# Solution

1. **Data Ingestion**  
   Upload raw datasets to **AWS S3** under a folder input/.
2. **Data Cleaning (Databricks / PySpark)**

Read raw data from S3  
Handle null values, standardize formats, and remove duplicates  
Validate data types

1. **Data Load to Redshift**  
   Define the target schema and tables in **AWS Redshift**Write cleaned DataFrames to Redshift using the JDBC connector
2. **Use Case Execution (Redshift SQL)**  
   Write SQL queries for each business use case  
   Store results in dedicated output tables under project\_output schema
3. **Automation & Version Control**  
   Store notebooks/scripts in **GitHub**Use **Databricks** to automate workflows.

# Use Cases

This solution addresses:

* Identifying the most claimed disease
* Subscribers under age 30 and their subgroup memberships
* Group with the most subgroups
* Most visited hospital
* Most subscribed subgroup
* Rejected claim count
* City generating the most claims
* Government vs. Private policy popularity
* Average monthly premium paid
* Most profitable policy group
* Cancer admissions under age 18
* High-cost claims with cashless insurance
* Knee surgeries in females 40+ in the past year

1. Database Design

## Tables Metadata Info with Pk/FK relationship –

#### a. Tables & Metadata with PK/FK

| **Table Name** | **Primary Key** | **Foreign Keys** |
| --- | --- | --- |
| patients | patient\_id | group\_id (FK) |
| subscribers | subscriber\_id | patient\_id (FK), group\_id (FK) |
| claims | claim\_id | patient\_id (FK), hospital\_id (FK) |
| group\_subgroup | group\_id | — |
| hospitals | hospital\_id | — |
| project\_output.\* | varies | Derived from above tables |

## 

# Technologies and Platforms to be used in this solution.

* **AWS S3** – Storage for raw input data
* **AWS Redshift** – Analytical data warehouse
* **Databricks**– PySpark environment for processing
* **PySpark** – Data cleansing, transformation
* **GitHub** – Version control and deployment
* **Jira** – Task and sprint management

## 