Data Model : Healthcare CCA Clinical Care Advance

Source 1 — Member & Eligibility

**Organize data** about patients, providers, insurance, claims, and quality measures into structured tables.

**Simulate a real-world healthcare system** (Medicare/Medicaid) with multiple connected data sources.

Designed a **healthcare data model** in SQL Server with 5 major sources, each containing 10 related tables (50 tables total).

* **Source 1 (Member & Eligibility):** stores patient details, insurance plans, and enrollment history.
* **Source 2 (Claims & Encounters):** records medical claims, billed services, diagnoses, procedures, and payments.
* **Source 3 (Care Management):** tracks care programs, goals, tasks, interventions, and quality gaps.
* **Source 4 (Provider Network):** holds provider, organization, specialty, contract, and network status data.
* **Source 5 (Reference Data):** contains standard medical code sets (ICD, CPT, NDC, POS, LOINC, measures, benefits).

**Source 1 — Member & Eligibility**

**Goal:** Store information about members (patients) and their insurance coverage (Medicare, Medicaid, etc.), so we can later connect them to claims, providers, and care programs.

**1) s1\_elig.Member → *Who the person is***

This is the **main patient table**. Every other table in this source links back to it.

**Columns**:

* **MemberID** → Primary Key (system-generated, unique ID for each member).
* **MemberBK** → Business Key (real-world ID like MBI = Medicare Beneficiary Identifier or Medicaid ID).
* **FirstName, LastName** → Member’s name.
* **DOB** → Date of birth.
* **Gender** → ‘M’, ‘F’, or other.
* **SSN** → Social Security Number (can be NULL in dummy data).

This identifies the person. All other tables (address, contact, enrollment, claims) need MemberID to tie back here.

**2) s1\_elig.MemberAddress → *Where the person lives***

Stores **addresses for members**. One member may have multiple addresses (home, mailing, previous).

**Columns**:

* **AddressID** → Primary Key (unique row for each address).
* **MemberID** → FK → s1\_elig.Member. Ties the address to the person.
* **AddressLine1/Line2** → Street address (line 1 required, line 2 optional).
* **City, State, ZipCode** → Standard address fields.

Useful for region-based analysis (e.g., members in Albany County).

**3) s1\_elig.MemberContact → *How to reach the person***

Stores phone and email.

**Columns**:

* **ContactID** → Primary Key.
* **MemberID** → FK → Member.
* **Phone, Email** → Contact details.

Needed for care management, outreach, communication.

**4) s1\_elig.Payer → *Who pays for healthcare***

This is the **insurance company / government agency**.

**Columns**:

* **PayerID** → Primary Key.
* **PayerName** → e.g., “Centers for Medicare & Medicaid Services” or “NY State Medicaid”.
* **PayerType** → Medicare, Medicaid, Commercial.

Members are enrolled into a **plan**, and every plan belongs to a **payer**.

**5) s1\_elig.InsurancePlan → *The specific insurance plan***

Each member enrolls in a plan offered by a payer.

**Columns**:

* **PlanID** → Primary Key.
* **PayerID** → FK → s1\_elig.Payer.
* **PlanName** → e.g., “Medicare Advantage HMO”.
* **PlanType** → Medicare, Medicaid, Dual.
* **CoverageLevel** → Gold/Silver/Standard (optional; for commercial plans).

Every claim must tie to a **plan**, so we know who pays it and what benefits apply.

**6) s1\_elig.Enrollment → *Monthly enrollment snapshots***

Tells us whether a member was enrolled in a plan during a specific month.

**Columns**:

* **EnrollmentID** → Primary Key.
* **MemberID** → FK → Member.
* **PlanID** → FK → InsurancePlan.
* **EnrollmentMonth** → Use 1st of the month (e.g., 2025-01-01 = Jan 2025).
* **IsActive** → 1 = Active, 0 = Inactive.

Helps us answer questions like “Was Alice active in Medicaid in Feb 2025?”

**7) s1\_elig.CoveragePeriod → *Continuous coverage spans***

Instead of month-by-month rows, this gives **start/end dates** of coverage.

**Columns**:

* **CoverageID** → Primary Key.
* **MemberID** → FK → Member.
* **PlanID** → FK → InsurancePlan.
* **StartDate, EndDate** → Continuous period covered. EndDate = NULL if still covered.

Easier for reporting long-term coverage (e.g., “Alice had continuous coverage from Jan–Jul 2025”).

**8) s1\_elig.DualStatus → *Medicare + Medicaid flag***

Indicates whether a person is “dual-eligible” (gets both Medicare & Medicaid).

**Columns**:

* **DualID** → Primary Key.
* **MemberID** → FK → Member.
* **StatusCode** → e.g., QMB (Qualified Medicare Beneficiary), SLMB (Specified Low-Income Medicare Beneficiary).
* **EffectiveDate** → When status began.

Dual eligibles often have special coverage and payment rules

**9) s1\_elig.SubsidyLIS → *Low-Income Subsidy status***

Tracks whether the member qualifies for **Medicare Part D drug subsidies**.

**Columns**:

* **LISID** → Primary Key.
* **MemberID** → FK → Member.
* **SubsidyLevel** → Full, Partial, None.
* **EffectiveDate** → Start date of subsidy.

Determines out-of-pocket drug costs.

**10) s1\_elig.MemberLanguagePref → *Preferred communication language***

Stores member’s language preference.

**Columns**:

* **PrefID** → Primary Key.
* **MemberID** → FK → Member.
* **Language** → English, Spanish, etc.
* **EffectiveDate** → When this preference started.

Needed for culturally appropriate care and member outreach.

**Big Picture (Source 1)**

* **Member** = central person table.
* **Payer & Plan** = insurance side.
* **Enrollment/Coverage** = when/where member is covered.
* **DualStatus & LIS** = extra flags for government programs.
* **Address/Contact/Language** = communication details.

Everything else in healthcare (claims, providers, care gaps) links back here.

**Source 2 — Claims & Encounters**

**Goal:** Record every healthcare service a member receives, whether it’s billed as a **claim** (financial record) or logged as a **clinical encounter** (medical event).

**Every time a patient gets healthcare (like a doctor visit, lab test, hospital stay, or medicine), it should be recorded.**

* If the provider sends a **bill to insurance**, it’s saved as a **claim** (money/financial record).
* If the service is just **noted in the medical system** (like a nurse’s visit, a vaccination, or a health check that isn’t billed), it’s saved as a **clinical encounter** (medical record).

So, whether it’s for **money (claims)** or for **medical history (encounters)**, both get recorded so the member’s full healthcare picture is tracked.

**1) s2\_claims.PlaceOfService → *Where the service happened***

This is a small reference table.

**Columns:**

* **POSCode** → Primary Key. E.g., 11=Office, 21=Inpatient, 23=ER.
* **POSDescription** → Text description (“Emergency Room – Hospital”).

Used by claims/encounters to show *location of care*.

**2) s2\_claims.ClaimHeader → *Claim “envelope”***

One row per claim (like an invoice).

**Columns:**

* **ClaimID** → Primary Key. Surrogate ID.
* **ClaimNumber** → Real-world claim number from payer.
* **MemberID** → FK → Member (Source 1).
* **PlanID** → FK → InsurancePlan (Source 1).
* **ClaimDate** → Date claim was filed.
* **TotalBilledAmount** → Sum of all line charges.
* **Status** → e.g., Submitted, Paid, Denied.
* **RenderingProviderNPI** → Doctor/provider NPI (later links to Source 4).
* **POSCode** → FK → PlaceOfService.

Holds the **summary info** for a claim.

**3) s2\_claims.ClaimLine → *Billed services***

Each claim has multiple lines (each service/procedure billed).

**Columns:**

* **ClaimID** → FK → ClaimHeader.
* **LineNo** → Line sequence within claim.
* **ServiceDate** → When service occurred.
* **POSCode** → FK → PlaceOfService.
* **Units** → Number of services (e.g., 2 lab tests).
* **RevenueCode** → Used in hospital billing.
* **BilledAmount** → Amount charged for this line.
* **RenderingProviderNPI** → Provider who performed the service.
* **PK:** (ClaimID, LineNo).

The “detailed bill” of the claim.

**4) s2\_claims.ClaimDiagnosis → *Diagnosis codes for a claim***

**Columns:**

* **ClaimID** → FK → ClaimHeader.
* **DxSeq** → Sequence (1=primary diagnosis, 2=secondary, etc.).
* **ICD10Code** → FK → Ref\_ICD10 (Source 5).

Lists *why the patient was treated*.

**5) s2\_claims.ClaimProcedure → *Procedures done on a claim line***

**Columns:**

* **ClaimID, LineNo** → FK → ClaimLine.
* **ProcSeq** → Procedure order (1=primary).
* **CPTCode** → FK → Ref\_CPT\_HCPCS (Source 5).

Lists *what was done* (doctor visits, x-rays, labs).

**6) s2\_claims.ClaimPayment → *How much was paid***

**Columns:**

* **PaymentID** → Primary Key.
* **ClaimID, LineNo** → FK → ClaimLine.
* **AllowedAmount** → Agreed price (per contract).
* **PaidAmount** → What payer actually paid.
* **PatientLiability** → Member’s share (copay, deductible).

Answers: “How much was charged vs paid?”

**7) s2\_claims.ClaimAdjustment → *Why payments differ***

**Columns:**

* **AdjustmentID** → Primary Key.
* **ClaimID, LineNo** → FK → ClaimLine.
* **GroupCode** → e.g., CO=Contractual Obligation, PR=Patient Responsibility.
* **ReasonCode** → e.g., 45=Charge exceeds fee schedule.
* **Amount** → Adjustment amount.

Explains *reductions/denials*.

**8) s2\_claims.Encounter → *Clinical event (not just billing)***

Captures actual medical visits, even if not billed as a claim.

**Columns:**

* **EncounterID** → Primary Key.
* **MemberID** → FK → Member.
* **PlanID** → FK → Plan.
* **ProviderNPI** → Doctor seen.
* **FacilityPOSCode** → FK → PlaceOfService.
* **EncounterDate** → Date of service.
* **AdmitDate, DischargeDate** → For inpatient stays.

Purely clinical — used for care quality reporting.

**9) s2\_claims.EncounterDiagnosis → *Encounter diagnoses***

**Columns:**

* **EncounterID** → FK → Encounter.
* **DxSeq** → Diagnosis sequence.
* **ICD10Code** → FK → Ref\_ICD10.

Lists *reasons for the encounter*.

**10) s2\_claims.EncounterProcedure → *Procedures during encounter***

**Columns:**

* **EncounterID** → FK → Encounter.
* **ProcSeq** → Procedure order.
* **CPTCode** → FK → Ref\_CPT\_HCPCS.

Records *what was done* clinically.

**Big Picture (Source 2)**

* **ClaimHeader + ClaimLine** = Billing record.
* **ClaimDiagnosis/Procedure** = Medical codes for why + what.
* **ClaimPayment/Adjustment** = Financial outcome.
* **Encounter + sub-tables** = Clinical record (not just money).
* **POS, NPI, ICD, CPT** = Codes to connect with reference data.

This source links **patients (Source 1)** to **providers (Source 4)**, and uses **codes (Source 5)** to classify services/diagnoses.

**Source 3: Care Management (CCA)**   
This source tracks programs (e.g., Diabetes), members’ care plans, goals, tasks, assessments, risk scores, care gaps, and interventions (touchpoints).

**Care gaps** = the **missing or overdue healthcare services** that a patient should have received, based on clinical guidelines.

In CCA or similar systems, **care gaps tables** track which members are missing recommended tests/checks.

Interventions = steps taken to fix health issues or prevent problems.

**1) s3\_cca.QualityMeasure — *catalog of quality measures***

**Purpose:** Master list of measures you track (HEDIS/Stars/CMS).  
**Columns:**

* **MeasureID** *(PK)* – Short code you use everywhere (e.g., QM001).
* **MeasureName** – Human title (“Diabetes: A1c control <8%”).
* **Owner** – Owner/program (e.g., HEDIS, Stars, CMS).  
  **Connects to:** CareGap.MeasureID.  
  **Example:** QM001 = Diabetes A1c control.

**2) s3\_cca.CCA\_User — *people who manage care***

**Purpose:** Staff directory of care managers, RNs, social workers.  
**Columns:**

* **UserID** *(PK)* – Unique staff user.
* **UserName** – “Renee RN”, “Carl CM”.
* **Role** – RN, Care Manager, SW, etc.  
  **Connects to:** CarePlan.AssignedUserID, CareTask.AssignedUserID, Intervention.UserID.  
  **Example:** UserID 1 = Renee RN.

**3) s3\_cca.CareProgram — *clinical programs***

**Purpose:** Programs members can join (Diabetes, CHF, CKD…).  
**Columns:**

* **ProgramID** *(PK)* – Unique program.
* **ProgramName** – “Diabetes”, “CHF”, “CKD”.
* **Description** – Short description of the program.  
  **Connects to:** CarePlan.ProgramID, Assessment.ProgramID.  
  **Example:** ProgramID 1 = Diabetes.

**4) s3\_cca.CarePlan — *member’s enrollment in a program***

**Purpose:** One row for a member participating in a specific program.  
**Columns:**

* **CarePlanID** *(PK)* – Unique plan row.
* **MemberID** *(FK → s1\_elig.Member)* – Which patient this plan belongs to.
* **ProgramID** *(FK → CareProgram)* – Which program (Diabetes/CHF/CKD).
* **PlanID** *(FK → s1\_elig.InsurancePlan, nullable)* – Optional insurance plan context.
* **StartDate / EndDate** – Dates the plan is active (EndDate NULL = still active).
* **Status** – Active / Closed.
* **AssignedUserID** *(FK → CCA\_User, nullable)* – Care manager/owner.  
  **Connects to:** CareGoal, CareTask, Intervention.  
  **Example:** Alice joins Diabetes program on 2025-01-10, assigned to Renee.

**5) s3\_cca.CareGoal — *targets inside a care plan***

**Purpose:** Specific clinical/behavioral goals under a care plan.  
**Columns:**

* **GoalID** *(PK)* – Unique goal.
* **CarePlanID** *(FK → CarePlan)* – Which plan it belongs to.
* **GoalText** – The goal (“Reduce A1c below 8%”).
* **TargetDate** – When you hope to achieve it.
* **Status** – Open / Met / NotMet.  
  **Connects to:** CareTask.GoalID (optional linkage from tasks to goals).  
  **Example:** Goal: “Schedule retinal eye exam” by 2025-03-31.

**6) s3\_cca.CareTask — *action items to achieve goals***

**Purpose:** Concrete tasks assigned to a user, tied to a plan (and sometimes a goal).  
**Columns:**

* **TaskID** *(PK)* – Unique task.
* **CarePlanID** *(FK → CarePlan)* – Which plan this task supports.
* **GoalID** *(FK → CareGoal, nullable)* – Optional: which goal it helps.
* **TaskText** – What to do (“Order A1c test”, “Diet education call”).
* **DueDate** – When to complete it.
* **Status** – Open / Done / Skipped.
* **AssignedUserID** *(FK → CCA\_User, nullable)* – Who owns it.  
  **Connects to:** Intervention.TaskID (what actually happened for this task).  
  **Example:** Task “Book eye exam” assigned to Ivy CM, due 2025-02-15.

**7) s3\_cca.Assessment — *surveys & screening results***

**Purpose:** Store standardized assessments (HRA, PHQ-9, Fall Risk) with scores.  
**Columns:**

* **AssessmentID** *(PK)* – Unique assessment row.
* **MemberID** *(FK → s1\_elig.Member)* – Patient assessed.
* **ProgramID** *(FK → CareProgram, nullable)* – If done within a program.
* **AssessmentDate** – When done.
* **AssessmentType** – HRA, PHQ9, FallRisk, etc.
* **Score** – Numeric score (scale depends on type).
* **Notes** – Optional comments (“baseline”, “improved diet”).  
  **Usage:** Track progress/risk changes over time.  
  **Example:** PHQ-9 score 6 = mild.

**8) s3\_cca.RiskScore — *risk stratification***

**Purpose:** Member’s risk score outputs from models (HCC, Readmission).  
**Columns:**

* **RiskID** *(PK)* – Unique row.
* **MemberID** *(FK → s1\_elig.Member)* – Patient scored.
* **RiskModel** – Model name (“HCC”, “Readmit”, “Internal”).
* **Score** – Numeric (e.g., 1.45 HCC).
* **RiskTier** – Low / Medium / High (derived from Score).
* **EffectiveDate** – Date the score is effective.  
  **Usage:** Prioritize outreach (e.g., High risk gets more touches).  
  **Example:** Maya HCC 1.80 → High.

**9) s3\_cca.CareGap — *quality gap status by period***

**Purpose:** For each member and measure, track whether a gap is **OPEN** or **CLOSED** for a defined time window.  
**Columns:**

* **GapID** *(PK)* – Unique gap row.
* **MemberID** *(FK → s1\_elig.Member)* – Patient.
* **MeasureID** *(FK → QualityMeasure)* – Which measure (QM001…).
* **PeriodStart / PeriodEnd** – Measurement window (e.g., CY 2025).
* **Status** – OPEN or CLOSED.
* **ClosedDate** – When it was closed (NULL if still open).  
  **Usage:** Measure compliance and program impact.  
  **Example:** Alice’s QM001 closed on 2025-02-16 after interventions.

**10) s3\_cca.Intervention — *what actually happened (touchpoints)***

**Purpose:** Log real actions taken by staff: calls, education, home visits.  
**Columns:**

* **InterventionID** *(PK)* – Unique touchpoint.
* **CarePlanID** *(FK → CarePlan)* – Which plan it supports.
* **TaskID** *(FK → CareTask, nullable)* – If tied to a specific task.
* **UserID** *(FK → CCA\_User, nullable)* – Who performed it.
* **InterventionDate** – When it happened.
* **InterventionType** – Call, Education, HomeVisit, Support, etc.
* **MinutesSpent** – Time spent.
* **Notes** – Free text (“Lab results discussed”).  
  **Usage:** Proves work was done; analyze workload & outcomes.  
  **Example:** Education session 25 min, “Class enrollment coaching”.

**How Source 3 connects (simple map)**

* **Member (S1)** ↔ CarePlan, Assessment, RiskScore, CareGap (all use MemberID).
* **Program** ↔ CarePlan, Assessment (context of the plan/assessment).
* **Users** ↔ CarePlan (assigned owner), CareTask (assigned), Intervention (performed).
* **Goals & Tasks** hang under a **CarePlan**; **Interventions** show work done.
* **QualityMeasure** ↔ CareGap for measure status (OPEN/CLOSED).

This creates a complete story: **member enrolls in program → goals & tasks are created → staff performs interventions → assessments & risk track progress → care gaps open/close over time.**

**Source 4: Provider Network**   
This source answers **who the doctors/providers are, where they practice, their specialties, and their contracts with payers/plans.**

**Source 4 — Provider Network**

**Goal:** Store all information about providers, their organizations, specialties, locations, and contracts.

**1) s4\_provider.Organization — *health systems or medical groups***

**Purpose:** Represents hospitals, health systems, or medical groups.

**Columns:**

* **OrgID** *(PK)* → Unique organization.
* **OrgName** → Name (“Capital Health System”).
* **TaxID** → Federal tax ID (for billing).

Connects to **Location** (where this org has sites), **Contract** (payer ↔ org contracts), **Affiliation** (providers belonging to this org).

**2) s4\_provider.Location — *physical sites***

**Purpose:** Actual clinic/hospital locations where care is delivered.

**Columns:**

* **LocationID** *(PK)* → Unique site.
* **OrgID** *(FK → Organization)* → Which org owns this location.
* **LocationName** → Friendly name (“Albany Clinic”).
* **AddressLine1/2, City, State, Zip** → Physical address.

A provider can work at multiple **locations**.

**3) s4\_provider.Provider — *individual doctors/clinicians***

**Purpose:** Core provider list.

**Columns:**

* **ProviderID** *(PK)* → Surrogate ID.
* **NPI** → National Provider Identifier (real-world ID).
* **ProviderName** → “Dr. Allan Carter”.
* **ProviderType** → MD, DO, NP, PA, etc.
* **IsActive** → 1 = Active, 0 = Not practicing.

Connects to **ClaimHeader / ClaimLine** (via NPI in Source 2), **ProviderSpecialty**, **ProviderLocation**, **Contract**, **Affiliation**.

**4) s4\_provider.ProviderLocation — *where providers practice***

**Purpose:** Connects providers to locations (many-to-many).

**Columns:**

* **ProviderID** *(FK → Provider)*.
* **LocationID** *(FK → Location)*.
* **StartDate / EndDate** → Time they practiced there.
* **PK** = (ProviderID,LocationID,StartDate).

Example: Dr. Carter practices at Albany Clinic since 2024-01-01.

**5) s4\_provider.Specialty — *types of specialties***

**Purpose:** Master list of specialties.

**Columns:**

* **SpecialtyCode** *(PK)* → Short code (IM, CAR, RAD).
* **SpecialtyName** → “Internal Medicine”, “Cardiology”.

Used to classify providers.

**6) s4\_provider.ProviderSpecialty — *provider ↔ specialty mapping***

**Purpose:** Many-to-many link between providers and specialties.

**Columns:**

* **ProviderID** *(FK → Provider)*.
* **SpecialtyCode** *(FK → Specialty)*.
* **PK** = (ProviderID,SpecialtyCode).

Example: Dr. Carter = Internal Medicine + Family Medicine.

**7) s4\_provider.Contract — *payer/plan ↔ org/provider contracts***

**Purpose:** Represents agreements for reimbursement.

**Columns:**

* **ContractID** *(PK)* → Unique contract.
* **PayerID** *(FK → s1\_elig.Payer)* → Who pays.
* **PlanID** *(FK → s1\_elig.InsurancePlan)* → Specific plan (nullable if payer-level).
* **ContractName** → e.g., “Capital Health – Medicare Advantage HMO”.
* **EffectiveDate / EndDate** → Valid period.
* **ContractScope** → ORG or PROVIDER (what this contract covers).
* **OrgID** *(nullable)* → If org-level contract.
* **ProviderID** *(nullable)* → If provider-specific contract.

Links insurance plans to providers/orgs.

**8) s4\_provider.ContractRate — *allowed $ for CPTs under a contract***

**Purpose:** Defines the negotiated rates for services.

**Columns:**

* **ContractID** *(FK → Contract)*.
* **CPTCode** *(FK → Ref\_CPT\_HCPCS in Source 5)*.
* **EffectiveDate** → When the rate applies.
* **AllowedAmount** → $ payer allows for that code.
* **PK** = (ContractID,CPTCode,EffectiveDate).

Example: Under Contract #1, CPT 99213 = $100.

**9) s4\_provider.NetworkStatus — *in-network/out-of-network status***

**Purpose:** Tells whether a provider is in-network for a plan in a given month.

**Columns:**

* **ProviderID** *(FK → Provider)*.
* **PlanID** *(FK → InsurancePlan)*.
* **Yyyymm** → Month (e.g., ‘202501’).
* **Status** → ‘IN’ or ‘OUT’.
* **PK** = (ProviderID,PlanID,Yyyymm).

Example: Dr. Carter in-network for Medicare HMO Jan–Mar 2025.

**10) s4\_provider.Affiliation — *providers linked to orgs***

**Purpose:** Connects a provider to an organization they are affiliated with.

**Columns:**

* **ProviderID** *(FK → Provider)*.
* **OrgID** *(FK → Organization)*.
* **StartDate / EndDate** → Affiliation period.
* **PK** = (ProviderID,OrgID,StartDate).

Example: Dr. Kim affiliated with Finger Lakes Hospital since 2025-01-01.

**Big Picture (Source 4)**

* **Organization** → health system / hospital.
* **Location** → physical sites under org.
* **Provider** → individual doctors.
* **Specialty** → what they practice.
* **ProviderLocation & ProviderSpecialty** → many-to-many mappings.
* **Contract & ContractRate** → legal/financial agreements with payers/plans.
* **NetworkStatus** → are they in/out-network this month?
* **Affiliation** → high-level link to orgs.

Together, Source 4 answers:

* “Who is this provider (NPI, specialty)?”
* “Where do they practice?”
* “Are they in network for this plan?”
* “What are their contracted rates?”

**Source 5: Reference Data**.  
This is the **“shared library”** of standard codes and definitions that other sources rely on.

**Source 5 — Reference Data**

**Goal:** Provide master lists of codes and categories used across the model (diagnosis, procedure, labs, drugs, benefits, regions, etc.).

**1) s5\_ref.Ref\_ICD10 — *diagnosis codes***

**Purpose:** Official list of ICD-10-CM diagnosis codes.

**Columns:**

* **ICD10Code** *(PK)* → “E11.9” (Diabetes without complication).
* **ShortDesc** → Short description.
* **Chapter** → ICD-10 chapter (“Endocrine”).
* **Category** → Subcategory (“Diabetes mellitus”).

Used in ClaimDiagnosis, EncounterDiagnosis.

**2) s5\_ref.Ref\_CPT\_HCPCS — *procedure codes***

**Purpose:** CPT/HCPCS procedure codes for services/tests.

**Columns:**

* **CPTCode** *(PK)* → “99213” (Office visit).
* **ShortDesc** → Short description.
* **Category** → E/M, Lab, Radiology, etc.

Used in ClaimProcedure, EncounterProcedure, ContractRate.

**3) s5\_ref.Ref\_POS — *place of service codes***

**Purpose:** Master reference for service locations.

**Columns:**

* **POSCode** *(PK)* → “11”, “21”.
* **POSDescription** → Office, Inpatient Hospital, ER.

Used in ClaimLine, Encounter, and also mirrored in s2\_claims.PlaceOfService.

**4) s5\_ref.Ref\_NDC — *drug codes***

**Purpose:** NDC (National Drug Code) master.

**Columns:**

* **NDC** *(PK)* → 11-digit string (e.g., 00093015001).
* **Generic** → Generic drug name.
* **Brand** → Brand name.
* **Strength** → Dosage strength (500 mg).
* **DosageForm** → Tablet, Capsule, Solution.
* **Route** → Oral, Subcutaneous, etc.

Could be tied to pharmacy claims if modeled.

**5) s5\_ref.Ref\_LOINC — *lab test codes***

**Purpose:** LOINC master for lab/clinical tests.

**Columns:**

* **LOINCCode** *(PK)* → e.g., “4548-4” (HbA1c).
* **Component** → What’s being measured.
* **Property** → Type of measure (MCnc, Qn).
* **Scale** → Qn (Quantitative), Ord (Ordinal).
* **Method** → Method used (optional).
* **CommonName** → Friendly name (“HbA1c”).

Useful for quality measures that depend on lab results.

**6) s5\_ref.Ref\_QualityMeasure — *quality measure definitions***

**Purpose:** Catalog of quality metrics (HEDIS, CMS Stars).

**Columns:**

* **MeasureID** *(PK)* → “QM001”.
* **MeasureName** → “Diabetes: A1c control <8%”.
* **Owner** → HEDIS, CMS, Stars.

Used in CareGap (Source 3).

**7) s5\_ref.Ref\_Benefit — *benefit categories***

**Purpose:** Master list of covered benefits.

**Columns:**

* **BenefitID** *(PK)*.
* **BenefitName** → “Inpatient Hospital”, “Pharmacy”.
* **BenefitType** → IP, OP, Rx, Dental, Vision.

Used to describe coverage for plans.

**8) s5\_ref.Ref\_PlanBenefit — *plan ↔ benefit mapping***

**Purpose:** Defines copays/coins/limits for a plan’s benefits.

**Columns:**

* **PlanID** *(FK → InsurancePlan)*.
* **BenefitID** *(FK → Ref\_Benefit)*.
* **Yyyymm** → Month (“202501”).
* **Copay** → Flat dollar (e.g., $20 office visit).
* **CoinsurancePct** → % of allowed ($).
* **LimitAmount** → Benefit caps ($500 annual vision).
* **PK** = (PlanID,BenefitID,Yyyymm).

Tells you “what coverage does this plan provide this month?”

**9) s5\_ref.Ref\_Geo\_ZipFips — *geographic mapping***

**Purpose:** Connects ZIP codes to counties, states, and regions.

**Columns:**

* **Zip** *(PK)* → ZIP Code.
* **CountyFIPS** → Federal county code.
* **CountyName** → “Albany County”.
* **State** → Two-letter (NY).
* **Region** → Region grouping (“Capital”).

Useful for geographic rollups and outreach targeting.

**10) s5\_ref.Ref\_AdjustmentCode — *CARC/RARC codes***

**Purpose:** Official adjustment/denial codes used in claims.

**Columns:**

* **GroupCode** → CO (Contractual), PR (Patient Responsibility), OA (Other Adjustment), R (Remark).
* **ReasonCode** → Specific reason (e.g., “45”).
* **Description** → Meaning of the code.
* **IsRARC** → 1 if it’s a remark code.
* **PK** = (GroupCode,ReasonCode).

Used in ClaimAdjustment (Source 2).

**Big Picture (Source 5)**

* **ICD, CPT, POS** = standard medical code sets.
* **NDC, LOINC** = drugs and labs.
* **QualityMeasure** = metrics used in care gaps.
* **Benefit & PlanBenefit** = insurance coverage rules.
* **Geo\_ZipFips** = geographic mapping.
* **AdjustmentCode** = reasons why payments differ.

This **reference library** ensures consistency across claims, encounters, providers, care management, and plans.

**Data Warehouse Layer**

After building the 5 source systems and populating them with dummy data, we designed a **Data Warehouse (DW) layer** on top of them. This DW is modeled in two approaches: **Star Schema** and **Snowflake Schema**.

The purpose of this DW layer is to:

* Provide **clean, consistent dimensions** for reporting and analysis.
* Store **facts (measures + keys)** at different grains (claims, encounters, eligibility, care gaps).
* Enable **business intelligence queries** (costs, utilization, care gap closure) without joining dozens of source tables.

**⭐ Star Schema (Denormalized)**

**Steps**

1. **Create DW schema (dw)**
   * Logical separation from source schemas.
   * All star schema objects live here.
2. **Define core Dimension tables**
   * DimDate (calendar, controlled DateSK = yyyymmdd).
   * DimMember (patient demographics).
   * DimPlan (plan + payer info in one table).
   * DimProvider (NPI, name, type).
   * DimDiagnosis (ICD10 codes).
   * DimProcedure (CPT/HCPCS codes).
   * DimPOS (place of service).
   * DimQualityMeasure (quality measure catalog).

These are **denormalized** — for example, Plan includes Payer details directly.

1. **Add “Unknown” rows**
   * Surrogate Key = 0.
   * Natural Key = -1 or "UNK".
   * Prevents fact load failures when data is missing or late.
2. **Populate Dimensions**
   * Load unique records from sources (e.g., Member from Source 1, Providers from Source 4, Codes from Source 5).
   * Use NOT EXISTS logic to avoid duplicates.
3. **Define Fact tables**
   * FactClaimsLine (grain = one billed line).
   * FactEncounters (grain = one encounter).
   * FactCareGaps (grain = member–measure–period).
   * FactEligibilityMonthly (grain = member–plan–month).

Facts store **surrogate keys (SKs)** + **measures**.

1. **Load Fact tables**
   * Join sources → dims to fetch SKs.
   * Use COALESCE to map missing lookups to Unknown SK.
2. **Validate**
   * Check row counts.
   * Confirm facts point to valid SKs.
   * Ensure no orphaned records.
3. **Index for performance**
   * Index on SKs commonly used in joins.
   * Speeds up BI queries.

**❄️ Snowflake Schema (Normalized)**

**Steps**

1. **Create DW schema (dw\_snow)**
   * Separate from both dw and source schemas.
   * Holds normalized version of dimensions.
2. **Define normalized Dimensions**
   * DimPlan → references DimPayer (separate Payer table).
   * DimProvider → linked via DimOrg and DimLocation.
   * DimSpecialty → linked via BridgeProviderSpecialty.
   * DimMember, DimDiagnosis, DimProcedure, DimPOS, DimDate, DimQualityMeasure remain similar.

More normalized, avoids storing redundant data in each dimension.

1. **Add “Unknown” rows**
   * Same logic as star schema (SK=0 safety net).
   * Each normalized table gets its own Unknown row.
2. **Populate Dimensions**
   * Load data in **hierarchical order**:
     + Payer → Plan.
     + Org → Location.
     + Provider → BridgeProviderSpecialty.
   * Reference tables (codes, measures) load as usual.
3. **Define Fact tables**
   * Same 4 fact tables as in star schema:
     + Claims Line, Encounters, Care Gaps, Eligibility.
   * Facts still reference the **lowest-level dims** (e.g., Plan, Provider, POS).
4. **Load Fact tables**
   * Join sources to lowest-level dims.
   * Snowflake rollups are done at query time (e.g., Plan → Payer).
5. **Validate**
   * Row counts, missing SKs.
   * Test joins across normalized dims.
6. **Index for performance**
   * Index on SKs.
   * Additional indexes on bridge tables for Provider–Specialty queries.

**Star vs. Snowflake (Summary)**

| **Aspect** | **Star Schema (dw)** | | **Snowflake Schema (dw\_snow)** |
| --- | --- | --- | --- |
| Dimension design | Denormalized (Plan has Payer info inside) | | Normalized (Plan points to Payer table) |
| Query performance | Faster, fewer joins | | Slower (more joins) |
| Storage | More redundant data | | Less redundancy |
| Complexity | | Easier to understand | More complex to navigate |
| Best for | | Reporting, BI dashboards | Data warehousing with strict normalization |