**Solution Design Document**

**1. Solution Overview**

The solution automates data ingestion, cleansing, transformation, analysis, and visualization for a health insurance provider. It processes multi-source CSV data via a Spark-based ETL pipeline, stores cleaned data in Redshift, and generates insights aligned with 13+ business use cases. The architecture supports modular development, data governance, and agile delivery.

**Step-by-Step Design:**

* Ingest raw CSVs from S3 bucket input-data/ into Databricks Bronze layer (raw zone).
* Validate nulls and duplicates, log data quality stats.
* Clean data (replace nulls with 'NA', drop duplicates).
* Load cleansed data to Redshift project\_staging schema using COPY via IAM.
* Generate business insights (13+ KPIs) using SQL queries.
* Store results in project\_output schema in Redshift.
* Produce visual dashboards for 5+ KPIs via Databricks.
* Maintain versioned PySpark codebase in GitHub with automated test pipeline.

**2. Use Cases Covered**

* FR 05a: Disease with maximum claims
* FR 05b: Subscribers <30 who subscribe any subgroup
* FR 05c: Group with maximum subgroups
* FR 05d: Hospital serving most patients
* FR 05e: Most subscribed subgroup
* FR 05f: Total rejected claims
* FR 05g: Cities with highest claim counts
* FR 05h: Government vs. Private group popularity
* FR 05i: Average monthly premium per subscriber
* FR 05j: Most profitable group (premium - claims)
* FR 05k: Patients <18 admitted for cancer
* FR 05l: Patients with cashless insurance and charges ≥ ₹50,000
* FR 05m: Female patients >40 with knee surgery in past year

**3. Database Design (Redshift)**

**Schema: project\_staging** (raw cleaned tables):

* patient\_records
* subscriber
* hospital
* group
* subgroup
* grpsubgrp
* disease

**Schema: project\_output** (final result tables/views):

* max\_claim\_disease
* young\_subscribers\_subgroup
* group\_max\_subgroups
* top\_hospital\_patients
* top\_subscribed\_subgroup
* total\_rejected\_claims
* claims\_by\_city
* policy\_type\_preference
* avg\_monthly\_premium
* most\_profitable\_group
* cancer\_patients\_under\_18
* cashless\_over\_50k
* female\_knee\_surgery

**4. Tables Metadata Info with PK/FK Relationships**

* subscriber(subscriber\_id) ⬌ patient\_records.subscriber\_id *(FK)*
* group(group\_id) ⬌ grpsubgrp.group\_id
* subgroup(subgroup\_id) ⬌ grpsubgrp.subgroup\_id and subscriber.subgroup\_id
* hospital(hospital\_id) ⬌ patient\_records.hospital\_id
* disease(disease\_id) ⬌ patient\_records.disease\_id

**5. Technologies and Platforms Used**

* **Cloud Storage**: AWS S3 (raw input)
* **Processing Engine**: PySpark (Databricks / AWS EMR)
* **Data Warehouse**: AWS Redshift
* **Project Management**: Jira (Agile Sprint)
* **Version Control**: GitHub
* **Visualization**: Databricks (Dashboards, Graphs)
* **Deployment/Integration**: CI/CD with GitHub PR validation