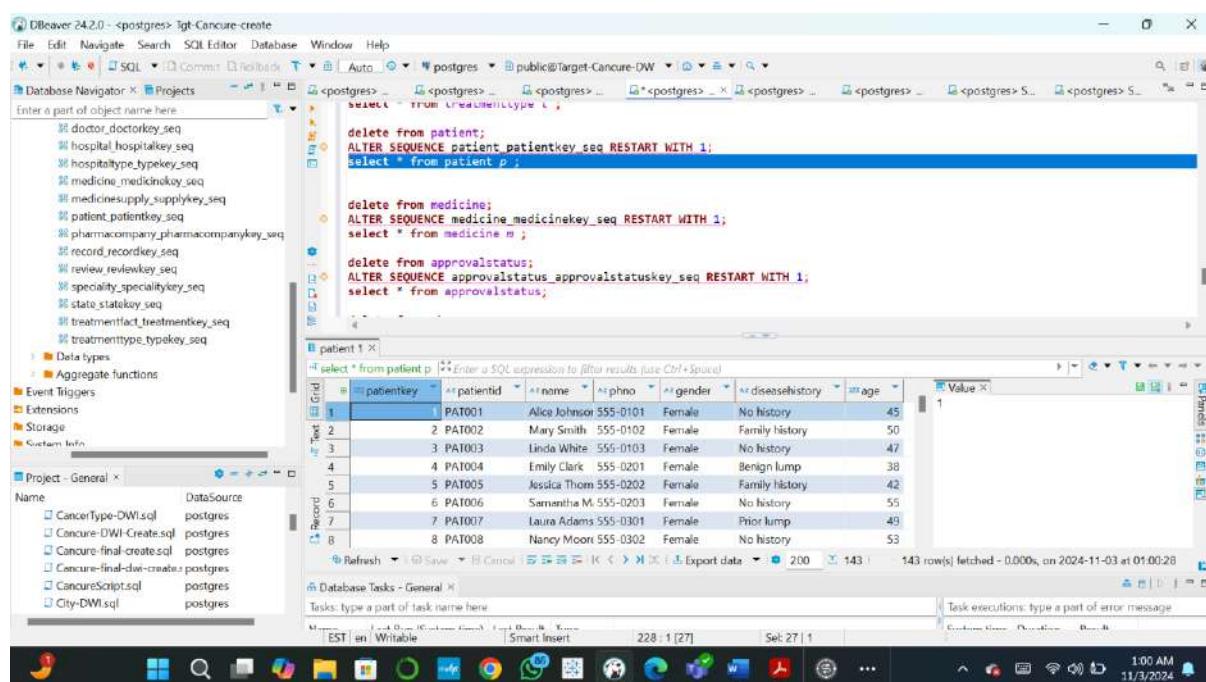
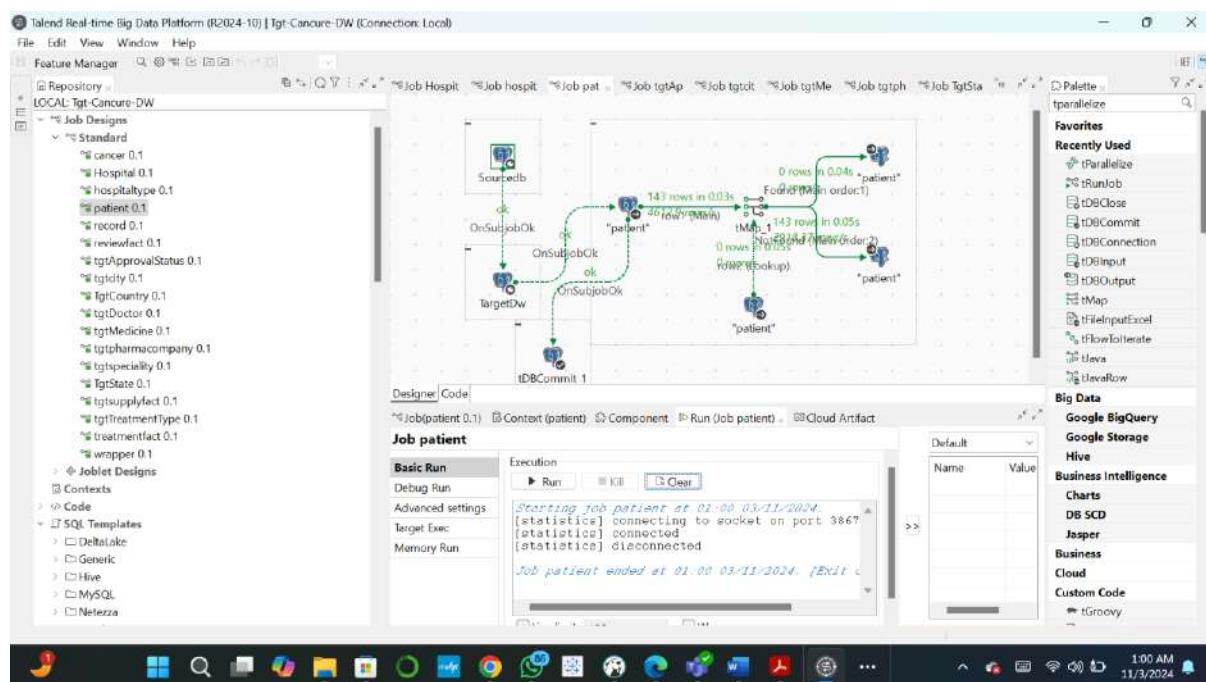
112 row(s) fetched - 0.002s, on 2024-11-03 at 00:59:45

Database Tasks - General X

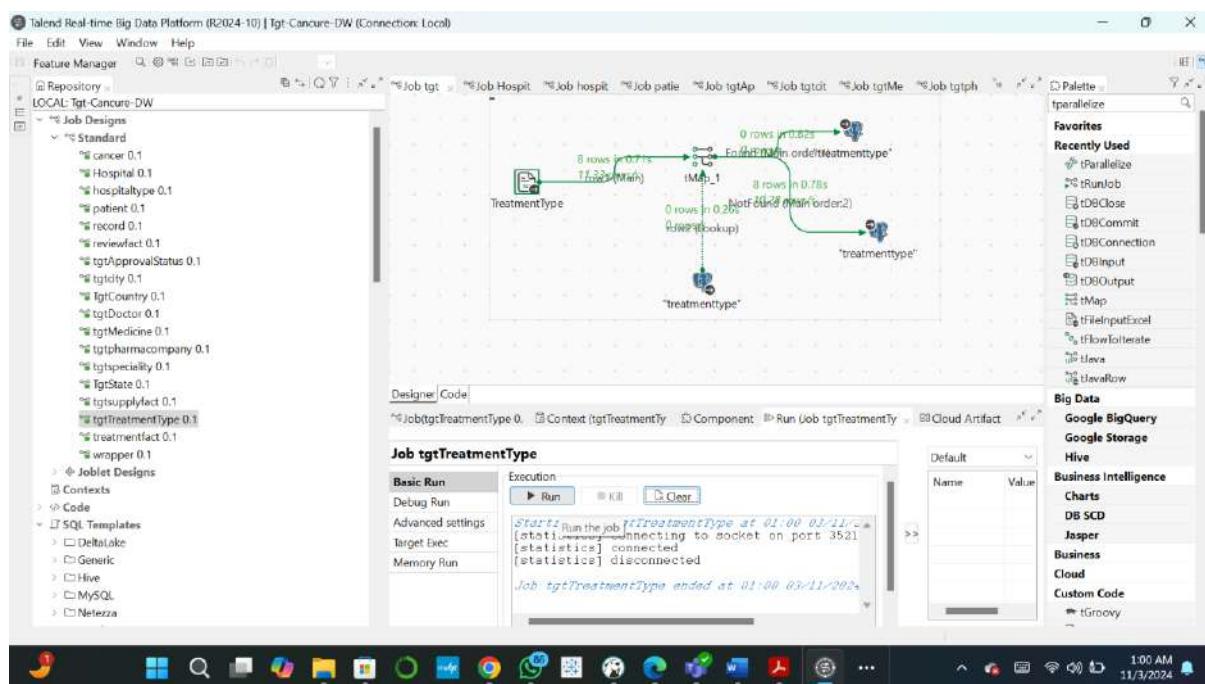
Tasks: type a part of task name here

EST en Writable Smart Insert 233 1 [28] Set 28 1

9. Patient:



10.TreatmentType:



DBeaver 24.2.0 - <postgres> Tgt-Cancure-create

File Edit Navigate Search SQL Editor Database Window Help

Database Navigator Projects

Enter a part of object name here

- doctor_doctorkey_seq
- hospital_hospitalkey_seq
- hospitality_typekey_seq
- medicine_medicinekey_seq
- medicineSupply_supplykey_seq
- patient_patientkey_seq
- pharmacyCompany_pharmacyCompanykey_seq
- record_recordkey_seq
- review_reviewkey_seq
- specialty_specialitykey_seq
- state_statekey_seq
- treatmentFact_treatmentkey_seq
- treatmentType_typekey_seq

Data types

Aggregate functions

Event Triggers

Extensions

Storage

System Info

Project - General

Name DataSource

- CancerType-DWI.sql postgres
- Cancure-DWI-Create.sql postgres
- Cancure-final-create.sql postgres
- Cancure-final-dwi-create.sql postgres
- CancureScript.sql postgres
- City-DWI.sql postgres

File Edit Navigate Search SQL Editor Database Window Help

Auto_ postgres public@Target-Cancure-DW

select from record;

delete from treatmenttype;

ALTER SEQUENCE treatmenttype_typekey_seq RESTART WITH 1;

select * from treatmenttype t ;

from patient;

ALTER SEQUENCE patient_patientkey_seq RESTART WITH 1;

select * from patient p ;

delete from medicine;

ALTER SEQUENCE medicine_medicinekey_seq RESTART WITH 1;

select * from medicine m ;

treatmenttype 1

typekey	type
1	Surgery
2	Radiation
3	Chemotherapy
4	Hormonal
5	Immunotherapy
6	Targeted
7	Palliative
8	Biologic

Column: gender: varchar(10)
Table: public.patient

Value 1

8 row(s) fetched - 0.000s, on 2024-11-03 at 01:01:08

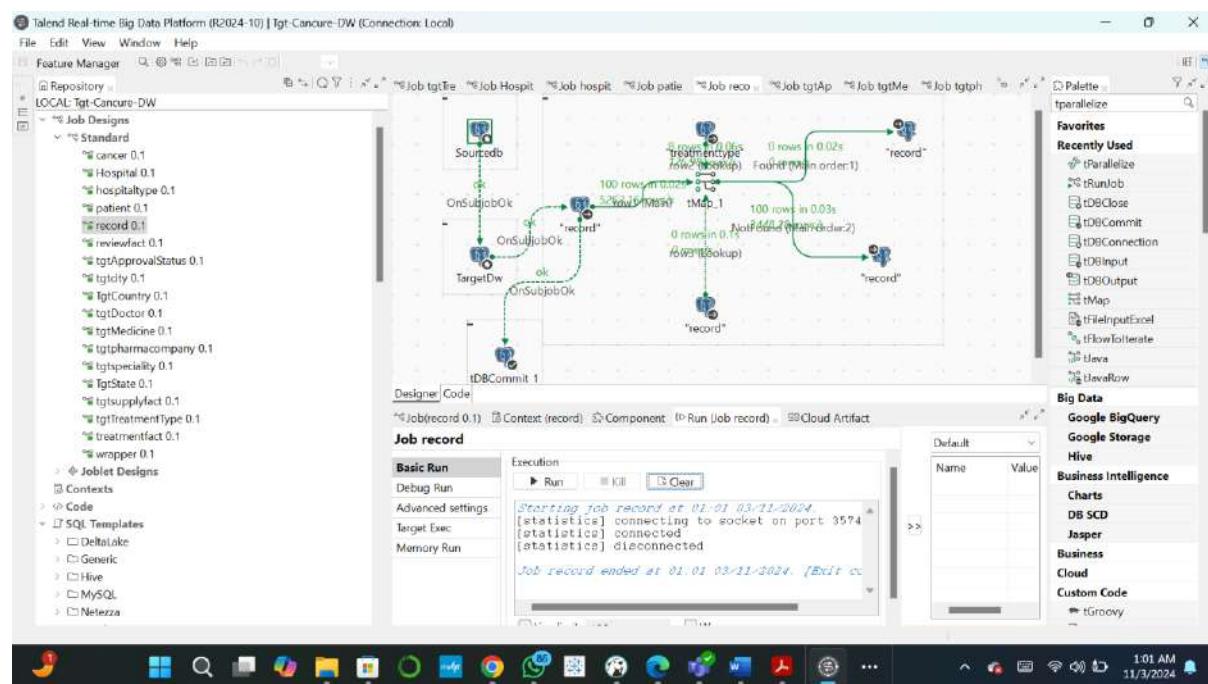
Database Tasks - General

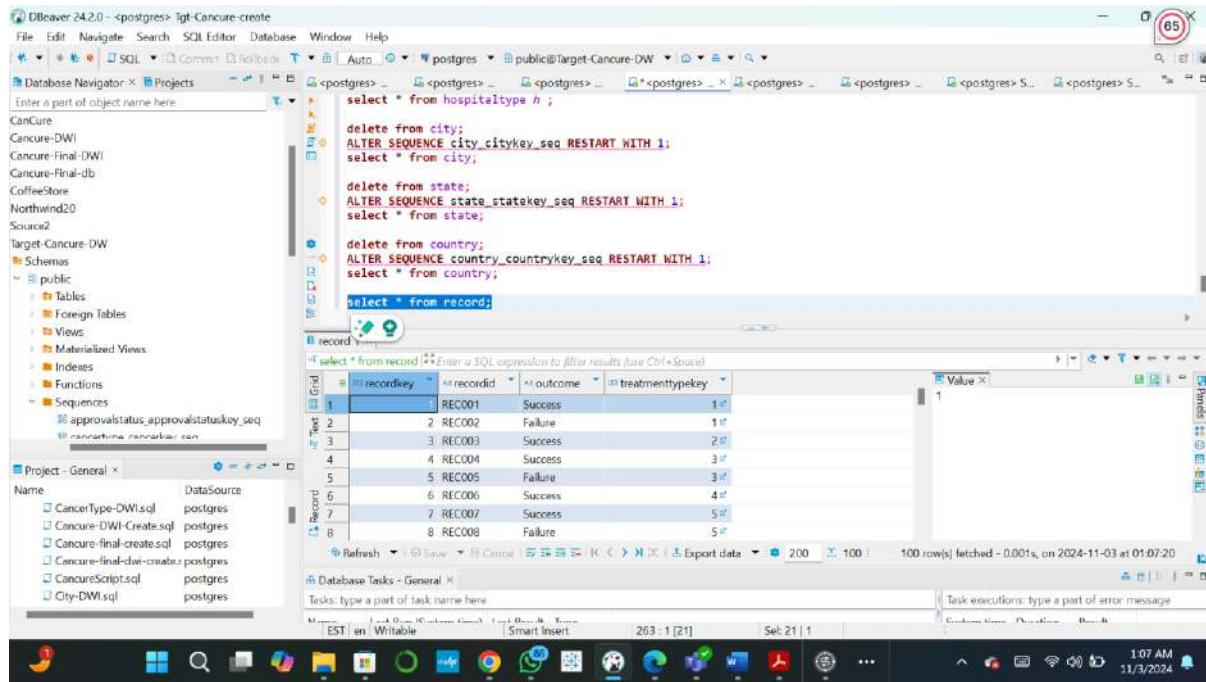
Tasks: type a part of task name here

EST is Writable Smart Insert 224 - 1 [31] Set 31 ↑ Task executions: type a part of error message

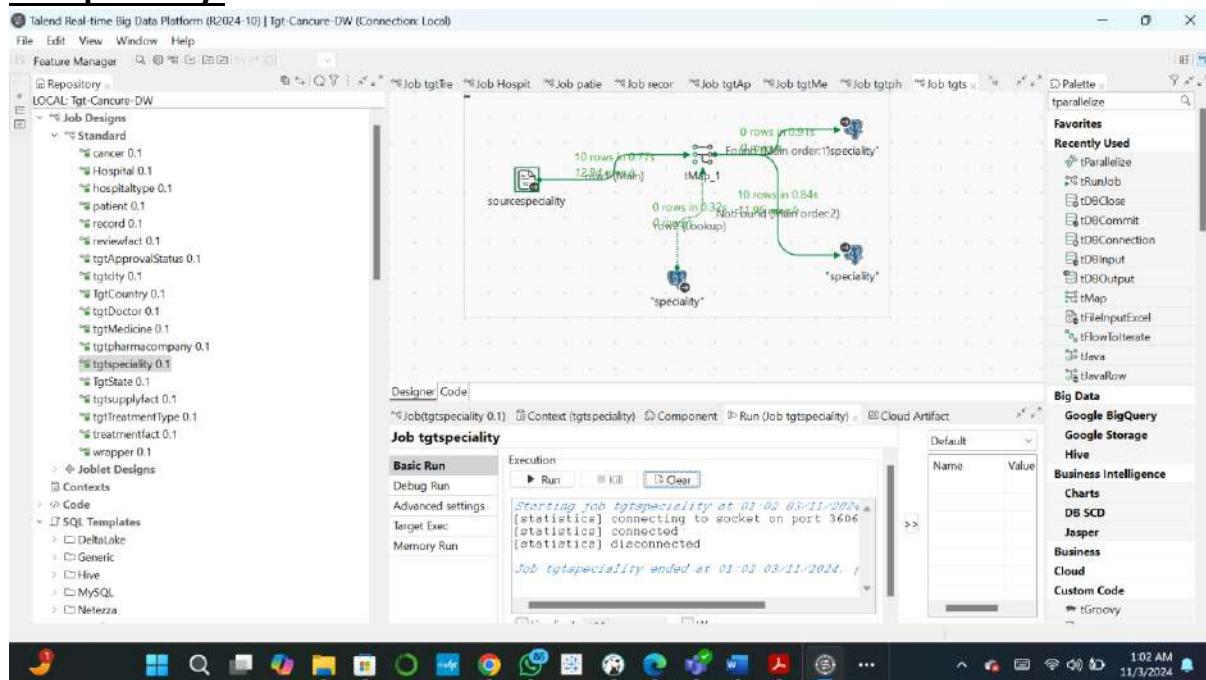
1.01 AM 11/3/2024

11. Record:





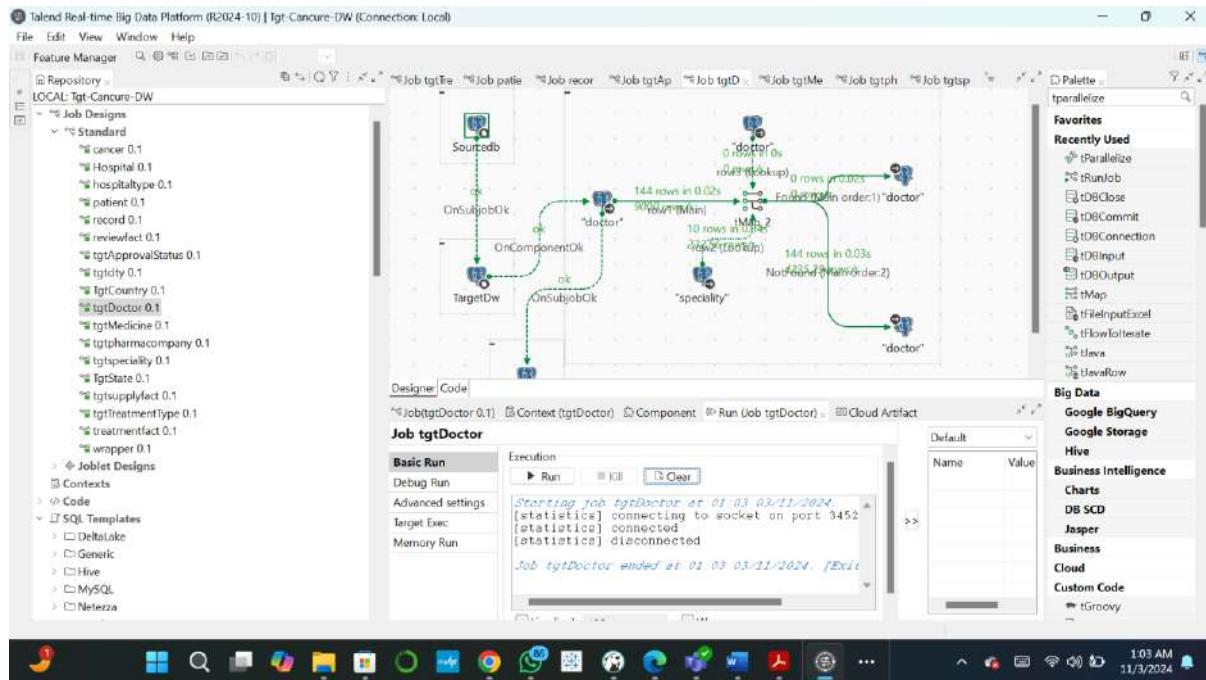
12. Speciality:



The screenshot shows the DBBeaver interface with the following details:

- File Bar:** File, Edit, Navigate, Search, SQL Editor, Database, Window, Help.
- Toolbar:** Back, Forward, Home, SQL, Comment, Revert, Auto, Postgres, public@Target-Cancure-DW.
- Database Navigator:** Shows projects, databases (CanCure, Cancure-DWI, Cancure-Final-DWI, Cancure-Final-db, CoffeeStore, Northwind20, Source2), and schemas (public).
- Project - General:** Lists files: CanCure-DWI.sql, Cancure-DWI-Create.sql, Cancure-Final-create.sql, Cancure-final-dwi-create.sql, CancureScript.sql, City-DWI.sql.
- SQL Editor:** Contains a sequence of SQL commands to drop and recreate tables (treatmentfact, doctor, specialty), sequences (cancerkey_seq, doctorkey_seq, specialitykey_seq), and data from records.
- Table View:** Shows the 'speciality' table with 10 rows. The first row, '1 Breast Oncology', is highlighted.
- Status Bar:** Shows 10 row(s) fetched < 0.002s on 2024-11-03 at 01:36:22.

13. Doctor:



DBeaver 24.2.0 - <postgres> Tgt-Cancure-create

File Edit Navigate Search SQL Editor Database Window Help

Database Navigator X Projects Enter a part of object name here

CanCure
Cancure-DWI
Cancure-Final-DWI
Cancure-Final-db
CoffeeStore
Northwind20
Source2
Target-Cancure-DW

Schemas public

Tables Foreign Tables Views Materialized Views Indexes Functions Sequences approvalstatus_approvalstatuskey_seq cancerthema_cancerkey_seq

Project - General

Name CancerType-DWI.sql DataSource postres Cancure-DWI-Create.sql postres Cancure-final-create.sql postres Cancure-final-dw-create.sql postres CancureScript.sql postres City-DWI.sql postres

SQL Editor

```

select * from medicinesupply m;

delete from cancertype;
ALTER SEQUENCE cancertype_cancerkey_seq RESTART WITH 1;
select * from cancertype;

delete from doctor;
ALTER SEQUENCE doctor_doctorkey_seq RESTART WITH 1;
select * from doctor;

delete from speciality;
ALTER SEQUENCE speciality_specialitykey_seq RESTART WITH 1;
select * from speciality;

delete from record;

```

doctor 1

	doctorday	doctordid	dr_name	dr_email	dr_yoe	specialitykey
1	DOC001	Dr. Emily Stone	estone@lacc	10	1	1
2	DOC002	Dr. James Wilson	jwilson@lacc	8	5	2
3	DOC003	Dr. Clara Osv	cowsal@lacc	5	3	3
4	DOC004	Dr. Richard Veeber	rveeber@lacc	15	2	4
5	DOC005	Dr. Miranda Maloley	mmaloley@lacc	12	4	5
6	DOC006	Dr. Alex Kase	akase@lacc	7	1	6
7	DOC007	Dr. Charlotte Kling	ckling@stcc	20	5	7
8	DOC008	Dr. Addison Montgome	amontgome@lacc	22	3	8

Grid Text Record Value

Refresh Save Cancel Export data 200 144 144 row(s) fetched - 0.001s, on 2024-11-03 at 01:03:30

Database Tasks - General

Tasks: type a part of task name here Task executions: type a part of error message

EST en Writable Smart Insert 210 52 5946 Set 0 | 0

10:3 AM 11/3/2024

14.Cancer:

Talend Real-time Big Data Platform (R2024-10) | Tgt-Cancure-DW (Connection: Local)

File Edit View Window Help

Feature Manager

LOCAL: Tgt-Cancure-DW

- Job Designs
 - Standard
 - Cancer 0.1
 - hospitaltype 0.1
 - patient 0.1
 - record 0.1
 - reviewfact 0.1
 - tgtApprovalStatus 0.1
 - tgtCity 0.1
 - tgtCountry 0.1
 - tgtDoctor 0.1
 - tgtMedicine 0.1
 - tgtPharmacyCompany 0.1
 - tghSpecialty 0.1
 - TgtState 0.1
 - tgtSupplyFact 0.1
 - tgtTreatmentType 0.1
 - treatmentType 0.1
 - wrapper 0.1
 - Joblet Designs
 - Contexts
 - Code
 - SQL Templates
 - DeltaLake
 - Generic
 - Hive
 - MySQL
 - Netezza

Designer Code

Job cancer

Basic Run Execution Run Kill Clear

Starting job cancer at 01:04 03/11/2024. [statistics] connecting to socket on port 3415 [statistics] connected [statistics] disconnected Job cancer ended at 01:04 03/11/2024. [Edit]

Advanced settings Target Exec Memory Run

Default Name Value

Palette

Favorites Recently Used

- Parallelize
- RunJob
- iOBClose
- iOBCommit
- iOBConnection
- iOBInput
- iOBOutput
 - iMap
 - FileInputExcel
 - FlowIterate
 - Java
 - JavaRow

Big Data Google BigQuery Google Storage Hive Business Intelligence

- Charts
- DB SCD
- Jasper
- Business
- Cloud
- Custom Code
 - Groovy

10:4 AM 11/3/2024

DBeaver 24.2.0 - <postgres> Tgt-Cancure-create

```

File Edit Navigate Search SQL Editor Database Window Help
Auto SQL Postgres public@Target-Cancure-DW
Database Navigator Projects
Enter a part of object name here
CanCure
Cancure-DWI
Cancure-Final-DWI
Cancure-Final-db
CoffeeStore
Northwind20
Source2
Target-Cancure-DW
Schemas
public
Tables
Foreign Tables
Views
Materialized Views
Indices
Functions
Sequences
approvalstatus_approvalstatuskey_seq
cancertype_cancertypekey_seq
Project - General
Name DataSource
CancerType-DWI.sql postgres
Cancure-DWI-Create.sql postgres
Cancure-final-create.sql postgres
Cancure-final-dwi-create.sql postgres
CancureScript.sql postgres
City-DWI.sql postgres

```

SQL Editor:

```

SELECT * FROM medicinupply m;

DELETE FROM canctype;
ALTER SEQUENCE canctype_cancerkey_seq RESTART WITH 1;
SELECT * FROM canctype;

DELETE FROM doctor;
ALTER SEQUENCE doctor_doctorkey_seq RESTART WITH 1;
SELECT * FROM doctor;

DELETE FROM speciality;
ALTER SEQUENCE speciality_specialitykey_seq RESTART WITH 1;
SELECT * FROM speciality;

DELETE FROM record;

```

Table Data:

	cancertype	stage
1	Breast Cancer	Stage I
2	Breast Cancer	Stage II
3	Breast Cancer	Stage III
4	Breast Cancer	Stage IV
5	Inflammatory Breast Cancer	Stage I
6	Inflammatory Breast Cancer	Stage II
7	Inflammatory Breast Cancer	Stage III
8	Inflammatory Breast Cancer	Stage IV

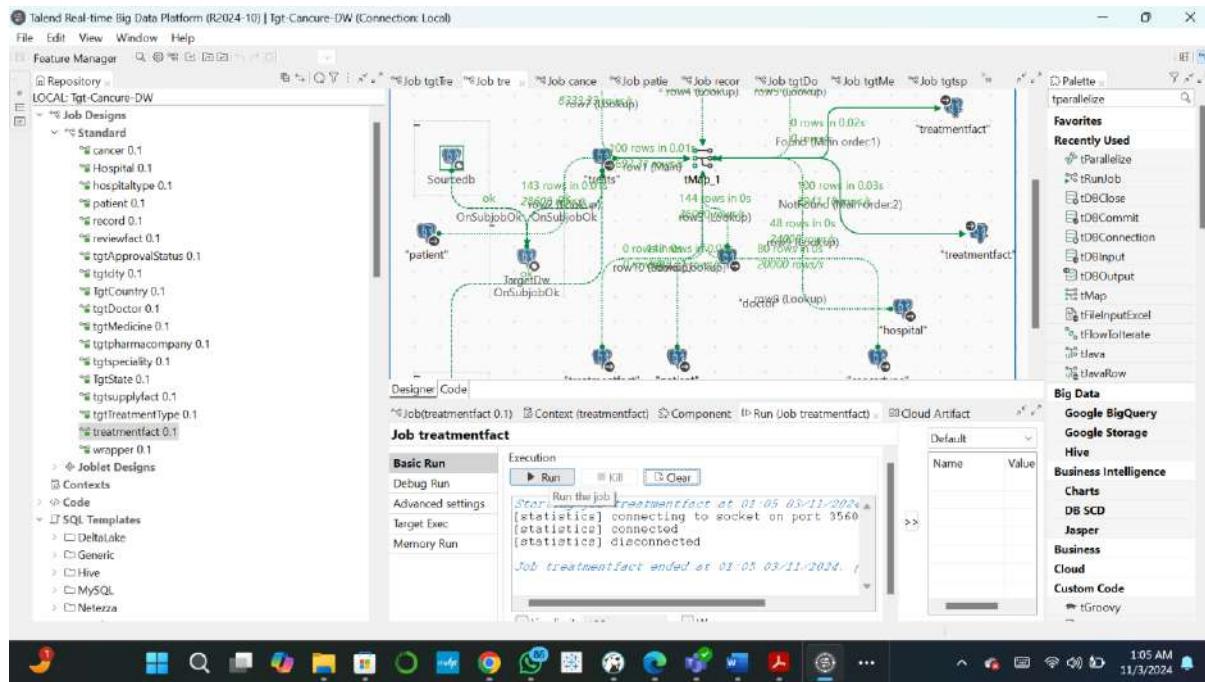
Database Tasks - General

Task executions: type a part of error message

Windows Taskbar: 10:4 AM 11/3/2024

Fact Tables:

15.TreatmentFact:



DBeaver 24.2.0 - <postgres> Tgt-Cancure-create

```

ALTER SEQUENCE medicinesupply_supplykey_seq RESTART WITH 1;
select * from medicinesupply m;

delete from treatmentfact;
ALTER SEQUENCE treatmentfact_treatmentkey_seq RESTART WITH 1;
select * from treatmentfact t;

delete from cancerstype;
ALTER SEQUENCE cancerstype_cancerkey_seq RESTART WITH 1;
select * from cancerstype;

delete from doctor;
ALTER SEQUENCE doctor_doctorkey_seq RESTART WITH 1;
select * from doctor;

```

treatmentfact.t

id	treatmentkey	hospitalkey	patientkey	patientid	name	phno	g
1	1	1	1	PAT001	Alice Johnson	555-0101	Fem
2	2	2	2	PAT002	Mary Smith	555-0102	Fem
3	3	1	3	PAT003	Linda White	555-0103	Fem
4	4	2	4	PAT004	Emily Clark	555-0201	Fem
5	5	3	5	PAT005	Jessica Thompson	555-0202	Fem
6	6	4	6	PAT006	Samantha Martin	555-0203	Fem
7	5	5	7	PAT007	Laura Adams	555-0301	Fem

Project - General

- Name: CancerType-DWI.sql
- DataSource: postres
- Cancure-DWI-Create.sql
- Cancure-final-create.sql
- Cancure-final-dw-create.sql
- CancureScript.sql
- City-DWI.sql
- City-DWI.sql

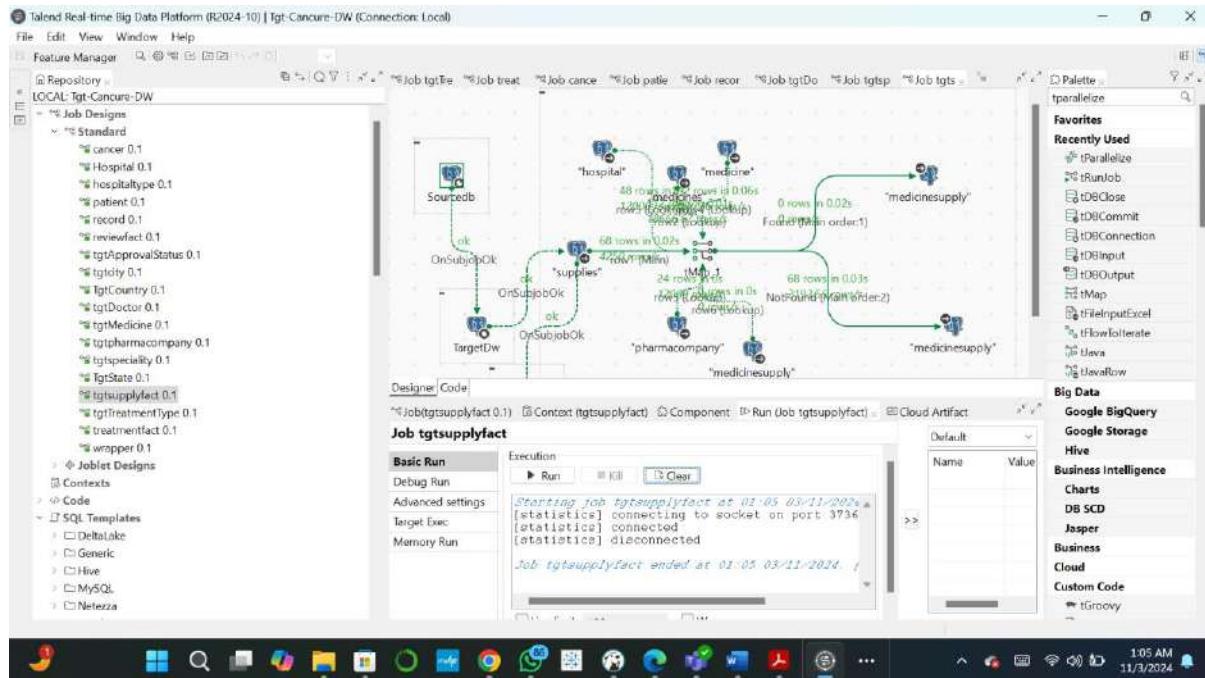
100 row(s) fetched - 0.000s, on 2024-11-03 at 01:05:22

Database Tasks - General

EST | en | Writable | Smart Insert | 203 | 1 [31] | Set 31 | 1 | Task executions: type a part of error message

10:5 AM 11/3/2024

16. SupplyFact:



DBeaver 24.2.0 - <postgres> Tgt-Cancure-create

File Edit Navigate Search SQL Editor Database Window Help

Database Navigator X Projects

Enter a part of object name here:

CanCure
Cancure-DWI
Cancure-Final-DWI
Cancure-Final-db
CoffeeStore
Northwind20
Source2
Target-Cancure-DW

Schemas

- public
 - Tables
 - Foreign Tables
 - Views
 - Materialized Views
 - Indices
 - Functions
 - Sequences
 - approvalstatus_approvalstatuskey_seq
 - cancuretype_cancerkey_seq

Project - General

Name: Data Source

- CancerType-DW.sql
- postres
- Cancure-DWI-Create.sql
- postres
- Cancure-final-create.sql
- postres
- CancureScript.sql
- postres
- City-DWI.sql
- postres

medicinesupply 1

select * from medicinesupply;

```

ALTER SEQUENCE medicinesupply_supplykey_seq RESTART WITH 1;
select * from medicinesupply #;

-- treatmentfact;
ALTER SEQUENCE treatmentfact_treatmentkey_seq RESTART WITH 1;
select * from treatmentfact t;

-- cancertype;
ALTER SEQUENCE cancertype_cancerkey_seq RESTART WITH 1;
select * from cancertype;

-- doctor;

```

medicineSupply 1

supplykey	hospitalkey	medicinekey	pharmaccompanykey	supplycost	quan	Value
1	25	8	1	900		1
2	31	8	1	900		
3	26	3	1	450		
4	31	3	1	450		
5	32	3	1	450		
6	37	publichospital	3	2	450	
7	33	9	1	630		

Refresh Style Cancel Export data 200 68 68 rows fetched - 0.001s, on 2024-11-03 at 01:06:12

Database Tasks - General

Tasks: type a part of task name here

EST an Writable Smart Insert 199 1 [32] Set 32 1

Task executions: type a part of error message

10:06 AM 11/3/2024

17. ReviewFact:

Talend Real-time Big Data Platform (R2024-10) | Tgt-Cancure-DW (Connection: Local)

File Edit View Window Help

Feature Manager

Repository LOCAL: Tgt-Cancure-DW

Job Designs

- Standard
 - cancer 0.1
 - Hospital 0.1
 - hospitality 0.1
 - patient 0.1
 - record 0.1
 - reviewfact 0.1
 - tgtApprovalStatus 0.1
 - tgtCtry 0.1
 - TgtCountry 0.1
 - tgtDoctor 0.1
 - tgtMedicine 0.1
 - tgtPharmCompany 0.1
 - tgtSpecialty 0.1
 - tgtState 0.1
 - tgtSupplyfact 0.1
 - tgtTreatmentType 0.1
 - treatmentfact 0.1
 - wrapper 0.1
- Joblet Designs
- Contexts
- Code
- SQL Templates
- DeltaLake
- Generic
- Hive
- MySQL
- Netezza

Designer Code

Job reviewfact

Basic Run

Execution

Starting job reviewfact at 01:06 03/11/2024 [statistics] connecting to socket on port 3406 [statistics] connected [statistics] disconnected

Job reviewfact ended at 01:06 03/11/2024. /Exe

Palletize

Favorites

Recently Used

- Parallelize
- tRunJob
- tDBClose
- tDBCommit
- tDBConnection
- tDBInput
- tDBOutput
- tMap
- tFileputExcel
- tFlowiterate
- tJava
- tJavaRow

Big Data

- Google BigQuery
- Google Storage
- Hive
- Business Intelligence
- Charts
- DB SCD
- Jasper
- Business
- Cloud
- Custom Code
- tGroovy

10:06 AM 11/3/2024

The screenshot shows the DBeaver interface with a SQL editor window containing a script to truncate and recreate tables in a PostgreSQL database. Below the editor is a results grid for the 'review' table, which has columns: reviewkey, patientkey, doctorkey, hospitalkey, reviewid, date, and comment. The data in the grid is as follows:

	reviewkey	patientkey	doctorkey	hospitalkey	reviewid	date	comment
1	1	1	1	1	REV001	2023-01-10	Excellent care!
2	2	2	2	2	REV002	2023-01-15	Good service
3	3	3	3	3	REV003	2023-01-20	Average experience
4	4	4	4	4	REV004	2023-02-10	Fantastic service
5	5	5	5	5	REV005	2023-02-15	Uncomfortable waiting time
6	6	6	6	6	REV006	2023-02-20	Professional staff
7	7	7	7	7	REV007	2023-03-10	Quick recovery

All the tables are successfully loaded into the Target DatawarehouseDB from the Relational Database through ETL.

Update Flow:

Hospital:

Before Update: Screenshot of the Hospital table in the data warehouse.

DBeaver 24.2.0 - <postgres> Tgt-Cancure-create

```

File Edit Navigate Search SQL Editor Database Window Help
Auto... SQL Commit Rollback
Database Navigator Projects
Enter a part of object name here
doctor_doctorkey_seq
hospital_hospitalkey_seq
hospitality_typekey_seq
medicine_medicinkey_seq
medicinesupply_supplykey_seq
patient_patientkey_seq
pharmaccompany_pharmaccompanykey_seq
record_recordkey_seq
review_reviewkey_seq
speciality_specialitykey_seq
state_statekey_seq
treatmentfact_treatmentkey_seq
treatmenttype_typekey_seq

Data types
Aggregate functions
Event Triggers
Extensions
Storage
Custom Info

Project - General
Name DataSource
CancerType-DWlsql postgres
Concure-DWI-Create.sql postgres
Concure-final-create.sql postgres
Concure-final-dw-create postgres
ConcureScript.sql postgres
City-DWI.sql postgres

hospital 1
select * from hospital

```

	hospitalkey	hospitalid	name	phone_number	oldemail
1	HOSP001	Los Angeles Cancer Center	123-456-7890	contact@lacc.co	
2	HOSP002	Los Angeles Oncology Institute	123-456-7891	info@lai.com	
3	HOSP003	San Francisco Cancer Treatment Cen	123-456-7892	contact@sftc.co	
4	HOSP004	Bay Area Cancer Hospital	123-456-7893	info@bach.com	
5	HOSP005	San Diego Oncology Center	123-456-7894	contact@sdonc	
6	HOSP006	Coastal Cancer Care	123-456-7895	info@ccc.com	
7	HOSP007	Houston Cancer Hospital	123-456-7896	contact@hch.co	

Refresh Save Cancel Export data 200 48 48 rows(s) fetched - 0.001s, on 2024-11-03 at 00:57:02

Database Tasks - General

Tasks: type a part of task name here

EST en Writable Smart Insert 245 - 1 [23] Set 23 | 1

Task executions: type a part of error message

12:57 AM 11/3/2024

Updating names of 2 records, in relational DB:

DBeaver 24.2.0 - <postgres> hospital-ref-final

```

File Edit Navigate Search SQL Editor Database Window Help
Auto... SQL Commit Rollback
Database Navigator Projects
Enter a part of object name here
45.79.206.143 - 45.79.206.143:3306
postgres - localhost:5432
Database
  CarCare
  Concure-DWI
  Concure-Final-DWI
  Concure-Final-db
  CoffeeStore
  Northwind20
  Source2
  Target-Concure-DW
  dummydb
  postgres
Administrator
System Info

Project - General
Name DataSource
CancerType-DWlsql postgres
Concure-DWI-Create.sql postgres
Concure-final-create.sql postgres
Concure-final-dw-create postgres
ConcureScript.sql postgres
City-DWI.sql postgres

hospital 1
select * from hospital

```

	hospitalid	name	phoneno	email	location	pty
9	HOSP011	Dallas Oncology Services	123-456-7900	contact@dos.com	Dallas	Publ
10	HOSP012	Metroplox Cancer Center	123-456-7901	info@mcc.com	Dallas	Priv
11	HOSP013	Miami Cancer Institute	123-456-7902	contact@mci.com	Miami	Publ
12	HOSP014	Florida Coastal Oncology	123-456-7903	info@fco.com	Miami	Priv

-- Ensure phone numbers and email addresses are unique and consistent for the real application.

```

UPDATE Hospital
SET Name = 'Los Angeles Cancer Research Institute'
WHERE HospitalID = 'HOSP001';

UPDATE Hospital
SET Name = 'Bay Area Cancer Research Centre'
WHERE HospitalID = 'HOSP004';

```

Refresh Save Cancel Open data 200 48 48 rows(s) fetched - 0.002s, on 2024-11-03 at 10:36:45

Database Tasks - General

Tasks: type a part of task name here

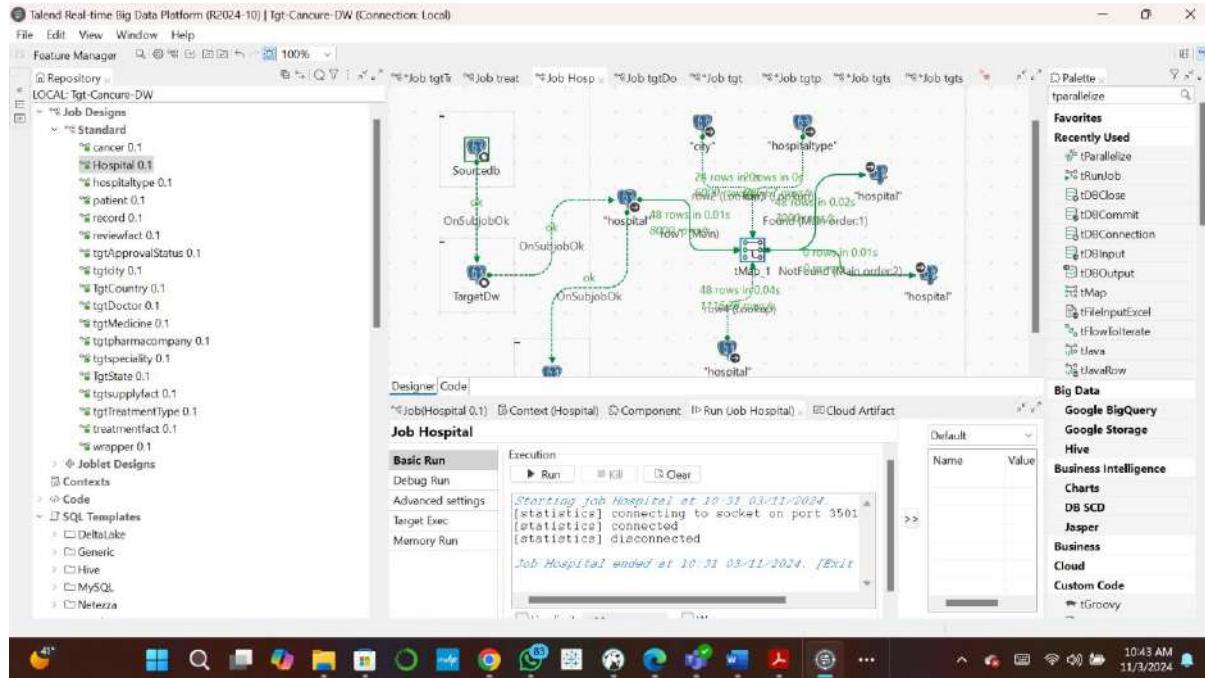
Name Last Run (System time) Last Result Type
Custom 2024-10-10 20:15:00 Success Data imp...

Task executions: type a part of error message

System time Duration Result

10:42 AM 11/3/2024

Executing the update flow



Running the “Select * from Hospital” in DatawarehouseDB to see if the update flow is successful,

hospitalkey	hospitalid	name	phone_number	oldemail
17	19	HOSP019	New York City Cancer Center	123-456-7908
18	1	HOSP001	Los Angeles Cancer Research Instn	123-456-7890
19	4	HOSP004	Bay Area Cancer Research Centre	123-456-7893
20	20	HOSP020	Empire State Oncology	123-456-7909
...

The change is successfully implemented, executed and reflected in the Data Warehouse DB. An **update of the Name** in hospitalID one and four is successfully reflecting

Pharma company:

Before Update,

The screenshot shows the DBBeaver interface with the following details:

- Database Navigator:** Shows various sequence definitions like doctor_doctorkey_seq, hospital_hospitalkey_seq, etc.
- SQL Editor:** Contains several ALTER SEQUENCE commands to restart sequences at 1.
- Data Grid:** Displays a table named "pharmacompany" with 8 rows, each containing a CompanyID, Name, Email, Rating, and CityKey.
- Task Bar:** Shows system icons and the date/time (11/3/2024, 12:57 AM).

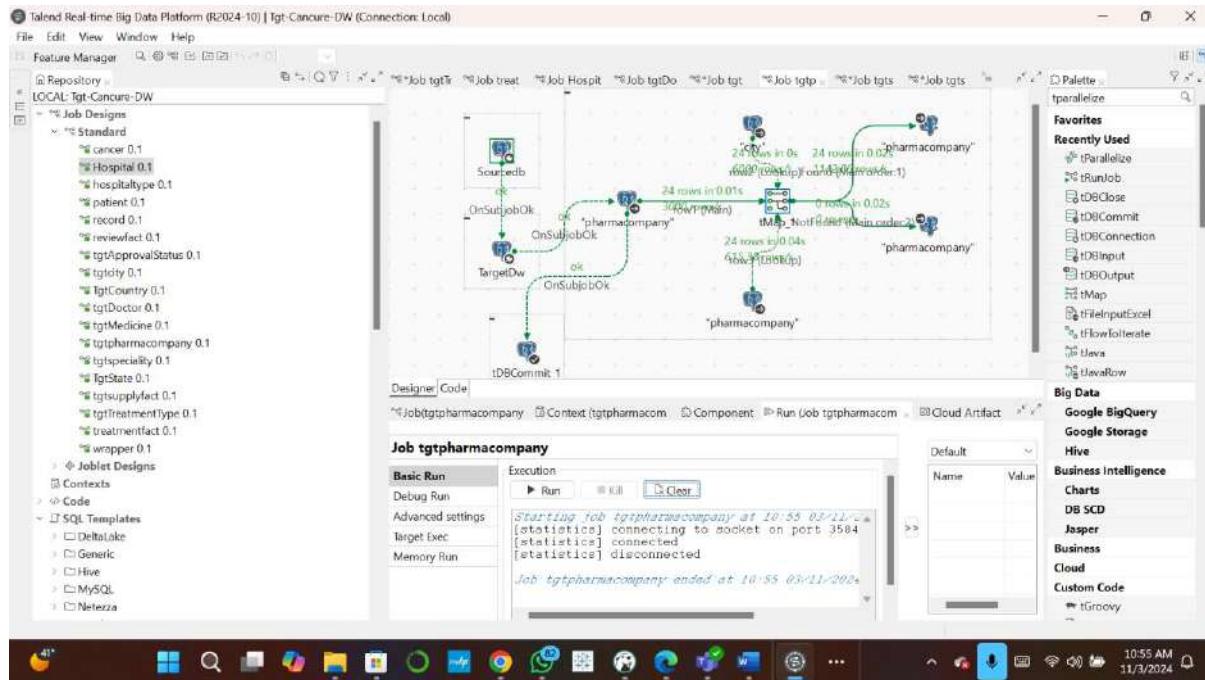
CompanyID	Name	Email	Rating	CityKey
PH001	Los Angeles	contact@lab	4	1
PH002	San Francisco	info@sfbio.c	5	2
PH003	San Diego	pt.contact@sdr	3	3
PH004	Houston	[San Francisco Biosciences]	4	4
PH005	Austin	Pharr.contact@ap	2	5
PH006	Dallas	Drug L info@ddd.c	4	6
PH007	Miami	Medic contact@mm	5	7
PH008	Orlando	Hea info@ohs.co	3	8

Updating the row1 email from 'contact@labiotech.com' to 'info@labiotech.com' in relational model,

The screenshot shows the DBBeaver interface with the following details:

- Database Navigator:** Shows the "postgres" database and its contents.
- SQL Editor:** Contains an UPDATE statement for the "pharmacompany" table, setting the "Email" field to "info@labiotech.com" where "CompanyID" is "PH001".
- Statistics:** Shows 1 updated row.
- Task Bar:** Shows system icons and the date/time (11/3/2024, 10:54 AM).

Running the update flow in Talend,



The update is reflecting in the pharma company dimension table, as we can see the updated email.

DBeaver 24.2.0 - <postgres> - Tgt-Concure create

The screenshot shows the DBeaver interface. The SQL Editor tab contains several SQL commands related to creating tables and sequences for the "pharmacompany" dimension table. The Database Navigator tab shows the database structure, including the "pharmacompany" table. The Table Viewer tab displays the contents of the "pharmacompany" table, which includes columns like companykey, companyid, name, email, rating, and citykey. The table currently has 24 rows. The bottom section shows a project tree and a Database Tasks panel.

Doctor:

Before the Update, the loaded data in the data warehouse shows the following output

The screenshot shows the DBeaver interface with the 'doctor' table selected in the SQL Editor. The table has columns: doctorkey, doctorid, fname, lname, email, myoe, and specialitykey. The data is as follows:

doctorkey	doctorid	fname	lname	email	myoe	specialitykey
1	DOC001	Dr. Emily	Stone	@lacc.com	10	1
2	DOC002	Dr. James	Wilson	@lacc.com	8	5
3	DOC003	Dr. Clara	Coswalt	@lacc.com	5	3
4	DOC004	Dr. Richard	Veeble	@lacc.com	15	2
5	DOC005	Dr. Miranda	Imbalance	@lacc.com	12	4
6	DOC006	Dr. Alex	Kare	@lacc.com	7	1
7	DOC007	Dr. Charlotte	Clings	@stfc.c	20	5
8	DOC008	Dr. Addison	Amontgome	@lacc.com	22	3

Now I am updating the the row in my relational database,

The screenshot shows the DBeaver interface with the following update query in the SQL Editor:

```
UPDATE Doctor
SET Name = 'Dr. Ragavi Murugavel', Email = 'rmurugavel@lacc.com'
WHERE DoctorID = 'DOC001';
```

The Statistics window shows:

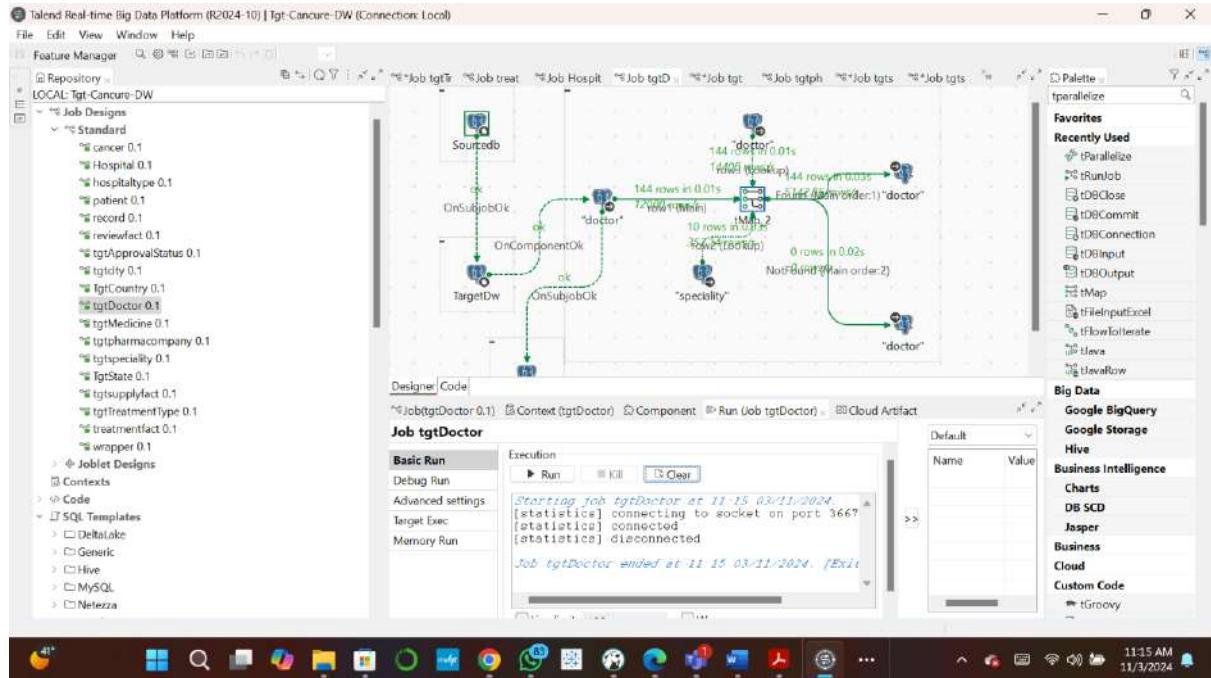
Name	Value
Updated Rows	1

The Database Tasks window shows the task was successful:

Name	Last Run (System time)	Last Result	Type
Custom 2024-10-10 20:15:00	Success	Data imp.	

Changing the name and email.

Job is executed,



The change is reflected in the datawarehouse table,

DBVisualer 24.0 - <postgres> Tgt-Cancure-create

Project - General

Name	DataSource
CancerType-DWI.sql	postgres
Concure-DWI-CREATE.sql	postgres
Concure-final-create.sql	postgres
Concure-final-dwi-create.sql	postgres
ConcureScript.sql	postgres
City-DWI.sql	postgres

Database Tasks - General

Name	Last Run [System time]	Last Result	Type
Custom 2024-10-10 20:15:00	Success		Data imp...

Medicines: Before Update

DBeaver 24.2.0 - <postgres> Tgt-Cancure-create

File Edit Navigate Search SQL Editor Database Window Help

Database Navigator X Projects Enter a part of object name here

medicine

doctor_doctorkey_seq
hospital_hospitalkey_seq
hospitality_typekey_seq
medicine_medicinekey_seq
medicinesupply_supplykey_seq
patient_patientkey_seq
pharmaccompany_pharmaccompanykey_seq
record_recordkey_seq
review_reviewkey_seq
specialty_specialitykey_seq
state_statekey_seq
treatmentfact_treatmentkey_seq
treatmenttype_typekey_seq

Data types Aggregate functions Event Triggers Extensions Storage System Info

Project - General X

Name DataSource

CancerType-DWI.sql postgres
Cancure-DWI-Create.sql postgres
Cancure-final-create.sql postgres
Cancure-final-dw-create.sql postgres
CancureScript.sql postgres
City-DWI.sql postgres

medicine X

select * from medicine m

medicinekey	medicined	name	no_of_trials	approvalstatuskey
1	MED1	OncBreast	5	1
2	MED2	OncOvarian	3	2
3	MED3	OncLeukem	4	1
4	MED4	OncGloroma	6	1
5	MED5	OncColore	3	2
6	MED6	OncProstati	5	1
7	MED7	OncMelano	4	3
8	MED8	OncNeurob	6	1

112 rows] fetched - 0.002s, on 2024-11-03 at 00:59:45

Database Tasks - General X

Tasks: type a part of task name here

EST | en | Writable | Smart Insert | 233 - 1 [28] | Set: 28 | 1

12:59 AM 11/3/2024

Now updating the rows in the relational model,

DBeaver 24.2.0 - <postgres> Medicines-rel-final

File Edit Navigate Search SQL Editor Database Window Help

Database Navigator X Projects Enter a part of object name here

45.79.206.143 - 45.79.206.143:3306

postgres - localhost:5433

Database Source2 Target-Cancure-DW

Medicines

select * from medicines;

UPDATE Medicines
SET Name = 'BreastCure', Cost = 550.00
WHERE MedicineID = 'MED1';

Statistics 1 X

Name Value

Updated Rows 1

Query UPDATE Medicines
SET Name = 'BreastCure', Cost = 550.00
WHERE MedicineID = 'MED1'

Start time Sun Nov 03 11:34:31 EST 2024

Finish time Sun Nov 03 11:34:31 EST 2024

Database Tasks - General X

Tasks: type a part of task name here

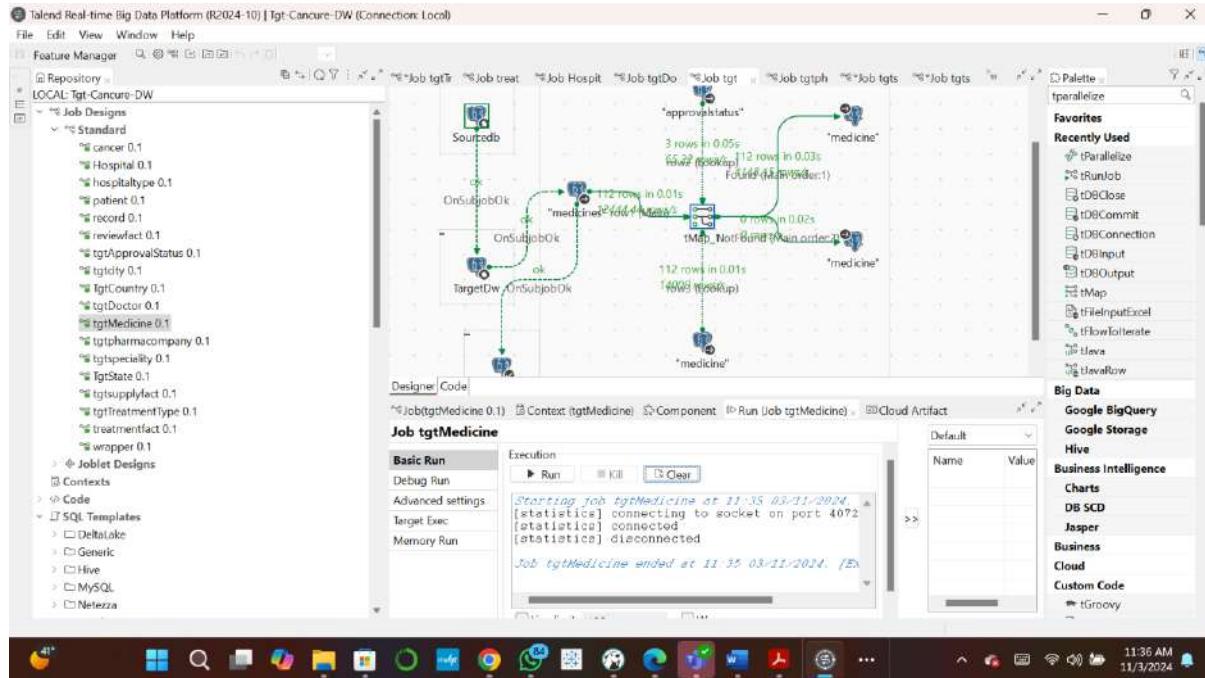
Name Last Run (System time) Last Result Type

Custom 2024-10-10 20:15:00 Success Data imp...

System time Duration Result

11:34 AM 11/3/2024

I am changing the name and cost for one record,
Running the medicine job



The changes are reflected in the medicine data warehouse table

DBVisualizer 24.2.0 - <postgres> Tgt-Cancure-create

File Edit Navigate Search SQL Editor Database Window Help

Database Navigator X Projects Enter a part of object name here

45.79.206.143 - 45.79.206.142:3306

postgres - localhost:5433

Database

CarCure

Cancure-DWI

Cancure-Final-DWI

Cancure-Final-db

CoffeeStore

Northwind20

Source2

Target-Cancure-DW

dummymdb

postgres

Administrator

System Info

Project - General

Name DataSource

Concure-DWI.sql postgres

Concure-DWI-CREATE.sql postgres

Concure-final-create.sql postgres

Concure-final-dwi-create.sql postgres

ConcureScript.sql postgres

City-DWI.sql postgres

<postgres> <postgres> <postgres> <postgres> <postgres> <postgres> <postgres> <postgres> <postgres> <postgres>

delete from patient;
ALTER SEQUENCE patient_patientkey_seq RESTART WITH 1;
select * from patient p;

delete from medicine;
ALTER SEQUENCE medicine_medicinekey_seq RESTART WITH 1;
select * from medicine m;

delete from approvalstatus;

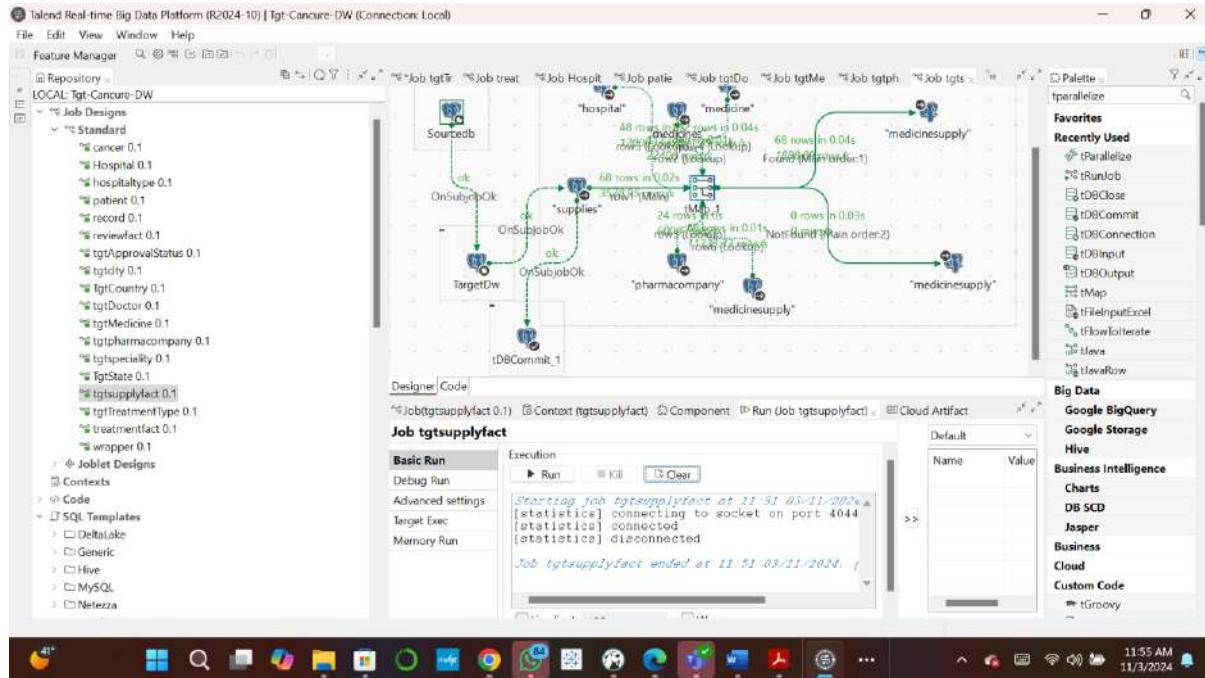
medicine 1

medicinkey	name	no_of_trials	approvalstatuskey
109	MED110	RM Neuroblast	8
110	MED111	RM Lymphoma	5
111	MED112	RM Thyroid	6
112	MED1	BreastCure	5

112 row(s) fetched - 0.002s, on 2024-11-03 at 11:35:38

The name change is reflected.

Now running the supply fact table,



The price changes is also reflected to 550 and correspondingly margin per unit becomes the selling price-supply cost(cost price)[1000-550]=450 which is the margin per cost for medicine key 1. The changes are reflected in the supplyfact table screenshot below.

```

delete from medicinesupply;
ALTER SEQUENCE medicinesupply_supplykey_seq RESTART WITH 1;
select * from medicinesupply m;

delete from treatmentfact;
ALTER SEQUENCE treatmentfact_treatmentkey_seq RESTART WITH 1;
select * from treatmentfact t;

delete from cancerType;

```

medicinesupply 1 ×

select * from medicinesupply m Enter a SQL expression to filter results (use Ctrl+Space)

	medicinekey	pharmaccompanykey	supplycost	quantity	sellingprice	marginperunit	leadtime
19	1	1	550	500	1,000	450	
20	2	1	610	400	1,000	390	
21	3	1	450	300	1,000	550	
22	4	1	720	500	1,000	280	
23	5	1	550	600	1,000	450	

Refresh Save Cancel Export data 200 68

Database Tasks - General

Patient:

Before Updating, the Datawarehouse table

DBeaver 24.2.0 - <postgres> Tgt-Cancure-create

File Edit Navigate Search SQL Editor Database Window Help

Database Navigator X Projects

Enter a part of object name here

doctor_doctorkey_seq
hospital_hospitalkey_seq
hospitaltype_typekey_seq
medicine_medicinakey_seq
medicinesupply_supplykey_seq
patient_patientkey_seq
pharmaccompany_pharmaccompanykey_seq
record_recordkey_seq
review_reviewkey_seq
speciality_specialitykey_seq
state_statekey_seq
treatmentfact_treatmentkey_seq
treatmenttype_typekey_seq

Data types
Aggregate functions
Event Triggers
Extensions
Storage
System Info

Project - General X

Name DataSource

- CancerType-DWI.sql postgres
- Cancure-DWI-Create.sql postgres
- Cancure-final-create.sql postgres
- Cancure-final-dw-create.sql postgres
- CancureScript.sql postgres
- City-DWI.sql postgres

SELECT * FROM treatmenttypekey

```
delete from patient;
ALTER SEQUENCE patient_patientkey_seq RESTART WITH 1;
select * from patient p;
```

```
delete from medicine;
ALTER SEQUENCE medicine_medicinekey_seq RESTART WITH 1;
select * from medicine m;
```

```
delete from approvalstatus;
ALTER SEQUENCE approvalstatus_approvalstatuskey_seq RESTART WITH 1;
select * from approvalstatus;
```

patient 1

	patientkey	patientid	fname	lname	gender	diseasehistory	age
Text	1	PAT001	Alice Johnson	555-0101	Female	No history	45
Text	2	PAT002	Mary Smith	555-0102	Female	Family history	50
Text	3	PAT003	Linda White	555-0103	Female	No history	47
Text	4	PAT004	Emily Clark	555-0201	Female	Benign lump	38
Text	5	PAT005	Jessica Thom	555-0202	Female	Family history	42
Text	6	PAT006	Samantha M.	555-0203	Female	No history	55
Text	7	PAT007	Laura Adams	555-0301	Female	Prior lump	49
Text	8	PAT008	Nancy Moon	555-0302	Female	No history	53

Refresh Save Cancel Export data 200 143 143 rows fetched - 0.000s, on 2024-11-03 at 01:00:28

Database Tasks - General X

Tasks: type a part of task name here

EST en Writable Smart Insert 228 1 [27] Sel: 27 | 1

Task executions: type a part of error message

100 AM 11/3/2024

Updating in a relational database, I am updating the name of the patient

DBeaver 24.2.0 - <postgres> Patient-rel-final

File Edit Navigate Search SQL Editor Database Window Help

Database Navigator X Projects

Enter a part of object name here

45.79.206.143 - 45.79.206.143:3306

postgres - localhost:5433

Database

- CanCure
- Cancure-DWI
- Cancure-Final-DWI
- Cancure-Final-db
- CoffeeStore
- Northwind20
- Source2
- Target-Cancure-DW
- dummymydb
- postgres

Administrator System Info

Project - General X

Name DataSource

- CancerType-DWI.sql postgres
- Cancure-DWI-Create.sql postgres
- Cancure-final-create.sql postgres
- Cancure-final-dw-create.sql postgres
- CancureScript.sql postgres
- City-DWI.sql postgres

SELECT * FROM patient;

```
UPDATE Patient
SET Name = 'Sarah Williams'
WHERE PatientID = 'PAT001';
```

Statistics 1 X

Name	Value
Updated Rows	1

Query

UPDATE Patient
SET Name = 'Sarah Williams'
WHERE PatientID = 'PAT001'

Start time Sun Nov 03 11:46:55 EST 2024

Finish time Sun Nov 03 11:46:55 EST 2024

Database Tasks - General X

Tasks: type a part of task name here

Name Last Run (System time) Last Result Type

Custom 2024-10-10 20:15:00 Success Data imp...

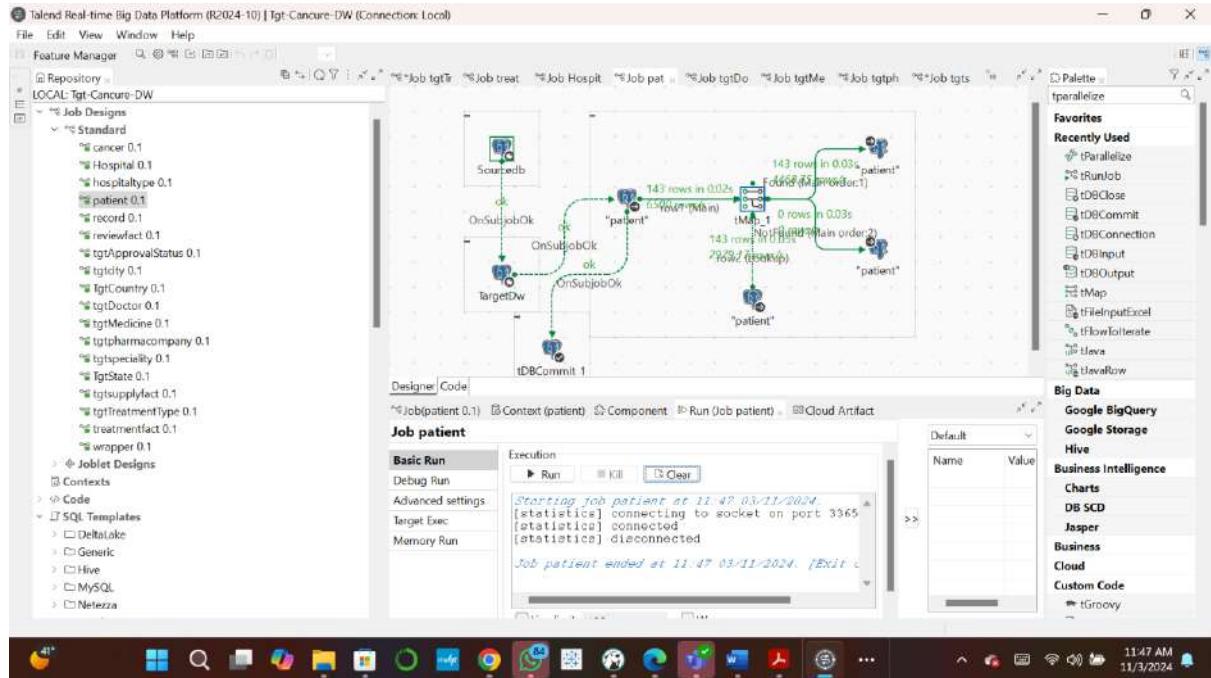
Task executions: type a part of error message

System time Duration Result

EST en Writable Smart Insert 240 1 [72] Sel: 72 | 3

11:46 AM 11/3/2024

Run the update job,



An update of the name is reflected in the data warehouse patient table,

patientkey	patientid	name	phno	gender	diseasehistory	age
141	PAT142	Raymond Clark	555-2007	Male	Difficulty swallowing	42
142	PAT143	Walter Lewis	555-2008	Male	Low pain	
143	PAT001	Sarah Williams	555-0101	Female	No history	

Thus the Update flows are working properly.

Control Flow:

Now we are again deleting all the data from our data warehouse and confirming if its deleted using the following sql queries,

```
delete from review ;
ALTER SEQUENCE review_reviewkey_seq RESTART WITH 1;
select * from review;
delete from medicinesupply;
ALTER SEQUENCE medicinesupply_supplykey_seq RESTART WITH 1;
select * from medicinesupply m ;
delete from treatmentfact;
ALTER SEQUENCE treatmentfact_treatmentkey_seq RESTART WITH 1;
select * from treatmentfact t ;
delete from cancertype;
ALTER SEQUENCE cancertype_cancerkey_seq RESTART WITH 1;
select * from cancertype;
delete from doctor;
ALTER SEQUENCE doctor_doctorkey_seq RESTART WITH 1;
select * from doctor;
delete from speciality;
ALTER SEQUENCE speciality_specialitykey_seq RESTART WITH 1;
select * from speciality;
delete from record;
ALTER SEQUENCE record_recordkey_seq RESTART WITH 1;

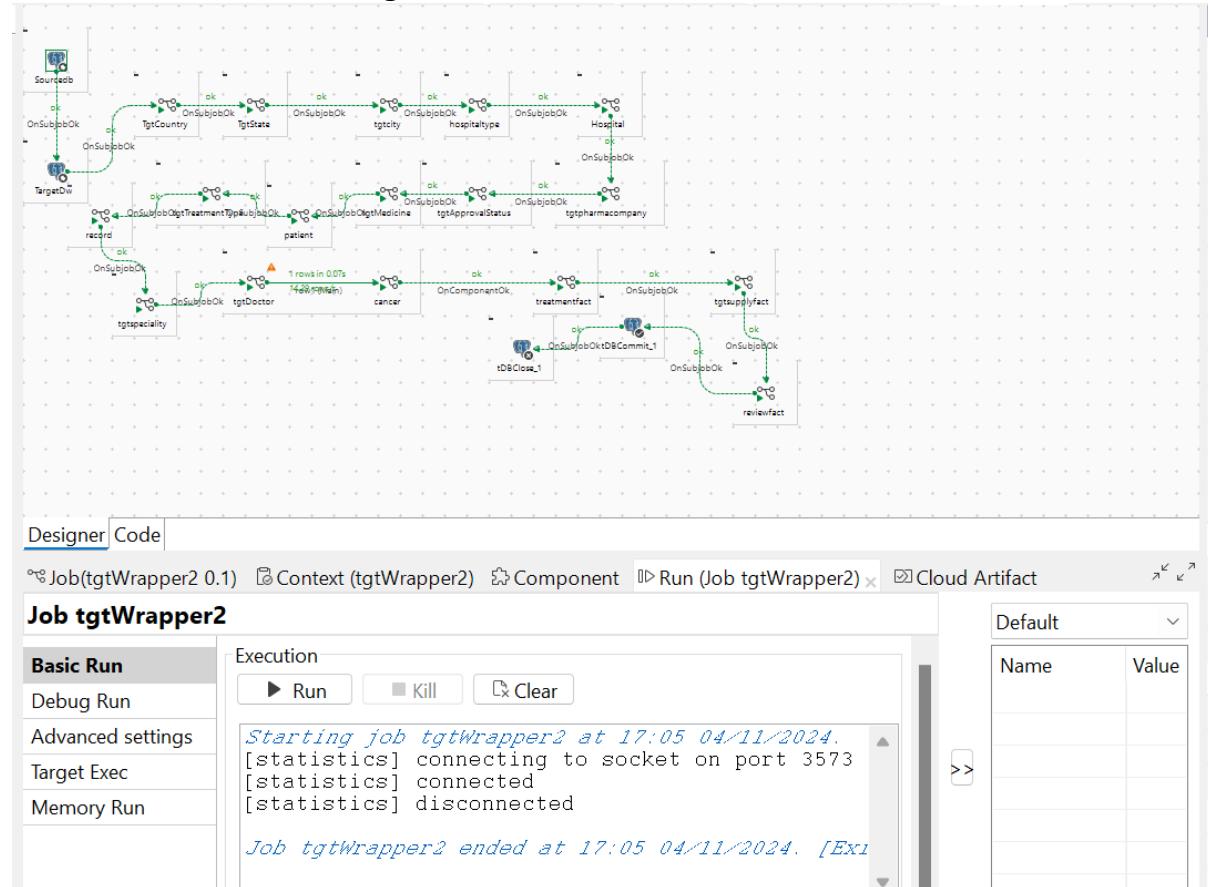
select * from record r ;
delete from treatmenttype;
ALTER SEQUENCE treatmenttype_typekey_seq RESTART WITH 1;
select * from treatmenttype t ;
delete from patient;
ALTER SEQUENCE patient_patientkey_seq RESTART WITH 1;
select * from patient p ;
delete from medicine;
ALTER SEQUENCE medicine_medicinekey_seq RESTART WITH 1;
select * from medicine m ;
delete from approvalstatus;
ALTER SEQUENCE approvalstatus_approvalstatuskey_seq RESTART WITH 1;
select * from approvalstatus;
delete from pharmaccompany;
ALTER SEQUENCE pharmaccompany_pharmaccompanykey_seq RESTART WITH 1;
```

```

select * from pharmaccompany;
delete from hospital;
ALTER SEQUENCE hospital_hospitalkey_seq RESTART WITH 1;
select * from hospital;
delete from hospitaltype;
ALTER SEQUENCE hospitaltype_typekey_seq RESTART WITH 1;
select * from hospitaltype h ;
delete from city;
ALTER SEQUENCE city_citykey_seq RESTART WITH 1;
select * from city;
delete from state;
ALTER SEQUENCE state_statekey_seq RESTART WITH 1;
select * from state;
delete from country;
ALTER SEQUENCE country_countrykey_seq RESTART WITH 1;
select * from country;

```

After this we are running the control flow,



After running it successfully all the data is populated in all the tables again, the proof is attached below,

```

DBeaver 24.2.0 - <postgres> Tgt-Cancure-create
File Edit Navigate Search SQL Editor Database Window Help
Auto... postgres public@Target-Cancure-DW
Database Navigator Projects
Enter a part of object name here
45.79.206.143 - 45.79.206.143:3306
postgres - localhost:5433

<postgres> public@Target-Cancure-DW
--> ALTER SEQUENCE city_citykey_seq RESTART WITH 1;
--> select * from city;
--> delete from state;
--> ALTER SEQUENCE state_statekey_seq RESTART WITH 1;
--> select * from state;
--> delete from country;
--> ALTER SEQUENCE country_countrykey_seq RESTART WITH 1;
--> select * from country;
-->

country 1
--> select * from country
--> Enter a SQL expression to filter results (use Ctrl+Space)
Grid
+-----+-----+
| id  | name |
+-----+-----+
| 1   | United States |
| 2   | India           |
+-----+-----+
Record 1
Row 1
1
2

Project - General
Name: CancerType-DWI.sql DataSource: postgres
Cancure-DWI-Create.sql postgres
Cancure-final-create.sql postgres
Cancure-final-dwi-create postgres
CancureScript.sql postgres
City-DWI.sql postgres

Database Tasks - General
Tasks: type a part of task name here
Name Last Run (System time) Last Result Type
Custor 2024-10-10 20:15:00 Success Data imp...
Task executions: type a part of error message
System time Duration Result
EST en Writable Smart Insert 261 : 1 (22) Sel 21 : 1
Export data 200 2 2 row(s) fetched - 0.000s, on 2024-11-04 at 17:40:36
5:40 PM 11/4/2024

DBeaver 24.2.0 - <postgres> Tgt-Cancure-create
File Edit Navigate Search SQL Editor Database Window Help
Auto... postgres public@Target-Cancure-DW
Database Navigator Projects
Enter a part of object name here
45.79.206.143 - 45.79.206.143:3306
postgres - localhost:5433

<postgres> public@Target-Cancure-DW
--> ALTER SEQUENCE city_citykey_seq RESTART WITH 1;
--> select * from city;
--> delete from state;
--> ALTER SEQUENCE state_statekey_seq RESTART WITH 1;
--> select * from state;
--> delete from country;
--> ALTER SEQUENCE country_countrykey_seq RESTART WITH 1;
--> select * from country;
-->

state 1
--> select * from state
--> Enter a SQL expression to filter results (use Ctrl+Space)
Grid
+-----+-----+-----+
| id  | statename | countrykey |
+-----+-----+-----+
| 1   | California | 1          |
| 2   | Texas      | 1          |
| 3   | Florida    | 1          |
| 4   | New York   | 1          |
| 5   | Illinois   | 1          |
+-----+-----+-----+
Record 1
Row 1
1
2
3
4
5

Project - General
Name: CancerType-DWI.sql DataSource: postgres
Cancure-DWI-Create.sql postgres
Cancure-final-create.sql postgres
Cancure-final-dwi-create postgres
CancureScript.sql postgres
City-DWI.sql postgres

Database Tasks - General
Tasks: type a part of task name here
Name Last Run (System time) Last Result Type
Custor 2024-10-10 20:15:00 Success Data imp...
Task executions: type a part of error message
System time Duration Result
EST en Writable Smart Insert 257 : 1 (20) Sel 21 : 1
Export data 200 8 8 row(s) fetched - 0.001s, on 2024-11-04 at 17:40:43
5:40 PM 11/4/2024

DBeaver 24.2.0 - <postgres> Tgt-Cancure-create
File Edit Navigate Search SQL Editor Database Window Help
Auto... postgres public@Target-Cancure-DW
Database Navigator Projects
Enter a part of object name here
45.79.206.143 - 45.79.206.143:3306
postgres - localhost:5433

<postgres> public@Target-Cancure-DW
--> ALTER SEQUENCE city_citykey_seq RESTART WITH 1;
--> select * from city;
--> delete from state;
--> ALTER SEQUENCE state_statekey_seq RESTART WITH 1;
--> select * from state;
--> delete from country;
--> ALTER SEQUENCE country_countrykey_seq RESTART WITH 1;
--> select * from country;
-->

city 1
--> select * from city
--> Enter a SQL expression to filter results (use Ctrl+Space)
Grid
+-----+-----+-----+-----+
| id  | cityname | population | statekey |
+-----+-----+-----+-----+
| 1   | Los Angeles | 380,000 | 1          |
| 2   | San Francisco | 880,000 | 1          |
| 3   | San Diego | 1,420,000 | 1          |
| 4   | Houston | 2,300,000 | 2          |
| 5   | Austin | 980,000 | 2          |
+-----+-----+-----+-----+
Record 1
Row 1
1
2
3
4
5

Project - General
Name: CancerType-DWI.sql DataSource: postgres
Cancure-DWI-Create.sql postgres
Cancure-final-create.sql postgres
Cancure-final-dwi-create postgres
CancureScript.sql postgres
City-DWI.sql postgres

Database Tasks - General
Tasks: type a part of task name here
Name Last Run (System time) Last Result Type
Custor 2024-10-10 20:15:00 Success Data imp...
Task executions: type a part of error message
System time Duration Result
EST en Writable Smart Insert 253 : 1 (21) Sel 21 : 1
Export data 200 24 24 row(s) fetched - 0.002s, on 2024-11-04 at 17:40:47
5:40 PM 11/4/2024

```

DBeaver 24.2.0 - <postgres> Tgt-Cancure-create

```

File Edit Navigate Search SQL Editor Database Window Help
Auto... SQL... Commit Rollback postgres public@Target-Cancure-DW
Database Navigator Projects
Enter a part of object name here
45.79.206.143 - 45.79.206.143:3306
postgres - localhost:5433

delete from hospital;
ALTER SEQUENCE hospital_hospitalkey_seq RESTART WITH 1;
select * from hospital;

delete from hospitaltype;
ALTER SEQUENCE hospitaltype_typekey_seq RESTART WITH 1;
select * from hospitaltype n;

delete from city;
ALTER SEQUENCE city_citykey_seq RESTART WITH 1;
select * from city;

```

hospitaltype 1

typkey	typename	Value
1	Public	1
2	Private	

Project - General

Name	DataSource
CancerType-DWI.sql	postgres
Cancure-DWI-Create.sql	postgres
Cancure-final-create.sql	postgres
Cancure-final-dwi-create.sql	postgres
CancureScript.sql	postgres
City-DWI.sql	postgres

EST en Writable Smart Insert 249: 1 [30] Set 30 | 1 5:40 PM 11/4/2024

DBeaver 24.2.0 - <postgres> Tgt-Cancure-create

```

File Edit Navigate Search SQL Editor Database Window Help
Auto... SQL... Commit Rollback postgres public@Target-Cancure-DW
Database Navigator Projects
Enter a part of object name here
45.79.206.143 - 45.79.206.143:3306
postgres - localhost:5433

delete from hospital;
ALTER SEQUENCE hospital_hospitalkey_seq RESTART WITH 1;
select * from hospital;

delete from hospitaltype;
ALTER SEQUENCE hospitaltype_typekey_seq RESTART WITH 1;
select * from hospitaltype n;

delete from city;
ALTER SEQUENCE city_citykey_seq RESTART WITH 1;
select * from city;

```

hospital 1

hospitalkey	hospitalid	name	phone_number	oldemail	newemail
1	HOSP002	Los Angeles Oncology Institute	123-456-7891	info@laoi.com	[NULL]
2	HOSP005	San Diego Oncology Center	123-456-7894	contact@sdcnc.com	[NULL]
3	HOSP006	Coastal Cancer Care	123-456-7895	info@ccc.com	[NULL]
4	HOSP007	Houston Cancer Hospital	123-456-7896	contact@hch.com	[NULL]

Project - General

Name	DataSource
CancerType-DWI.sql	postgres
Cancure-DWI-Create.sql	postgres
Cancure-final-create.sql	postgres
Cancure-final-dwi-create.sql	postgres
CancureScript.sql	postgres
City-DWI.sql	postgres

EST en Writable Smart Insert 245: 1 [25] Set 25 | 1 5:41 PM 11/4/2024

DBeaver 24.2.0 - <postgres> Tgt-Cancure-create

```

File Edit Navigate Search SQL Editor Database Window Help
Auto... SQL... Commit Rollback postgres public@Target-Cancure-DW
Database Navigator Projects
Enter a part of object name here
45.79.206.143 - 45.79.206.143:3306
postgres - localhost:5433

delete from approvalstatus;
ALTER SEQUENCE approvalstatus_approvalstatuskey_seq RESTART WITH 1;
select * from approvalstatus;

delete from pharmacompany;
ALTER SEQUENCE pharmacompany_pharmacompanykey_seq RESTART WITH 1;
select * from pharmacompany;

delete from hospital;
ALTER SEQUENCE hospital_hospitalkey_seq RESTART WITH 1;

```

pharmacompany 1

pharmacompanykey	companyid	name	email	rating	citykey
1	Ph002	San Francisco Biomedics	info@fbio.com	5	2
2	Ph003	San Diego Pharma Innovations	contact@dpic.com	3	3
3	Ph004	Houston Research Pharma	info@hrp.com	4	4
4	Ph005	Austin Pharmaceuticals	contact@ap.com	2	5
5	Ph006	Dallas Drug Development	info@dd.com	4	6

Project - General

Name	DataSource
CancerType-DWI.sql	postgres
Cancure-DWI-Create.sql	postgres
Cancure-final-create.sql	postgres
Cancure-final-dwi-create.sql	postgres
CancureScript.sql	postgres
City-DWI.sql	postgres

EST en Writable Smart Insert 241: 1 [28] Set 28 | 1 5:41 PM 11/4/2024

DBeaver 24.2.0 - <postgres> Tgt-Concure-create

```

File Edit Navigate Search SQL Editor Database Window Help
Auto Commit Rollback
Database Navigator Projects
Enter a part of object name here
45.79.206.143 - 45.79.206.143:3306
postgres - localhost:5433

delete from approvalstatus;
ALTER SEQUENCE approvalstatus_approvalstatuskey_seq RESTART WITH 1;
select * from approvalstatus;

delete from pharmaccompany;
ALTER SEQUENCE pharmaccompany_pharmaccompanykey_seq RESTART WITH 1;
select * from pharmaccompany;

delete from hospital;
ALTER SEQUENCE hospital_hospitalkey_seq RESTART WITH 1;

```

approvalstatus 1

Grid	approvalstatuskey	status
1	1	Approved
2	2	On hold
3	3	Unapproved

Record

Project - General

- Name
 - CancerType-DWI.sql
 - Concure-DWI-Create.sql
 - Concure-final-create.sql
 - Concure-final-dwi-create.sql
 - ConcureScript.sql
 - City-DWI.sql
- DataSource

Database Tasks - General

Name	Last Run (System time)	Last Result	Type
Custom 2024-10-10 20:15:00	Success	Data imp...	

EST on Writable Smart Insert 237 : 1 [29] Set 29 | 1 5:41 PM 11/4/2024

DBeaver 24.2.0 - <postgres> Tgt-Concure-create

```

File Edit Navigate Search SQL Editor Database Window Help
Auto Commit Rollback
Database Navigator Projects
Enter a part of object name here
45.79.206.143 - 45.79.206.143:3306
postgres - localhost:5433

ALTER SEQUENCE patient_patientkey_seq RESTART WITH 1;
select * from patient p;

delete from medicine;
ALTER SEQUENCE medicine_medicinekey_seq RESTART WITH 1;
select * from medicine m;

delete from approvalstatus;
ALTER SEQUENCE approvalstatus_approvalstatuskey_seq RESTART WITH 1;
select * from approvalstatus;

```

medicine 1

Grid	medicinekey	medined	name	no_of_trials	approvalstatuskey
1	1	MED2	OncOOverien	3	20
2	2	MED3	OncLeukemia	4	12
3	3	MED4	OncGloMe	6	12
4	4	MED5	OncColorRectal	3	20
5	5	MED6	OncProstate	5	12

Record

Project - General

- Name
 - CancerType-DWI.sql
 - Concure-DWI-Create.sql
 - Concure-final-create.sql
 - Concure-final-dwi-create.sql
 - ConcureScript.sql
 - City-DWI.sql
- DataSource

Database Tasks - General

Name	Last Run (System time)	Last Result	Type
Custom 2024-10-10 20:15:00	Success	Data imp...	

EST on Writable Smart Insert 233 : 1 [26] Set 26 | 1 5:41 PM 11/4/2024

DBeaver 24.2.0 - <postgres> Tgt-Concure-create

```

File Edit Navigate Search SQL Editor Database Window Help
Auto Commit Rollback
Database Navigator Projects
Enter a part of object name here
45.79.206.143 - 45.79.206.143:3306
postgres - localhost:5433

ALTER SEQUENCE patient_patientkey_seq RESTART WITH 1;
select * from patient p;

delete from medicine;
ALTER SEQUENCE medicine_medicinekey_seq RESTART WITH 1;
select * from medicine m;

delete from approvalstatus;
ALTER SEQUENCE approvalstatus_approvalstatuskey_seq RESTART WITH 1;
select * from approvalstatus;

```

patient 1

Grid	patientkey	patientid	name	phone	gender	diseasehistory	age
3	PA1004	Emily.Clerk	555-0201	Female	Benign lump	38	
4	PA1005	Jessica.Thompson	555-0202	Female	Family history	42	
5	PA1006	Samantha.Martin	555-0203	Female	No history	55	
6	PA1007	Laura.Adams	555-0301	Female	Prior lump	49	
7	PA1008	Nancy.Moore	555-0302	Female	No history	53	

Record

Project - General

- Name
 - CancerType-DWI.sql
 - Concure-DWI-Create.sql
 - Concure-final-create.sql
 - Concure-final-dwi-create.sql
 - ConcureScript.sql
 - City-DWI.sql
- DataSource

Database Tasks - General

Name	Last Run (System time)	Last Result	Type
Custom 2024-10-10 20:15:00	Success	Data imp...	

EST on Writable Smart Insert 228 : 1 [27] Set 27 | 1 5:41 PM 11/4/2024

DBeaver 24.2.0 - <postgres> Tgt-Concure-create

```

File Edit Navigate Search SQL Editor Database Window Help
Auto... <postgres> public@target-Concure-DW ...
Database Navigator Projects
Enter a part of object name here
45.79.206.143 - 45.79.206.143:3306
postgres - localhost:5433

<postgres> 
select * from record r;
delete from treatmenttype;
ALTER SEQUENCE treatmenttype_typekey_seq RESTART WITH 1;
select * from treatmenttype t;

delete from patient;
ALTER SEQUENCE patient_patientkey_seq RESTART WITH 1;
select * from patient p;

treatmenttype t |
select * from treatmenttype t; 25 Enter a SQL expression to filter results (use Ctrl+Space)
Grid
Old
1 Surgery
2 Radiation
3 Chemotherapy
4 Hormonal
5 Immunotherapy
Record 1
Tgt-Concure-create
Refresh Save Cancel Export data 200 8 8 row(s) fetched - 0.001s, on 2024-11-04 at 17:41:38
Database Tasks - General X
Tasks: type a part of task name here
Name Last Run (System time) Last Result Type
Custor 2024-10-10 20:15:00 Success Data imp...
Task executions: type a part of error message
System time Duration Result
EST on Writable Smart Insert 224 | 1 [33] Set 33 | 1
5:41 PM 11/4/2024
Project - General X
Name
CancerType-DWI.sql postgres
Concure-DWI-Create.sql postgres
Concure-final-create.sql postgres
Concure-final-dwi-create.sql postgres
ConcureScript.sql postgres
City-DWI.sql postgres
DataSource

```

DBeaver 24.2.0 - <postgres> Tgt-Concure-create

```

File Edit Navigate Search SQL Editor Database Window Help
Auto... <postgres> public@target-Concure-DW ...
Database Navigator Projects
Enter a part of object name here
45.79.206.143 - 45.79.206.143:3306
postgres - localhost:5433

<postgres> 
select * from speciality;
delete from record;
ALTER SEQUENCE record_recordkey_seq RESTART WITH 1;
select * from record r;
delete from treatmenttype;
ALTER SEQUENCE treatmenttype_typekey_seq RESTART WITH 1;
select * from treatmenttype t;
delete from patient;

record 1 |
select * from record r; 25 Enter a SQL expression to filter results (use Ctrl+Space)
Grid
recordkey recordid outcome treatmenttypekey
1 RECO01 Success
2 RECO02 Failure
3 RECO03 Success
4 RECO04 Success
5 RECO05 Failure
Record 1
Tgt-Concure-create
Refresh Save Cancel Export data 200 100 100 row(s) fetched - 0.000s, on 2024-11-04 at 17:41:52
Database Tasks - General X
Tasks: type a part of task name here
Name Last Run (System time) Last Result Type
Custor 2024-10-10 20:15:00 Success Data imp...
Task executions: type a part of error message
System time Duration Result
EST on Writable Smart Insert 220 | 1 [24] Set 24 | 1
5:41 PM 11/4/2024
Project - General X
Name
CancerType-DWI.sql postgres
Concure-DWI-Create.sql postgres
Concure-final-create.sql postgres
Concure-final-dwi-create.sql postgres
ConcureScript.sql postgres
City-DWI.sql postgres
DataSource

```

DBeaver 24.2.0 - <postgres> Tgt-Cancure-create

```

File Edit Navigate Search SQL Editor Database Window Help
Auto... SQL Commit Rollback
Database Navigator Projects
Enter a part of object name here
45.79.206.143 - 45.79.206.143:3306
postgres - localhost:5433

ALTER SEQUENCE doctor_doctorkey_seq RESTART WITH 1;
select * from doctor;

delete from speciality;
ALTER SEQUENCE speciality_specialitykey_seq RESTART WITH 1;
select * from speciality;

delete from record;
ALTER SEQUENCE record_recordkey_seq RESTART WITH 1;
select from record;
select * from record r;

```

speciality 1

specialitykey	speciality
1	Breast Oncology
2	Gynecologic Oncology
3	Pediatric Oncology
4	Hematologic Oncology
5	Neuro Oncology

Project - General

- Name DataSource
- CancerType-DWI.sql postgres
- Concure-DWI-Create.sql postgres
- Concure-final-create.sql postgres
- Concure-final-dwi-create.sql postgres
- ConcureScript.sql postgres
- City-DWI.sql postgres

EST en Writable Smart Insert 215 - 1 [25] Set 25 | 1 542 11/4/2024

DBeaver 24.2.0 - <postgres> Tgt-Cancure-create

```

File Edit Navigate Search SQL Editor Database Window Help
Auto... SQL Commit Rollback
Database Navigator Projects
Enter a part of object name here
45.79.206.143 - 45.79.206.143:3306
postgres - localhost:5433

ALTER SEQUENCE cancerstype_cancerkey_seq RESTART WITH 1;
select * from cancerstype;

delete from doctor;
ALTER SEQUENCE doctor_doctorkey_seq RESTART WITH 1;
select * from doctor;

delete from speciality;
ALTER SEQUENCE speciality_specialitykey_seq RESTART WITH 1;
select * from speciality;

```

doctor 1

doctorkey	doctor	name	email	yoe	specialitykey
1	DOC002	Dr. James Wilson	jwilson@lcc.com	8	5
2	DOC003	Dr. Clara Oswald	cowsold@lcc.com	5	3
3	DOC004	Dr. Richard Webber	rwebber@lcc.com	15	2
4	DOC005	Dr. Miranda Bailey	m.bailey@lcc.com	12	4
5	DOC006	Dr. Alex Karev	akarev@lcc.com	7	1

Project - General

- Name DataSource
- CancerType-DWI.sql postgres
- Concure-DWI-Create.sql postgres
- Concure-final-create.sql postgres
- Concure-final-dwi-create.sql postgres
- ConcureScript.sql postgres
- City-DWI.sql postgres

EST en Writable Smart Insert 211 - 1 [21] Set 21 | 1 542 11/4/2024

DBeaver 24.2.0 - <postgres> Tgt-Cancure-create

File Edit Navigate Search SQL Editor Database Window Help

Auto... SQL Commit Rollback Database public@Target-Cancure-DW

Database Navigator Projects Enter a part of object name here 45.79.206.143 - 45.79.206.143:3306 postgres - localhost:5433

```

delete from treatmentfact;
ALTER SEQUENCE treatmentfact_treatmentkey_seq RESTART WITH 1;
select * from treatmentfact t;

delete from cancertype;
ALTER SEQUENCE cancertype_cancerkey_seq RESTART WITH 1;
select * from cancertype;

delete from doctor;
ALTER SEQUENCE doctor_doctorkey_seq RESTART WITH 1;
select * from doctor;

```

cancertype 1

	cancerkey	cancertype	stage
1	1	Breast Cancer	Stage I
2	2	Breast Cancer	Stage II
3	3	Breast Cancer	Stage III
4	4	Breast Cancer	Stage IV
5	5	Inflammatory Breast Cancer	Stage I

Project - General

Name Data Source

- CancerType-DWI.sql postgres
- Concure-DWI-Create.sql postgres
- Concure-final-create.sql postgres
- Concure-final-dwi-create.sql postgres
- ConcureScript.sql postgres
- City-DWI.sql postgres

EST en Writable Smart Insert 207 : 1 [27] Set 27 | 1 542 11/4/2024

Database Tasks - General

Name	Last Run (System time)	Last Result	Type
Custom 2024-10-10 20:15:00	Success	Data imp...	

Task executions: type a part of error message

System time Duration Result

72

DBeaver 24.2.0 - <postgres> Tgt-Cancure-create

File Edit Navigate Search SQL Editor Database Window Help

Auto... SQL Commit Rollback Database public@Target-Cancure-DW

Database Navigator Projects Enter a part of object name here 45.79.206.143 - 45.79.206.143:3306 postgres - localhost:5433

```

delete from treatmentfact;
ALTER SEQUENCE treatmentfact_treatmentkey_seq RESTART WITH 1;
select * from treatmentfact t;

delete from cancertype;
ALTER SEQUENCE cancertype_cancerkey_seq RESTART WITH 1;
select * from cancertype;

delete from doctor;
ALTER SEQUENCE doctor_doctorkey_seq RESTART WITH 1;
select * from doctor;

```

treatmentfact 1

	treatmentkey	hospitalkey	patientkey	patontid	name	phno	gender	discashi
1	1	46	1	1#PAT001	Sarah Williams	555-0101	Female	No history
2	2	1	2	1#PAT002	Mary Smith	555-0102	Female	Family histo
3	3	46	2	2#PAT003	Unda White	555-0103	Female	No history
4	4	1	3	3#PAT004	Emily Clark	555-0201	Female	Benign lump

Project - General

Name Data Source

- CancerType-DWI.sql postgres
- Concure-DWI-Create.sql postgres
- Concure-final-create.sql postgres
- Concure-final-dwi-create.sql postgres
- ConcureScript.sql postgres
- City-DWI.sql postgres

EST en Writable Smart Insert 203 : 1 [31] Set 31 | 1 542 11/4/2024

Database Tasks - General

Name	Last Run (System time)	Last Result	Type
Custom 2024-10-10 20:15:00	Success	Data imp...	

Task executions: type a part of error message

System time Duration Result

72

DBeaver 24.2.0 - <postgres> Tgt-Concure-create

```

File Edit Navigate Search SQL Editor Database Window Help
Auto... SQL Commit Rollback <postgres> public@Target-Concure-DW
Database Navigator Projects Enter a part of object name here
45.79.206.143 - 45.79.206.143:3206
postres - localhost:5433

delete from medicinesupply;
ALTER SEQUENCE medicinesupplykey_seq RESTART WITH 1;
select * from medicinesupply m;

delete from treatmentfact;
ALTER SEQUENCE treatmentfact_treatmentkey_seq RESTART WITH 1;
select * from treatmentfact t;

delete from cancertype;
ALTER SEQUENCE cancertype_cancerkey_seq RESTART WITH 1;
select * from cancertype c;

```

medicinesupply 1

medicinekey	hospitalkey	medicinekey	pharmaccompanykey	supplycost	quantity	sellingprice
1	22	7	24	900	100	1.
2	30	74	24	900	100	1.
3	23	2	24	450	120	1.
4	28	27	24	450	120	1.

Record Test

Refresh Save Cancel K X Export data 200 X 68 68 row(s) fetched - 0.000s, on 2024-11-04 at 17:42:30

Project - General

Name DataSource

- CancerType-DWI.sql postgres
- Concure-DWI-Create.sql postgres
- Concure-final-create.sql postgres
- Concure-final-dwi-create.sql postgres
- ConcureScript.sql postgres
- City-DWI.sql postgres

EST en Writable Smart Insert 199.1 [32] Set 32 | 1

Database Tasks - General

Tasks: type a part of task name here

Name	Last Run (System time)	Last Result	Type
Custom 2024-10-10 20:15:00	Success	Data imp...	

Task executions: type a part of error message

System time Duration Result

72 5:42 PM 11/4/2024

DBeaver 24.2.0 - <postgres> Tgt-Concure-create

```

File Edit Navigate Search SQL Editor Database Window Help
Auto... SQL Commit Rollback <postgres> public@Target-Concure-DW
Database Navigator Projects Enter a part of object name here
45.79.206.143 - 45.79.206.143:3206
postres - localhost:5433

delete from medicinesupply m;
select * from review r;
select * from treatmentfact t;

delete from review;
ALTER SEQUENCE review_reviewkey_seq RESTART WITH 1;
select * from review;

delete from medicinesupply;
ALTER SEQUENCE medicinesupplykey_seq RESTART WITH 1;
select * from medicinesupply m;

```

review 1

reviewkey	patientkey	doctorkey	hospitalkey	reviewid	date	comments
1	143	144	46	REV001	2023-01-10	Excellent care and attention to detail.
2	14	1	1	REV002	2023-01-15	Good service, but the wait time was longer than expected.
3	2	2	46	REV003	2023-01-20	Average experience, room for improvement.
4	3	3	17	REV004	2023-02-10	Fantastic staff and successful treatment.

Record Test

Refresh Save Cancel K X Export data 200 X 50 1

Project - General

Name DataSource

- CancerType-DWI.sql postgres
- Concure-DWI-Create.sql postgres
- Concure-final-create.sql postgres
- Concure-final-dwi-create.sql postgres
- ConcureScript.sql postgres
- City-DWI.sql postgres

EST en Writable Smart Insert 195.1 [21] Set 21 | 1

Database Tasks - General

Tasks: type a part of task name here

Name	Last Run (System time)	Last Result	Type
Custom 2024-10-10 20:15:00	Success	Data imp...	

Task executions: type a part of error message

System time Duration Result

72 5:42 PM 11/4/2024

Additional Features:

SCD: Type 1

The update flow is essentially the implementation of Type 1 SCDs because it updates the attributes of the dimensions as and when needed and removes the old data.

SCD: Type 3

Now, implementing Type 3 SCD, where only **the previous historical value is stored**, for example, in the hospital dimension, we store both the old and the new email of the dimension.

Updating the email address of Hospital 3 in relational database table,

The screenshot shows the DBBeaver 24.2.0 interface with a PostgreSQL connection named 'postgres' active. The main window displays a SQL editor with the following query:

```
-UPDATE Hospital
SET Name = 'Bay Area Cancer Research Centre'
WHERE HospitalID = 'HOSP004';

select * from hospital;

-UPDATE Hospital
SET Email = 'newcontact@sfc.com'
WHERE HospitalID = 'HOSP003';
```

Below the query editor, the 'Statistics 1' panel shows the execution details:

Name	Value
Updated Rows	1
Query	UPDATE Hospital SET Email = 'newcontact@sfc.com' WHERE HospitalID = 'HOSP003';
Start time	Mon Nov 04 13:50:44 EST 2024
Finish time	Mon Nov 04 13:50:44 EST 2024

The 'Database Tasks - General' panel lists a task named 'Custom' with the following details:

Name	Last Run (System time)	Last Result	Type
Custom	2024-10-10 20:15:00	Success	Data imp...

At the bottom, the system tray shows the date as 11/4/2024.

Before Updating Hospital data warehouse dimension table,

DBeaver 24.2.0 - <postgres> Tgt-Concure-create

```

File Edit Navigate Search SQL Editor Database Window Help
Auto... SQL Connect Rollback
Database Navigator Projects Enter a part of object name here
45.79.206.143 - 45.79.206.143:3306
postres - localhost:5433
<postgres> <postgres> <postgres> <postgres> <postgres> <postgres> <postgres> <postgres>
select * from pharmaccompany;

delete from hospital;
ALTER SEQUENCE hospital_hospitalkey_seq RESTART WITH 1;
select * from hospital;

delete from hospitaltype;
ALTER SEQUENCE hospitaltype_typeskey_seq RESTART WITH 1;
select * from hospitaltype h;

delete from city;
ALTER SEQUENCE city_id_seq RESTART WITH 1;
select * from city;

```

hospital 1

hospitalkey	hospitalid	name	phone_number	oldemail	newemailid	city	Value
45	45	HOSP044	Augusta Cancer Treatment Cent 123-456-7933	info@actc.com	[NULL]		1
46	46	HOSP001	Los Angeles Cancer Research Inc 123-456-7890	contact@lacc.com	[NULL]		
47	47	HOSP004	Bay Area Cancer Research Centr 123-456-7893	info@bach.com	[NULL]		
48	48	HOSP003	San Francisco Cancer Treatment 123-456-7892	contact@sftc.com	[NULL]		

48 row(s) fetched - 0.002s, on 2024-11-04 at 13:50:05

Project - General

Name Data Source

- CancerType-DWI.sql postgres
- Concure-DWI-Create.sql postgres
- Concure-final-create.sql postgres
- Concure-final-dwi-create.sql postgres
- ConcureScript.sql postgres
- City-DWI.sql postgres

EST | en | Writable | Smart Insert | 245 - 1 [23] | Sel: 23 | 1 | 151 | 11/4/2024 | 71

As we can see the newemailid column of the hospital 3 is null.

After running the update flow job,

Talend Real-time Big Data Platform (R2024-10) | Tgt-Concure-DW (Connection: Local)

File Edit View Window Help

Feature Manager

Repository LOCAL: Tgt-Concure-DW

Job Designs

Sources: Sourcedb

Targets: TargetDw

Components: Job Hosp., Job hospital, Job tgtid, Job TgtCo, Job TgtD, Job TgtSta, Job tgtsu, Job wrapp

Execution: Starting job Hospital at 13:51 04/11/2024 [statistics] connecting to socket on port 4009 [statistics] connected [statistics] disconnected

Job Hospital

Basic Run

Execution: Job Hospital ended at 13:51 04/11/2024. [Exit]

T-map configuration:

The screenshot shows the Talend Real-time Big Data Platform interface, specifically the tMap component. It displays a graphical mapping between three input rows (row1, row2, row3) and various output columns. The mapping includes expressions for columns like hospitalkey, name, phone, email, location, typekey, citykey, cityname, population, statekey, oldemail, newemail, and typokey. The 'Expression editor' and 'Schema editor' tabs are open at the bottom, showing the detailed definitions for each column.

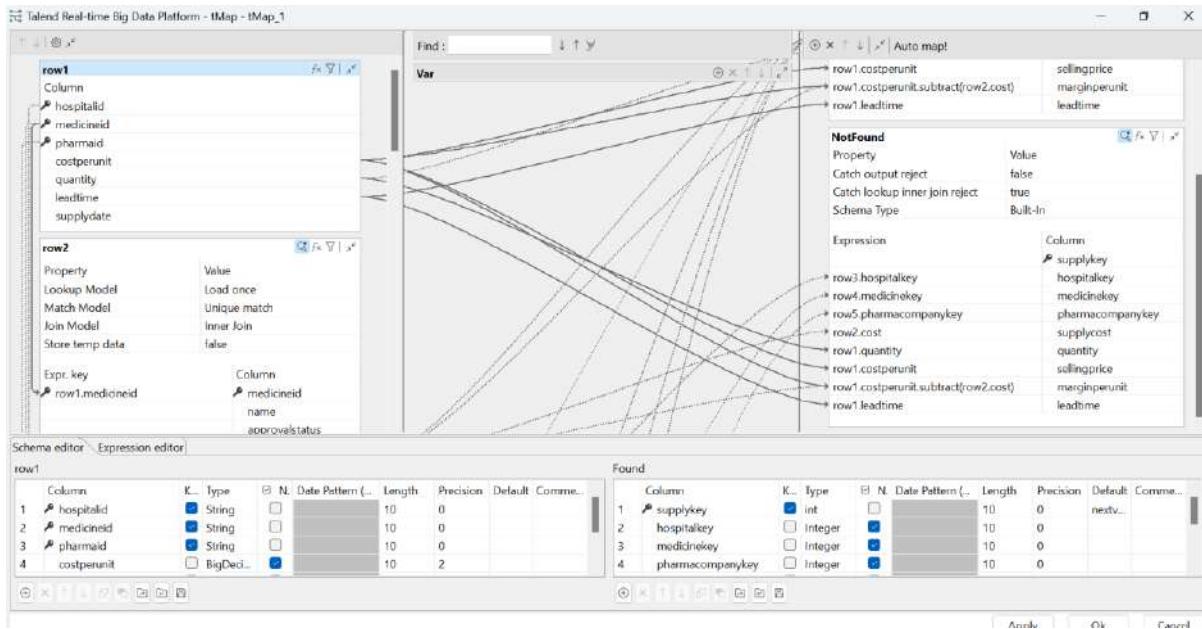
Updated data warehouse hospital dimension table with both old and new email information for hospital3, thus type 3 SCD is successful.

The screenshot shows the DBeaver 24.0.0 interface with a database session for PostgreSQL. The session window displays several SQL commands for updating the hospital dimension table. These include ALTER SEQUENCE, SELECT statements, and DELETE operations. The results pane shows a table of hospital records with columns like hospitalid, name, phone_number, oldemail, and newemail. The table data is as follows:

Row	hospitalid	name	phone_number	oldemail	newemail
45	HOSP044	AspiraCare Cancer Treatment Center	123-456-7893	info@aspiracare.com	newinfo@aspiracare.com
46	HOSP001	Los Angeles Cancer Research Inc	123-456-7890	contact@lacrc.com	(null)
47	HOSP004	Bay Area Cancer Research Center	123-456-7899	info@bacrc.com	newinfo@bacrc.com
48	HOSP002	San Francisco Cancer Treatment	123-456-7892	contact@sfcrc.com	(null)

Calculated Measures:

We have implemented a calculated measure of our supplies fact table named MarginPerUnit which is the difference between the other two measures of cost price and selling price: The T-map shows the implementation of the calculated/derived measure Marginperunit



DataSources:

LOCAL: Tgt-Cancure-DW

- > **Snowflake**
- > **Teradata**
- > **Vertica**
- Metadata**
 - Db Connections**
 - Sourcedb 0.1**
 - Queries**
 - Synonym schemas**
 - > **Table schemas**
 - View schemas**
 - TargetDw 0.1**
 - Queries**
 - Synonym schemas**
 - > **Table schemas**
 - View schemas**
- File delimited**
- File positional**
- File regex**
- File XML**
- XLS File Excel**
 - > **SourceApprovalStatus 0.1**
 - > **sourcacity 0.1**
 - > **sourcecountry 0.1**
 - > **sourcehospitaltype 0.1**
 - > **sourcespecialty 0.1**
 - > **state 0.1**
 - > **TreatmentType 0.1**
- LDIF File Idif**
- JSON File Json**

item selected

Excel and OLTP database are my two sources of data in this project to populate 17 tables of my datawarehouse of these 17 tables. 7 tables are populated from Excel sheets and the rest 10 tables are populated from the OLTP database. From the above screenshots we can see my sourcedb(OLTP Database) and all the 7 excel

sheets.

Transformations:

Some of the transformations done in the project are,

1. Dates to int(duration):

The screenshot shows the Talend Real-time Big Data Platform interface, specifically the tMap component. The Var panel displays a complex mapping between source columns (row1 and row2) and target columns. A formula is being used to calculate the duration between start and end dates. The formula is: $(int)(row7.endday.getTime() - row7.startday.getTime())/(1000*60*60*24)$. The Schema editor shows the schema for both row1 and the resulting row1 output, which includes columns like treatmentkey, hospitalkey, patientkey, and patientid.

In the treatment fact table, the start and end days are converted to duration in int using the highlighted formula.

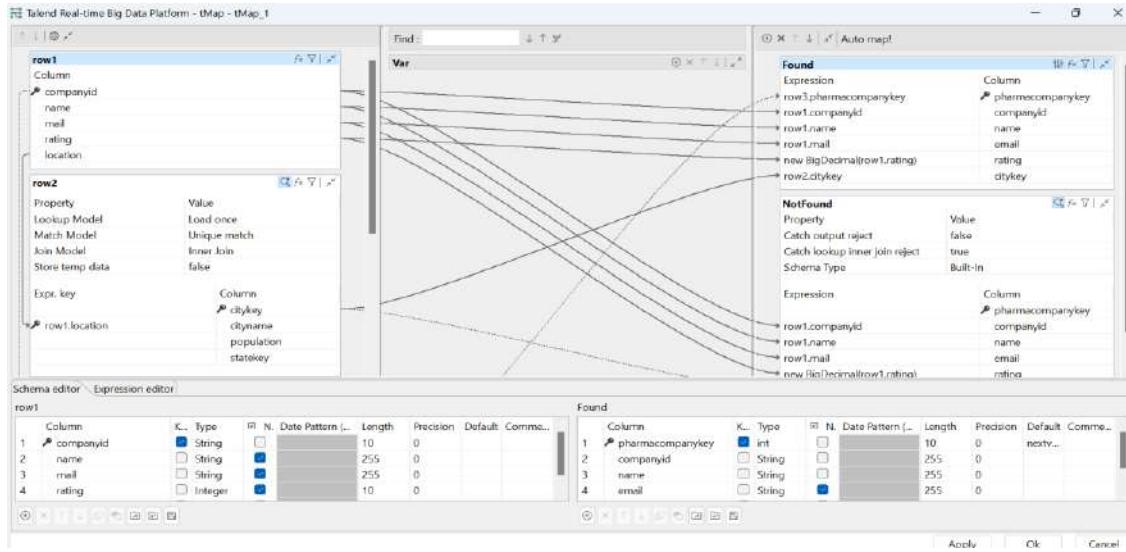
We can see the calculated duration in the below table

select * from treatmentfact t <small>Enter a SQL expression to filter results (use Ctrl+Space)</small>								
Grid	key	cancerkey	doctorkey	medicinekey	bill	duration	insurance	
21	5	7	37	21	30,000	92	25,000	
22	7	8	40	22	15,000	92	12,000	
23	8	8	43	23	32,000	92	28,000	
24	1	9	23	24	22,000	30	18,000	
25	2	9	28	25	20,000	30	16,000	
26	1	9	29	26	22,000	30	18,000	
27	3	10	46	27	24,000	92	19,000	

2. Converting int to big decimal:

In the pharma company dimension table, we converted the int data of the rating column in our source relational database to the big decimal datatype of rating in our datawarehouse database for a better precision

The same can be seen in the T-map below,

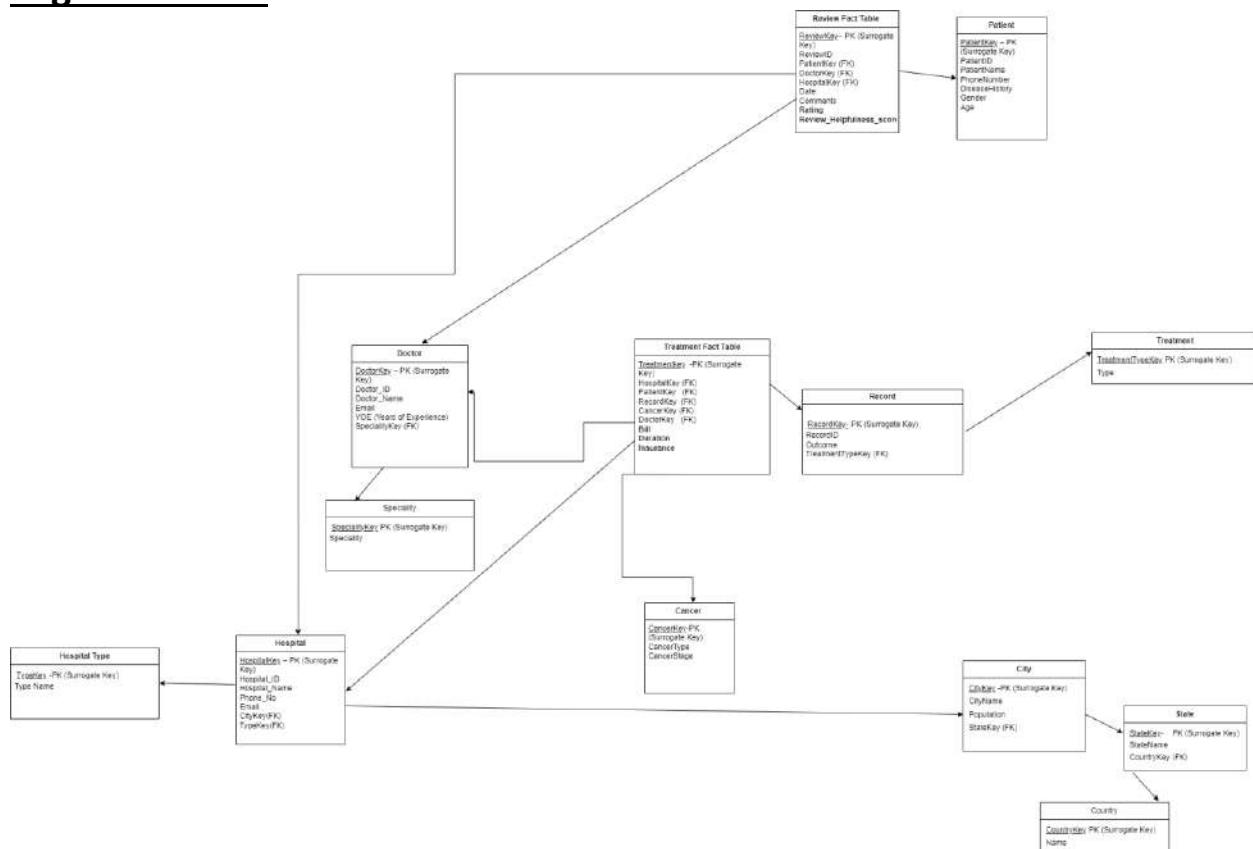


Project 2:

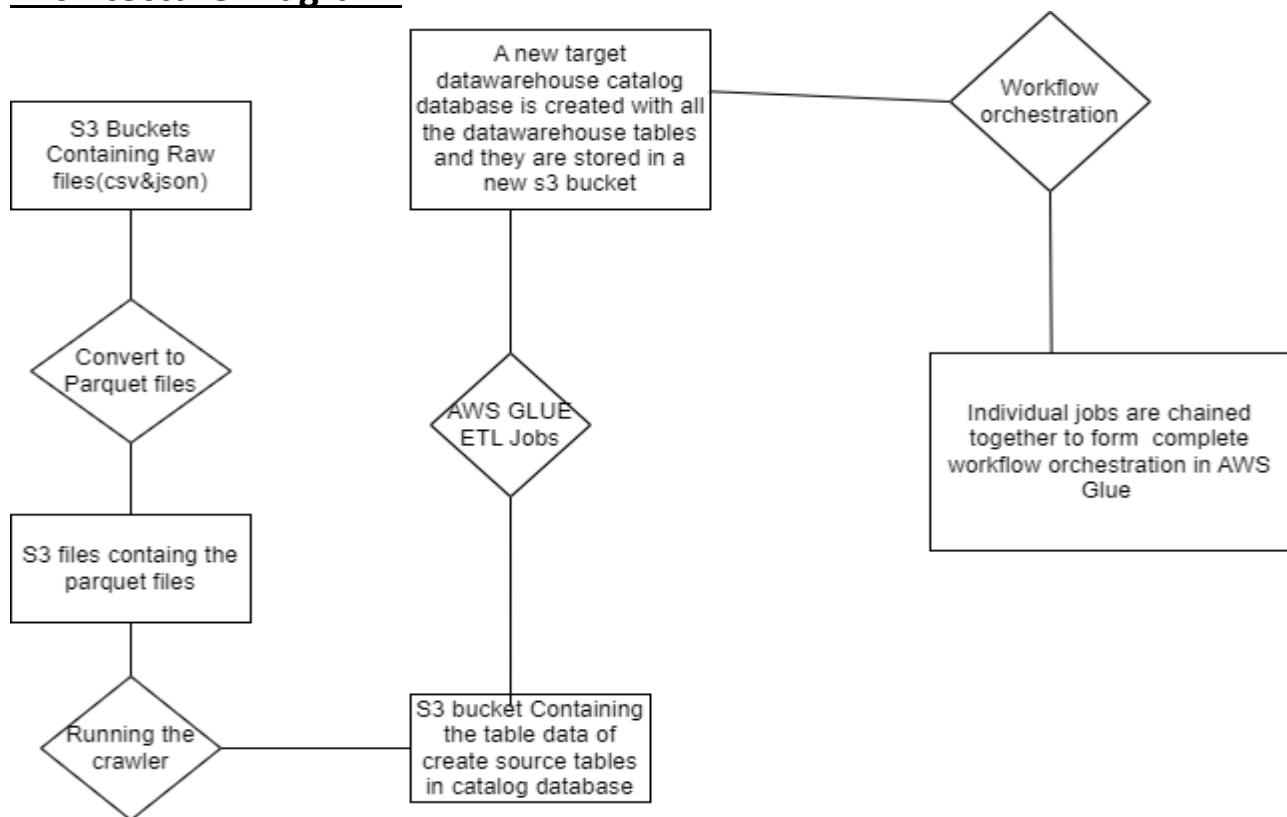
Cloud Implementation:

As mentioned in the previous milestone we have cut down on our on-prem project and excluded the supplies part hence the updated Logical Model:

Logical Model:



Architecture Diagram:



Implementation:

Data Sources:

First, we convert the data stored in our relational tables into CSV and then load it in S3 Bucket. In addition to that we have some files in JSON format, and all the necessary conversions are done using **Python**.

```
In [3]: import pandas as pd

file_path = file_path = r'C:\Users\aravi\OneDrive\Desktop\DataSource\COUNTRY.xlsx'
df = pd.read_excel(file_path)

json_path = r'C:\Users\aravi\OneDrive\Documents\JSON_data_source\country.json'
df.to_json(json_path, orient='records', lines=True)

print(f'File converted and saved as {json_path}')

File converted and saved as C:\Users\aravi\OneDrive\Documents\JSON_data_source\country.json
```

```
In [4]: import pandas as pd

file_path = file_path = r'C:\Users\aravi\OneDrive\Desktop\DataSource\state.xlsx'
df = pd.read_excel(file_path)

json_path = r'C:\Users\aravi\OneDrive\Documents\JSON_data_source\state.json'
df.to_json(json_path, orient='records', lines=True)

print(f'File converted and saved as {json_path}')

File converted and saved as C:\Users\aravi\OneDrive\Documents\JSON_data_source\state.json
```

```
In [5]: import pandas as pd

file_path = file_path = r'C:\Users\aravi\OneDrive\Desktop\DataSource\city.xlsx'
df = pd.read_excel(file_path)

json_path = r'C:\Users\aravi\OneDrive\Documents\JSON_data_source\city.json'
df.to_json(json_path, orient='records', lines=True)

print(f'File converted and saved as {json_path}')

File converted and saved as C:\Users\aravi\OneDrive\Documents\JSON_data_source\city.json
```

```
In [6]: M import pandas as pd

df = pd.read_excel(file_path)

json_path = r'C:\Users\aravi\OneDrive\Documents\JSON_data_source\approvalstatus.json'
df.to_json(json_path, orient='records', lines=True)

print(f'File converted and saved as {json_path}')
File converted and saved as C:\Users\aravi\OneDrive\Documents\JSON_data_source\approvalstatus.json

In [7]: M import pandas as pd

file_path = file_path = r'C:\Users\aravi\OneDrive\Desktop\DataSource\HospitalType.xlsx'

df = pd.read_excel(file_path)

json_path = r'C:\Users\aravi\OneDrive\Documents\JSON_data_source\hospitaltype.json'
df.to_json(json_path, orient='records', lines=True)

print(f'File converted and saved as {json_path}')
File converted and saved as C:\Users\aravi\OneDrive\Documents\JSON_data_source\hospitaltype.json

In [8]: M import pandas as pd

file_path = file_path = r'C:\Users\aravi\OneDrive\Desktop\DataSource\Speciality.xlsx'

df = pd.read_excel(file_path)

json_path = r'C:\Users\aravi\OneDrive\Documents\JSON_data_source\speciality.json'
df.to_json(json_path, orient='records', lines=True)

print(f'File converted and saved as {json_path}')
File converted and saved as C:\Users\aravi\OneDrive\Documents\JSON_data_source\speciality.json
```

```
In [10]: import pandas as pd

file_path = file_path = r'C:\Users\aravi\OneDrive\Desktop\DataSource\TreatmentType.xlsx'

df = pd.read_excel(file_path)

json_path = r'C:\Users\aravi\OneDrive\Documents\JSON_data_source\treatmenttype.json'
df.to_json(json_path, orient='records', lines=True)

print(f'File converted and saved as {json_path}')
```

File converted and saved as C:\Users\aravi\OneDrive\Documents\JSON_data_source\treatmenttype.json

```
In [11]: import pandas as pd

file_path = r'C:\Users\aravi\OneDrive\Desktop\DataSource\COUNTRY.xlsx'

df = pd.read_excel(file_path)

csv_path = r'C:\Users\aravi\OneDrive\Documents\CSV_data_source\country.csv'
df.to_csv(csv_path, index=False)

print(f'File converted and saved as {csv_path}')
```

File converted and saved as C:\Users\aravi\OneDrive\Documents\CSV_data_source\country.csv

```
In [12]: import pandas as pd

file_path = r'C:\Users\aravi\OneDrive\Desktop\DataSource\State.xlsx'

df = pd.read_excel(file_path)

csv_path = r'C:\Users\aravi\OneDrive\Documents\CSV_data_source\state.csv'
df.to_csv(csv_path, index=False)

print(f'File converted and saved as {csv_path}')
```

File converted and saved as C:\Users\aravi\OneDrive\Documents\CSV_data_source\state.csv

```
In [13]: import pandas as pd
file_path = r'C:\Users\aravi\OneDrive\Desktop\Datasources\city.xlsx'

df = pd.read_excel(file_path)
csv_path = r'C:\Users\aravi\OneDrive\Documents\CSV_data_source\city.csv'
df.to_csv(csv_path, index=False)

print(f'File converted and saved as {csv_path}')
File converted and saved as C:\Users\aravi\OneDrive\Documents\CSV_data_source\city.csv

In [14]: import pandas as pd
file_path = r'C:\Users\aravi\OneDrive\Desktop\Datasources\Approvalstatus.xlsx'
df = pd.read_excel(file_path)
csv_path = r'C:\Users\aravi\OneDrive\Documents\CSV_data_source\approvalstatus.csv'

df.to_csv(csv_path, index=False)

print(f'File converted and saved as {csv_path}')
File converted and saved as C:\Users\aravi\OneDrive\Documents\CSV_data_source\approvalstatus.csv

In [15]: import pandas as pd
file_path = r'C:\Users\aravi\OneDrive\Desktop\Datasources\HospitalType.xlsx'

df = pd.read_excel(file_path)
csv_path = r'C:\Users\aravi\OneDrive\Documents\CSV_data_source\hospitaltype.csv'
df.to_csv(csv_path, index=False)

print(f'File converted and saved as {csv_path}')
File converted and saved as C:\Users\aravi\OneDrive\Documents\CSV_data_source\hospitaltype.csv

In [16]: import pandas as pd
file_path = r'C:\Users\aravi\OneDrive\Desktop\Datasources\Speciality.xlsx'
df = pd.read_excel(file_path)
csv_path = r'C:\Users\aravi\OneDrive\Documents\CSV_data_source\speciality.csv'
df.to_csv(csv_path, index=False)

print(f'File converted and saved as {csv_path}')
File converted and saved as C:\Users\aravi\OneDrive\Documents\CSV_data_source\speciality.csv

In [17]: import pandas as pd
file_path = r'C:\Users\aravi\OneDrive\Desktop\Datasources\TreatmentType.xlsx'
df = pd.read_excel(file_path)
csv_path = r'C:\Users\aravi\OneDrive\Documents\CSV_data_source\treatmenttype.csv'

df.to_csv(csv_path, index=False)

print(f'File converted and saved as {csv_path}')
File converted and saved as C:\Users\aravi\OneDrive\Documents\CSV_data_source\treatmenttype.csv
```

Now these CSV and JSON files are loaded into an s3 bucket named [cancurerawsouce](#). These files act as our source.

cancurerawsource [Info](#)

[Objects](#) [Properties](#) [Permissions](#) [Metrics](#) [Management](#) [Access Points](#)

Objects (2) [Info](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

Name	Type	Last modified	Size	Storage class
covidfiles/	Folder	-	-	-
parquetfiles/	Folder	-	-	-

Now these raw CSV files and JSON files are converted to parquet format using AWS Glue jobs and is stored in another bucket named cancureprocessedbucket.

cancureprocessedbucket [Info](#)

[Objects](#) [Properties](#) [Permissions](#) [Metrics](#) [Management](#) [Access Points](#)

Objects (3) [Info](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

Name	Type	Last modified	Size	Storage class
parquet-files-csv/	Folder	-	-	-
parquet-files-json/	Folder	November 22, 2024, 12:01:10 (UTC-05:00)	0 B	Standard
parquet-files-json/_folders/	Folder	-	-	-

The AWS Glue jobs used to convert the raw files are

[csvparquet](#) [Glue ETL](#) [Script](#) [11/22/2024, 2:30:49 AM](#) [4.0](#)

[jsonparquet](#) [Glue ETL](#) [Script](#) [11/22/2024, 2:23:49 AM](#) [4.0](#)

[Actions](#) [Save](#) [Run](#)

[csvparquet](#)

[Script](#) [Job details](#) [Runs](#) [Data quality](#) [Schedules](#) [Version Control](#) [Upgrade analysis - preview](#)

Script [Info](#)

```

1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7 import logging
8
9 # Create Logging
10 logger = logging.getLogger()
11 logger.setLevel(logging.INFO)
12
13 # Get the JOB_NAME argument
14 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
15
16 # Initialize Spark and glue contexts
17 sc = SparkContext()
18 glueContext = GlueContext(sc)
19 spark = glueContext.spark_session
20 job = Job(glueContext)
21 job.init(args['JOB_NAME'], args)
22
23 # List of all csv files to process
24 files = [
25     {
26         "path": "s3://cancurerawsource/csvFiles/assignments.csv",
27         "output_path": "s3://cancureprocessedbucket/parquet-files-csv/assignments/"
28     },
29     {
30         "path": "s3://cancurerawsource/csvFiles/cancer_200011111204.csv",
31         "output_path": "s3://cancureprocessedbucket/parquet-files-csv/cancer/"
32     },
33     {
34         "path": "s3://cancurerawsource/csvFiles/doctor_200011111204.csv",
35         "output_path": "s3://cancureprocessedbucket/parquet-files-csv/doctor/"
36     }
37 ]

```

Python - Line 1, Call 1 | 0 Errors | 0 Warnings | 0

```

1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7
8 # Set the job name argument
9 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
10
11 # Initialize spark and glue contexts
12 sc = SparkContext()
13 glueContext = GlueContext(sc)
14 spark = glueContext.create_spark_session
15 job = Job(glueContext)
16 job.init(args['JOB_NAME'], args)
17
18 # Process each file
19 files = [
20     {
21         "path": "s3://concurrawsource/jsonfiles/approvalstatus.json",
22         "output_path": "s3://concurprocessedbucket/parquet-files-json/approvalstatus/"
23     },
24     [
25         {
26             "path": "s3://concurrawsource/jsonfiles/city.json",
27             "output_path": "s3://concurprocessedbucket/parquet-files-json/city/"
28         },
29     ],
30     [
31         {
32             "path": "s3://concurrawsource/jsonfiles/hospitaltype.json",
33             "output_path": "s3://concurprocessedbucket/parquet-files-json/hospitaltype/"
34         },
35     ],
36     [
37         {
38             "path": "s3://concurrawsource/jsonfiles/speciality.json",
39             "output_path": "s3://concurprocessedbucket/parquet-files-json/speciality/"
40         },
41     ],
42     [
43         {
44             "path": "s3://concurrawsource/jsonfiles/country.json",
45         }
46     ]
47 ]

```

Python In 1, Col 1 | Errors 0 | Warnings 0

So these jobs are our first transformation and it is stored in another bucket as mentioned.

Crawlers:

Now we build crawlers to crawler over these two parquet folders to create our source relational catalog database.

Crawlers

A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

Name	Status	Schedule	Last run	Last run timestamp	Log	Table changes from last run
jsoncrawler	Ready		Success	November 22, 2024 at 07:37:05	View log	12 created
jpcsvcrawler	Ready		Success	November 22, 2024 at 07:44:46	View log	7 created

csvcrawler

Crawler properties		Database		State	
Name	IAM role	Table	Table	Table	Table
csvcrawler	Lambda	Lambda, Service	Lambda	READY	READY
Description	Security configuration	Lake Formation configurations	Table prefixes		
Maximum table threshold					
Advanced settings					

Crawler runs [1]

The list of crawler runs for this crawler.

Run date	Start time (UTC)	End time (UTC)	Filter by a date and time range	Status	DPU hours	Table changes
November 22, 2024	November 22, 2024 at 07:37:09	November 22, 2024 at 07:38:27		Completed	01 min 21 s	12 table changes, 0 partition changes

Crawler properties

Name: jsoncrawler

Maximum table threshold: -

Advanced settings

Crawler runs

Start time (UTC)	End time (UTC)	Status	DPU hours	Table changes
November 22, 2024 at 01:48:45	November 22, 2024 at 07:45:58	Completed	01 min 12 s	0.039 2 table changes, 0 partition changes

And once both the crawlers are run.
We have a source catalog database.
There were total of 12 CSV files and 7 JSON files,
Which are converted to corresponding tables in the source catalog database.

Announcing new optimization features for Apache Iceberg tables
Optimize storage for Apache Iceberg tables with automatic snapshot retention and synch file deletion. Learn more

Database properties

Name: cancure_source

Tables (19)

Name	Database	Location	Classification	Deprecated	View data	Data quality	Column statistics
apponerstatus	cancure_source	s3://cancureprocessedbucket/parquet-f	Parquet	-	Table data	View data quality	View statistics
assignment	cancure_source	s3://cancureprocessedbucket/parquet-f	Parquet	-	Table data	View data quality	View statistics
cancer	cancure_source	s3://cancureprocessedbucket/parquet-f	Parquet	-	Table data	View data quality	View statistics
city	cancure_source	s3://cancureprocessedbucket/parquet-f	Parquet	-	Table data	View data quality	View statistics
country	cancure_source	s3://cancureprocessedbucket/parquet-f	Parquet	-	Table data	View data quality	View statistics
doctor	cancure_source	s3://cancureprocessedbucket/parquet-f	Parquet	-	Table data	View data quality	View statistics
hospital	cancure_source	s3://cancureprocessedbucket/parquet-f	Parquet	-	Table data	View data quality	View statistics
heighttype	cancure_source	s3://cancureprocessedbucket/parquet-f	Parquet	-	Table data	View data quality	View statistics
medicines	cancure_source	s3://cancureprocessedbucket/parquet-f	Parquet	-	Table data	View data quality	View statistics
patient	cancure_source	s3://cancureprocessedbucket/parquet-f	Parquet	-	Table data	View data quality	View statistics
pharmacampeny	cancure_source	s3://cancureprocessedbucket/parquet-f	Parquet	-	Table data	View data quality	View statistics
posts	cancure_source	s3://cancureprocessedbucket/parquet-f	Parquet	-	Table data	View data quality	View statistics
record	cancure_source	s3://cancureprocessedbucket/parquet-f	Parquet	-	Table data	View data quality	View statistics
reviews	cancure_source	s3://cancureprocessedbucket/parquet-f	Parquet	-	Table data	View data quality	View statistics
specialty	cancure_source	s3://cancureprocessedbucket/parquet-f	Parquet	-	Table data	View data quality	View statistics
state	cancure_source	s3://cancureprocessedbucket/parquet-f	Parquet	-	Table data	View data quality	View statistics
supplies	cancure_source	s3://cancureprocessedbucket/parquet-f	Parquet	-	Table data	View data quality	View statistics
treat	cancure_source	s3://cancureprocessedbucket/parquet-f	Parquet	-	Table data	View data quality	View statistics
treatmenttype	cancure_source	s3://cancureprocessedbucket/parquet-f	Parquet	-	Table data	View data quality	View statistics

Now the data source is loaded and source catalog database is created, we now create ETL AWS Glue jobs to transform and load the data in the target data warehouse database.
There are a total 13 jobs including for all dimensions and fact tables.
We have followed the AWS Glue script editor to construct these jobs,

We have jobs for all the tables,

Your jobs (17) Info					
<input type="checkbox"/> Filter jobs by property			Actions Run job		
<input type="checkbox"/>	Job name	Type	Created by	Last modified	AWS Glue version
<input type="checkbox"/>	approvstatus_target_job	Glue ETL	Script	11/22/2024, 10:05:23 PM	4.0
<input type="checkbox"/>	treatment_fact_job	Glue ETL	Visual	11/22/2024, 6:43:56 PM	4.0
<input type="checkbox"/>	reviews_fact_target	Glue ETL	Script	11/22/2024, 4:26:32 PM	4.0
<input type="checkbox"/>	record_target_job	Glue ETL	Script	11/22/2024, 4:10:36 PM	4.0
<input type="checkbox"/>	treatmentnyc_target_job	Glue ETL	Script	11/22/2024, 3:52:13 PM	4.0
<input type="checkbox"/>	patient_target_job	Glue ETL	Script	11/22/2024, 3:45:03 PM	4.0
<input type="checkbox"/>	cancer_target_job	Glue ETL	Script	11/22/2024, 3:18:01 PM	4.0
<input type="checkbox"/>	doctor_target_job	Glue ETL	Script	11/22/2024, 5:08:58 PM	4.0
<input type="checkbox"/>	specialty_target_job	Glue ETL	Script	11/22/2024, 3:02:29 PM	4.0
<input type="checkbox"/>	hospital_target_job	Glue ETL	Script	11/22/2024, 2:53:05 PM	4.0

Your jobs (17) Info					
Filter jobs by property			Actions		
Job name	Type	Created by	Last modified	AWS Glue version	
hospitality_target_job	Glue ETL	Script	11/22/2024, 2:41:39 PM	4.0	
city_target_job	Glue ETL	Script	11/22/2024, 2:36:33 PM	4.0	
state_target_job	Glue ETL	Script	11/22/2024, 2:29:47 PM	4.0	
country_target_job	Glue ETL	Script	11/22/2024, 12:40:55 PM	4.0	
update_txs	Glue ETL	Script	11/22/2024, 12:15:19 PM	4.0	
experiments	Glue ETL	Script	11/22/2024, 2:30:49 AM	4.0	
bonaparte	Glue ETL	Script	11/22/2024, 2:23:49 AM	4.0	

Pipeline Tools:

So in order to build and check this pipeline we have used various tools like S3 Bucket, AWS Glue jobs, Workflow Orchestration, Crawlers and Athena and AWS Catalog databases.

Now seeing the individual jobs,

Individual Jobs:

Country Job:

The screenshot shows the AWS Glue Data Catalog interface. The top navigation bar includes 'Actions', 'Save', and 'Run' buttons. The left sidebar lists 'Getting started', 'ETL jobs', 'Visual ETL', 'Notebooks', 'Machine learning', 'Data Catalog tables', 'Data connectors', 'Data Catalog', 'Datasets', 'Tables', 'Storage schema registries', 'Schemas', 'Generated', 'Crawlers', 'Codelines', and 'Catalog settings'. Under 'Data Integration and ETL', there are 'Legacy pages' and a 'Create new' button. The main content area is titled 'country_target_job' and contains tabs for 'Script', 'Logs', 'Job details', 'Runs', 'Data quality', 'Schedules', 'Version Control', and 'Upgrade analysis - preview'. The 'Script' tab is selected, displaying Python code for the job. The code imports various AWS Glue modules and defines a transformation function that reads from a specific S3 bucket ('s3://etl-data-lake/region/usa/2023/01/01/00000') and writes to a target table ('country_target'). It also handles errors and logs information. The bottom status bar indicates the script has 1 line, 1 error, 0 warnings, and 0 errors.

```

if targetDF:
    query = """
        SELECT
            ROW_NUMBER() OVER (ORDER BY Name ASC) AS CountryKey,
            Name
        FROM source_table src
        LEFT ANTI JOIN target_table tgt
        ON TRIM(src.Name) = TRIM(tgt.Name) -- Match based on Name column
        WHERE src.Name IS NOT NULL AND TRIM(src.Name) != ''
    """
else:
    query = """
        SELECT
            ROW_NUMBER() OVER (ORDER BY Name ASC) AS CountryKey,
            Name
        FROM source_table
        WHERE Name IS NOT NULL AND TRIM(Name) != ''
    """

```

This part of the code matches and sees if any duplicate data is there in the incoming data so that duplicate data is not inserted every time the job is run and we also add a surrogate key.

And it is stored in another target data warehouse S3 bucket named **cancuretargetdw**.

```

filtered_new_records_DF = DynamicFrame.fromDF(filtered_new_records, glueContext, "filtered_new_records_DF")

:s3sink = glueContext.getSink(
    path="s3://cancuretargetdw/country/",
    connection_type="s3",
    updateBehavior="LOG",
    partitionKeys=[],
    enableUpdateCatalog=True,
    transformation_ctx="s3sink"
)

:s3sink.setCatalogInfo(catalogDatabase="cancure_target", catalogTableName="country_target")
:s3sink.setFormat("glueparquet", compression="snappy")
:s3sink.writeFrame(filtered_new_records_DF)

job.commit()

```

And the job is run,

Run status	Retries	Start time (Local)	End time (Local)	Duration	Capacity (DPUs)	Worker type	Size version
<input checked="" type="radio"/> Successful	0	11/22/2024 12:44:54	11/22/2024 12:46:19	1 m 17 s	10 DPUs	G.1X	4.0
<input type="radio"/> Successful	0	11/22/2024 12:40:51	11/22/2024 12:42:14	1 m 15 s	10 DPUs	G.1X	4.0
<input type="radio"/> Successful	0	11/22/2024 12:21:13	11/22/2024 12:22:41	1 m 18 s	10 DPUs	G.1X	4.0
<input type="radio"/> Successful	0	11/22/2024 04:03:32	11/22/2024 04:05:28	1 m 42 s	10 DPUs	G.1X	4.0

State Job:

Now are doing the same for the state job,

We are adding the surrogate keys and transferring it to the state datawarehouse table.

And the job is run

And the job is run,

Run status	Retries	Start time (Local)	End time (Local)	Duration	Capacity (DBU/h)	Worker type	Glue version
Successful	0	11/22/2024 14:22:13	11/22/2024 14:32:56	1 m 26 s	10 DBUs	G_1X	4.0
Successful	0	11/22/2024 14:39:54	11/22/2024 14:51:20	1 m 16 s	10 DBUs	G_1X	4.0
Successful	0	11/22/2024 14:08:04	11/22/2024 14:08:28	1 m 14 s	10 DBUs	G_1X	4.0

City:

AWS Glue

Last modified on 1/10/2024, 8:00:22 PM Actions Save Run

Getting started
ETL jobs
Visual ETL
Notebooks
AWS Lambda monitoring
Data Catalog tables
Data connections
AWS Lambda infrastructure

► Data Catalog

- Datasets
- Tables
- Version schema registries
- Serializers
- Connections
- CloudWatch Metrics
- CloudWatch Metrics
- Catalog entries

► Data integration and ETL

- Legacy pages

Metrics View Documentation AWS Marketplace

Enable compact mode Enable new navigation

Script [Edit details](#) [Items](#) [Data quality](#) [Schedules](#) [Version Control](#) [Upgrade analysis - preview](#)

city_target_job

```
1 # Import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7 from awsglue.dynamicframe import DynamicFrame
8
9 # Set sparkcontext(gluestream, mary, session, transformation_ctx) as dynamicframe
10 ssc = SparkContext()
11 ssc.addFile('s3://getresolvedoptions\(result, transformation\_ctx\)
12 result = glueContext.createSession(ssc)
13 return DynamicFrame.fromDF(result, glueContext, transformation_ctx)
14
15 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
16 sc = SparkContext()
17 spark = GlueContext(sc)
18 spark = glueContext.createSession()
19 job = Job(gluestream)
20 job.init(args['JOB_NAME'], args)
21
22 # Load source data from AWS glue data catalog
23 source = glueContext.create_dynamic_frame.from_catalog(
24     database='source_table',
25     table_name='city',
26     transformation_ctx='sourceDF')
27
28 # If sourceDF.isLocal() == False
29 #     raise exception("source table is empty or not loaded")
30
31 if True:
32     target = glueContext.create_dynamic_frame.from_catalog(
33         database='source_target',
34         table_name='city_target',
35         transformation_ctx='targetDF')
36
37
38
39 # targetDF.show()
40
41 query = "SELECT
42     ROW_NUMBER() OVER (ORDER BY src.CityName ASC) AS CityKey,
43     src.CityName,
44     CAST(src.Population AS INT) AS Population,
45     CAST(src.StateKey AS INT) AS StateKey
46 FROM source_table src
47 LEFT ANTI JOIN target_table tgt
48 ON TRIM(src.CityName) = TRIM(tgt.CityName) AND src.statekey = tgt.statekey
49 WHERE src.cityname IS NOT NULL AND TRIM(src.CityName) != ''"
50
51
52 else:
53     query = """
54     SELECT
55         ROW_NUMBER() OVER (ORDER BY CityName ASC) AS CityKey,
56         CityName,
57         CAST(Population AS INT) AS Population,
58         CAST(StateKey AS INT) AS StateKey
59     FROM source_table
60     WHERE CityName IS NOT NULL AND TRIM(CityName) != ''
61
62
63
64 filtered_new_records = spark.sql(query)
65
66
67 filtered_new_records_DF = DynamicFrame.fromDF(filtered_new_records, glueContext, "filtered_new_records_DF")
68
69
70 # Write the new records to the target table in ss and update the glue catalog
71 ssSink = glueContext.getSink(
72     path="s3://getcurrenttargetdir\(city\)/",
73     connection\_type='s3',
74     updateBehavior='Update',
75     partitionKeys=\[\],
76     enableUpdateCatalog=True,
```

The same is done for the city datawarehouse table.

And the job is run,

Job runs (1/3) View details											Run started at 11/22/2024, 14:16:33 PM	Actions	Save	Run	
											View details	Stop job run	Troubleshoot with AI	Table View	Card View
Filter job runs by property											November 21, 2024 at 16:57:49				
Run status	#	Retries	Start time (Local)	End time (Local)	Duration	Capacity (DPUs)	Worker type	Glue version							
● Succeeded	0	0	11/22/2024 14:36:58	11/22/2024 14:37:58	1 m 10 s	10 DPUs	G-1X	4.0							
● Succeeded	0	0	11/22/2024 14:07:40	11/22/2024 14:09:12	1 m 24 s	10 DPUs	G-1X	4.0							
● Succeeded	0	0	11/22/2024 04:32:11	11/22/2024 04:33:50	1 m 29 s	10 DPUs	G-1X	4.0							

HospitalTypeJob:

AWS Glue

Sending started
ETL jobs
AWS ETL
Notebooks
Lambda monitoring
Data Catalog tables
Data conversions
AWS Lambda functions
Data integration and ETL
Schemas
Connections
Crawlers
Overrides
Glue settings
Legacy pages

Run now (2)
Documentation (2)
AWS Marketplace

Enable console mode
Enable new navigation

Last modification 1/23/2024, 8:45:30 PM Actions Save Run

hospitalitytype_target_job

Script Job Data quality Schedules Version Control Upgrade analysis - preview

```
Script info
1 #!/usr/bin/python
2 from awsglue.transforms import *
3 from awsglue.dynamicframe import DynamicFrame
4 from awsglue.context import GlueContext
5 from awsglue.job import Job
6 from awsglue.dynamicframe import DynamicFrame
7
8
9 *def sparkSession(glueContext, args, session, transformation_ctx) -> DynamicFrame:
10     for alias, frame in mapping.items():
11         if alias == "source_table":
12             result = glueContext.create_dynamic_frame.from_options(
13                 options={"connection_type": "s3", "path": "s3://cancretargettdw/hospitalitytype/"}, 
14                 transformation_ctx=transformation_ctx)
15             return result
16     return None
17
18     args = getResolvedOptions(sc, ["JOB_NAME"])
19     sc = SparkContext()
20     glueContext = GlueContext(sc)
21     spark = glueContext.spark_session
22     job = Job(glueContext)
23     job.init(args["JOB_NAME"], args)
24
25     # Load source data from AWS Data Catalog
26     sourceDF = glueContext.create_dynamic_frame.from_catalog(
27         database="cancretarget",
28         table_name="hospitalitytype",
29         transformation_ctx="sourceDF")
30
31
32     # If sourceDF.count() == 0:
33     #     raise Exception("Source table is empty or not loaded.")
34
35
36     # Trig
37     targets = glueContext.create_dynamic_frame.from_catalog(
38         database="cancretarget",
39         table_name="hospitalitytype_target",
40         transformation_ctx="targetDF")
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
69
70
71
72
73
74
75
76
77
78
79
79
80
81
82
83
83
84
85
86
87
88
89
89
90
91
92
93
94
95
96
97
98
99
99
100
101
102
103
104
105
106
107
108
109
109
110
111
112
113
114
115
116
117
118
119
119
120
121
122
123
124
125
126
127
128
129
129
130
131
132
133
134
135
136
137
138
139
139
140
141
142
143
144
145
146
147
148
149
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
169
170
171
172
173
174
175
176
177
178
179
179
180
181
182
183
184
185
186
187
188
189
189
190
191
192
193
194
195
196
197
198
199
199
200
201
202
203
204
205
206
207
208
209
209
210
211
212
213
214
215
216
217
218
219
219
220
221
222
223
224
225
226
227
228
229
229
230
231
232
233
234
235
236
237
238
239
239
240
241
242
243
244
245
246
247
248
249
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2
```

For hospitaltype datawarehouse table also the surrogate key is added and loaded. Now the job is run ,

Job runs (1/4) Info										Last updated (UTC)	View details	Stop job run	Troubleshoot with AI
Run status	Retries	Start time (Local)	End time (Local)	Duration	Capacity (DPUs)	Worker type	Glue						
Success Successed	0	11/22/2024 14:41:44	11/22/2024 14:43:15	1m 20s	10 DPUs	G.1X	4.0						
Failure Failed	0	11/22/2024 14:19:46	11/22/2024 14:23:15	1m 18s	10 DPUs	G.1X	4.0						
Success Successed	0	11/22/2024 14:16:32	11/22/2024 14:18:05	1m 22s	10 DPUs	G.1X	4.0						
Success Successed	0	11/22/2024 04:34:39	11/22/2024 04:36:08	1m 21s	10 DPUs	G.1X	4.0						

Hospital Job:

```

if targetDF:
    query = """
        SELECT
            ROW_NUMBER() OVER (ORDER BY h.hospitalid ASC) AS HospitalKey, -- Generate sequential HospitalKey
            h.hospitalid,
            h.name,
            h.phoneno AS Phone_Number,
            h.email AS EmailID,
            c.citykey AS CityKey, -- Replace location with CityKey from city_table
            t.typekey AS Typekey -- Replace type with TypeKey from type_table
        FROM hospital_table h
        LEFT ANTI JOIN target_table tgt
            ON TRIM(h.hospitalid) = TRIM(tgt.hospitalid) -- Avoid duplicates
        INNER JOIN city_table c
            ON TRIM(h.location) = TRIM(c.cityname) -- Match location with cityname
        INNER JOIN type_table t
            ON TRIM(h.type) = TRIM(t.typename) -- Match type with typename
        WHERE h.hospitalid IS NOT NULL AND TRIM(h.hospitalid) != ''
    """
else:
    query = """
        SELECT
            ROW_NUMBER() OVER (ORDER BY h.hospitalid ASC) AS HospitalKey,
            h.hospitalid,
            h.name,
            h.phoneno AS Phone_Number,
            h.email AS EmailID,
            c.citykey AS CityKey, -- Replace location with CityKey from city_table
            t.typekey AS Typekey -- Replace type with TypeKey from type_table
        FROM hospital_table h
        INNER JOIN city_table c
            ON TRIM(h.location) = TRIM(c.cityname) -- Match location with cityname
        INNER JOIN type_table t
            ON TRIM(h.type) = TRIM(t.typename) -- Match type with typename
        WHERE h.hospitalid IS NOT NULL AND TRIM(h.hospitalid) != ''
    """

```

Now for the hospital table we join the citytable and the hospitaltype table to map the corresponding citykey and hospitaltypekey in the hospital datawarehouse table.

```

    www

filtered_new_records = spark.sql(query)

resultDynamicFrame = DynamicFrame.fromDF(filtered_new_records, glueContext, "resultDynamicFrame")

sasSink = glueContext.getSink(
    path="s3://canonstartgsd/hospital/",
    connection_type="s3",
    updateBehavior="UPDATE_IN_DATABASE",
    partitionKeys=[],
    enablePartitioning=True,
    transformation_ctx="sasSink"
)
sasSink.setCatalogEntry(catalogDatabase="canon_target", catalogTableName="hospital_target")
sasSink.setFormat("json", "glueparquet", compression="SNAPPY")
sasSink.writeFrame(resultDynamicFrame)

# Commit the job
job.commit()

```

And the job is run,

The screenshot shows the AWS Glue Job Runs page for the 'hospital_target_job'. The page header includes navigation tabs for Script, Job details, Burns, Data quality, Schedules, Version Control, and Upgrade analysis - preview. On the right, there are buttons for Actions, Save, and Run. A note at the top right indicates the last modified time as 11/12/2024, 2:11:08 AM.

Job runs (1/4) (1)

Run status	Version	Start time (Local)	End time (Local)	Duration	Capacity (DPUs)	Worker type	Glue version
Succeeded	0	11/12/2024 14:56:08	11/12/2024 14:57:40	1m 25s	10 DPUs	G.TK	4.0
Succeeded	0	11/12/2024 14:58:16	11/12/2024 14:59:12	1m 21s	10 DPUs	G.TK	4.0
Failed	0	11/12/2024 14:48:38	11/12/2024 14:49:25	8s	10 DPUs	G.TK	4.0
Succeeded	0	11/12/2024 11:09:33	11/12/2024 11:10:08	1m 27s	10 DPUs	G.TK	4.0

Last modified at 11/12/2024, 2:11:08 AM

Actions ▾ Save Run

View details Stop job run Transition with AI Table View Card View

Speciality Job:

The screenshot shows the AWS Glue Data Catalog interface. On the left, there's a sidebar with navigation links like 'Getting Started', 'AWS Glue', 'Jobs', 'Tables', 'Workflows', 'Jobs run recently', 'Data Catalog tables', 'Data connections', 'Variables (versioned)', 'Data Catalog', 'Database', 'Tables', 'Create schema explorer', 'Schemas', 'Connections', 'Data types', 'Classifiers', 'Catalog storage', 'Data integration and ETL', and 'Legacy pages'. Below that are 'Where Next', 'Documentation', 'AWS Marketplace', and two checkboxes for 'Enable compact mode' and 'Enable new navigation'. The main area has tabs for 'Script', 'Job details', 'Runs', 'Data quality', 'Schedules', 'Version Control', and 'Upgrade analysis - preview'. The 'Script' tab is selected, displaying a large block of Python code for the 'speciality_target_job'. The code performs various Glue operations such as creating databases, tables, and partitions, and inserting data into them. It includes logic to handle source and target tables, and it handles errors by inserting all source records into the target table if the target table does not exist.

```
1 insert into
2   from aodglue_transfers.job
3   into aodglue_mltsc import getsourceconnections
4   using aodglue_mltsc.getworklist
5   into aodglue_mltsc.create_table_ifnotexists
6   from aodglue_db import id
7   into aodglue_mltsc.insertsynonyms
8
9   # and speciality_mltsc.insert, query, mapping, transformation_ctx) -> dynamicframe:
10   # for alias, frame in mapping.items():
11   #     frame.settable().createReplaceTempView(alias)
12   result = glueContext.create_dynamic_frame.from_options(
13       connection_type="redshift",
14       database="speciality",
15       args={ "redshiftEdition": "single-node", "port": "3000", "host": "host" },
16       transformation_ctx="speciality_mltsc")
17
18   # and gettables(options=args, ["3000", "host"])
19   # spark = glueContext.spark_session
20   # job = Job(glueContext, "spark_usage")
21   # job.addResolvedSource("spark_usage", args)
22
23   # sources = glueContext.create_dynamic_frame.from_catalog(
24   #     database="excar_source",
25   #     table_name="speciality",
26   #     transformation_ctx="spark_usage")
27
28   # try:
29   #     targetdt = glueContext.create_dynamic_frame.from_catalog(
30   #         database="excar_target",
31   #         table_name="speciality_target",
32   #         transformation_ctx="targetdt")
33   #     targetdt.create_external_table()
34   # except Exception as e:
35   #     print(e)
36   #     print("targetdt table does not exist, proceeding to insert all source records...")
37   targetdt = None
```

```

if targetDF:
    query = """
        SELECT
            ROW_NUMBER() OVER (ORDER BY src.Speciality ASC) AS SpecialityKey, -- Generate unique keys starting from 1
            src.Speciality
        FROM source_table src
        LEFT ANTI JOIN target_table tgt
        ON TRIM(src.Speciality) = TRIM(tgt.Speciality) -- Exclude already present records
        WHERE src.Speciality IS NOT NULL AND TRIM(src.Speciality) != ''
    """
else:
    query = """
        SELECT
            ROW_NUMBER() OVER (ORDER BY Speciality ASC) AS SpecialityKey, -- Generate unique keys starting from 1
            Speciality
        FROM source_table
        WHERE Speciality IS NOT NULL AND TRIM(Speciality) != ''
    """

# Execute the query
filtered_new_records = spark.sql(query)

# Convert the filtered records back to a Glue DynamicFrame
filtered_new_records_DF = DynamicFrame.fromDF(filtered_new_records, glueContext, "filtered_new_records_DF")

# Write the new records to the target table in S3 and update the Glue Catalog
s3Sink = glueContext.getSink(
    path="s3://cancuretargetdw/speciality/",
    connection_type="s3",
    updateBehavior="LOG",
    partitionKeys=[],
    enableUpdateCatalog=True,
    transformation_ctx="s3Sink"
)

```

hon Ln 42, Col 1 Errors: 0 | Warnings: 0

As discussed in all the other tables above we add surrogate key for the speciality table and load it in the speciality datawarehouse table.
Now the job is run.

Run status	Retries	Start time (Local)	End time (Local)	Duration	Capacity (DPU)	Worker type	Glue version
Success	0	11/22/2024 15:02:55	11/22/2024 15:04:51	1m 25s	10 DPU	G.1X	4.0
Success	0	11/22/2024 10:20:22	11/22/2024 10:22:04	1m 21s	10 DPU	G.1X	4.0

Doctor Job:

doctor_target_job

Script Job details Runs Data quality Schedules Version Control Upgrade analysis - preview

Last modified on 1/12/2021, 10:03:11 AM Action Save Run

```

1 import sys
2 from awsglue.transforms import *
3 from awsglue.dynamic import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7 from awsglue.dynamicframe import DynamicFrame
8
9 # add sparkSQLQuery(glueContext, query, mapping, transformation_ctx) as dynamicFrame:
10 #     for alias, frame in mapping.items():
11 #         frame.withColumnRenamed("name", alias).create_dynamic_frame(glueContext)
12 #     result = glueContext.create_dynamic_frame.from_options(query)
13 #     return DynamicFrame.fromDF(result, glueContext, transformation_ctx)
14
15 try:
16     args = getResolvedOptions(sys.argv, ["JOB_NAME"])
17     glueContext = GlueContext()
18     spark = glueContext.spark_session
19     job = Job(glueContext)
20     job.init(args["JOB_NAME"], args)
21
22     # create glueContext.create_dynamic_frame.from_catalog(
23     #     database="cancure_source",
24     #     table_name="doctor",
25     #     transformation_ctx="doctor")
26
27     # specialityDf = glueContext.create_dynamic_frame.from_catalog(
28     #     database="cancure_target",
29     #     table_name="speciality_target",
30     #     transformation_ctx="speciality")
31
32     # try:
33
34     # if targetDF:
35     #     query = """
36     #         SELECT
37     #             ROW_NUMBER() OVER (ORDER BY d.doctorid ASC) AS DoctorKey, -- Generate sequential DoctorKey
38     #             d.doctorid, -- Retain Doctor_ID
39     #             d.name AS Doctor_Name, -- Map Name to Doctor_Name
40     #             d.email AS Email, -- Retain Email
41     #             CAST(d.yoe AS INT) AS YOE, -- Cast YOE to INT
42     #             s.specialitykey -- Map speciality to SpecialityKey
43     #         FROM doctor_table d
44     #         LEFT ANTI JOIN target_table t
45     #             ON TRIM(d.doctorid) = TRIM(t.doctorid) -- Avoid duplicates
46     #         INNER JOIN speciality_table s
47     #             ON TRIM(d.speciality) = TRIM(s.speciality) -- Match speciality with speciality_key
48     #         WHERE d.doctorid IS NOT NULL AND TRIM(d.doctorid) != ''
49     #     """
50
51     # else:
52     #     query = """
53     #         SELECT
54     #             ROW_NUMBER() OVER (ORDER BY d.doctorid ASC) AS DoctorKey, -- Generate sequential DoctorKey
55     #             d.doctorid, -- Retain Doctor_ID
56     #             d.name AS Doctor_Name, -- Map Name to Doctor_Name
57     #             d.email AS Email, -- Retain Email
58     #             CAST(d.yoe AS INT) AS YOE, -- Cast YOE to INT
59     #             s.specialitykey -- Map speciality to SpecialityKey
60     #         FROM doctor_table d
61     #         INNER JOIN speciality_table s
62     #             ON TRIM(d.speciality) = TRIM(s.speciality) -- Match speciality with speciality_key
63     #         WHERE d.doctorid IS NOT NULL AND TRIM(d.doctorid) != ''
64     #     """
65
66
67     filtered_new_records = spark.sql(query)
68
69
70     # ---#
71
72     #84     filtered_new_records = spark.sql(query)
73
74
75     #87     resultDynamicFrame = DynamicFrame.fromDF(filtered_new_records, glueContext, "resultDynamicFrame")
76
77
78     #90     s3Sink = glueContext.getSink(
79     #91         path="s3://cancuretargetglue/doctor/",
80     #92         connection_type="s3",
81     #93         update_behavior="LOG",
82     #94         partition_keys=[],
83     #95         enableUpdateCatalog=True,
84     #96         transformation_ctx="s3sink"
85     #97     )
86     #98     s3Sink.setCatalogInfo(catalogDatabase="cancure_target", catalogTableName="doctor_target")
87     #99     s3Sink.setFormat("glueparquet", compression="snappy")
88     #100    s3Sink.writeFrame(resultDynamicFrame)
89
90     # Commit the job
91     job.commit()
92
93 
```

Now we join the speciality data warehouse and doctor source table to map the speciality key in the doctor datawarehouse table and we also add surrogate key to the doctor table.

Now the job is run.

doctor_target_job										
Script Job details Runs Data quality Schedules Version Control Upgrade analysis - preview										
Last update 1 min ago View details Cross job view Troubleshoot with AI Table View Card View ...										
Job runs (1/2) View										
<input checked="" type="checkbox"/> After job runs for unsupervised learning										
Run status	Duration	Start time (Local)	End time (Local)	Duration	Capacity (DPU)	Worker type	File version			
● Succeeded	0	11/22/2024 15:09:17	11/22/2024 15:09:49	1 min 32 s	10 DPU	0.1K	4.0			
● Succeeded	0	11/22/2024 10:34:31	11/22/2024 10:35:27	1 min 29 s	10 DPU	0.1K	4.0			

Cancer Job:

Script 101

```

49 sourceDF.toDF().createOrReplaceTempView("source_table")
50
51
52 * if targetDF1:
53     query = """
54         SELECT
55             ROW_NUMBER() OVER (ORDER BY src.stage, src.cancertype) AS Cancerkey, -- Generate surrogate key
56             src.stage, -- Keep the Stage column
57             src.cancertype -- Keep the Cancer Type column
58         FROM source_table src
59         LEFT ANTI JOIN target_table tgt
60         ON TRIM(src.stage) = TRIM(tgt.stage) AND TRIM(src.cancertype) = TRIM(tgt.cancertype) -- Exclude duplicates
61         WHERE src.stage IS NOT NULL AND src.cancertype IS NOT NULL
62             AND TRIM(src.stage) != '' AND TRIM(src.cancertype) != '' -- Exclude NULL or blank rows
63     """
64
65 * else:
66     query = """
67         SELECT
68             ROW_NUMBER() OVER (ORDER BY stage, cancertype) AS Cancerkey, -- Generate surrogate key
69             stage, -- Keep the Stage column
70             cancertype -- Keep the Cancer Type column
71         FROM source_table
72         WHERE stage IS NOT NULL AND cancertype IS NOT NULL
73             AND TRIM(stage) != '' AND TRIM(cancertype) != '' -- EXCLUDE NULL OR blank rows
74     """
75
76 filtered_new_records = spark.sql(query)
77
78
79 filtered_new_records_DF = DynamicFrame.fromDF(filtered_new_records, glueContext, "filtered_new_records_DF")
80
81
82 ussink = glueContext.getSink(
83     path="s3://canceretarget/cancer/",
84     type="AmazonRedshift"
85 )

```

```
64     WHERE stage IS NOT NULL AND cancertype IS NOT NULL
65         AND TRIM(stage) != '' AND TRIM(cancertype) != '' -- Exclude NULL or blank rows
66     """
67
68
69 filtered_new_records = spark.sql(query)
70
71
72 filtered_new_records_DF = DynamicFrame.fromDF(filtered_new_records, glueContext, "filtered_new_records_DF")
73
74
75 s3Sink = glueContext.getSink(
76     path="s3://cancuretargetdw/cancer/",
77     connection_type="s3",
78     updateBehavior="LOG",
79     partitionKeys=[],
80     enableUpdateCatalog=True,
81     transformation_ctx="s3sink"
82 )
83 s3Sink.setCatalogInfo(catalogDatabase="cancure_target", catalogTableName="cancer_target")
84 s3Sink.setFormat("glueparquet", compression="snappy")
85 s3Sink.writeFrame(filtered_new_records_DF)
86
87 # Commit the job
88 job.commit()
89
```

Python Ln 74, Col 1 Errors: 0 Warnings: 0

And similar to other tables we add surrogate key and load the data to the cancer data warehouse table.

Now the job is run,

Treatment type Job:

```

sourceDF.toDF().createOrReplaceTempView("source_table")

if targetDF:
    query = """
        SELECT
            ROW_NUMBER() OVER (ORDER BY src.TreatmentType ASC) AS TreatmentKey, -- Generate unique keys starting from 1
            src.TreatmentType -- Include the existing TreatmentType column
        FROM source_table src
        LEFT ANTI JOIN target_table tgt
        ON TRIM(src.TreatmentType) = TRIM(tgt.TreatmentType) -- Exclude already present records
        WHERE src.TreatmentType IS NOT NULL AND TRIM(src.TreatmentType) != '' -- Exclude NULL or blank rows
    """
else:
    query = """
        SELECT
            ROW_NUMBER() OVER (ORDER BY TreatmentType ASC) AS TreatmentKey, -- Generate unique keys starting from 1
            TreatmentType -- Include the existing TreatmentType column
        FROM source_table
        WHERE TreatmentType IS NOT NULL AND TRIM(TreatmentType) != '' -- Exclude NULL or blank rows
    """

# Create a sink
s3sink = glueContext.getSink(
    path="s3://cancuretargetdw/treatmenttype/",
    connection_type="s3",
    updateBehavior="LOG",
    partitionKeys=[],
    enableUpdateCatalog=True,
    transformation_ctx="s3sink"
)
s3sink.setCatalogInfo(catalogDatabase="cancure_target", catalogTableName="treatmenttype_target")
s3sink.setFormat("glueparquet", compression="snappy")
s3sink.writeFrame(filtered_new_records_DF)

# Commit the job
job.commit()

```

Now we add the surrogate key to the treatment type datawarehouse table and load the data.

Now the job is run,

Run ID	Status	Start Time (Local)	End Time (Local)	Duration	Capacity (DPUs)	Worker Type	Glue version
1	Successful	11/22/2024 13:32:16	11/22/2024 13:33:44	1m 14s	10 DPUs	G.1X	4.0
2	Successful	11/22/2024 04:21:40	11/22/2024 04:23:03	1m 15s	10 DPUs	G.1X	4.0

Record Job:

Run ID	Status	Start Time (Local)	End Time (Local)	Duration	Capacity (DPUs)	Worker Type	Glue version
1	Successful	11/22/2024 04:21:40	11/22/2024 04:23:03	1m 15s	10 DPUs	G.1X	4.0

```

Script Info
45 * If targetDF
46   query = """
47     SELECT
48       ROW_NUMBER() OVER (ORDER BY src.recordid ASC) AS RecordKey, -- Generate unique keys starting from
49       src.recordID AS RecordID, -- Include RecordID
50       src.treatmentType AS TreatmentType, -- Include TreatmentType
51       src.outcome AS Outcome -- Include outcome
52     FROM source_table src
53     LEFT ANTI JOIN target_table tgt
54     ON TRIM(src.recordID) = TRIM(tgt.RecordID) -- Exclude already present records
55     WHERE src.recordID IS NOT NULL AND TRIM(src.recordID) != '' -- Exclude NULL or blank rows
56   """
57 * else:
58   query = """
59     SELECT
60       ROW_NUMBER() OVER (ORDER BY recordid ASC) AS RecordKey, -- Generate unique keys starting from 1
61       recordid AS RecordID, -- Include RecordID
62       treatmenttype AS TreatmentType, -- Include TreatmentType
63       outcome AS outcome -- Include outcome
64     FROM source_table
65     WHERE recordid IS NOT NULL AND TRIM(recordid) != '' -- Exclude NULL or blank rows
66   """
67
68
69 filtered_new_records = spark.sql(query)
70
71
72 filtered_new_records_DF = DynamicFrame.fromDF(filtered_new_records, glueContext, "filtered_new_records_DF")
73
74
75 s3Sink = glueContext.getSink(
76   path="s3://cancuretargetdb/record/",
77   connection_type="s3",
78   updateBehavior="追加",
79   partitionKeys=[],
80   enableUpdateCatalog=True,
81   transformation_ctx="s3Sink"
82 )

```

Now we add the surrogatekey to the record datawarehouse table and load the data.
Now the job is run,

Job runs (1/1) Info										Last modified Edit	View details	Stop job run	Rechedule next with AI	Table View	Card View
Run state	Run ID	Run name	Start time (Local)	End time (Local)	Duration	Capacity (DPU)	Instance type	Git version							
Success	1	11/22/2024 16:10:48	11/22/2024 16:12:15	1m 19s	10 DPU	G.1x	4.0								

Patient Job:

AWS Glue

Getting started
ETL jobs
Visual ETL
Notebooks
Job monitoring
Data catalog
Data connectors
Workflow (orchestrator)
Data Catalog
Databases
Tables
Schema evolution registries
Schemas
Connections
Crawlers
Glaucers
Catalog settings

Data Integration and ETL
Legacy pages

What's New
Documentation
AWS Marketplace

Enable compact mode
Enable new navigation

patient_target_job

Script Job details Runs Data quality Schedules Version Control Upgrade analysis - preview

```
Script: #!/usr/bin/python
# Standard library imports
1 import sys
2 import json
3 import logging
4 import time
5 from builtins import str
6
7 # Third party imports
8 from glue import transformation
9 from glue import exceptions
10 from pyarrow import parquet
11 from pyarrow import json as pa_json
12 from pyarrow import Table
13 from pyarrow import Schema
14 from pyarrow import DynamicFrame
15
16 # Local imports
17 from spark import glueContext, spark, mapping, transformation_cbs
18
19 for alias, frame in mapping.read():
20     alias, frame = alias, frame.replace("temp", "alias")
21     frame.toDF().createTempView(alias)
22     result = glueContext.create_dynamic_frame.from_options(
23         frame,
24         transformation_cbs[alias],
25         "transformed",
26         transformation_cbs[alias])
27     result = transformation_cbs[alias](result)
28     result.write.parquet(f"/tmp/{alias}_out")
29
30     print(f"Transformed {alias} to {f'/tmp/{alias}_out'}")
31
32     target_name = f'{alias}_target'
33     target_table = f'{alias}_target'
34     target_db = f'{alias}_target'
35     target_cbs = f'{alias}_target'
36
37     target_cbs.create_view_if_not_exists(target_table) == target_cbs.target_view
38
39 except Exception as e:
```

```

51
52 ▼ if targetDF:
53     query = """
54         SELECT
55             ROW_NUMBER() OVER (ORDER BY src.patientid) AS PatientKey, -- Generate surrogate key
56             src.patientid AS PatientID, -- Include PatientID
57             src.name AS Name, -- Include Name
58             src.phoneno AS PhNo, -- Include Phone Number
59             src.gender AS Gender, -- Include Gender
60             src.diseasehistory AS DiseaseHistory, -- Include Disease History
61             CAST(src.age AS INT) AS Age -- Cast Age as INT
62         FROM source_table src
63         LEFT ANTI JOIN target_table tgt
64         ON TRIM(src.patientid) = TRIM(tgt.PatientID) -- Avoid duplicates based on PatientID
65         WHERE src.patientid IS NOT NULL AND TRIM(src.patientid) != '' -- Exclude NULL or blank rows
66     """
67 ▼ else:
68     query = """
69         SELECT
70             ROW_NUMBER() OVER (ORDER BY patientid) AS PatientKey, -- Generate surrogate key
71             patientid AS PatientID, -- Include PatientID
72             name AS Name, -- Include Name
73             phoneno AS PhNo, -- Include Phone Number
74             gender AS Gender, -- Include Gender
75             diseasehistory AS DiseaseHistory, -- Include Disease History
76             CAST(age AS INT) AS Age -- Cast Age as INT
77         FROM source_table
78         WHERE patientid IS NOT NULL AND TRIM(patientid) != '' -- Exclude NULL or blank rows
79     """
80
81
filtered_new_records_DF = DynamicFrame.fromDF(filtered_new_records, glueContext, "filtered_new_records_DF")

s3Sink = glueContext.getSink(
    path="s3://cancuretargetdw/patient/",
    connection_type="s3",
    updateBehavior="LOG",
    partitionKeys=[],
    enableUpdateCatalog=True,
    transformation_ctx="s3Sink"
)
s3sink.setCatalogInfo(catalogDatabase="cancure_target", catalogTableName="patient_target")
s3sink.setFormat("glueparquet", compression="snappy")
s3sink.writeFrame(filtered_new_records_DF)

# Commit the job
job.commit()

```

The same process is done for the patient job.

The job is run,

Run status	Workers	Start time (Local)	End time (Local)	Duration	Capacity (DPUs)	Worker type	Glue version
Successful	0	11/22/2024 15:45:05	11/22/2024 15:46:40	1m 20s	10 DPUs	G.1K	4.0
Successful	0	11/22/2024 10:45:31	11/22/2024 10:46:55	1m 14s	10 DPUs	G.1K	4.0

Fact Tables:

Review Fact:

AWS Glue

Getting started
ETL jobs
Visual ETL
Notebooks
Jobs run monitoring
Data Catalog tables
Data catalog metrics
Performance instrumentation

▪ Data Catalog
Database
Tables
String schema registries
Schemas
Connections
Cursors
Gauges
Catalog settings

▶ Data integration and ETL

▶ Legacy pages

Whitelabel
Documentation
AWS Marketplace

Enable compact mode
Enable non-navigable

reviews_fact_target

Script Job details Runs Data quality Schedules Version Control Upgrade analysis - preview

```
1 import sys
2 from awsglue.transforms import *
3 from awsglue.dynamic import *
4 from awsglue.context import *
5 from awsglue.job import *
6 from awsglue.datacatalog import *
7 from awsglue.datasync import *
8
9 # define sparkcontext, s3n, mapping, transformation_ddbs
10 # for alias, frame is mapping item
11 # Pre-requisites: creating glue context and alias
12 result = + sparkcontext
13 result.setTempTable("review")
14 return result
15 result.datasync.fetch(results, glucoset, transformation_ddbs)
16
17 args = glucoset.getOptions().getArgs()
18
19 glucoset = GlueContext(sc)
20 s3n = glueContext.s3().createBucket("s3n")
21 bob = "bob" + glucoset.getTempTable()
22 jin = "jin" + glucoset.getTempTable()
23 jin.setArgv("--name", "jin", args)
24
25 # Load source
26 reviewsource = glucoset.create_dynamic_frame.from_catalog(
27     database="source_source",
28     table_name="review",
29     transformation_ctx="reviewsource"
30 )
31
32 transformsource_ddb = glucoset.create_dynamic_frame.from_catalog(
33     database="source_source",
34     table_name="transform",
35     transformation_ctx="transformsource"
36 )
37
38 postjoin = glucoset.create_dynamic_frame.from_catalog(
39     database="source_source",
40     table_name="postjoin",
41     transformation_ctx="postjoin"
42 )
43
44 postjoin = glucoset.create_dynamic_frame.from_catalog(
45     database="source_source",
46     table_name="postjoin",
47     transformation_ctx="postjoin"
48 )
49
50 postjoin = glucoset.create_dynamic_frame.from_catalog(
51     database="source_target",
52     table_name="postjoin",
53     transformation_ctx="postjoin"
54 )
```

reviews_fact_target

Last modified on 11/22/2024, 4:25:22 PM Actions Save

Script Job details Runs Data quality Schedules Version Control Upgrade analysis - preview

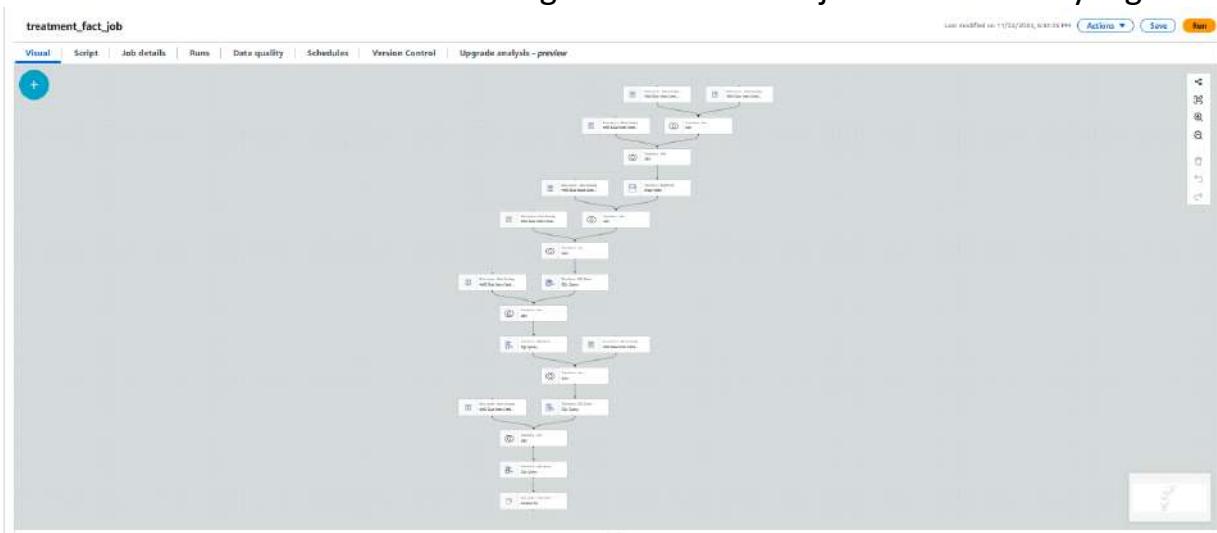
```
Script info
72 w // reviewFactDF
73     query = """
74     SELECT
75         ROW_NUMBER() OVER (ORDER BY r.reviewId ASC) AS reviewacy,
76         p.patientkey,
77         d.doctorkey,
78         h.hospitalkey,
79         r.reviewId AS reviewId,
80         CAST(r.date AS DATE) AS Date,
81         r.comments AS comments,
82         CAST(r.rating AS INT) AS Rating,
83         CAST(r.likes AS INT) AS ReviewerLikesScore
84     FROM reviews r
85     LEFT JOIN posts p ON r.patientid = p.patientid
86     LEFT JOIN doctors d ON d.doctorid = p.doctorid
87     LEFT JOIN patients p ON p.patientid = p.patientid
88     LEFT JOIN hospitals h ON h.hospitalid = p.hospitalid
89     LEFT JOIN 2018_target_table t ON r.reviewId = t.reviewId
90     """
91 
92     query = """
93     SELECT
94         ROW_NUMBER() OVER (ORDER BY r.reviewId ASC) AS reviewacy,
95         p.patientkey,
96         d.doctorkey,
97         h.hospitalkey,
98         r.reviewId AS reviewId,
99         CAST(r.date AS DATE) AS Date,
100        p.comments AS comments,
101        CAST(r.rating AS INT) AS Rating,
102        CAST(r.likes AS INT) AS ReviewerLikesScore
103    FROM reviews r
104    LEFT JOIN posts p ON r.patientid = p.patientid
105    LEFT JOIN doctors d ON d.doctorid = p.doctorid
106    LEFT JOIN patients p ON p.patientid = p.patientid
107    LEFT JOIN hospitals h ON h.hospitalid = p.hospitalid
108    """
109 
```

Now for the reviews fact table in addition to the source table we join the patient, hospital and doctor fact table to map their respective keys. And we also added the surrogate key for the review fact table.

The job is run.

Treatment Fact Table:

For treatment fact table we are using visual AWS Glue job since it is very big:



For the treatment fact table in addition to the source table we map various data warehouse tables like patient, doctor, hospital, cancer, record in order to map their corresponding keys and we also add a surrogate key to the treatment fact table.

Thus all the jobs are successfully executed and the tables are loaded in the target database,

Now a general explanation of individual jobs are completed now the following are the transformations done in the process,

Transformations:

We have done multiple transformations in many tables,

City Table:

```
if targetDF:
    query = """
        SELECT
            ROW_NUMBER() OVER (ORDER BY src.CityName ASC) AS CityKey,
            src.CityName,
            CAST(src.Population AS INT) AS Population,
            CAST(src.StateKey AS INT) AS StateKey
        FROM source_table src
        LEFT ANTI JOIN target_table tgt
        ON TRIM(src.CityName) = TRIM(tgt.CityName) AND src.StateKey = tgt.StateKey
        WHERE src.CityName IS NOT NULL AND TRIM(src.CityName) != ''
    """

else:
    query = """
        SELECT
            ROW_NUMBER() OVER (ORDER BY CityName ASC) AS CityKey,
            CityName,
            CAST(Population AS INT) AS Population,
            CAST(StateKey AS INT) AS StateKey
        FROM source_table
        WHERE CityName IS NOT NULL AND TRIM(CityName) != ''
    """

filtered_new_records = spark.sql(query)

# filtered_new_records_DF = DynamicFrame.fromDF(filtered_new_records, glueContext, "filtered_new_records_DF")

# Write the new records to the target table in s3 and update the glue catalog
s3sink = glueContext.getSink(
    path="s3://cancloudtargetdw/city/",
    connection_type="s3",
    updateBehavior="OOB",
    partitionKeys=[],
    enableUpdateCatalog=True,
    transformation_ctx="s3KLine"
)
```

We have transformed the datatype of the incoming population column from string to int before loading it into the city datawarehouse table.

Doctor table

```

▼ if targetDF:
    query = """
        SELECT
            ROW_NUMBER() OVER (ORDER BY d.doctorid ASC) AS DoctorKey, -- Generate sequential DoctorKey
            d.doctorid, -- Retain Doctor_ID
            d.name AS Doctor_Name, -- Map Name to Doctor_Name
            d.email AS Email, -- Retain Email
            CAST(d.yoe AS INT) AS YOE, -- Cast YOE to INT
            s.specialitykey -- Map speciality to SpecialityKey
        FROM doctor_table d
        LEFT ANTI JOIN target_table t
            ON TRIM(d.doctorid) = TRIM(t.doctorid) -- Avoid duplicates
        INNER JOIN speciality_table s
            ON TRIM(d.speciality) = TRIM(s.speciality) -- Match speciality with speciality_key
        WHERE d.doctorid IS NOT NULL AND TRIM(d.doctorid) != ''
    """
    """
▼ else:
    query = """
        SELECT
            ROW_NUMBER() OVER (ORDER BY d.doctorid ASC) AS DoctorKey, -- Generate sequential DoctorKey
            d.doctorid, -- Retain Doctor_ID
            d.name AS Doctor_Name, -- Map Name to Doctor_Name
            d.email AS Email, -- Retain Email
            CAST(d.yoe AS INT) AS YOE, -- Cast YOE to INT
            s.specialitykey -- Map speciality to SpecialityKey
        FROM doctor_table d
        INNER JOIN speciality_table s
            ON TRIM(d.speciality) = TRIM(s.speciality) -- Match speciality with speciality_key
        WHERE d.doctorid IS NOT NULL AND TRIM(d.doctorid) != ''
    """
    """

filtered_new_records = spark.sql(query)

```

similarly even in the doctor table the Years of experience (YOE) column is transformed from string to int before loading it into the doctor data warehouse table.

Patient:

```

51
52 ▼ if targetDF:
53     query = """
54         SELECT
55             ROW_NUMBER() OVER (ORDER BY src.patientid) AS PatientKey, -- Generate surrogate key
56             src.patientid AS PatientID, -- Include PatientID
57             src.name AS Name, -- Include Name
58             src.phoneno AS PhNo, -- Include Phone Number
59             src.gender AS Gender, -- Include Gender
60             src.diseasehistory AS DiseaseHistory, -- Include Disease History
61             CAST(src.age AS INT) AS Age -- Cast Age as INT
62         FROM source_table src
63         LEFT ANTI JOIN target_table tgt
64             ON TRIM(src.patientid) = TRIM(tgt.PatientID) -- Avoid duplicates based on PatientID
65             WHERE src.patientid IS NOT NULL AND TRIM(src.patientid) != '' -- Exclude NULL or blank rows
66     """
67 ▼ else:
68     query = """
69         SELECT
70             ROW_NUMBER() OVER (ORDER BY patientid) AS PatientKey, -- Generate surrogate key
71             patientid AS PatientID, -- Include PatientID
72             name AS Name, -- Include Name
73             phoneno AS PhNo, -- Include Phone Number
74             gender AS Gender, -- Include Gender
75             diseasehistory AS DiseaseHistory, -- Include Disease History
76             CAST(age AS INT) AS Age -- Cast Age as INT
77         FROM source_table
78             WHERE patientid IS NOT NULL AND TRIM(patientid) != '' -- Exclude NULL or blank rows
79     """
80
81

```

Here the datatype of the age column is transformed from string to int before loading it into the patient datawarehouse table.

Review Fact:

```

reviews_fact_target
Script | Job details | Runs | Data quality | Schedules | Version Control | Upgrade analysis - preview
Last modified on 11/12/2020, 4:29:12 PM Actions Save Help

Script (100)
72 -- 34 reviewsFACT
73   -- 34 reviewsFACT
74   SELECT
75     ROW_NUMBER() OVER (ORDER BY r.reviewsID ASC) AS reviewkey,
76     r.patientkey,
77     r.doctorkey,
78     h.hospitalkey,
79     r.visitid AS visitkey,
80     CAST(r.date AS DATE) AS Date,
81     r.comments AS comments,
82     CAST(r.rating AS INT) AS Rating,
83     CAST(r.likes AS INT) AS ReviewerfulnessScore
84   FROM reviews r
85   LEFT JOIN patients po ON r.patientkey = po.patientkey
86   LEFT JOIN doctors d ON r.doctorkey = d.doctorkey
87   LEFT JOIN hospitals h ON r.hospitalid = h.hospitalid
88   LEFT JOIN target_table t ON r.visitid = t.visitid
89   ...
90   ...
91   * else
92     query = """
93   SELECT
94     ROW_NUMBER() OVER (ORDER BY r.reviewsID ASC) AS reviewkey,
95     r.patientkey,
96     r.doctorkey,
97     h.hospitalkey,
98     r.visitid AS visitkey,
99     CAST(r.date AS DATE) AS Date,
100    r.comments AS comments,
101    CAST(r.rating AS INT) AS Rating,
102    CAST(r.likes AS INT) AS ReviewerfulnessScore
103   FROM reviews r
104   LEFT JOIN patients po ON r.visitid = po.visitid
105   LEFT JOIN doctors d ON r.doctorkey = po.doctorkey
106   LEFT JOIN patients p ON p.visitid = po.visitid
107   LEFT JOIN hospitals h ON r.hospitalid = po.hospitalid
108   ...
109   ...

```

Similarly the datatype of rating and likes are transformed from string to int before loading in the target table.

Treatment Fact Table:

```

FROM mydatasource
;
SQLQuery_node1732318126664 = sparkSqlQuery(glueContext, query = sqlQuery3, mapping = {"myDataSource":Join_node1732318034202}, transformation_ctx = "SQLQuery_node1732318126664")
+ Script generated for node Join
Join_node1732318247109 - Join.apply(Frame1-AWSGlueDataCatalog_node1732318239446, frame2-SQLQuery_node1732318126664, keys1={"hospitalid"}, keys2={"hospitalid"}, transformation_ctx="Join_node1732318247109")
+ Script generated for node SQL QUERY
sqlQuery9 = """
SELECT
  ROW_NUMBER() OVER (ORDER BY Patientkey ASC) AS Treatmentkey, -- Generate surrogate key for treatmentFact
  Hospitalkey, -- Map Hospitalkey
  Patientkey, -- Map Patientkey
  Cancerkey, -- Map Cancerkey
  Doctorkey, -- Map Doctorkey
  Recordkey, -- Map Recordkey
  CAST(Bill AS DECIMAL(10, 2)) AS Bill, -- Cast Bill to DECIMAL
  DATEDIFF(CAST(date AS DATE), '1970-MM-DD'), TO_DATE(startday, 'yyyy-MM-dd')) AS Duration, -- calculate duration
  CAST(Insurance AS DECIMAL(10, 2)) AS Insurance -- Cast Insurance to DECIMAL
FROM
  myDataSource
;
SQLQuery_node1732318443700 = sparkSqlQuery(glueContext, query = sqlQuery8, mapping = {"myDataSource":Join_node1732318247109}, transformation_ctx = "SQLQuery_node1732318443700")
+ Script generated for node Join
Join_node1732318443700 - Join.apply(Frame1-AWSGlueDataCatalog_node1732318239446, frame2-SQLQuery_node1732318443700, keys1={"Treatmentkey"}, keys2={"Treatmentkey"}, transformation_ctx="Join_node1732318443700")

```

As we can see we have done multiple transformations in the treatment fact table. **The startday and endday string coming from the source table is converted to date and then subtracted to derive duration in int.**

Similarly the bill and the insurance string fields from the source table is converted to decimal before loading into the treatment fact table.

Now since all the tables are loaded into the datawarehouse catalog database named **cancure_target**, the below is the screenshot.

The screenshot shows the AWS Glue Database console. On the left, there's a sidebar with navigation links like 'Getting started', 'AWS Glue', 'Tables', 'Streams', 'Ingestions', 'Jobs', 'Runners', 'Data connectors', 'Data connections', 'Metrics (by job)', 'Data Catalog', 'Data integration and ETL', and 'Legacy pages'. The main area is titled 'cancure_target' and shows 'Database properties' with a 'Name' of 'cancure_target'. Below this is a table titled 'Tables (15)' listing 15 tables: 'country_target', 'city_target', 'country_target', and 'city_target'. Each table has columns for 'Name', 'Database', 'Location', 'Classification', 'Deprecated', 'View data', 'Data quality', and 'Customer statistics'. A note at the top says 'Announcing new optimization features for Apache Hive tables'.

The corresponding S3 bucket location is also shown.

The screenshot shows the Amazon S3 console. On the left, there's a sidebar with 'Amazon S3', 'Buckets', 'Actions', 'Access Grants', 'Access Policies', 'Object Lambda Access Points', 'Multi-Region Access Points', 'Batch Operations', 'Open Access Analyzer for S3', 'Bucket Public Access settings for this account', 'Storage Lens', 'Default rules', 'Storage class prices', 'AWS Organizations settings', 'Feature spotlight', and 'AWS Marketplace for S3'. The main area is titled 'cancuretargetdw' and shows the 'Objects (14)' page. It lists 14 objects: 'appointments', 'city', 'country', 'doctor', 'hospital', 'instrument', 'instrumentcat', 'patient', 'report', 'specialist', 'state', 'surgeon', and 'treatmentcat'. Each object has columns for 'Name', 'Type', 'Last modified', 'Size', and 'Storage class'.

Now checking the outputs with AWS Athena,
As we can see all tables are loaded **with surrogate keys and required structure**.
For each table:

Country:

The screenshot shows the AWS Athena console. On the left, there's a sidebar with 'Data', 'Data source', 'Amazon Athena', 'Database', 'cancure_target', 'Tables and views', and 'Views (0)'. The main area has tabs for 'Editor', 'Recent queries', 'Saved queries', and 'Settings'. There are three open queries: 'Query 11', 'Query 12', and 'Query 13'. Query 13 is selected and contains the SQL command: 'SELECT * FROM `redshiftdevdb`.`cancure_target`.`country_target` limit 200;'. Below the queries is a 'Results' section with a table titled 'Results (2)'. The table has two rows, both with the value '1' in the 'id' column and 'India' and 'United States' in the 'name' column. The bottom of the screen shows performance metrics: 'Time in queue: 67 ms', 'Peak time: 452 ms', and 'Data scanned: 0.13 KB'.

State:

The screenshot shows the AWS Glue Data Catalog interface. On the left, the sidebar displays the Data source (Amazon Redshift) and destination (Secure_target). The Tables and views section lists various tables such as cancer_target, city_target, country_target, doctor_target, hospital_target, hospitaltype_target, patient_target, record_target, reviewTarget, similarity_target, state_target, treatmentTarget, and treatmenttype_target. The Views section is empty. The main area shows a query editor with the following SQL command:

```
SELECT * FROM "redshiftcatalog"."Secure_target"."state_target" limit 10;
```

The results table contains the following data:

#	statekey	statename	countrykey
1	1	California	1
2	2	Florida	1
3	3	Georgia	1
4	8	Texas	1
5	4	Illinois	1
6	5	New York	1

Time in queue: 107 ms Run time: 466 ms Data scanned: 0.29 kB

City:

The screenshot shows the AWS Glue Data Catalog interface. The sidebar and table list are identical to the previous state screenshot. The main area shows a query editor with the following SQL command:

```
SELECT * FROM "redshiftcatalog"."Secure_target"."city_target" limit 10;
```

The results table contains the following data:

#	citykey	cityname	population	statekey
1	1	Atlanta	510000	1
2	2	Augusta	100000	1
3	3	Audubon	990000	2
4	4	Balboa	262000	4
5	5	Chicago	2790000	1
6	6	Dallas	1340000	2

Time in queue: 24 ms Run time: 127 ms Data scanned: 0.71 kB

HospitalType:

The screenshot shows a database query interface with the following details:

- Data Source:** AnonCatalog
- Database:** cancer_target
- Tables and views:** A sidebar lists tables such as cancer_target, clinc_target, country_target, doctor_target, hospital_target, hospitaltype_target, patient_target, record_target, reviewfact_target, specialty_target, state_target, treatmentfact_target, and treatmenttype_target.
- Query:** SELECT * FROM "AnonCatalog", "cancer_target", "HospitalType_Target" LIMIT 10;
- Results:** The results show two rows of data:

	typetkey	typename
1	1	Private
2	2	Public
- Performance Metrics:** Time in query: 50 ms, Run time: 401 ms, Data scanned: 0.13 KB.

Hospital:

The screenshot shows a database query interface with the following details:

- Data Source:** AnonCatalog
- Database:** cancer_target
- Tables and views:** A sidebar lists tables such as cancer_target, clinc_target, country_target, doctor_target, hospital_target, hospitaltype_target, patient_target, recordfact_target, specialty_target, state_target, treatmentfact_target, and treatmenttype_target.
- Query:** SELECT * FROM "cancer_target", "Hospital_Target" LIMIT 10;
- Results:** The results show six rows of data:

#	HospitalKey	HospitalID	Name	Phone_Number	EmailID	CityKey	TypeKey
1	1	HOSP001	Los Angeles Cancer Research Institute	(123)456-1000	contact@acrc.com	1	2
2	2	HOSP002	Los Angeles Oncology Institute	(123)456-2001	info@aoi.com	2	1
3	3	HOSP003	San Francisco Cancer Treatment Center	(123)456-2002	newcontact@sfcc.com	12	2
4	4	HOSP004	Bay Area Cancer Research Centre	(123)456-2003	info@bach.com	13	1
5	5	HOSP005	San Diego Oncology Center	(123)456-2004	contact@sdoc.com	14	2
6	6	HOSP006	Cancer Care Center	(123)456-2005	info@ccc.com	15	1
- Performance Metrics:** Time in query: 103 ms, Run time: 431 ms, Data scanned: 2.44 KB.

Speciality:

The screenshot shows a database query interface with the following details:

- Data Source:** AnonCatalog
- Database:** cancer_target
- Tables and views:** A sidebar lists tables such as cancer_target, clinc_target, country_target, doctor_target, hospital_target, hospitaltype_target, patient_target, record_target, reviewfact_target, specialty_target, state_target, treatmentfact_target, and treatmenttype_target.
- Query:** SELECT * FROM "cancer_target", "Speciality_Target" LIMIT 10;
- Results:** The results show six rows of data:

#	specialtykey	specialty
1	1	Breast Oncology
2	2	Dermatology Oncology
3	3	Gastrointestinal Oncology
4	4	Genitourinary Oncology
5	5	Gynecologic Oncology
6	6	Head and Neck Oncology
- Performance Metrics:** Time in query: 112 ms, Run time: 423 ms, Data scanned: 0.22 KB.

Doctor:

Data

Data source: AndDataCatalog

Tables and views: [Create](#)

Tables (11): [View table definitions](#)

- Cancer_target
- City_target
- Country_target
- Doctor_target
- Hospital_target
- Hospitaltype_target
- parent_target
- recent_target
- residencefact_target
- speciality_target
- state_target
- TreatmentFact_target
- TreatmentType_target

Views (0):

SQL: In 1. Col 1

Run again [Revert](#) [Cancel](#) [Clear](#) [Create](#)

Query results: [Completed](#)

Results (10)

#	doctorid	doctorid	doctor_name	email	age	speciality
1	1	D00201	Dr. Rajesh Munganti	rman@gmail.com	10	1
2	4	D00204	Dr. Victoria Weber	vweber@icloud.com	18	5
3	5	D00205	Dr. Horatio Bailey	hbailey@icloud.com	12	2
4	6	D00206	Dr. Alice Karp	akarp@icloud.com	2	3
5	7	D00207	Dr. Charlotte King	cking@icloud.com	20	8
6	8	D00208	Dr. Addison Montgomery	amontgomery@icloud.com	22	9

Time in query: 151 ms Run time: 203 ms Data scanned: 5.12 KB

[Copy](#) [Download results](#)

© 2014 Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

TreatmentType:

Data

Data source: AndDataCatalog

Tables and views: [Create](#)

Tables (11): [View table definitions](#)

- Cancer_target
- City_target
- Country_target
- Doctor_target
- Hospital_target
- Hospitaltype_target
- parent_target
- recent_target
- residencefact_target
- speciality_target
- state_target
- TreatmentFact_target
- TreatmentType_target

Views (0):

SQL: In 1. Col 1

Run again [Revert](#) [Cancel](#) [Clear](#) [Create](#)

Query results: [Completed](#)

Results (8)

#	treatmentkey	treatmenttype
1	1	Biologic
2	2	Chemotherapy
3	3	Horror
4	8	Targeted
5	4	Immunotherapy
6	5	Pelvic

Time in query: 131 ms Run time: 452 ms Data scanned: 0.23 KB

[Copy](#) [Download results](#)

Record:

Cancer:

The screenshot shows a database interface with a sidebar for 'Data' and 'Tables and views'. The main area displays a SQL query and its results. The query is:

```
SELECT * FROM "cancer_target", "treatment_target" LIMIT 10;
```

The results table has columns: ID, casenumber, and treatmenttype. The data is as follows:

ID	casenumber	treatmenttype
1	1	Surgery
2	2	Surgery
3	3	Radiation
4	4	Chemotherapy
5	5	Chemotherapy
6	6	Hormonal

Time in query: 0.00 ms Run time: 957 ms Data scanned: 1.20 KB

Cancer:

The screenshot shows a database interface with a sidebar for 'Data' and 'Tables and views'. The main area displays a SQL query and its results. The query is:

```
SELECT * FROM "cancer_target", "treatment_target" LIMIT 10;
```

The results table has columns: ID, casenumber, and stage. The data is as follows:

ID	casenumber	stage	cancertype
1	1	Stage I	Bladder Cancer
2	2	Stage I	Breast Cancer
3	3	Stage I	Cervical Cancer
4	4	Stage I	Laryngeal Cancer
5	5	Stage I	Leukemia
6	6	Stage I	Leukemia (Childhood)

Time in query: 106 ms Run time: 523 ms Data scanned: 0.06 KB

Patient:

Data

Data source: AnalyticsCatalog

Dataset: cancer_target

Tables and views:

- Tables (10)
 - Cancer_target
 - City_target
 - Country_target
 - Doctor_target
 - Hospital_target
 - HospitalType_target
 - Patient_target
 - Record_target
 - ReviewFact_target
 - Speciality_target
- Views (0)

SQL: In 1. Col 1

Run again | Delete | Cancel | Clear | Create |

Query results | Query stats

Results (10)

#	patientkey	patientid	name	phone	gender	dischargestatus	age
1	PA1001	Sarah Williams	555-CHE1	Female	No history	45	
2	PA1002	Mary Smith	555-CHE2	Female	Family history	50	
3	PA1003	Linda White	555-CHE3	Female	No history	47	
4	PA1004	Emily Clark	555-CHE4	Female	Diagnosed	55	
5	PA1005	Mark Thompson	555-CHE5	Male	Family history	42	
6	PA1006	Samantha Martin	555-CHE6	Female	No history	58	

Time in query: 101 ms Run time: 1.293 sec Data scanned: 545.05 kB

Copy | Download results |

Reviews Fact Table:

Data

Data source: AnalyticsCatalog

Dataset: cancer_target

Tables and views:

- Tables (10)
 - Cancer_target
 - City_target
 - Country_target
 - Doctor_target
 - Hospital_target
 - HospitalType_target
 - Patient_target
 - Record_target
 - ReviewFact_target
 - Speciality_target
- Views (0)

SQL: In 1. Col 1

Run again | Delete | Cancel | Clear | Create |

Query results | Query stats

Results (10)

#	reviewkey	patientkey	doctorkey	hospitalkey	reviewid	date	comments	rating	reviewhighlightrate
1	1	1	1	1	REV001	2023-01-10	Excellent care and attention to detail.	5	100
2	2	2	2	2	REV002	2023-01-15	Good service, but the wait time was long.	4	85
3	3	3	3	1	REV003	2023-01-20	Average experience, room for improvement.	3	70
4	4	4	4	2	REV004	2023-02-10	Fantastic staff and successful treatment.	5	140
5	5	5	5	3	REV005	2023-02-15	Uncomfortable stay, room could be cleaner.	2	45
6	6	6	6	4	REV006	2023-02-20	Professional doctors. Highly recommended.	4	95

Time in query: 101 ms Run time: 401 ms Data scanned: 3.12 kB

Copy | Download results |

Treatment Fact Table:

Data

Data source: AnalyticsCatalog

Dataset: cancer_target

Tables and views:

- Tables (10)
 - Cancer_target
 - City_target
 - Country_target
 - Doctor_target
 - Hospital_target
 - HospitalType_target
 - Patient_target
 - Record_target
 - ReviewFact_target
 - Speciality_target
- Views (0)

SQL: In 1. Col 1

Run again | Delete | Cancel | Clear | Create |

Query results | Query stats

Results (10)

#	insuranckey	hospitalkey	patientkey	doctorkey	recovkey	bill	duration	insurance
1	1	1	1	2	1	12000.00	90	8000.00
2	2	2	2	2	2	12000.00	90	5000.00
3	3	1	1	2	2	12000.00	90	8000.00
4	4	2	4	22	4	15000.00	85	10000.00
5	5	3	5	22	5	15000.00	85	10000.00
6	6	4	6	22	6	15000.00	85	8000.00

Time in query: 101 ms Run time: 420 ms Data scanned: 3.27 kB

Copy | Download results |

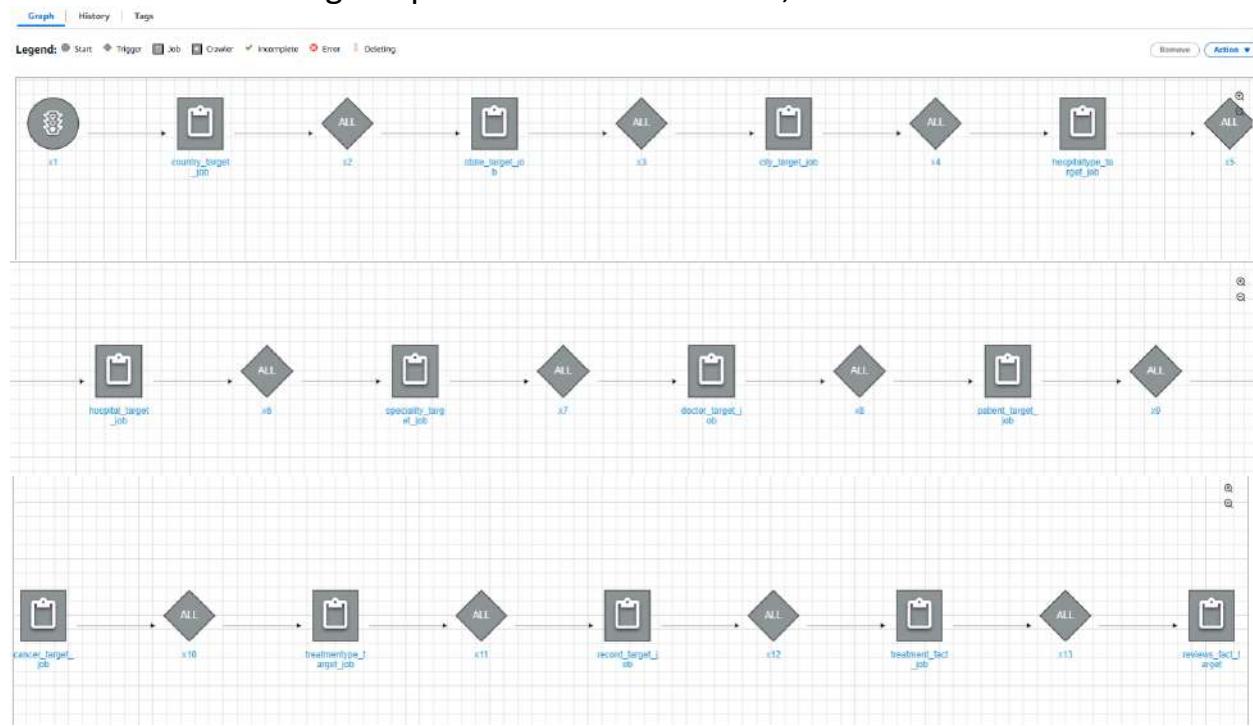
Thus all tables (dimensions and facts) are loaded properly.

Complete Workflow Pipeline:

We have also built a complete pipeline of the workflows and ran it , we are not running it again in the other account as it is incurring a lot of cost.



Now we are attaching the pictures of the workflow, below



After running this job the entire database got populated again with all the tables.

Thus we have also presented the logical architecture diagram and also the individual jobs and the final complete workflow orchestration pipeline diagram and loaded the data using both the methods. All the jobs without any dependency are added in the beginning followed by jobs which are dependent on the other jobs with treatment and review fact tables coming at the end.

Therefore we have satisfied all the requirements by **including 4+ tools to build the pipeline, presenting the architecture**(Individual jobs, logical architecture diagram, entire workflow pipeline), **Using programming language** like python to convert my source files into the required format and **using two different sources**, and **executing and loading all the jobs,pipeline and tables successfully**.

Conclusion:

The CanCure project successfully demonstrates the implementation of on-premise and cloud-based solutions for comprehensive data warehouses in the healthcare sector. By implementing a robust ETL pipeline using Talend and in the cloud using AWS Glue, S3 buckets, and Athena, the project captures and processes critical data, enabling meaningful insights. Dashboards created for treatment, reviews, and supplies provide actionable KPIs, such as treatment duration, hospital performance, and insurance coverage, empowering patients and stakeholders. This initiative bridges gaps in cancer care by improving transparency, enhancing decision-making, and driving better patient outcomes. The CanCure system exemplifies how technological advancements can revolutionize healthcare data analytics, setting a benchmark for future healthcare projects.

PRESENTATION MATERIAL:

SLIDE 1:



SLIDE 2

This slide is a two-column layout. The left column is a blurred photograph of a person's hands typing on a laptop keyboard. The right column contains text and a bulleted list. The text "Problem statement" is in bold. The bulleted list describes the gap in cancer treatment information and the project's aim to fill it.

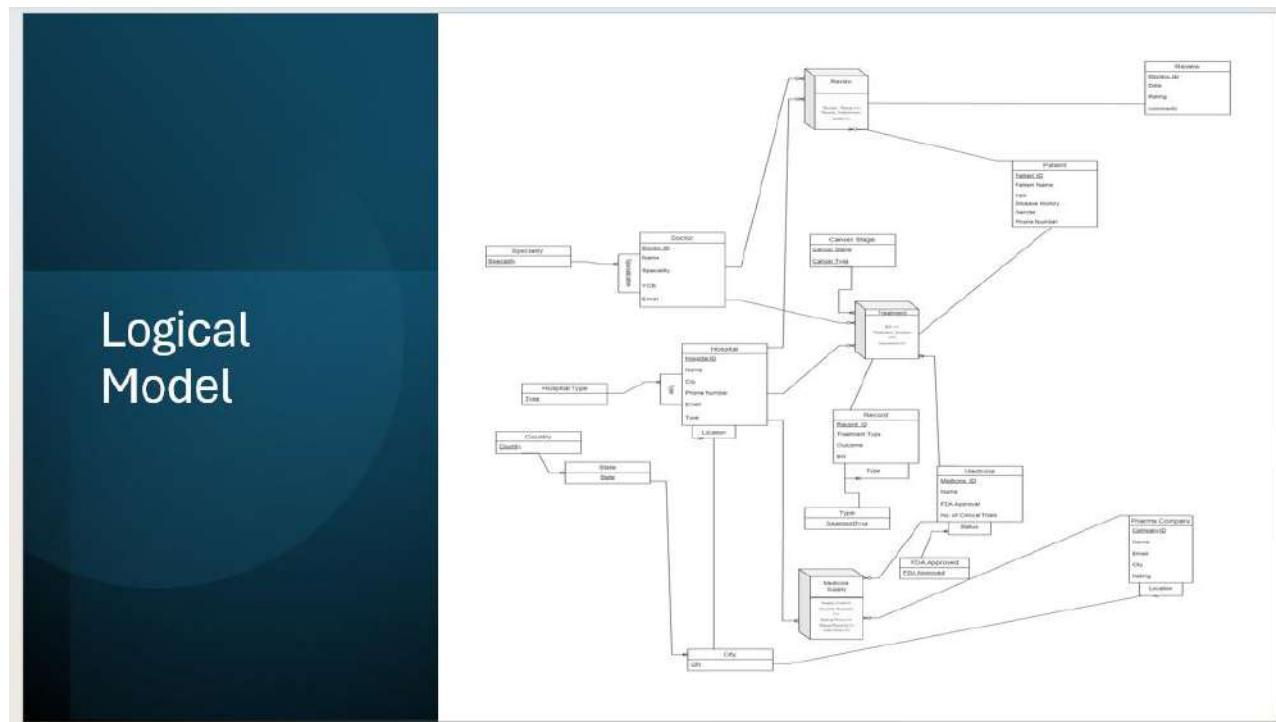
Problem statement

- Cancer patients and previvors often lack crucial, transparent information necessary for informed decision-making regarding treatments, complicating their care journey. There's a significant gap in accessible data on treatment effectiveness, hospital and physician performance, and medication efficacy. The CanCure project aims to fill this gap by developing a comprehensive database that provides detailed insights into cancer care dynamics, enabling better patient choices and improving treatment outcomes.

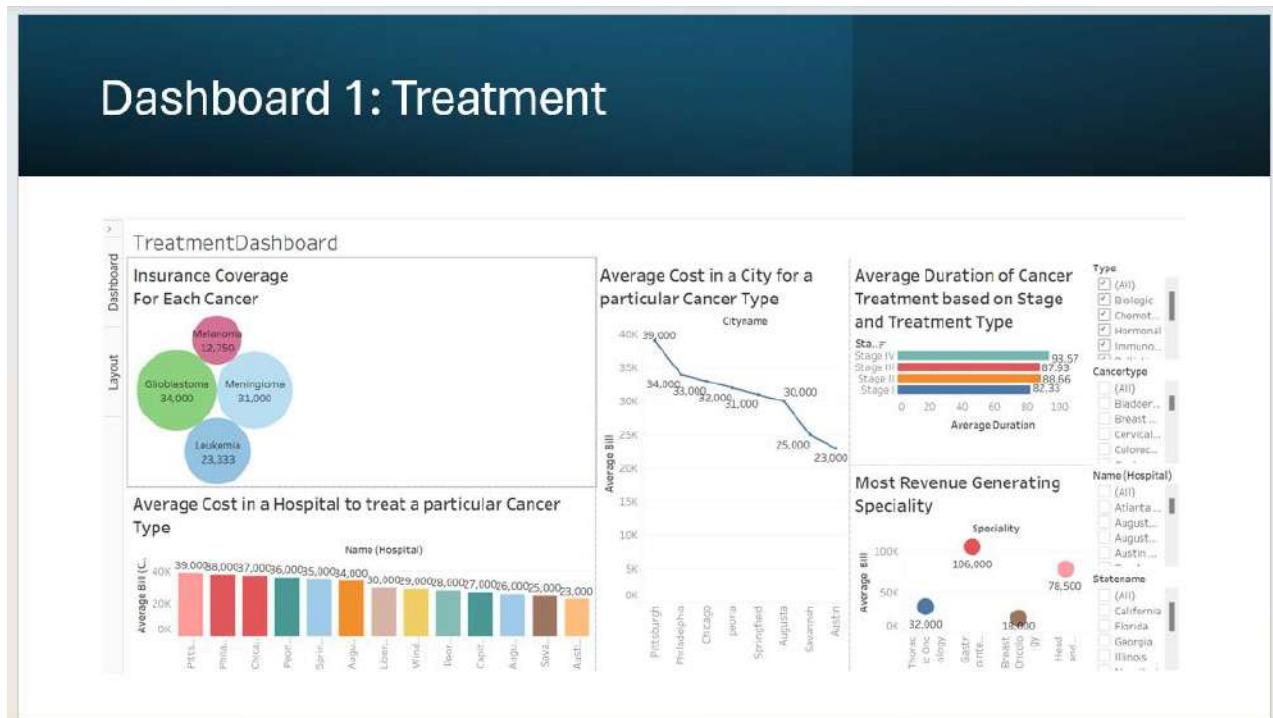
SLIDE 3:



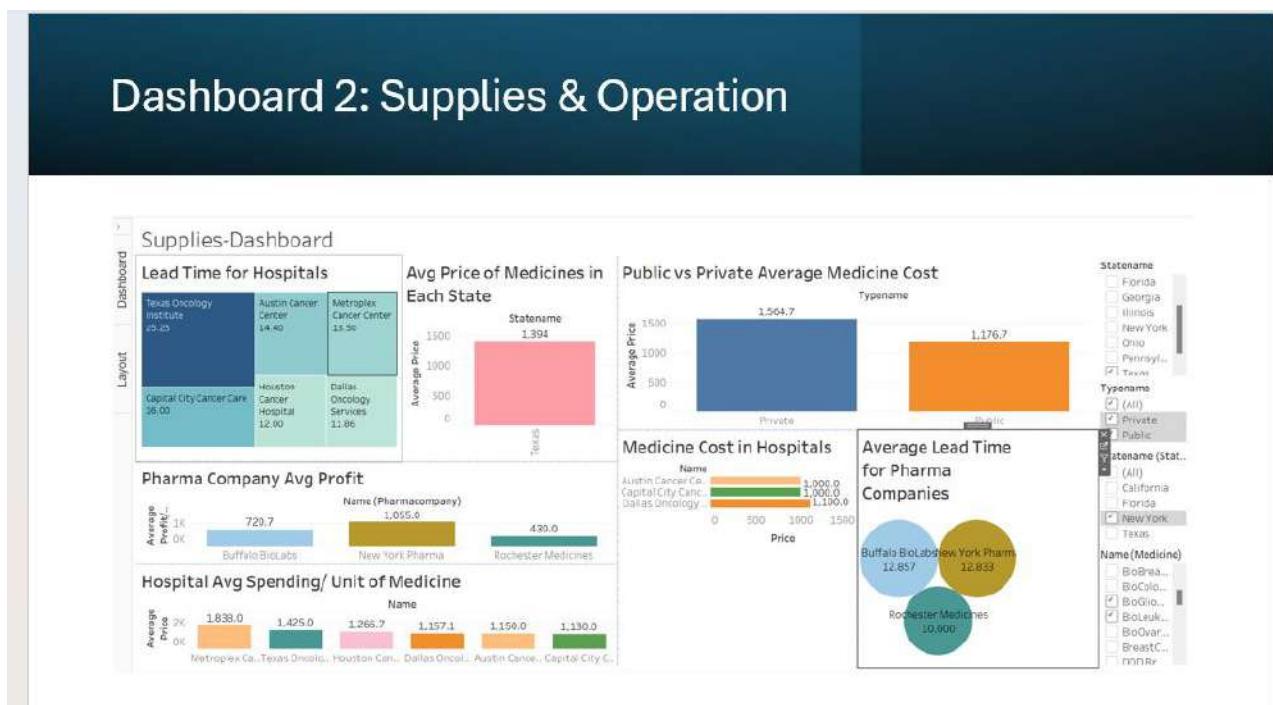
SLIDE 4:



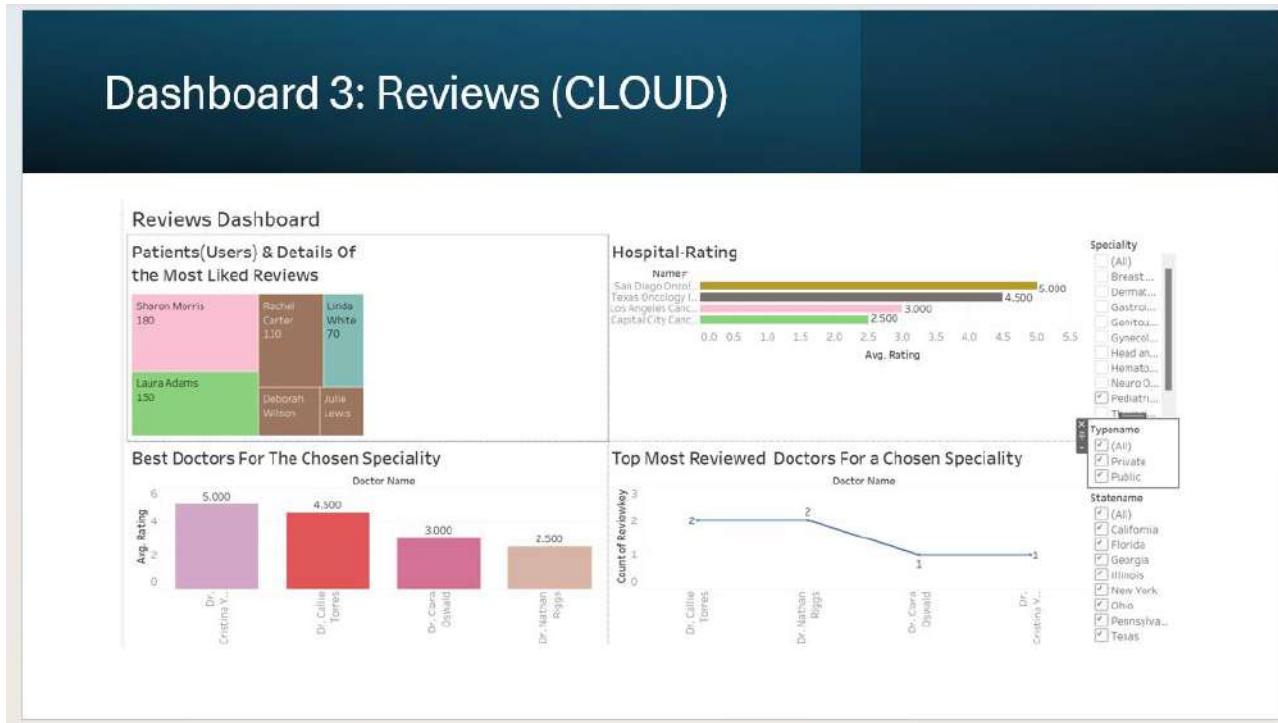
SLIDE 5:



SLIDE 6:



SLIDE 7:



SLIDE 8:

THANK YOU!