NSBM Green University
Faculty of Computing
BSc (Hons) Data Science

DS403.3- Big Data Programming

Final Report

Health Insurance Analytics System Using Lambda Architecture on Microsoft Azure

(A Scalable Solution for Real-Time and Batch Data Processing in the Healthcare Sector)

Group - 02

Student ID	Student Name
24490	MRK Karunathilaka
24572	MKIM Rohana
24614	GAAS Ganegoda

Module Lecturer: Mr. Adhil Rushdy

Table of Contents

1. Introduction & Objectives

The health insurance sector in Sri Lanka faces challenges in analyzing large volumes of data, which impacts fraud detection, pricing, and operational efficiency. This project aims to address these issues by developing a **Health Insurance Analytics System** using **Lambda Architecture** on **Microsoft Azure**, enabling the processing of both historical and real-time claims data to provide enhanced insights and support informed decision-making.

The primary objectives of the project are:

- To design and implement a scalable analytics system capable of handling over one million historical records and real-time data streams from diverse healthcare sources.
- To integrate and process data from Medicare and Medicaid datasets, such as drug spending, durable medical equipment usage, and hospital claims, to generate comprehensive healthcare cost insights.
- To provide healthcare providers with detailed, region-specific insights into treatment patterns, drug prescription trends, and medical equipment referrals.
- To develop an interactive, user-friendly dashboard that visualizes key metrics (e.g., spending by drug, equipment reimbursement, hospital costs) for stakeholders such as insurers, healthcare providers, and policymakers.
- To ensure cost-effective and scalable processing by leveraging Azure services such as Azure Data Lake, Azure Synapse Analytics, Azure Stream Analytics, and Power BI.

2. Data Sources

This project uses data from the **Centers for Medicare & Medicaid Services (CMS)**, a U.S. government agency that manages health programs like Medicare and Medicaid. CMS collects detailed information about healthcare services and spending, which helps researchers understand how money is used in healthcare.

1. Medicaid Spending by Drug (2018–2022): Medicaid Spending by Drug

This dataset shows how much Medicaid spends on outpatient drugs prescribed to patients. It includes the average cost per dosage, spending trends over the years, drug manufacturers, and information about drug uses. The spending data reflects total reimbursements to pharmacies, including fees but excludes rebates.

2. Medicare Durable Medical Equipment, Devices & Supplies by Referring Provider (2018–2022): *Medicare DME Data Dictionary*

This dataset contains information about Medicare spending on durable medical equipment (like wheelchairs and oxygen devices) by the healthcare providers who referred these items. It helps understand the types of equipment prescribed and how much Medicare reimburses for them.

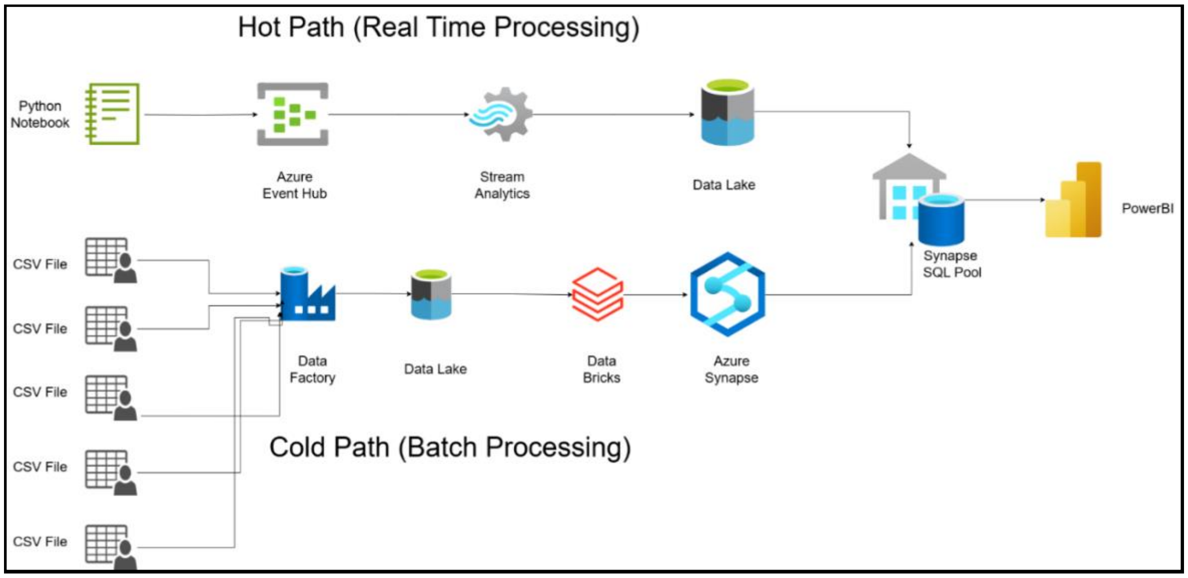
3. Medicare Hospital Spending by Claim (2018–2022): *Medicare Hospital Spending by Claim*

This dataset provides detailed data on Medicare spending for hospital claims. It includes information on services provided, payment amounts, hospital details, helping to analyze hospital costs and spending patterns, etc.

3. Architecture

Used Architecture → Lambda Architecture

Lambda Architecture is employed in this project to efficiently handle both historical (batch) and real-time healthcare data, enabling a robust and scalable analytics system.



Lambda Architecture Main Components,

- **Batch Layer** → Responsible for processing large volumes of historical data in batches to generate accurate views over long time periods.
- **Speed Layer** → Handles real-time or near real-time data streams to provide low-latency updates and immediate insights.
- **Serving Layer** → Combines outputs from both the batch and speed layers to deliver a unified and complete view of the data to end users.

Additional Layer: Presentation Layer

- A **Presentation Layer** is incorporated to transform the processed data into actionable insights for end users via interactive dashboards and visualizations.

4. Technologies and Services Used

For Batch Processing:

- **Azure Data Factory** – For orchestrating and automating data movement and transformation pipelines.
- **Azure Data Lake Storage** – Used as a centralized repository for storing raw and processed batch data.
- **Azure Databricks** – For scalable data processing, cleansing, and transformation using Apache Spark.
- **Azure Synapse Analytics** – For querying and analyzing large datasets, combining both batch and structured data processing.

For Real-Time Processing:

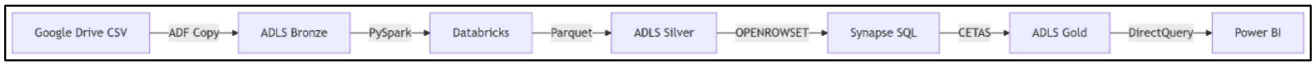
- **Azure Event Hub** – Ingests real-time data streams, such as incoming claims or updates from healthcare systems.
- **Azure Stream Analytics** – Processes and analyzes data in motion with low latency.
- **Azure Data Lake Storage** – Stores intermediate and processed real-time data for downstream analysis.
- **Azure Synapse Analytics** – Integrates and queries both real-time and historical data for unified analytics.

For Data Representation:

- **Microsoft Power BI** – Used to create interactive dashboards and visual reports for stakeholders, enabling drill-downs, trend analysis, and decision support.

5. Batch Layer

The **Batch Layer** in our Lambda Architecture is responsible for processing historical data at scale. This layer follows a three-step pipeline that moves raw data through transformation and structuring phases to make it analytics-ready.



Step 01: Transferring CSV Files to the Data Lake (Bronze Container)

- A **dynamic pipeline** was implemented in Azure Data Factory (ADF) to ingest and organize CSV files into the **Bronze Layer** (raw data zone) of Azure Data Lake Storage.
- The pipeline included the following activities:
  - **Lookup Activity** – Reads values from a JSON array to determine the folders/files to process.
  - **ForEach Activity** – Iterates through the array items.
  - **Copy Activity** – Transfers CSV files into corresponding folder paths in the Bronze container based on folder names.
- **References:**
  - JSON File for Lookup Activity
  - Dynamic Pipeline Copy Activity

Step 02: Transforming Data and Loading into the Silver Container

- A **single-node Azure Databricks cluster** was used to transform the raw data. The cluster was configured with **auto-termination** after 10 minutes of inactivity to minimize cost.
- A **Python notebook** was developed to:
  - Clean and preprocess the data
  - Handle missing values, formatting, and schema standardization
  - Load the transformed data into the **Silver Layer** (processed data zone) of Azure Data Lake Storage

- **References:**
  - Silver Layer(Data_transformation) Notebook
  - Dynamic Batch Processing Pipeline

Step 03: Creating Views and External Tables in Synapse (Gold Layer)

- A dedicated **"gold" schema** was created in **Azure Synapse Analytics** to represent curated and query-ready datasets.
- For each dataset in the **Silver Layer**, we created **views** using the OPENROWSET function to read data directly from Parquet files.

Example:

```
OPENROWSET(

   BULK 'https://mylambdadatalake.blob.core.windows.net/silver/Claims/',

   FORMAT = 'PARQUET'

) AS View1Claims
```

- External tables were then created from these views to persist data in the **Gold Layer**. This allowed Power BI to access optimized, structured datasets for reporting.

- **Steps to Create External Tables:**

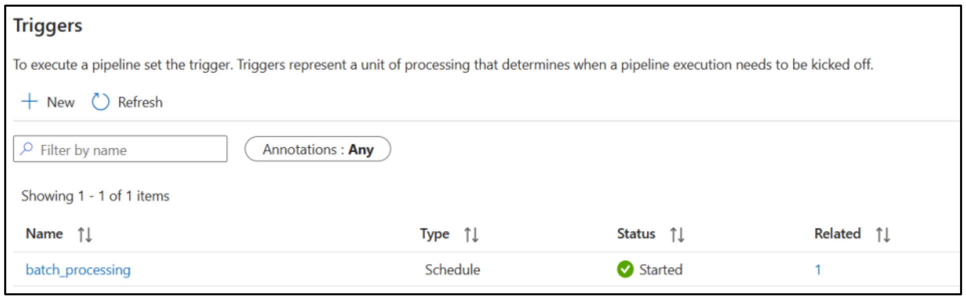
1. Create **Database Scoped Credentials**
2. Define **External Data Sources**
3. Set up an **External File Format** for Parquet files
4. Use views as the source for creating **external tables**

- **References:**

    - Create Gold Schema
    - Create Views for each dataset
    - Create External Tables

Automation:

- A **scheduled trigger** was configured in Azure Data Factory to **run the batch processing pipeline daily**, ensuring the system is regularly updated with new data.


Triggers

To execute a pipeline set the trigger. Triggers represent a unit of processing that determines when a pipeline execution needs to be kicked off.

Showing 1 - 1 of 1 items

Name	Type	Status	Related
batch_processing	Schedule	✓ Started	1

6. Speed Layer

The **Speed Layer** of the Lambda Architecture enables real-time processing of Medicare claims data, with a batch size of 20 records every 60 seconds. This layer utilizes Azure services to ingest, process, store, and integrate streaming data with batch-processed data. The goal is to deliver low-latency analytics to stakeholders including insurance companies, healthcare providers, and government agencies.

Step 01: Synthetic Data Generation

- A Python notebook was developed to simulate real-time Medicare claim submissions by generating synthetic claims data in **JSON** format every 2 seconds.
- The generated data includes fields such as ClaimID and ClaimAmount.
- The notebook uses the **Azure Event Hubs SDK** to send events into **Azure Event Hubs**, ensuring a continuous data stream for testing.
- This synthetic approach allows for flexible testing without relying on live data sources, though **rate control** was essential to manage high-frequency data generation efficiently.
- **References:**
    - Synthetic Data Generation

Step 02: Real-Time Data Ingestion with Azure Event Hubs

- **Azure Event Hubs** acts as the ingestion layer, receiving JSON-formatted claim data.
- The Event Hub is configured within the lambda-claims-eventhub namespace in the lambda-arch-rg resource group.
- It uses **4 partitions** to support parallel data streams, ensuring high throughput and low latency.
- **Checkpointing** was enabled to maintain reliable delivery and handle potential network interruptions, especially under high data velocity conditions.

Step 03: Stream Processing with Azure Stream Analytics

- The **Azure Stream Analytics job** (claims-stream-job) connects to the Event Hub to process incoming records.
- It filters out null or invalid values and ensures schema consistency before writing to the data lake.
- Cleaned records are stored in the **Silver Layer** of **Azure Data Lake Storage Gen2** in **Parquet format**.
- The job was scaled to **3 Streaming Units** to address processing bottlenecks.
- Schema validation was implemented in the source notebook to guarantee that data adheres to expected structures.

Step 04: Data Storage in Azure Data Lake (Silver Container)

- Processed real-time data is saved in the **silver container**, organized by **date and hour** for optimized access.
- Using **Parquet format** reduces storage overhead and enhances query performance.
- Data is **batched and written every 60 seconds**, minimizing file fragmentation and improving downstream processing efficiency.
- Partitioning strategy was carefully designed to balance storage and performance, though query optimization required iterative tuning.

Step 05: Real-Time Data Access with Azure Synapse Analytics

- **Azure Synapse Analytics Serverless SQL Pool** provides real-time access to data in the silver container.
- A **Synapse view** is created using OPENROWSET to query Parquet files stored in date/hour partitions.
- This setup enables efficient querying of streaming data and integrates seamlessly with batch data pipelines.
- Cost control was managed through optimized partitioning and by minimizing the amount of scanned data during query execution.

Step 06: Merging Batch and Real-Time Data

- Real-time data (from the Synapse view) is merged with historical batch data (from the gold container) into a unified **external table** in Synapse Analytics.
- **Deduplication** based on ClaimID and SubmissionTime ensures no duplicate entries exist in the final output.
- Schema alignment between batch and real-time datasets was carefully managed to prevent integration issues.
- The resulting unified dataset provides a comprehensive view that supports both **historical analysis** and **real-time insights**, enhancing the analytical capabilities of the overall system.

7. Serving Layer

The **Serving Layer** acts as the unified access point for both historical (batch) and real-time data. This layer delivers integrated, queryable datasets to downstream services such as Power BI for analytics and visualization.

Step 01: Creating Views Using **OPENROWSET()**

- In this step, we created **views** for both **real-time** and **batch** datasets using the OPENROWSET() function in **Azure Synapse Analytics**.
- These views serve as logical representations of the Parquet files stored in the Silver Layer for both streaming and batch data.
- **References:**
    - View for the real-time data in Medicare dataset
    - View for the batch data in Medicare dataset

This separation of views allows us to manage and validate each data stream independently before merging them.

Step 02: Creating an External Table for Unified Access

- After defining the views, we created an **external table** to combine both real-time and batch data sources using a UNION operation.
- This external table is stored in the **Gold Layer** and is used for querying and reporting via Power BI dashboards.

```
CREATE EXTERNAL TABLE gold.ext_medicare_ds
WITH
(
   LOCATION = 'ext_medicare_ds',
   DATA_SOURCE = source_gold,
   FILE_FORMAT = format_parquet
)
AS
SELECT * FROM gold.Medicare_DME_DS
UNION
SELECT * FROM gold.RealTime_Medicare_DME_DS;
```

- This unified external table allows downstream systems to access a **comprehensive view** of Medicare Durable Medical Equipment data, regardless of whether it originated from historical or real-time pipelines.

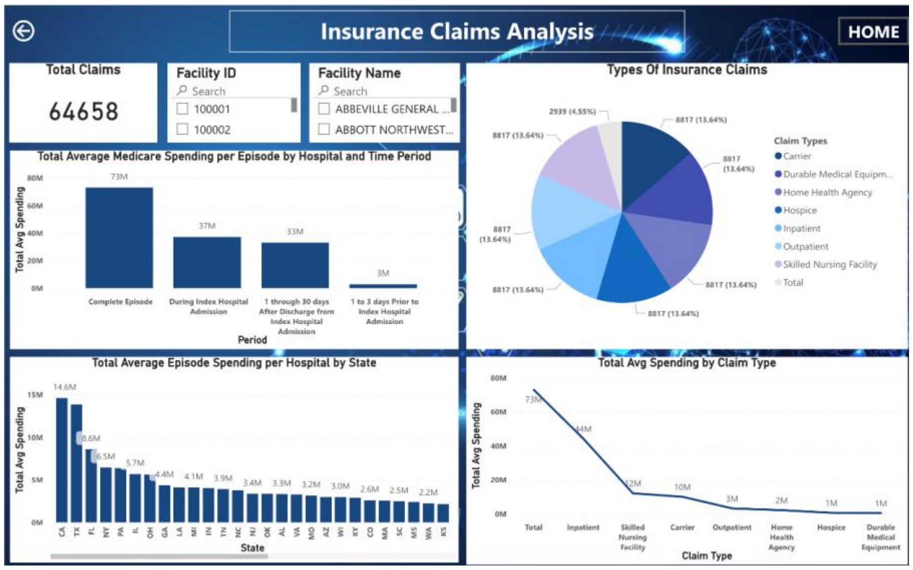
8. Presentation Layer: Power BI Dashboards

Our analysis is presented through interactive Power BI dashboards, connected to the cleaned data in the **gold** layer via Azure Synapse Analytics. These dashboards offer visual insights into insurance, drug, and Medicare claims.

Home Page

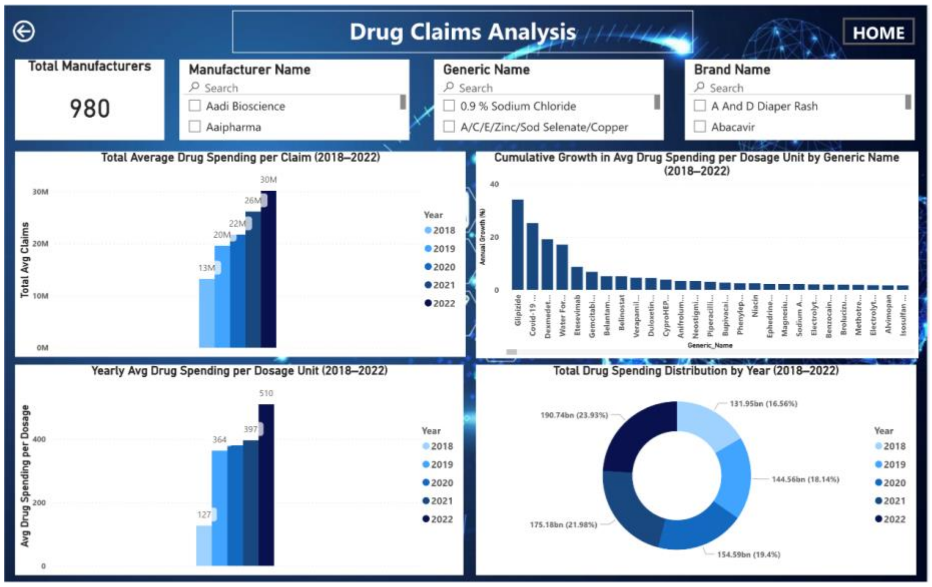
The Home Page provides easy access to the three main analysis areas: Insurance Claims, Drug Claims, and Medicare Claims.

Insurance Claims Analysis


Insurance Claims Analysis
HOME
Total Claims
64658
Facility ID
Search
100001
100002
Facility Name
Search
ABBEVILLE GENERAL ...
ABBOTT NORTHWEST...
Types Of Insurance Claims
Claim Types
Carrier
Durable Medical Equipm...
Home Health Agency
Hospice
Inpatient
Outpatient
Skilled Nursing Facility
Total
Total Average Medicare Spending per Episode by Hospital and Time Period
Total Avg Spending
Period
Complete Episode
During Index Hospital Admission
1 through 30 days After Discharge from Index Hospital Admission
1 to 3 days Prior to Index Hospital Admission
Total Average Episode Spending per Hospital by State
State
Total Avg Spending by Claim Type
Claim Type

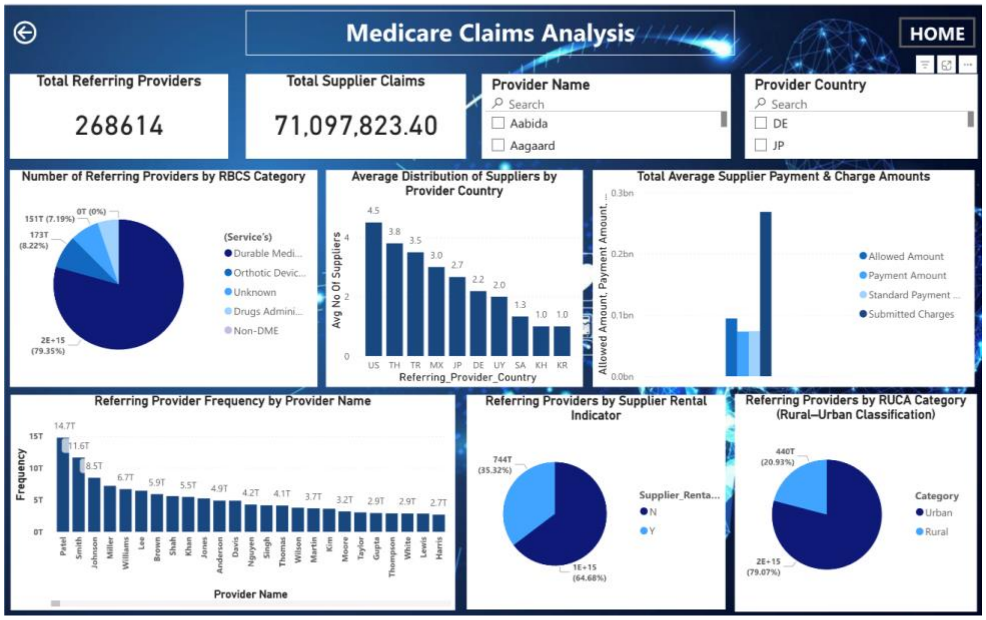
This dashboard shows key metrics for insurance claims, including total claims, claim types, average spending per hospital stay phase, spending by state, and spending trends by claim type. Filters for facility are available.

Drug Claims Analysis


Drug Claims Analysis
HOME
Total Manufacturers
980
Manufacturer Name
Search
Aadi Bioscience
Aaipharma
Generic Name
Search
0.9 % Sodium Chloride
A/C/E/Zinc/Sod Selenate/Copper
Brand Name
Search
A And D Diaper Rash
Abacavir
Total Average Drug Spending per Claim (2018–2022)
Total Avg Claims
Cumulative Growth in Avg Drug Spending per Dosage Unit by Generic Name (2018–2022)
Annual Growth (%)
Generic_Name
Yearly Avg Drug Spending per Dosage Unit (2018–2022)
Avg Drug Spending per Dosage
Total Drug Spending Distribution by Year (2018–2022)
Year
2018
2019
2020
2021
2022

This dashboard focuses on Medicaid drug spending, displaying total manufacturers, average spending per claim and dosage unit over time, cumulative spending growth by drug, and spending distribution by year. Filters for drug and manufacturer are included.

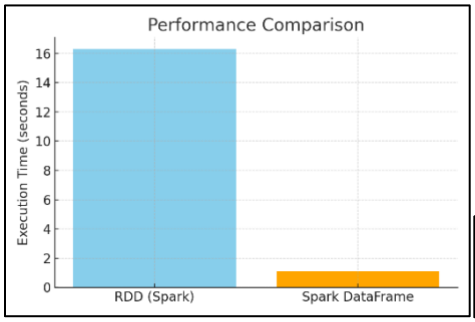
Medicare Claims Analysis

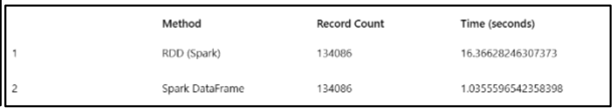

Medicare Claims Analysis
HOME
Total Referring Providers
268614
Total Supplier Claims
71,097,823.40
Provider Name
Search
Aabida
Aagaard
Provider Country
Search
DE
JP
Number of Referring Providers by RBCS Category
Average Distribution of Suppliers by Provider Country
Total Average Supplier Payment & Charge Amounts
Referring Provider Frequency by Provider Name
Referring Providers by Supplier Rental Indicator
Referring Providers by RUCA Category (Rural–Urban Classification)

This dashboard analyzes Medicare claims, showing the number of referring providers and supplier claims, provider types, supplier distribution by country, payment and charge amounts, referral frequency by provider, and provider location (urban/rural). Filters for provider and country are available.

9. Performance Comparison: MapReduce vs. Spark

- To compare the performance of **MapReduce** and **Spark**, we used a **Databricks notebook**.
- We simulated MapReduce using **Spark RDDs**.
- The task was to **count records** in the Medicare_DME_DS dataset where Average_Supplier_Medicare_Payment_Amount > 100.
- **Reference**: Used Notebook
- **Observation**: Spark performed the operation **faster** than the RDD-based MapReduce simulation due to **in-memory processing**.





	Method	Record Count	Time (seconds)
1	RDD (Spark)	134086	16.36628246307373
2	Spark DataFrame	134086	1.0355596542358398

10. Challenges & Solutions

Challenge	Solution
Google Drive file corruption (>100MB)	Split files into chunks smaller than 100MB.
High cost of Databricks cluster	Enabled auto-termination and used a single-node cluster configuration.
Dynamic pipeline complexity in ADF	Used parameterized JSON files with Lookup and ForEach activities.
Lack of real-time Medicare data	Generated synthetic data using a Python notebook (every 2 seconds).
Measuring MapReduce vs. Spark performance	Used a Databricks notebook with both RDD (Spark) and native Spark APIs.

11. References

- Git Repository: https://github.com/isurumadhushan24572/Lambda_Architecture_Health_Insurance_Analysis
- Data Source: https://data.cms.gov/
- Microsoft Learn: https://learn.microsoft.com/en-us/
- Power BI: https://powerbi.microsoft.com/
- Dashboard: Health Insurance Analysis Dashboard
- Preview Video: BDP_Preview_Video_Group_02.mp4

12. Team Contribution

Student ID	Student Name	Contribution
24490	M.R.K. Karunathilaka	Data Source & Presentation Layer (Power BI)
24572	M.K.I.M. Rohana	Batch Layer & Serving Layer
24614	G.A.A.S. Ganegoda	Speed Layer