# Solution - *Explain your solution here in a step by step manner.*

The solution involves a comprehensive data processing and analytics system designed to enhance revenue and customer understanding for the Health Care insurance company by leveraging Big Data Ecosystem analysis. Below is a step-by-step explanation of the solution:

**Step 1: Data Ingestion**

* **Data Sources:** Data is ingested from various sources, including internal systems, web scraping, third-party data providers, and more.
* **Storage:** The ingested data is stored in AWS S3 in the input-data folder.

**Step 2: Data Cleaning**

* **Null Handling:** Identify and count null values in the datasets, replacing them with "NA" where applicable.
* **Duplicate Removal:** Detect and remove any duplicate records from the datasets.
* **Data Cleansing for Specific Datasets:** Clean the data for key datasets such as Patients, Subscribers, Claims, and Group\_Subgroup to ensure consistency and accuracy before further processing.

**Step 3: Data Transformation**

* **Schema Design:** Develop a schema to structure the cleaned data for loading into AWS Redshift.
* **Transformation:** Transform the cleaned data into a suitable format, partition, and index it in Redshift to optimize query performance.

**Step 4: Data Loading**

* **Loading into Redshift:** Load the transformed data into corresponding tables in AWS Redshift. The tables are designed to support efficient storage and retrieval for subsequent analysis.

**Step 5: Data Analysis**

* **Predefined Queries:** Execute specific analytical queries on the data, such as identifying the disease with the maximum claims, analyzing subscriber demographics, determining the most popular policy groups, and calculating rejected claims.
* **Result Storage:** Store the results of these analyses in the Project-Output schema in Redshift for easy access and reporting.

**Step 6: Visualization and Reporting**

* **Dashboard Creation:** Use Databricks to create visualizations of the analysis results. These dashboards provide insights into customer behavior, claim patterns, and other critical metrics.
* **Report Generation:** Generate and export reports from Databricks for further analysis and decision-making.

**Step 7: Integration and Deployment**

* **Jira for Project Management:** Integrate with Jira to track project progress, user stories, and tasks.
* **GitHub for Version Control:** Maintain all source code, including data processing and analysis scripts, in a GitHub repository for version control.
* **Deployment on AWS:** Deploy the solution on AWS EMR or Databricks, ensuring scalability and performance.

# Use Cases - *List down all the use cases on which this solution will be applicable.*

The solution applies to various use cases, including:

1. **Revenue Optimization:**
   * Analyze claim data to identify the most and least profitable customer segments.
   * Provide insights to optimize pricing strategies and enhance revenue streams.
2. **Customer Segmentation:**
   * Segment customers based on demographics, policy preferences, and claim history to develop targeted marketing strategies.
3. **Claims Management:**
   * Identify trends in claim rejections and optimize the claim approval process.
   * Determine common reasons for claim rejections and develop strategies to reduce them.
4. **Healthcare Provider Analysis:**
   * Evaluate hospital performance by analyzing patient data, identifying the most frequented hospitals, and determining patient satisfaction levels.
5. **Policy Group Analysis:**
   * Assess the performance of different policy groups and subgroups to optimize product offerings and improve customer retention.
6. **Data-driven Decision Making:**
   * Provide actionable insights to leadership teams for strategic planning and operational improvements.
7. Database Design - List down all possible db(Redshift) tables here

## Tables Metadata Info with Pk/FK relationship -

## ER diagram – *Optional*

The database design involves creating a set of tables in AWS Redshift to store the cleaned and transformed data. Below are the possible tables along with their metadata:

**3.1 Tables Metadata Info with PK/FK Relationships**

1. **Patients Table**
   * **Columns:** patient\_id (PK), patient\_name, age, gender, city, state, disease\_id (FK), hospital\_id (FK)
   * **Relationships:** disease\_id references Diseases table, hospital\_id references Hospitals table.
2. **Subscribers Table**
   * **Columns:** subscriber\_id (PK), name, age, gender, city, state, group\_id (FK)
   * **Relationships:** group\_id references Groups table.
3. **Claims Table**
   * **Columns:** claim\_id (PK), patient\_id (FK), subscriber\_id (FK), disease\_id (FK), hospital\_id (FK), claim\_amount, claim\_status
   * **Relationships:** patient\_id references Patients table, subscriber\_id references Subscribers table, disease\_id references Diseases table, hospital\_id references Hospitals table.
4. **Diseases Table**
   * **Columns:** disease\_id (PK), disease\_name, description
5. **Hospitals Table**
   * **Columns:** hospital\_id (PK), hospital\_name, city, state
6. **Groups Table**
   * **Columns:** group\_id (PK), group\_name
   * **Relationships:** group\_id references Subgroups table.
7. **Subgroups Table**
   * **Columns:** subgroup\_id (PK), subgroup\_name, group\_id (FK)
   * **Relationships:** group\_id references Groups table.
8. **Project-Output Table**
   * **Columns:** Various columns depending on the analysis results, such as analysis\_id (PK), result\_description, metric\_value, etc.

# Technologies and Platforms to be used in this solution -*List down list of technologies like spark, aws and databricks etc.*

The solution leverages a variety of technologies and platforms to handle data ingestion, processing, analysis, and visualization:

1. **AWS S3:** For storing raw and cleaned data.
2. **AWS Redshift:** For structured data storage, processing, and query optimization.
3. **Apache Spark and PySpark:** For data processing and transformation tasks, particularly for handling large datasets.
4. **Databricks:** For data analysis, visualization, and creating dashboards.
5. **Jira:** For project management, including tracking user stories and tasks.
6. **GitHub:** For version control and collaborative development.
7. **AWS EMR:** For running big data workloads and Spark jobs.
8. **Docker:** For consistent development environments using containers.
9. **AWS Glue:** (If applicable) For managing ETL processes and data cataloging.
10. **Python:** For scripting data processing, transformation, and analysis logic.

## 