# Solution

The **"Healthcare Insurance Competitor Analysis Data Pipeline for Customer Attraction and Revenue Growth"** is a strategic initiative designed to empower the company by harnessing the power of big data analytics. The primary objective of this initiative is to drive revenue growth and gain deeper insights into customer behavior and market dynamics. The company is currently facing challenges, such as stagnant revenue growth and limited visibility into customer preferences and competitive market trends.

This project aims to tackle these challenges by developing an advanced data pipeline capable of processing vast amounts of both structured and unstructured data. By analyzing this data, the company can uncover valuable insights into customer behavior, identify trends in healthcare needs, and pinpoint high-value customers. Additionally, this solution will enable the company to create personalized insurance products and tailored offerings that meet the specific needs of different customer segments, fostering greater customer satisfaction and loyalty.

In summary, the implementation of this data pipeline will not only help the company optimize its pricing strategies and identify new revenue opportunities, but it will also provide the tools needed to make data-driven decisions, ultimately leading to sustained growth and a stronger competitive position in the healthcare insurance market.

The solution for the data pipeline will be in following steps:

### 1.1 Data Ingestion:

The first step involves collecting data related to insurance claims, subscribers, their associated groups and subgroups, patient information, and hospital details. This data has been previously gathered through web scraping and third-party data providers by other software teams. The data must first be identified, analysed, and then designated as the raw data source for the data pipeline. Once the raw data has been assessed, it will be uploaded to AWS S3, which will serve as the data lake for this project. AWS S3 will provide a scalable and secure storage solution, allowing for the efficient handling of large volumes of data.

### 1.2 Data Wrangling:

Raw data often contains issues such as incorrect, incomplete, and duplicate records, as well as unsupported or inconsistent column names. The data wrangling process will address these problems through a series of PySpark transformations. These transformations will include removing duplicate records, filling missing values with suitable placeholders (e.g., "NA"), and renaming columns to adhere to a standardized naming convention. The goal is to ensure that the data is clean, consistent, and ready for further processing. Data wrangling will help streamline the data, making it easier to perform accurate analysis and transformations later in the pipeline.

### 1.3 Data Transformations:

Once the data has been cleaned, the next step is to perform the necessary transformations to generate the final output datasets. The transformations required will be based on the specifications outlined in the Requirements Specification Document. These transformations may include joining different datasets, calculating aggregate values, grouping by specific key columns, and filtering out irrelevant data.

For example, consider the requirement to calculate the "Average Monthly Premium" for subscribers. Upon reviewing the raw datasets, we find that subscriber information is stored in the subscriber.csv file, while the monthly premium associated with each subgroup is stored in the subgroup.csv file. Each subscriber is categorized under a particular subgroup, such as "Deficiency Diseases" or "Accidents."

To achieve the desired transformation, we will follow these steps:

Dataset Join: Join the subscriber and subgroup datasets based on the common key column subgroup\_id. This will allow us to assign the appropriate monthly premium to each subscriber, based on the subgroup they belong to.

Aggregation: After assigning the monthly premium to each subscriber, group the data by subscriber\_id and calculate the average monthly premium for each subscriber.

This approach will ensure that the transformed data aligns with the specific business requirements and provides meaningful insights, such as the average cost incurred by subscribers for their insurance coverage.

These transformations will be applied to other use cases in the pipeline as well, following similar logic to achieve the desired outcomes for all requirements.

1.4. Data Loading:

The final step in the data pipeline process involves loading the refined dataset, which has undergone cleaning and necessary transformations, into Amazon Redshift. This structured data will be organized into well-defined tables, making it easily accessible for querying and analysis.

Amazon Redshift, acting as the data warehouse, will store this processed data, ensuring high performance and scalability for large-scale analytics. Access to the data in Redshift will be provided on a need-to-know basis to various stakeholders, including executives, data analysts, data scientists, marketing team members, and finance personnel. These stakeholders will be able to utilize the data to perform detailed analyses, generate insights, and drive informed decision-making across different departments of the organization.

By providing this level of access, the company ensures that each team can harness the power of the data for their specific needs, whether it's for strategic planning, marketing campaigns, financial forecasting, or customer engagement initiatives.

# Use Cases

#### **2.1** Executives

**Use Case:** Dashboard and Report Review for Strategic Decision-Making  
**Description:** Executives will leverage dashboards and reports generated from the data pipeline to monitor and evaluate key performance indicators (KPIs) such as revenue growth, customer retention rates, and competitive positioning. These insights will support their strategic decision-making, ensuring that decisions are aligned with the company’s long-term objectives and growth plans. The data provided will help executives assess overall business performance and steer the company’s strategic direction effectively.

#### **2.2 Data Analysts**

**Use Case:** Data Extraction and Analysis for Operational Insights  
**Description:** Data analysts will interact with the data pipeline to extract and transform data from various sources. They will conduct analyses to identify critical metrics such as the top diseases by claim volume, the most profitable insurance groups, and other essential indicators. The insights derived from these analyses will be utilized to guide marketing strategies, sales efforts, and product development initiatives. Data analysts will ensure that the information provided is accurate and actionable for the other departments.

#### **2.3 Data Scientists**

**Use Case:** Model Training and Predictive Analysis  
**Description:** Data scientists will utilize the data pipeline to access and analyze data for building predictive models. These models will be used to forecast customer behavior, assess risk levels, and refine pricing strategies. Additionally, data scientists will leverage the data to develop personalized insurance products and loyalty programs that cater to specific customer segments. The goal is to enhance customer engagement and optimize insurance offerings based on predictive insights.

#### **2.4 Marketing Teams**

**Use Case:** Personalized Marketing Campaigns and Strategy Development  
**Description:** Marketing teams will use insights from the data pipeline to design and implement personalized marketing campaigns. For instance, they might identify young subscribers under the age of 30 and create targeted offers for this demographic. Additionally, they will analyze regions with high claim volumes to tailor insurance packages accordingly. The data will also reveal preferences for different types of insurance (e.g., government vs. private), enabling the marketing team to adjust strategies and effectively target their audience.

#### **2.5 Finance Teams**

**Use Case:** Profitability Tracking and Pricing Optimization  
**Description:** Finance teams will leverage the data pipeline to monitor profitability metrics, such as identifying the most profitable insurance groups and analyzing trends in average monthly premiums. This information will assist in adjusting pricing models, optimizing revenue streams, and making informed budgetary recommendations to the executive team. The financial insights provided will be crucial for strategic financial planning and resource allocation.

#### **2.6 Customer Service Teams**

**Use Case:** Enhanced Customer Service and Support  
**Description:** Customer service teams will access the data pipeline to quickly retrieve comprehensive information about a customer’s insurance plan, claim history, and previous interactions with the company. This capability will enable them to offer tailored solutions, address customer issues more efficiently, and improve overall service quality. By having detailed customer data at their fingertips, customer service representatives can enhance the customer experience and resolve inquiries promptly.

# Database Design

### 3.1. Tables Metadata

#### 3.1.1 Source Tables

##### claims

* **Primary Key:** claim\_id
* **Foreign Keys:** sub\_id (references subscriber.sub\_id), patient\_id (references patient\_records.patient\_id)
* **Columns:**
  + claim\_id (INTEGER) - Unique identifier for the claim
  + patient\_id (INTEGER) - Identifier for the patient
  + disease\_name (VARCHAR) - Name of the disease
  + sub\_id (VARCHAR) - Identifier for the subscriber
  + claim\_or\_rejected (VARCHAR) - Status of the claim (e.g., Yes, No, Not available)
  + claim\_type (VARCHAR) - Type of the claim (e.g. claims of policy, claims of fact, claims of value)
  + claim\_amount (FLOAT) - Amount of the claim
  + claim\_date (DATE) - Date of the claim

##### subscriber

* **Primary Key:** sub\_id
* **Foreign Keys:** subgrp\_id (references subgroup.subgrp\_id)
* **Columns:**
  + sub\_id (VARCHAR) - Unique identifier for the subscriber
  + first\_name (VARCHAR) - Subscriber's first name
  + last\_name (VARCHAR) - Subscriber's last name
  + street (VARCHAR) - Street address
  + birth\_date (DATE) - Date of birth
  + gender (VARCHAR) - Gender
  + phone (VARCHAR) - Phone number
  + country (VARCHAR) - Country
  + city (VARCHAR) - City
  + zip\_code (VARCHAR) - Zip code
  + subgrp\_id (VARCHAR) - Identifier for the subgroup
  + elig\_ind (VARCHAR) - Eligibility indicator (Eligible: Yes or No)
  + eff\_date (DATE) - Effective date
  + term\_date (DATE) - Termination date

##### group

* **Primary Key:** grp\_id
* **Columns:**
  + country (VARCHAR) - Country where the group operates
  + premium\_written (FLOAT) - Total premium written
  + zipcode (VARCHAR) - Zip code for the group
  + grp\_id (INTEGER) - Unique identifier for the group
  + grp\_name (VARCHAR) - Name of the group
  + grp\_type (VARCHAR) - Type of the group
  + city (VARCHAR) - City
  + year (INTEGER) - Year of the data

##### subgroup

* **Primary Key:** subgrp\_id
* **Columns:**
  + subgrp\_id (VARCHAR) - Unique identifier for the subgroup
  + subgrp\_name (VARCHAR) - Name of the subgroup
  + monthly\_premium (FLOAT) - Monthly premium for the subgroup

##### grpsubgrp

* **Primary Key:** Composite Key (subgrp\_id, grp\_id)
* **Foreign Keys:** subgrp\_id (references subgroup.subgrp\_id), grp\_id (references group.grp\_id)
* **Columns:**
  + subgrp\_id (VARCHAR) - Identifier for the subgroup
  + grp\_id (INTEGER) - Identifier for the group

##### patient\_records

* **Primary Key:** patient\_id
* **Foreign Keys:** disease\_name (references disease.disease\_name), hospital\_id (references hospital.hospital\_id)
* **Columns:**
  + patient\_id (INTEGER) - Unique identifier for the patient
  + patient\_name (VARCHAR) - Name of the patient
  + patient\_gender (VARCHAR) - Gender of the patient
  + patient\_birth\_date (DATE) - Date of birth
  + patient\_phone (VARCHAR) - Phone number
  + disease\_name (VARCHAR) - Name of the disease
  + city (VARCHAR) - City
  + hospital\_id (INTEGER) - Identifier for the hospital

##### disease

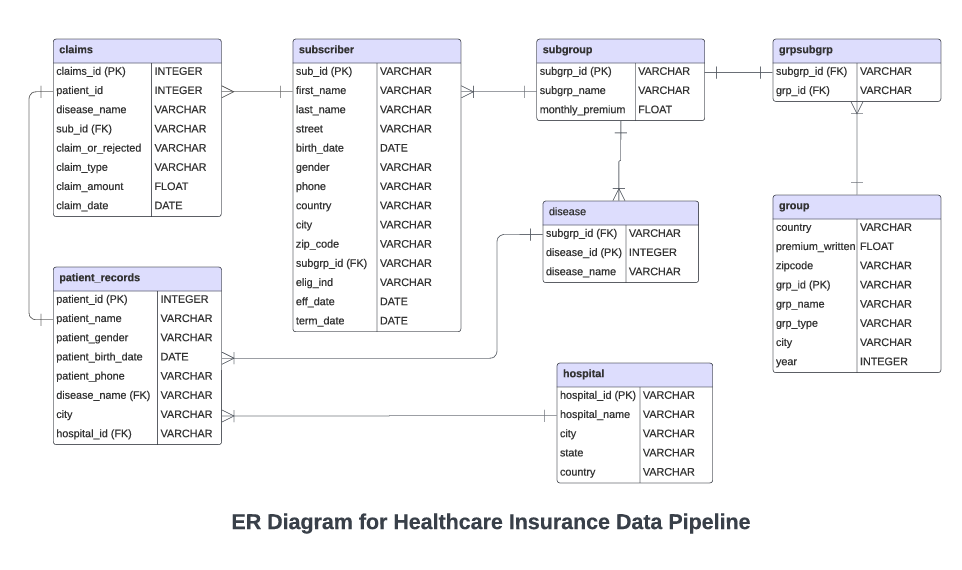
* **Primary Key:** disease\_id
* **Foreign Keys:** subgrp\_id (references subgroup.subgrp\_id)
* **Columns:**
  + subgrp\_id (VARCHAR) - Identifier for the subgroup
  + disease\_id (INTEGER) - Unique identifier for the disease
  + disease\_name (VARCHAR) - Name of the disease

##### hospital

* **Primary Key:** hospital\_id
* **Columns:**
  + hospital\_id (INTEGER) - Unique identifier for the hospital
  + hospital\_name (VARCHAR) - Name of the hospital
  + city (VARCHAR) - City
  + state (VARCHAR) - State
  + country (VARCHAR) – Country

#### 3.1.2 ER Diagram

Below ER Diagram shows the relationship between all the raw tables.



#### 3.1.3 Output Tables

These tables will be derived from the source tables to meet the specified requirements.

##### disease\_claim\_count

* **Columns:**
  + disease\_name (VARCHAR) - Name of the disease
  + claim\_count (INTEGER) - Number of claims for the disease

##### subscribers\_age

* **Columns:**
  + sub\_id (VARCHAR) - Subscriber ID
  + first\_name (VARCHAR) - Subscriber's first name
  + last\_name (VARCHAR) - Subscriber's last name
  + age (INTEGER) – Subscriber’s age

##### group\_subgroup\_count

* **Columns:**
  + grp\_id (INTEGER) - Group ID
  + grp\_name (VARCHAR) - Group name
  + subgroup\_count (INTEGER) - Number of subgroups

##### hospital\_patient\_count

* **Columns:**
  + hospital\_id (INTEGER) - Hospital ID
  + hospital\_name (VARCHAR) - Hospital name
  + patient\_count (INTEGER) - Number of patients served

##### subgroup\_subscription\_count

* **Columns:**
  + subgrp\_name (VARCHAR) - Name of the subgroup
  + subscription\_count (INTEGER) - Number of times subscribed

##### city\_claim\_count

* **Columns:**
  + city (VARCHAR) - City
  + claim\_count (INTEGER) - Number of claims from the city

##### group\_subscriber\_count

* **Columns:**
  + grp\_type (VARCHAR) – Subscriber’s group type (e.g., Government, Private)
  + subscriber\_count (INTEGER) - Number of subscribers

##### group\_premium

* **Columns:**
  + grp\_id (INTEGER) - Group ID
  + grp\_name (VARCHAR) - Group name
  + premium (FLOAT) – Premium paid by the subscriber under the group

##### patients\_disease

* **Columns:**
  + patient\_id (INTEGER) - Patient ID
  + patient\_name (VARCHAR) - Patient name
  + patient\_birth\_date (DATE) – Patient date of birth
  + disease\_name (VARCHAR) – Disease of the patient

##### patients\_claims

* **Columns:**
  + patient\_id (INTEGER) - Patient ID
  + patient\_name (VARCHAR) - Patient name
  + disease\_name (VARCHAR) – Patient’s disease
  + sub\_id (VARCHAR) – Subscriber’s id for the patient
  + claim\_id (INTEGER) – Claim ID
  + claim\_type (VARCHAR) – Claim type
  + claim\_amount (FLOAT) – Claim amount
  + claim\_date (DATE) – Claim date

##### insurance\_claims\_metrics

* **Columns:**
  + metrics\_id (INTEGER) – Metrics ID

##### metrics\_name (VARCHAR) – metric\_name e.g. **total\_rejected\_claims\_count**, **subscriber\_ average\_monthly\_premium**

* + metrics\_desc (VARCHAR) – Description of metrics
  + metrics\_value (INTEGER) - Patient name

# Technologies and Platforms

###### AWS S3 (Amazon Simple Storage Service)

* **Purpose**: Acts as the data lake for storing raw data.
* **Features**:
  + **Scalability**: Can handle large volumes of data with auto-scaling.
  + **Durability**: Ensures high durability with multiple redundancy options.
  + **Security**: Offers encryption options for data at rest and in transit.
  + **Data Access**: Provides APIs and SDKs for easy data upload and retrieval.

###### PySpark

* **Purpose**: Performs data wrangling to clean and preprocess raw data.
* **Features**:
  + **DataFrame API**: Allows for high-level data manipulation and processing.
  + **Transformations**: Supports operations such as removing duplicates, filling null values, and renaming columns.
  + **Fault Tolerance**: Ensures reliability with RDD lineage and checkpointing.
  + **Integration**: Works well with large datasets and integrates seamlessly with other big data tools.

###### Databricks

* **Purpose**: Provides an integrated environment for running PySpark transformations and performing complex data processing.
* **Features**:
  + **Collaborative Workspace**: Offers notebooks and dashboards for collaborative data analysis.
  + **Optimized Runtime**: Features optimized Apache Spark runtime for better performance.
  + **Integration**: Integrates with AWS S3 for accessing and processing data stored in the data lake.
  + **Scalability**: Scales resources dynamically based on workload requirements.

###### Amazon Redshift

* **Purpose**: Serves as the data warehouse for storing and querying the refined data.
* **Features**:
  + **Columnar Storage**: Uses columnar storage format to optimize query performance.
  + **Scalability**: Provides high scalability and performance for large-scale analytics.
  + **SQL Interface**: Offers a SQL-based interface for querying and analysis.
  + **Integration**: Easily integrates with various ETL tools and data sources.

**Supporting Tools**

###### Jira

* **Purpose**: Project management and tracking.
* **Features**:
  + **Issue Tracking**: Manages tasks, bugs, and project progress.
  + **Agile Boards**: Supports Scrum and Kanban methodologies.
  + **Reporting**: Provides dashboards and reports to track project metrics and status.

###### GitHub

* **Purpose**: Code version control and collaboration.
* **Features**:
  + **Repository Management**: Hosts and manages code repositories.
  + **Pull Requests**: Facilitates code reviews and collaboration.
  + **Issue Tracking**: Manages issues and bugs within the codebase.