Solution Design Document

## 1. Solution

Step by step:

1. **Configure S3 Access**
   * Store AWS credentials in Databricks Secrets.
   * In your notebook, retrieve and set fs.s3a.access.key and fs.s3a.secret.key via spark.sparkContext.\_jsc.hadoopConfiguration().
2. **Load Raw Data**
   * Read CSV/JSON files from S3 into Spark DataFrames (e.g., spark.read.csv, spark.read.json).
3. **Clean & Prepare Data**
   * Drop duplicates: .dropDuplicates()
   * Handle nulls: .na.drop() or .na.fill()
   * Cast columns to proper types: .withColumn(... .cast(...))
   * Rename columns to snake\_case for consistency.
4. **Compute Analytics**
   * **Use Case 1:** Group claims\_clean by disease\_name, count claims, sort descending, select top.
   * **Use Case 2:** Group patient\_clean by hospital\_id, count distinct patient\_id, sort descending, join hospital\_clean for names, select top.
5. **Write Results**
   * Write each result DataFrame as Parquet to S3: .write.mode("overwrite").parquet("s3a://.../ucXX").
6. **(Optional) Load into Redshift**
   * Create external tables on S3 Parquet or COPY into Redshift internal tables for BI consumption.

## 2. Use Cases

* **UC1:** Identify the disease with the highest number of claims.
* **UC2:** Identify the hospital serving the most distinct patients.

## 3. Database Design (Amazon S3)

### 3.1 disease

* **Description:** Reference list of diseases.
* **Columns:**
  + disease\_id INT PRIMARY KEY
  + disease\_name VARCHAR NOT NULL
  + subgroup\_id VARCHAR

### 3.2 claims

* **Description:** Insurance claim records.
* **Columns:**
  + claim\_id BIGINT PRIMARY KEY
  + disease\_name VARCHAR NOT NULL
  + patient\_id BIGINT NOT NULL
  + claim\_amount INT NOT NULL
  + claim\_date DATE
  + claim\_rejected\_flag BOOLEAN
* **FK:** (disease\_name) → disease(disease\_name)

### 3.3 hospital

* **Description:** Hospital master data.
* **Columns:**
  + hospital\_id VARCHAR PRIMARY KEY
  + hospital\_name VARCHAR
  + city VARCHAR
  + state VARCHAR
  + country VARCHAR

### 3.4 patient\_records

* **Description:** Patient admission details.
* **Columns:**
  + patient\_id BIGINT PRIMARY KEY
  + patient\_name VARCHAR
  + patient\_gender VARCHAR
  + patient\_birth\_date DATE
  + patient\_phone VARCHAR
  + disease\_name VARCHAR
  + city VARCHAR
  + hospital\_id VARCHAR
* **FKs:**
  + hospital\_id → hospital(hospital\_id)
  + disease\_name → disease(disease\_name)

### 3.5 ER Diagram (Optional)

A simple ER diagram illustrating relationships:

[disease] 1───⋆ [claims]  
[disease] 1───⋆ [patient\_records]  
[hospital] 1───⋆ [patient\_records]

## 4. Technologies & Platforms

* **Apache Spark (PySpark)** on Databricks for ETL and analytics.
* **AWS S3** for raw data storage and Parquet output.
* **Amazon Redshift** for data warehousing.
* **Databricks Notebooks** for interactive development and testing.
* **AWS IAM & Databricks Secrets** for secure credential management.
* **Git/GitHub** for version control of code and notebooks.
* **Databricks Jobs** or **Airflow** for scheduling and orchestration.