**1. Finding and Analyzing Data**

Chosen Dataset:

Name: Healthcare Dataset

Source: [Kaggle Healthcare Dataset](https://www.kaggle.com/datasets/prasad22/healthcare-dataset)

Description: This chosen dataset contains comprehensive healthcare records with details about patients, their medical conditions, treatment, and hospital information.

1.1 Analyze the Dataset

Technical Observations

Schema:

Columns:

Name (String) Insurance Provider (String)

Age (Integer) Billing Amount (Float)

Gender (String) Room Number (Integer)

Blood Type (String) Admission Type (String)

Medical Condition (String) Discharge Date (Date)

Date of Admission (Date) Medication (String)

Doctor (String) Test Results (String)

Hospital (String)

Data Quality:

Redundant Values: I checked for redundant values in column ‘Name’.

Inconsistencies: Standardized the format of names and dates and ensure consistency in categorical data like Gender and Blood Type.

Non-Technical Observations

Purpose: To analyze patient records to gain insights into medical conditions, hospital performance, statistics on insurance providers and treatment effectiveness.

Insights: Potential insights include the distribution of medical conditions, average billing amounts, common medications, and patient demographics.

Potential Improvements

Data Cleaning: Handling missing values and treat outliers.

Data Enrichment: The outcome could have been more productive if the data had more contextually relevant information such as geographical data and additional patient demographics.

2. Architectural Diagram

A diagram of a computer software

Description automatically generated

3. Data Pipeline Creation

3.1 Data Ingestion

Method: Created an HTTP linked service in ADF to ingest data from GitHub to ADLS.

Implementation: I utilized a Copy Activity in ADF to transfer data from GitHub to ADLS.

3.2 Data Transformation (Azure Databricks)

Process: Firstly, I link ADLS to an Azure Databricks notebook.

Transformations done in the notebook on the personal compute cluster:

Drop unnecessary columns (e.g., Room Number, Admission Type).

Standardize names (e.g., convert all names to proper case).

Added IDs or rearranged column positions for consistency.

Storage Format: Saved all the transformed data to ADLS in Parquet format for efficient storage.

3.3 Data Validation and Backup

Validation: Then, I added a Metadata Activity in the pipeline to check for file existence.

Debugging: Also implemented breakpoints for debugging.

Backup: Moreover, I used an If Activity to copy processed files to a backup container.

4. Loading and Analysis in Synapse Analytics and Visualization

Loading Data: After creating a backup in the previous step, I then loaded the transformed data into Synapse Analytics and created an external table for querying.

Analysis: Performed analysis using SQL queries. Some of the example queries include:

Count of patients with normal test results.

Number of patients per hospital.

Highest insurance provided by the insurance provider.

Visualization in Power BI

Connection: Then, I connected Synapse Analytics (using serverless SQL pool) endpoints to Power BI.

Data Transformation: Performed data type checks and transformations in Power Query.

Visualizations: Used bar charts, slicers, and add new columns (e.g., age group) in Power Query to visualize insights.

5. Conclusion

Advantages:

Streamlined Data Processing: My data pipeline efficiently automates the process of data ingestion, transformation, and storage, reducing manual effort and the risk of errors.

Improved Data Quality: Through standardization and removal of redundant columns, the overall quality of the data has been significantly enhanced.

Scalability: Utilizing Azure services like Azure Data Factory, Azure Databricks, and Synapse Analytics ensures that the pipeline can handle large volumes of data and scale as needed.

Efficient Storage and Querying: Storing transformed data in Parquet format within Azure Data Lake Storage and querying via Synapse Analytics ensures efficient data storage and fast query performance.

Enhanced Insights: The end-to-end pipeline enables comprehensive analysis and visualization of healthcare data, providing valuable insights for decision-making.

Improvements:

Data Cleaning: The data cleaning process can be further improved by implementing more advanced techniques for outlier detection and imputation of missing values.

Data Enrichment: The dataset could be enriched with additional contextually relevant information, such as geographical data and additional patient demographics to provide deeper insights.

User-Friendly Visualizations: Further refinement and additional problem statements can we added to create multiple reports in Power BI dashboards to include more interactive elements and user-friendly features can enhance the usability and accessibility of the insights.

Benefits:

Improved Decision-Making: The insights gained from the analysis can help healthcare providers and administrators make informed decisions about patient care, resource allocation, and operational improvements.

Cost Savings: By automating data processing and leveraging cloud-based services, an organization can save costs associated with manual data handling and on-premises infrastructure.