

Part 1 — Dimension Model + Approach

Dimensional Model

- **Dim_Patient** (Patient_Key, First_Name, Last_Name, Date_of_Birth)
Key = natural patient_id.
- **Dim_Doctor** (Doctor_Key, Doctor_Name, Specialty)
Key = natural doctor_id.
- **Dim_Date** (Date_Key, Full_Date, Year, Month, Day)
Key = YYYYMMDD.
- **Dim_Diagnosis** (Diagnosis_Key, Diagnosis_Code, Diagnosis_Description)
Key = natural DiagnosisCode (one row/code).
- **Fact_Visit** (Visit_Key, Patient_Key, Doctor_Key, Date_Key, Amount)
Keys are natural (visit_id → Visit_Key).
- **Bridge_Visit_Diagnosis** (Visit_Key, Diagnosis_Key, Diagnosis_Sequence, Is_Primary)
Resolves a many-to-many between visits and diagnoses; preserves order and primary flag.

Business requirements

1. “**For a given patient, on a given date, for a doctor, list all diagnoses**”
Join Fact_Visit → Bridge_Visit_Diagnosis → Dim_Diagnosis filtered by Patient/Date/Doctor.
The **bridge** allows multiple diagnoses per visit in sequence with a primary marker.
2. “**How many patients were diagnosed with a given diagnosis_code?**”
Count distinct Patient_Key by Diagnosis_Key using Bridge_Visit_Diagnosis → Fact_Visit.

Part 2 – Delta Live Tables Pipeline & ETL Implementation

Hospital Visit Analysis using Databricks DLT

Dataset Used

We used the **Synthea Synthetic Electronic Health Records (EHR)** dataset, which contains realistic patient, provider, and encounter data.

The following CSV files were uploaded into a Databricks **Volume**:

- patients.csv – basic patient demographics
- providers.csv – doctor information (name, specialty)
- encounters.csv – visit records with encounter type and cost
- conditions.csv – diagnosis codes and descriptions

All files were stored under
/Volumes/hospital_cg/visit-check/raw.

Overall Architecture

The project was implemented as a three-layer Delta Live Tables (DLT) pipeline plus a Gold SQL layer for analytics.

Layer	Purpose	Storage
Bronze	Ingest raw CSV data using Auto Loader and DLT streaming	/Volumes/hospital_cg/visit-check/raw
Silver	Create cleaned and standardized views with proper data types	SQL Views (silver_*_v)
Gold	Build analytical tables (Dimensions + Fact + Bridge)	SQL Tables (dim_*, fact_visit, bridge_visit_diagnosis)

3. Implementation Steps

Step 1 – Bronze Layer Creation

- Created a **DLT Python notebook** (DLT_Hospital_Pipeline) that uses dlt.table() to load all CSV files as streaming Delta tables.
- Each table (e.g., bronze_patients, bronze_providers, bronze_encounters, bronze_conditions) was ingested from the volume path using Auto Loader.

Step 2 – Silver Layer (Views)

- Created SQL views to clean and standardize the Bronze data.
- Converted IDs to string for consistent joins.
- Extracted key columns for each entity:
 - **Patients:** Id, First, Last, Birthdate
 - **Doctors:** Id, Name, Specialty
 - **Encounters:** Id, Patient, Provider, Start, Cost, Code, Description
 - **Visit-Diagnosis:** Linked each visit with diagnosis codes and generated sequence + primary flag.

Step 3 – Gold Layer (Dimensional Model)

Built six analytical tables using CREATE OR REPLACE TABLE:

1. **dim_date** – dates extracted from encounter and diagnosis timestamps
2. **dim_patient** – patient dimension using natural key patient_id
3. **dim_doctor** – doctor dimension using natural key doctor_id
4. **dim_diagnosis** – unique diagnosis codes with names and descriptions
5. **fact_visit** – main fact table recording each visit, doctor, patient, date, and amount
6. **bridge_visit_diagnosis** – many-to-many bridge between visits and diagnoses with diagnosis_sequence and is_primary

This follows a **Star Schema + Bridge Table** design.

Step 4 – Pipeline Job Setup

Created a Databricks Job with 2 tasks:

1. **Task 1 – Bronze Ingestion:** Runs the DLT notebook.
2. **Task 2 – Gold Build:** Runs the SQL notebook containing Gold table definitions.

Dependency: Task 2 depends on Task 1.

So, each time the Job runs,

- Bronze is updated.
- Silver views always query the current data.
- But Gold tables must be rebuilt periodically.

Conclusion

- This project demonstrated a complete ETL workflow in Databricks using Delta Live Tables.
- The Bronze layer handles continuous ingestion.
- Silver views standardize and clean data.
- Gold tables implement a star schema with a bridge for multi-diagnosis visits.
- The pipeline is fully automated and supports incremental updates whenever new records are inserted into the Bronze tables.