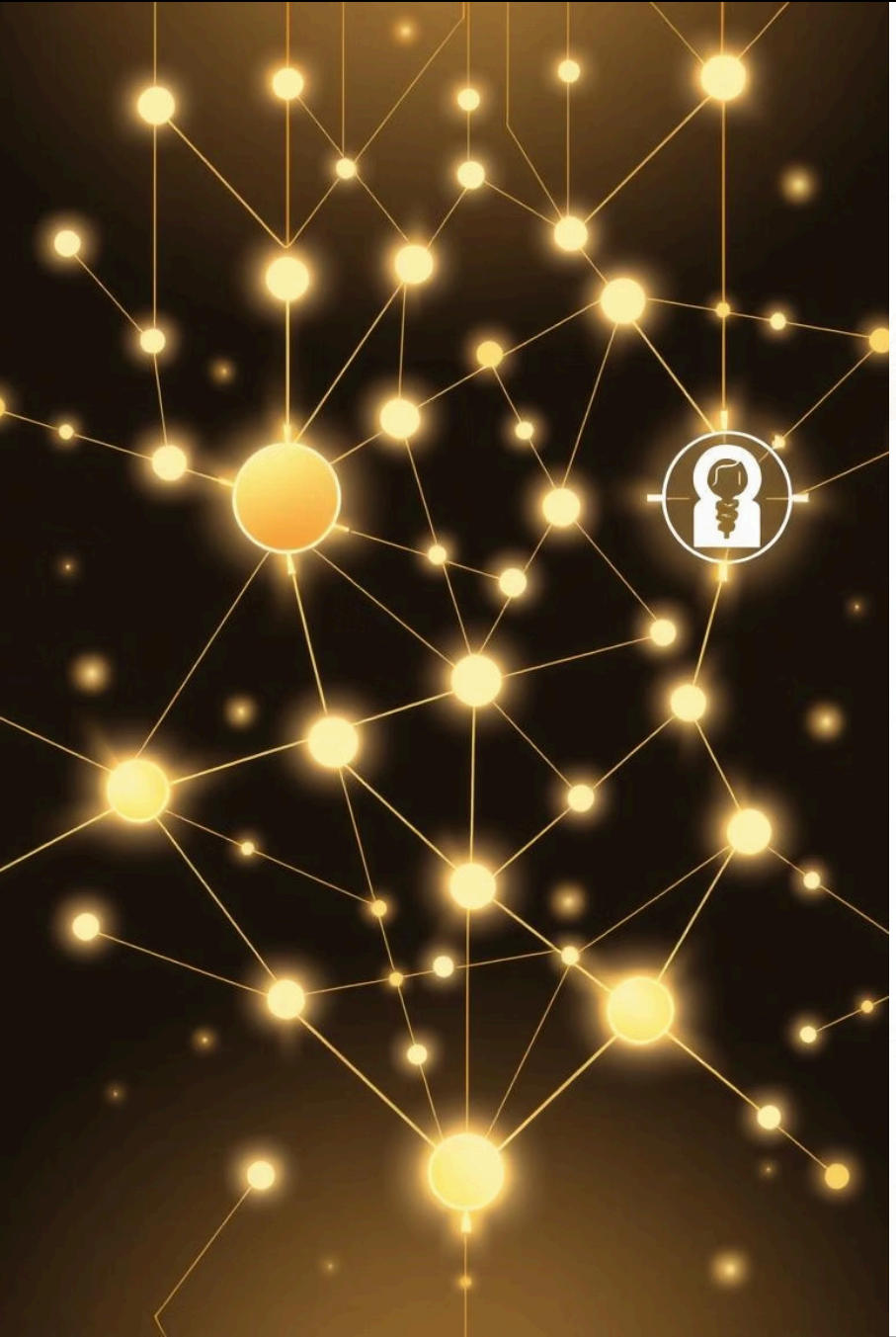


Healthcare Revenue Cycle Management (RCM) - Data Pipeline

End-to-End Data Engineering Project

Presented by: Vasavi Anusha Botta

Tools: Python, SQL, MySQL, BigQuery, Pandas

Project Overview: Optimising Healthcare Revenue

This project simulates a real-world scenario focused on optimising healthcare revenue cycle management. Our solution consolidates fragmented data from two disparate hospital systems into a unified, actionable format.

At its core, it leverages a robust, scalable ETL pipeline designed to ingest, process, and transform raw data. The ultimate output is business-ready analytics, housed securely within Google BigQuery, providing immediate insights for decision-makers.

The Business Challenge:

Fragmented Systems

Two hospitals operated with entirely separate and incompatible data systems, leading to a disjointed view of patient information and billing.

Manual Claim Delays

Predominantly manual claim processing resulted in significant delays, high error rates, and increased administrative overhead, directly impacting revenue.

No Historical Tracking

A critical lack of historical patient data tracking meant no audit trail for changes, hindering analysis of patient journeys and treatment effectiveness.

Revenue Leakage

These issues combined to create substantial revenue leakage and expose the organisations to significant compliance risks due to inconsistent data practices.

Technical Objectives: Building a Robust Data Foundation

1

Data Integration

Integrate diverse data sources, specifically MySQL databases from both hospitals and supplementary CSV files containing claims data.

2

Data Quality

Implement rigorous validation, cleansing, and unification processes to ensure high data quality and consistency across all datasets.

3

Historical Tracking

Apply Slowly Changing Dimensions (SCD) Type 2 methodology to accurately track and maintain a comprehensive historical record of patient information.

4

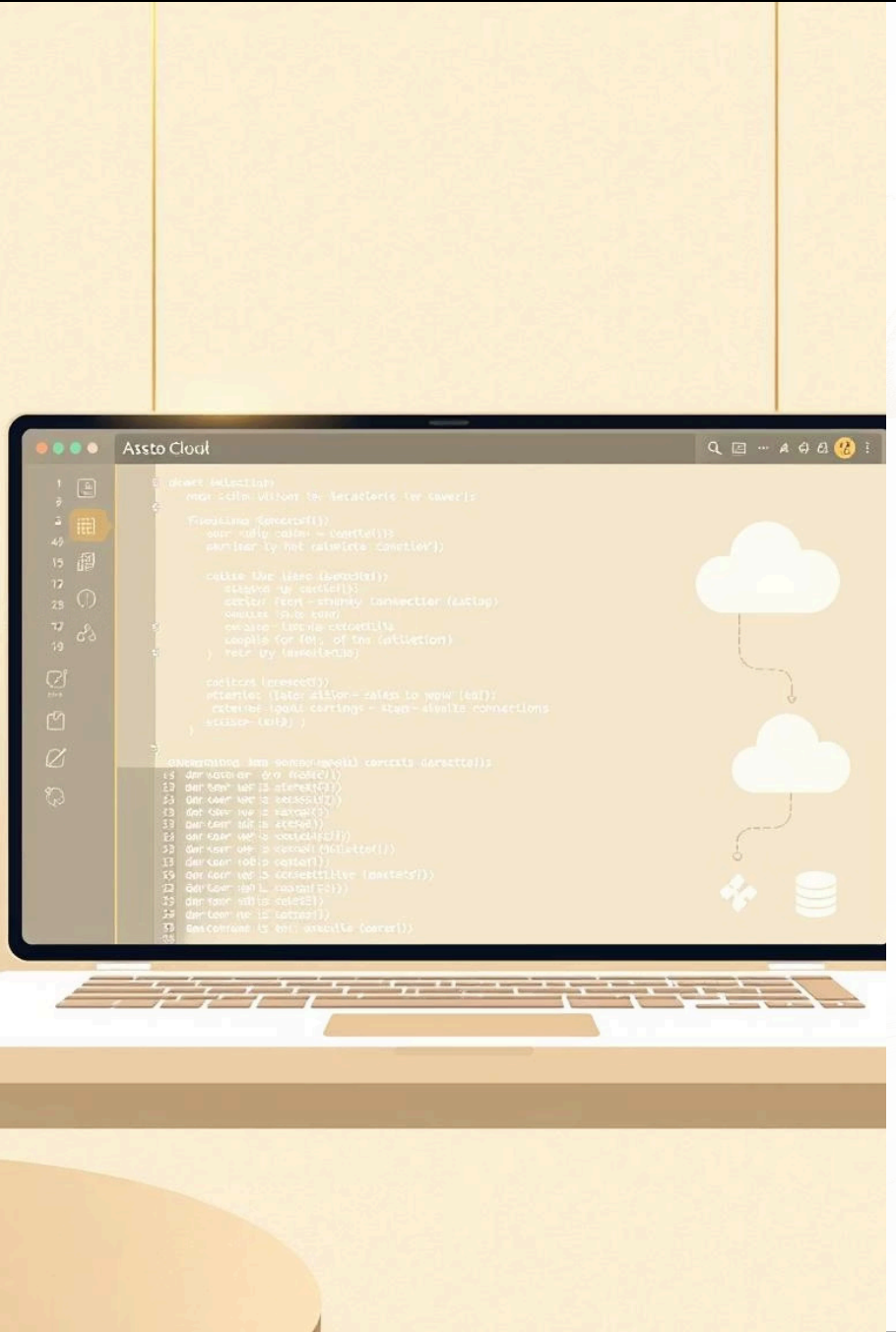
Cloud Data Warehousing

Load the transformed, high-quality data into Google BigQuery, leveraging its scalability and analytical capabilities for efficient storage and querying.

5

KPI Enablement

Structure the data in BigQuery to seamlessly enable tracking of key performance indicators (KPIs), providing real-time insights for revenue cycle management.



Technology Stack: Core Tools for the Pipeline

Programming & Databases

- **Python:** The primary language for ETL logic, scripting, and data manipulation.
- **SQL:** Essential for database interactions, querying, and data definition.
- **MySQL:** Source databases for patient and hospital operational data.
- **BigQuery:** The cloud data warehouse for final storage and analytics.

Libraries & Tools

- **Pandas:** For powerful data manipulation and analysis within Python.
- **SQLAlchemy:** ORM for abstracting database interactions and connections.
- **Google-Cloud-BigQuery:** Python client library for BigQuery integration.
- **VSCode & Google Colab:** Development environments for coding and prototyping.

ETL Pipeline Phases: A Step-by-Step Approach

1 **Environment Setup**

Configuring development and production environments, including database connections and cloud credentials.

2 **Data Extraction**

Retrieving raw data from disparate source systems (MySQL and CSV files).

3 **Transformation**

Cleaning, standardising, and enriching data to prepare it for analysis.

4 **Dimensional Modeling**

Designing the data warehouse schema using a star schema approach.

5 **SCD Type 2**

Implementing Slowly Changing Dimensions to manage historical data for patient records.

6 **BigQuery Load**

Loading the transformed and modeled data into Google BigQuery tables.

7 **Analytics & Reporting**

Enabling business intelligence tools and dashboards for KPI tracking and insights.

Data Extraction: Sourcing Information

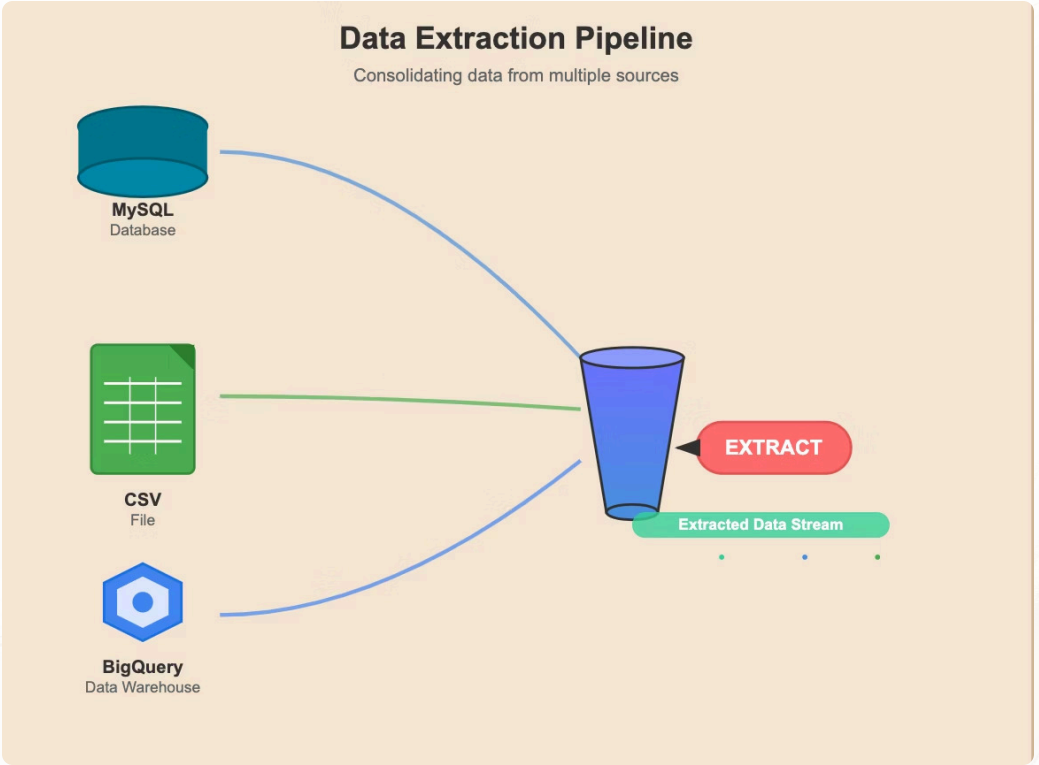
The extraction phase involves gathering data from its primary sources:

MySQL Databases: Data from both Hospital A and Hospital B's operational systems, including patient demographics, appointments, and treatment records.

CSV Files: Supplementary claims data, which often arrives in flat file formats due to external billing systems or legacy processes.

A dedicated `DataExtractor` class was developed to manage connections, query execution, and robust error handling during the retrieval process, ensuring data integrity from the outset.

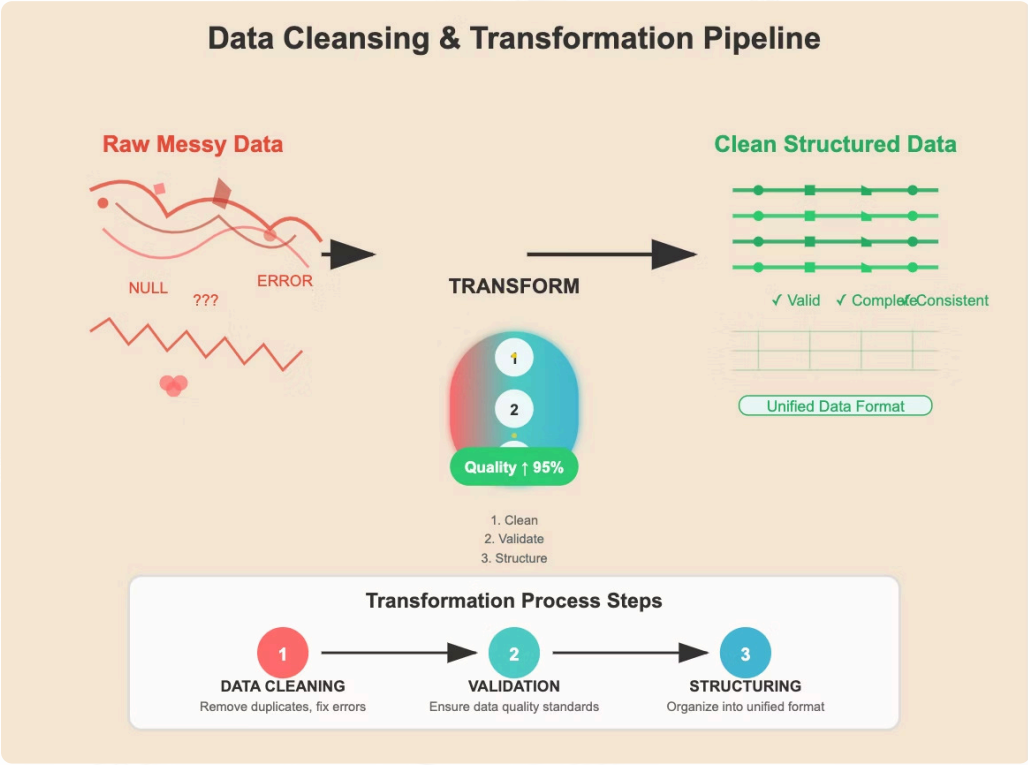
No Duplicate Datasets are used for the transactions and all are taken from given hospital dataset only.



Extraction Output :

```
(base) apple@MacBook-Pro required_data_for_p3_healthcare_RCM % python /Users/apple/Desktop/Healthcare/required_data_for_p3_healthcare_RCM/phase2_output.py

--- Phase 2: Data Extraction & Combination (Raw Counts + Preview) ---

✅ patients: 10000 rows before cleaning
      PatientID FirstName LastName MiddleName          SSN  ...         DOB                                    Address ModifiedDate HospitalName Updated_Date
0  HOSP1-000001      Rick    Russo          U  188-23-9828  ...  1937-06-04           Unit 0915 Box 7064, DPO AA 82777   2020-05-25   Hospital A          NaN
1  HOSP1-000002   Gregory   Graham          B  730-45-8217  ...  1937-06-10   9864 Gibson Islands, Danielside, KY 99809   2021-06-05   Hospital A          NaN
2  HOSP1-000003      Mary     Ryan          H  348-14-7947  ...  1926-08-09  6194 Joseph Turnpike, North Juan, OH 46800   2024-09-06   Hospital A          NaN

[3 rows x 12 columns]
--------------------------------------------------------------------------------------------------------------
✅ providers: 55 rows before cleaning
    ProviderID FirstName LastName      Specialization   DeptID         NPI HospitalName
0  H1-PROV0001   Brandon   Harper            Oncology  DEPT018  3086631719   Hospital A
1  H1-PROV0002      Luke    Clark  Emergency Medicine  DEPT006  1265568618   Hospital A
2  H1-PROV0003   Brandon   Austin          Pediatrics  DEPT014  3152052791   Hospital A
--------------------------------------------------------------------------------------------------------------
✅ transactions: 20000 rows before cleaning
  TransactionID EncounterID     PatientID ProviderID   DeptID   VisitDate ServiceDate  ... LineOfBusiness MedicaidID  MedicareID  InsertDate  ModifiedDate PaymentStatus HospitalNa
me
0   TRANS000001   ENC001204  HOSP1-002372   PROV0456  DEPT002  2024-08-02  2024-05-25  ...     Commercial  MEDI24173  MCARE19466  2021-01-16    2021-12-27           NaN   Hospital
 A
1   TRANS000002   ENC000029  HOSP1-002329   PROV0321  DEPT013  2024-05-02  2024-09-14  ...       Self-Pay  MEDI63110  MCARE97946  2021-02-06    2022-01-26           NaN   Hospital
 A
2   TRANS000003   ENC001088  HOSP1-004636   PROV0405  DEPT007  2024-07-25  2024-03-04  ...       Medicaid  MEDI83622  MCARE77469  2022-05-09    2021-09-22           NaN   Hospital
 A

[3 rows x 23 columns]
--------------------------------------------------------------------------------------------------------------
✅ encounters: 20000 rows before cleaning
  EncounterID     PatientID EncounterDate EncounterType ProviderID DepartmentID  ProcedureCode InsertedDate ModifiedDate HospitalName
0   ENC000001  HOSP1-001127    2020-01-14     Inpatient   PROV0133      DEPT019          97099   2023-03-08   2020-03-01   Hospital A
1   ENC000002  HOSP1-003842    2021-01-21    Outpatient   PROV0147      DEPT013          19272   2022-05-01   2022-02-05   Hospital A
2   ENC000003  HOSP1-001372    2021-10-29  Telemedicine   PROV0444      DEPT004          65512   2023-06-13   2022-08-11   Hospital A
--------------------------------------------------------------------------------------------------------------
✅ claims: 20000 rows before cleaning
      ClaimID TransactionID     PatientID EncounterID ProviderID   DeptID ServiceDate  ... ClaimStatus PayorType  Deductible  Coinsurance  Copay  InsertDate  ModifiedDate
0  CLAIM000001   TRANS007657  HOSP1-001026   ENC002703   PROV0190  DEPT009  2024-07-22  ...     Pending  Self-pay      279.32       170.77  13.01  2022-06-15    2024-03-03
1  CLAIM000002   TRANS004722  HOSP1-000299   ENC001313   PROV0260  DEPT002  2024-05-21  ...    Approved   Private      250.16       100.11  11.37  2023-11-13    2023-05-15
2  CLAIM000003   TRANS001815  HOSP1-003085   ENC009891   PROV0469  DEPT013  2024-05-31  ...      Denied   Private      241.32        99.26  38.21  2021-11-25    2021-05-25

[3 rows x 18 columns]
--------------------------------------------------------------------------------------------------------------
(base) apple@MacBook-Pro required_data_for_p3_healthcare_RCM %
```

Transformation: Enhancing Data Quality



This critical phase involves several steps to refine the extracted data:

- **Standardisation:** Uniform formatting of contact information like phone numbers and email addresses, and consistent casing for names.
- **Derived Attributes:** Calculation of patient age from date of birth and deriving patient status (e.g., active, discharged).
- **ID Unification:** Merging and standardising patient IDs across disparate systems to create a single, unified patient identifier.
- **Cleansing & Correction:** Identification and removal of duplicate records, and correction of any inconsistencies or errors within the datasets.

Transformation Output:

```
(base) apple@MacBook-Pro required_data_for_p3_healthcare_RCM % python /Users/apple/Desktop/Healthcare/required_data_for_p3_he
althcare_RCM/phase3_transformation.py

✅ Patients cleaned: 9985 rows
💾 Saved: /Users/apple/Desktop/Healthcare/required_data_for_p3_healthcare_RCM/output/cleaned_patients.csv
✅ Providers cleaned: 55 rows
💾 Saved: /Users/apple/Desktop/Healthcare/required_data_for_p3_healthcare_RCM/output/cleaned_providers.csv
✅ Transactions cleaned: 10000 rows
💾 Saved: /Users/apple/Desktop/Healthcare/required_data_for_p3_healthcare_RCM/output/cleaned_transactions.csv
✅ Encounters cleaned: 10000 rows
💾 Saved: /Users/apple/Desktop/Healthcare/required_data_for_p3_healthcare_RCM/output/cleaned_encounters.csv
✅ Claims cleaned: 10000 rows
💾 Saved: /Users/apple/Desktop/Healthcare/required_data_for_p3_healthcare_RCM/output/cleaned_claims.csv
(base) apple@MacBook-Pro required_data_for_p3_healthcare_RCM %
```

Dimensional Modeling: Star Schema Design

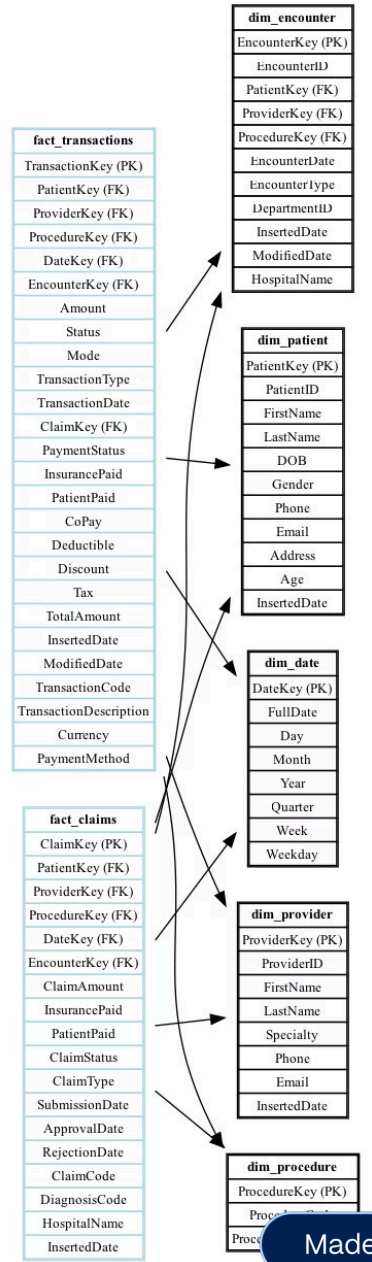
To enable efficient analytics and reporting, a star schema approach was implemented in the data warehouse. This design separates business process data into fact tables and contextual attributes into dimension tables, simplifying queries and improving performance.

Dimensions

- **Patient Dimension:** Patient demographics, contact info, and historical changes.
- **Provider Dimension:** Doctor and practitioner details.
- **Date Dimension:** Granular date information for time-based analysis.
- **Procedure Dimension:** Details of medical procedures performed.
- **Encounter Dimension:** Information about each patient visit or interaction.

Fact Tables

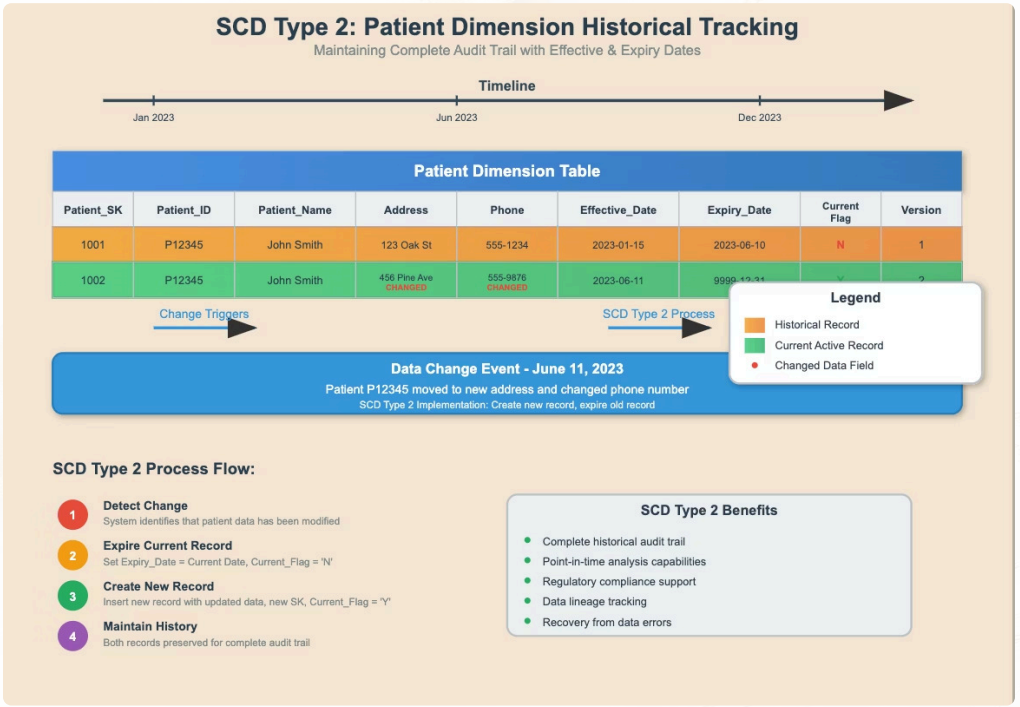
- **Claims Fact:** Details on submitted and processed claims, including amounts and statuses.
- **Transactions Fact:** Financial transactions related to patient care and billing.



Dimensional Tables Output:

```
(base) apple@MacBook-Pro required_data_for_p3_healthcare_RCM % python /Users/apple/Desktop/Healthcare/required_data_for_p3_he
althcare_RCM/phase4_output.py
✅ dim_patient: 9985 rows — saved to dim_patient.csv
✅ dim_provider: 55 rows — saved to dim_provider.csv
✅ dim_date: 1761 rows — saved to dim_date.csv
✅ dim_procedure: 996 rows — saved to dim_procedure.csv
✅ fact_transactions: 10000 rows — saved to fact_transactions.csv
✅ fact_claims: 10000 rows — saved to fact_claims.csv
(base) apple@MacBook-Pro required_data_for_p3_healthcare_RCM %
```

SCD Type 2: Tracking Patient History



To maintain a complete historical record of patient information, Slowly Changing Dimension (SCD) Type 2 was implemented for the Patient Dimension.

This method allows us to track changes to patient attributes over time without overwriting previous data. Key fields added to the patient dimension table include:

`effective_date`: The start date of the record's validity.

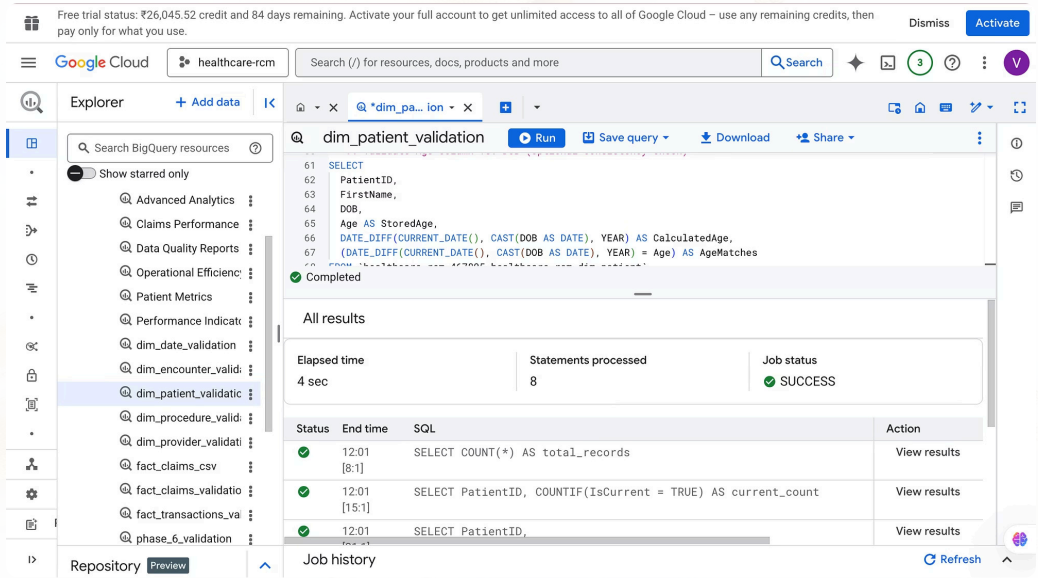
`expiry_date`: The end date of the record's validity (NULL for the current record).

`is_current`: A boolean flag indicating if this is the most current record for the patient.

`version`: A numerical identifier for each version of the patient record.

This approach ensures a full audit trail, crucial for compliance and long-term analytical insights into patient journeys.

SCD Version-2 Output:

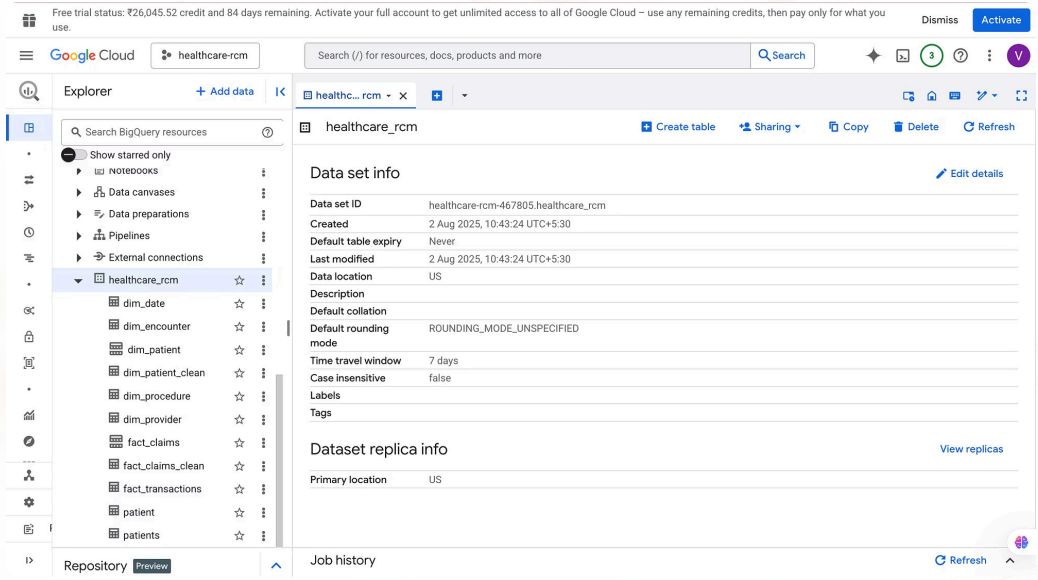


Big Query Validation

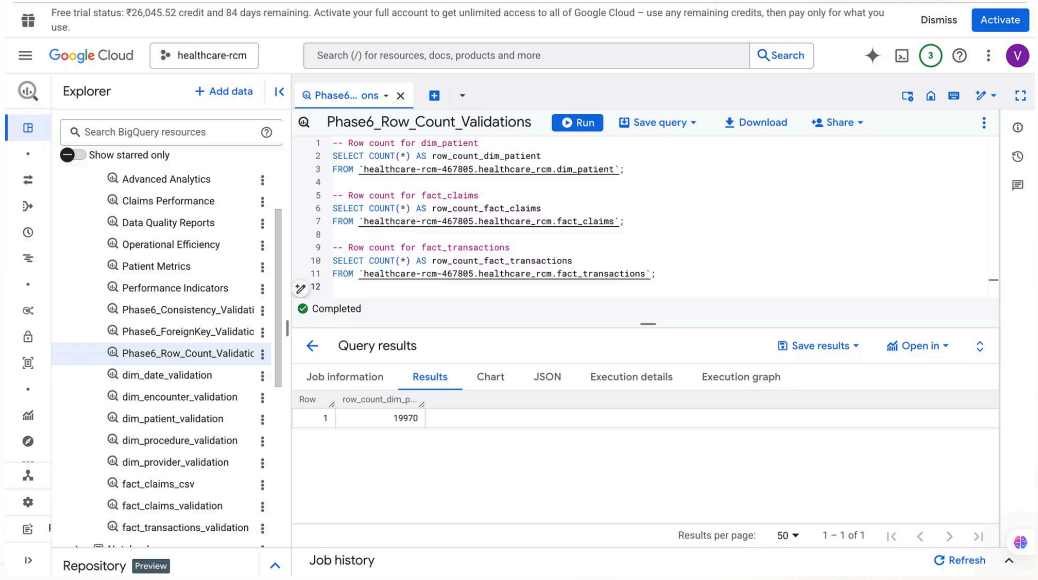
```
(base) apple@MacBook-Pro etl % python scd2.py
✅ Data Loaded Successfully
📋 Historical Columns: ['PatientID', 'FirstName', 'MiddleName', 'LastName', 'Gender', 'DOB', 'Age', 'hospital', 'StartDate',
'EndDate', 'IsCurrent', 'Version']
📋 New Data Columns: ['PatientID', 'FirstName', 'LastName', 'MiddleName', 'SSN', 'PhoneNumber', 'Gender', 'DOB', 'Address', '
ModifiedDate', 'hospital', 'Age']
/Users/apple/Desktop/Healthcare/required_data_for_p3_healthcare_RCM/etl/scd2.py:105: FutureWarning: Setting an item of incomp
atible dtype is deprecated and will raise an error in a future version of pandas. Value '2025-08-08' has dtype incompatible w
ith datetime64[ns], please explicitly cast to a compatible dtype first.
  self.historical_df.loc[idx, 'EndDate'] = today
➕ New Records: 0
🔄 Updated Records: 9985
✅ Unchanged Records: 0
📦 Final SCD2 file written to: /Users/apple/Desktop/Healthcare/required_data_for_p3_healthcare_RCM/output/dim_patient_history
.csv
✅ Data loaded to BigQuery table: healthcare-rcm-467805.healthcare_rcm.dim_patient
(base) apple@MacBook-Pro etl %
```

SCD Version-2 Upload to Big Query

Big Query Integration Output

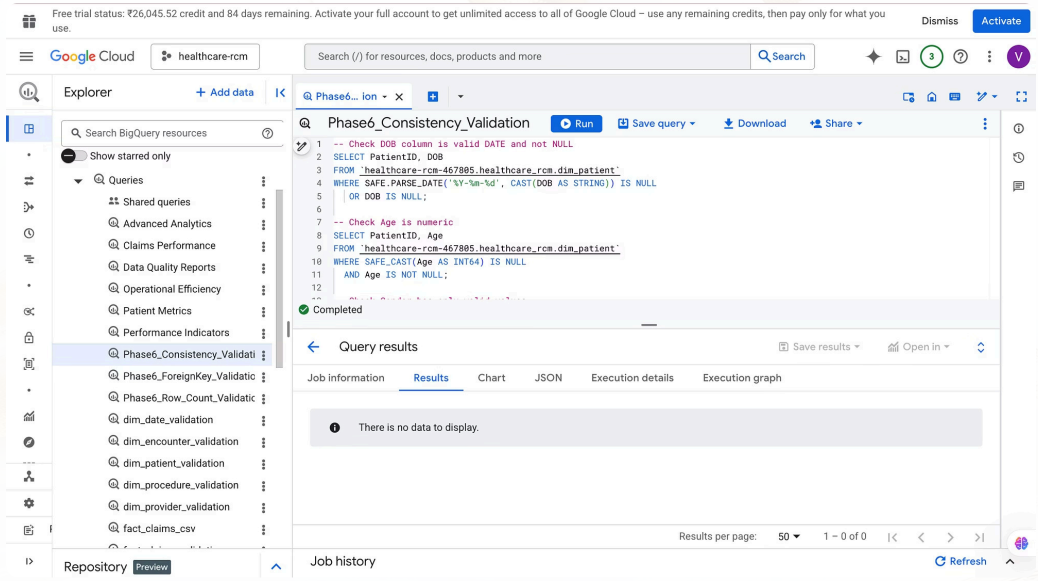


Load Dimensional And Fact Tables in Big Query

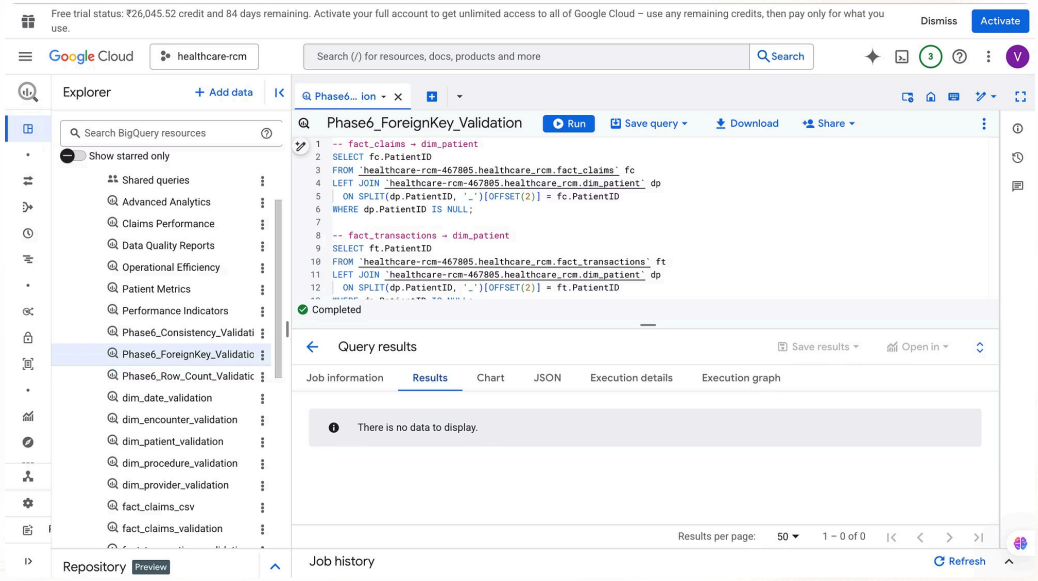


Row_Count_Validations

Big Query Integration Output



Consistency Validation

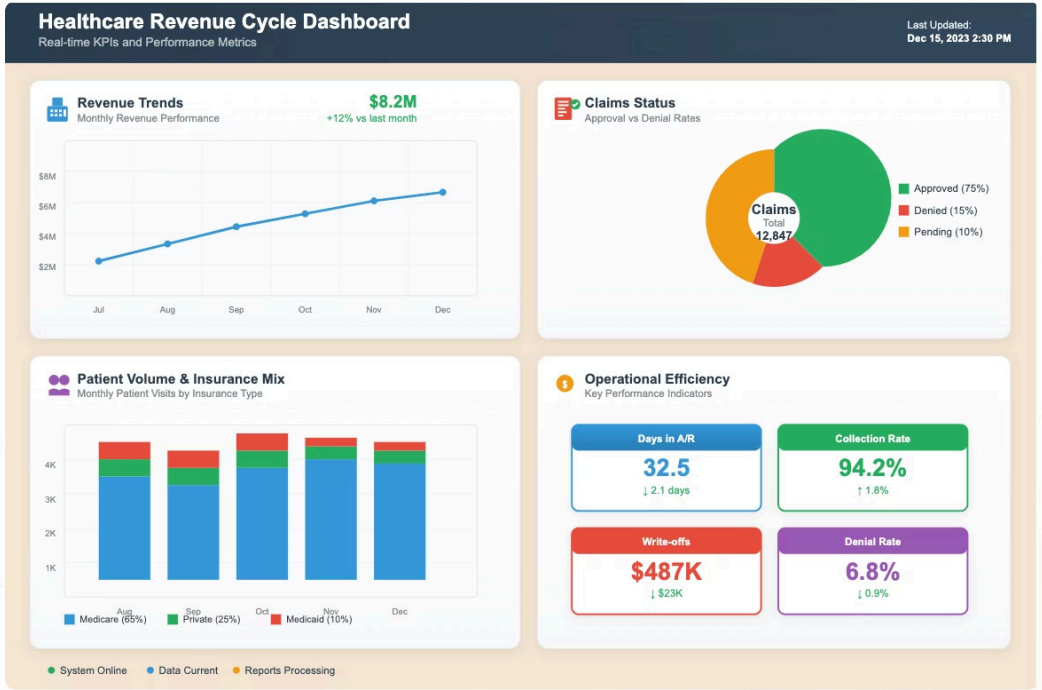


Foreign Key Validation

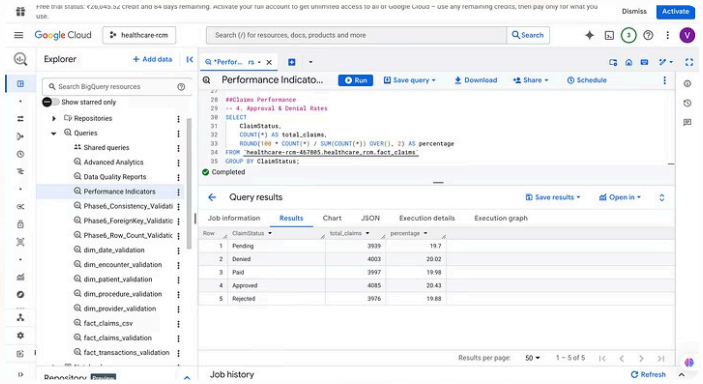
BigQuery KPIs for RCM Insights:

In the final phase of the data pipeline, we loaded the clean, validated dimensional model into **Google BigQuery**.

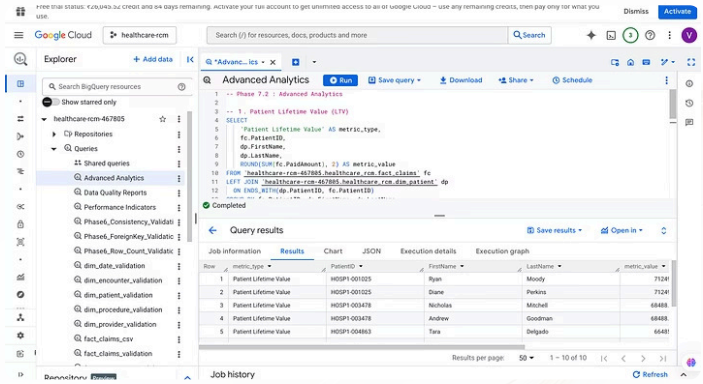
- **Revenue Metrics:**
  - Total revenue by month, provider, and hospital
  - Revenue trend analysis over time
- ✅ **Claim Performance:**
  - **Approval rate:** % of claims approved on first submission
  - **Denial rate:** % of rejected claims and reasons
  - **Avg. processing time:** Time between submission and settlement
- 👥 **Patient Metrics:**
  - Patient volume over time
  - Insurance mix (private, public, self-pay)
  - Demographics: age, gender, location
- ⚙️ **Operational Efficiency:**
  - **Days in A/R (Accounts Receivable):** Avg. days to collect revenue
  - **Collection rate:** % of billed amount actually collected
  - **Write-offs:** Total value of unpaid claims



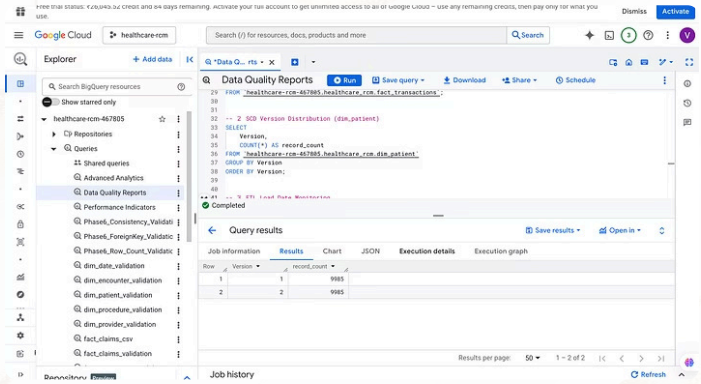
RCM Analytics Output :



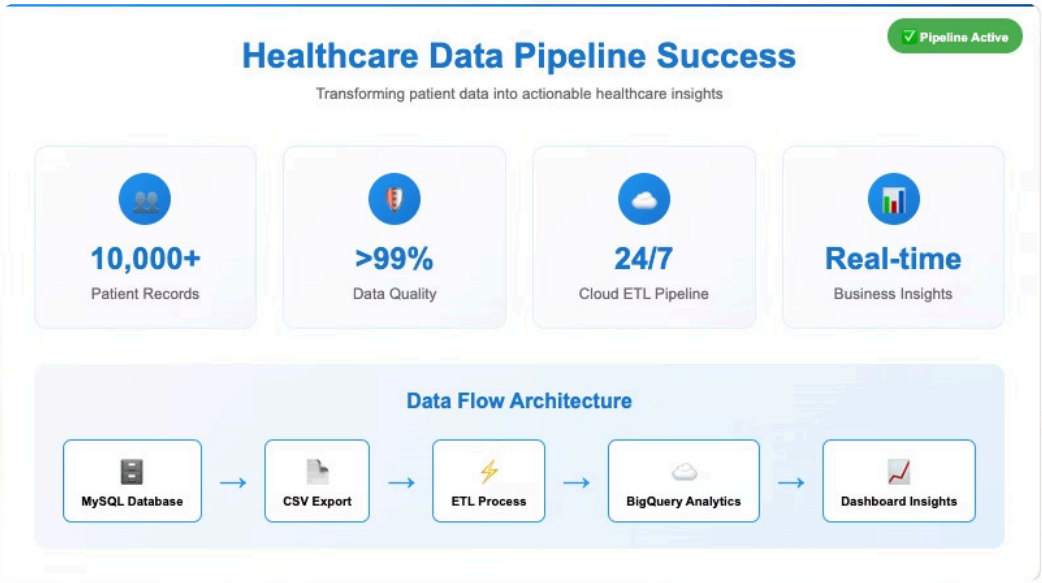
Performance Indicators



Advanced Analytics.



Data Quality Reports



Project Impact & Readiness

Scale & Quality:

- Processed **10,000+ patient records**
- Achieved **>99% data quality** through validation, cleansing, and error handling

🔁 **ETL Pipeline:**

- End-to-end automated pipeline from **MySQL + CSV to BigQuery**
- Includes historical tracking using **Slowly Changing Dimensions (SCD Type 2)**

☁️ **Cloud Integration:**

- Fully integrated with **Google BigQuery** for scalable querying and fast performance
- Star schema enables efficient joins and slicing for analytics

📈 **Business Value:**

- Enables **data-driven decision-making** for hospital finance and operations teams
- Supports KPI tracking, denial reduction, and revenue optimization

THANK YOU