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

**STAGING**

**S3 setup** : Created dedicated S3 Bucket **capstone-proj** for storing project data. Inside the bucket we create 3 separate directories –

* + - * 1. raw-data/ : for storing unprocessed raw data
        2. input-data/ : for storing cleaned data to be used for processing
        3. redshift-dir/ : redshift directory to store redshift table information

**Databrick setup** : Create a compute cluster to access necessary computational resources for executing Spark jobs on databricks notebooks. Set the AWS S3 access and secret keys for a Spark session using the hadoopConfiguration method.

*spark.sparkContext.\_jsc.hadoopConfiguration().set("fs.s3a.access.key", <"my access key">)*

*spark.sparkContext.\_jsc.hadoopConfiguration().set("fs.s3a.secret.key", <"my secret key">)*

**Redshift setup** : Create redshift schema with name ***project\_output*** which will dedicatedly store redshift output tables in redshift database.

**ETL FOR DATA CLEANING:**

**Extract from AWS S3**

Read the raw data from S3 and create a dataframe on the raw data set provided.

**Data Cleaning**

Next, we perform data cleaning on the raw data by fixing or removing incorrect, corrupted, duplicate, mislabelled or incomplete data within the given dataset. For consistencies, NULL values were replaced with below mentioned values:

String -> NA

Integer -> 0

Dates -> 19000101

**Load transformed data into S3**

The final cleaned data is pushed into AWS S3 to be used as input data for processing S3 directory named as input-data. Each table will be stored in the individual table directory created inside input-data earlier.

**ETL FOR DATA PROCESSING (Implementation and Testing)**

Using Databricks notebook and pySpark, we will test the given use cases to check the accuracy of the designed code.

1. **Extract input data from AWS S3**

Read the input data from S3 and create a dataframe on the table data.

1. **Transform input data into redshift tables**

The designed model will be applied to the provided use cases to check the accuracy of the model as in one of the use cases below. Example:

A screenshot of a computer

Description automatically generated

A white background with black text

Description automatically generated

Output ->

1. **Load into Redshift Database Schema**

The final output data is pushed into Redshift database under the database schema ***project\_output***. Each table will be stored as an individual table in redshift database as below.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

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

## 1. Which disease has a maximum number of claims

## 2. Find those Subscribers having age less than 30 and they subscribe any subgroup

## 3. Find out which group has maximum subgroups

## 4. Find out hospital which serve most number of patients

## 5. Find out which subgroups subscribe most number of times

## 6. Find out total number of claims which were rejected

## 7. From where most claims are coming (city)

## 8. Which groups of policies subscriber subscribe mostly Government or private

## 9. Average monthly premium subscriber pay to insurance company

## 10. Find out Which group is most profitable

## 11. List all the patients below age of 18 who are admit for cancer

1. Database Design - List down all possible db(Redshift) tables here

## Tables Metadata Info with Pk/Fk relationship –

**Usecase 1**: **Usecase 2:**

A screen shot of a computer code

Description automatically generated A screenshot of a computer code

Description automatically generated

**Usecase 3**: **Usecase 4**:A screenshot of a computer code

Description automatically generated A screenshot of a computer

Description automatically generated

**Usecase 5**: **Usecase 6**:

A screenshot of a computer code

Description automatically generated A screenshot of a computer screen

Description automatically generated

**Usecase 7**: **Usecase 8**:

A screenshot of a computer code

Description automatically generated A screen shot of a computer code

Description automatically generated

**Usecase 9**: **Usecase 10**:

A screenshot of a computer

Description automatically generated A screen shot of a computer

Description automatically generated

**Usecase 11**:

A screenshot of a computer code

Description automatically generated

## ER diagram – *Optional*

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

## AWS S3 -> as a data warehouse to store unprocessed data, input data and Redshift tables.

## AWS Redshift -> as a database storage for the redshift output tables and visualization.

## AWS Glue -> as a database storage for the processed input tables and visualization.

## Databricks community version ->

## Pyspark ->

## Jira ->

## GitHub ->