# Solution:

* 1. Environment and data set creation
     1. Set up AWS account and create an S3 bucket named “input data”.
     2. Upload the given sample data to the “input-data” folder in the S3 bucket.
     3. Set up AWS redshift, Databricks and EMR studio.
     4. Create a GitHub repository to manage code
  2. Cleaning Activity
     1. Create PySpark scripts in data bricks to clean the data. Use PySpark’s DataFrame API
     2. Use the following command to connect your databricks to s3 and add your access key and secret key
        1. spark.sparkContext.\_jsc.hadoopConfiguration().set("fs.s3a.access.key", "")
        2. spark.sparkContext.\_jsc.hadoopConfiguration().set("fs.s3a.secret.key", "")
     3. Load the datasets from the S3 bucket into PySpark DataFrames
     4. Check for null values in each dataset using .isNull() functiom
     5. Count the total null values for each column using the **.agg()** function.
     6. Replace null values with "NA" or appropriate values using the .fillna() function.
     7. Drop duplicate records using the .dropDuplicates() function.
     8. Clean the required datasets: Patients, Subscriber, Claims, Group\_subgroup.
     9. Create cleaned DataFrames for each cleaned dataset.c.
  3. Redshift table Schema design
     1. Create a schema that shows the structure of redshift structure table for each cleaned DataFrames. It should include table names, column names, data types, primary key, foreign key.
     2. Use Create Table command to create new tables.
  4. Data Loading to RedShift
     1. Load cleaned data from DataBricks to redshift tables based on
     2. Use the following command and update the command with proper credentials
        1. spark.sparkContext.\_jsc.hadoopConfiguration().set("fs.s3a.access.key", "")
        2. spark.sparkContext.\_jsc.hadoopConfiguration().set("fs.s3a.secret.key", "")
        3. df.write.format("redshift").option("url", "jdbc:redshift://default-workgroup.556646946508.us-east-1.redshift-serverless.amazonaws.com:5439/dev").option("dbtable", "test.capstone -project").option("aws\_iam\_role", "").option("driver","com.amazon.redshift.jdbc42.Driver").option("tempdir", "s3a:// / ").option("user", "admin").option("password", " mode("overwrite").save()
  5. Analysis and Result Creation
     1. Create a schema “Project-output”.
     2. Write a SQL on those redshift tables (cleaned data table)
     3. One destination table for each use case
  6. Testing and Deploying
     1. Test your SQL queries in databricks to visualize the output of SQL
     2. Push the tested code to GitHub repository.
     3. Deploy it to AWS EMR using that code

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

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

## Tables Metadata Info with Pk/FK relationship -

## ER diagram - *Optional*

# Technologies and Platforms to be used in this solution

AWS S3

AWS Redshift

Databricks

AWS EMR Studio

Pyspark

Jira

GitHub