# **Solution Design Document**

#### **Solution**

The solution involves building a robust big data pipeline using AWS services, Databricks, and PySpark for data ingestion, cleansing, transformation, and analysis of healthcare insurance data. The steps are as follows:

1. **Data Ingestion**:
   * Collect data from various sources, including competitor data obtained via scraping, and third-party sources.
   * Store the raw data in an AWS S3 bucket under a structured folder (input-data) for further processing.
2. **Data Cleansing and Preprocessing**:
   * Load raw data from S3 into Databricks for preprocessing.
   * Conduct data cleansing activities such as null value replacement, duplicate record removal, and data validation (e.g., ensuring data format consistency).
   * Cleaned data is then uploaded to AWS Redshift for structured storage in respective tables.
3. **Data Transformation and Aggregation**:
   * Execute transformation processes using PySpark to perform aggregations and filter operations (e.g., identify most frequent disease claims, filter subscribers by age and subgroup).
   * Generate financial insights by calculating metrics like average monthly premiums and the most profitable policy groups.
4. **Data Storage in Redshift**:
   * Store each cleaned and transformed dataset in separate Redshift tables, following a structured schema.
   * Data tables for entities such as Patients, Subscribers, Claims, and Group\_Subgroup will be used for easy retrieval and analysis.
5. **Analytics and Result Generation**:
   * Execute SQL queries on Redshift to derive insights, such as the city with the most claims, hospital serving the most patients, policy group preferences, and more.
   * Store each query result in designated tables within a Project-Output schema in Redshift for organised output storage.
6. **Integration with Databricks for Visualization**:
   * Visualise key insights (e.g., top diseases by claims, subgroup popularity, average premium by age group) using Databricks’ visualisation capabilities, allowing for more accessible data interpretation and sharing.
7. **Deployment and Code Management**:
   * Maintain PySpark code in a GitHub repository for version control.
8. **Documentation and Sprint Planning**:
   * Document requirements specifications and solution design in GitHub, along with Jira sprint plans for organised task management and timely project completion.

#### **2. Use Cases**

The following use cases are covered in this solution:

* Determine the disease with the maximum number of claims.
* Identify subscribers under age 30 who have subscribed to any subgroup.
* Find the group with the most subgroups.
* Identify hospitals with the highest number of patients served.
* Track the most frequently subscribed subgroups.
* Count the total number of rejected claims.
* Determine the city with the highest claim submissions.
* Analyse policy preference (Government vs. Private) among subscribers.
* Calculate the average monthly premium paid by subscribers.
* Identify the most profitable policy group.
* List patients under age 18 admitted for cancer.
* Identify patients with cashless insurance and charges over Rs. 50,000.
* List female patients over age 40 who had knee surgery in the past year.

#### 

#### **3. Database Design**

The solution’s database design in Redshift will include the following tables:

1. **Patients**
   * Stores patient demographics and medical history.
2. **Subscribers**
   * Contains subscriber details, including age, gender, policy group, and subgroup.
3. **Claims**
   * Maintains claims details, including claim status, disease, hospital, and charges.
4. **Group\_Subgroup**
   * Contains information on groups and subgroups for analysis of policy preferences.
5. **Output Tables (in Project-Output schema)**
   * Dedicated tables for each specific use case, storing precomputed analysis results for easy retrieval and reporting.

#### **4. Technologies and Platforms to be Used in this Solution**

* **AWS S3**: Storage of raw data and processed data files for easy access and retrieval.
* **AWS Redshift**: Centralised database for structured data storage, supporting complex queries and data aggregation.
* **Databricks**: Platform for running PySpark transformations, visualisations, and development in a big data environment.
* **PySpark**: Data processing and transformation tool for handling large datasets, enabling efficient data cleaning, filtering, and aggregation.
* **Jira**: Project management tool for sprint planning, tracking, and organising user stories.
* **GitHub**: Repository for version control, housing PySpark scripts and project documentation.

# 

# 

## 