National College of Ireland

Project Submission Sheet

Student Name:	Surya Chandra Raju Kurapati, Nisarga Holenarasipur Ramesh		
Student ID:	x233969620, x24101028		
Programme:	MSc in Data Analytics (MSCDAD_B)	**Year:**	Jan 2025-2026
Module:	Data Intensive and Scalable Systems		
Lecturer:	Jaswinder Singh		
Submission Due Date:	16-Jul-2025		
Project Title:	Healthcare Insurance Claims Processing: Leveraging Big Data for Policy Renewal Analysis (Descriptive Analytics) and Fraud Detection (predictive Analytics)		
Word Count:	2811 words		

Signature:	Surya Chandra Raju Kurapati, Nisarga Holenarasipur Ramesh
Date:	16-July-2025

Office Use Only	
Signature:	

Date:	
Penalty Applied (if applicable):	

AI Acknowledgement Supplement

Data Intensive and Scalable Systems

Healthcare Insurance Claims Processing: Leveraging Big Data for Policy Renewal Analysis (Descriptive Analytics) and Fraud Detection (predictive Analytics)

Your Name/Student Number	Course	Date
Surya Chandra Raju Kurapati	MSc in Data Analytics	16-Jul-2025
Nisarga Holenarasipur Ramesh	MSc in Data Analytics	16-Jul-2025

This section is a supplement to the main assignment, to be used if AI was used in any capacity in the creation of your assignment; if you have queries about how to do this, please contact your lecturer. For an example of how to fill these sections out, please click here.

AI Acknowledgment

This section acknowledges the AI tools that were utilized in the process of completing this assignment.

Tool Name	Brief Description	Link to tool
Notebooklm	Used to understand key concepts of claims fraud detection and examples in detail	https://notebooklm.google.com/

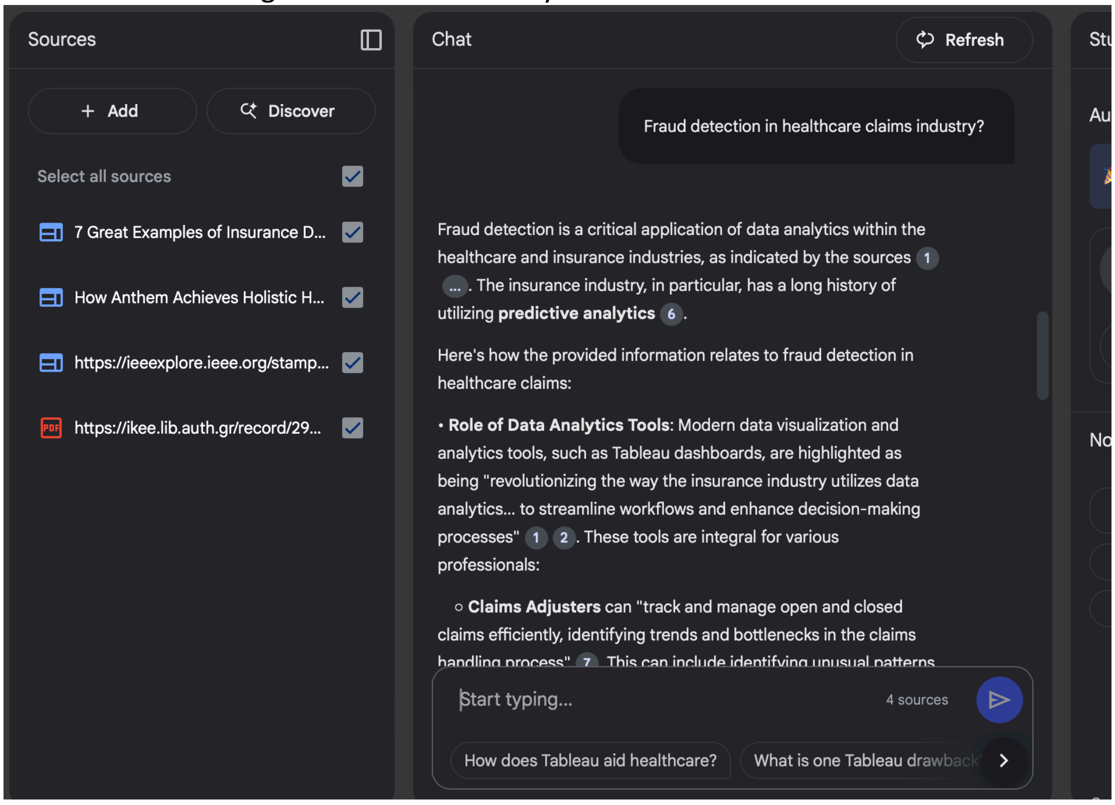
Description of AI Usage

This section provides a more detailed description of how the AI tools were used in the assignment. It includes information about the prompts given to the AI tool, the responses received, and how these responses were utilized or modified in the assignment. **One table should be used for each tool used**.

[Insert Tool Name]	
Upon collecting various sources and references relevant to claims fraud detection, used notebooklm to study various content, key objectives of the use case.	
Fraud Detection in healthcare claims industry	Machine Learning techniques can be used to detect outliers in claims adjudication and flag the fraudulent claim submissions.

Evidence of AI Usage

This section includes evidence of significant prompts and responses used or generated through the AI tool. It should provide a clear understanding of the extent to which the AI tool was used in the assignment. Evidence may be attached via screenshots or text.


Sources
Add
Discover
Select all sources
7 Great Examples of Insurance D...
How Anthem Achieves Holistic H...
https://ieeexplore.ieee.org/stamp...
https://ikee.lib.auth.gr/record/29...
Chat
Refresh
Start typing...
4 sources
How does Tableau aid healthcare?
What is one Tableau drawback

Fraud detection in healthcare claims industry?

Fraud detection is a critical application of data analytics within the healthcare and insurance industries, as indicated by the sources 1 .... The insurance industry, in particular, has a long history of utilizing **predictive analytics** 6.

Here's how the provided information relates to fraud detection in healthcare claims:

- **Role of Data Analytics Tools:** Modern data visualization and analytics tools, such as Tableau dashboards, are highlighted as being "revolutionizing the way the insurance industry utilizes data analytics... to streamline workflows and enhance decision-making processes" 1 2. These tools are integral for various professionals:
  - **Claims Adjusters** can "track and manage open and closed claims efficiently, identifying trends and bottlenecks in the claims handling process" 7. This can include identifying unusual patterns

Additional Evidence:

[Place evidence here]

Additional Evidence:

[Place evidence here]

Healthcare Insurance Claims Processing: Leveraging Big Data for Policy Renewal Analysis (Descriptive Analytics) and Fraud Detection (Predictive Analytics)

1st Surya Chandra Raju Kurapati
MSc in Data Analytics
National College of Ireland
Dublin, Ireland
x23396920@student.ncirl.ie

2nd Nisarga Holenarasipur Ramesh
MSc in Data Analytics
National College of Ireland
Dublin, Ireland
x24101028@student.ncirl.ie

***Abstract*—The Insurance Industry generates vast quantities of claims and policy data, calling for a need of scalable big data engineering solutions for analytics and reporting. This paper presents an end-to-end architecture for a daily healthcare claims processing system designed to support both descriptive analytics (strategic dashboards for policy renewals) and prescriptive analytics (fraud detection). The architecture is an integration of various tools and technologies such as Apache Spark for distributed in-memory computation, Hadoop tools such as DistCp for high-volume data transfer, HDFS for persistent storage, and Apache Airflow for workflow orchestration. The system processes terabyte-scale datasets daily via parallel in-memory execution, with results persisted to a Hive data warehouse. Key noticeable innovations include optimized Spark configurations, a four-stage ETL pipeline (raw, stage, processed, consumption), and pre-aggregated materialized views for analytics. Evaluation demonstrates that the solution can reliably process over 10 million daily records within a 4-5-hour SLA, reducing costs over 70% compared to traditional single threaded computation.**

***Index Terms*—Healthcare, Claims Batch Processing, Big Data, Spark, Hadoop, HDFS, ETL, DistCp, Airflow, Tableau, Fraud Detection, HDFS, Policy Renewal**

I. INTRODUCTION

Healthcare Insurance providers, for instance, Anthem handles over 1.4 PB of data and process 10-20 TB on daily basis [1]. This volume and velocity of data are critical for decision-making use cases like policy renewals and fraud migration. This paper presents an architecture that processes structured healthcare claims data using a big data pipeline that supports two primary use cases:

1. **Policy Renewal Progress Dashboard** that highlights gross written premium (GWP) trends at risk, empowering account managers, underwriters, space and space financial analyst to drive timely policy renewals and enhance customer retention [4].
2. **Fraud detection pipeline** that uses machine learning to identify anomalies in submitted claims, enabling early fraud detection while adhering to private regulations [5].

These use cases are supported by a multi-layer data pipeline designed for high throughput performance, scalability, any point-of-failure, and real time insight generation.

II. LITERARTURE REVIEW

The growing reliance on data-driven practices in health-care insurance is well-documented. Pareek et al. discuss the limitations of legacy systems in managing unstructured and high-velocity data [2].

Past studies and industry reports [7][8] emphasize the need for high-throughput processing platforms for healthcare datasets. Traditional ETL tools fail to meet the volume and performance demands. Research indicates Spark outperforms MapReduce due to in-memory computing [9]. Machairdou emphasizes the role of BI platforms like Tableau in visualizing insurance KPIs [3]. A tableau dashboard example by Quantize Analytics highlights how GWP-based visualizations support strategic decision-making [4].

On fraud detection, Kenyon and Eloff propose a scalable big data framework for insurance fraud prediction using supervised and unsupervised ML techniques [5]. Their approach validates the necessity for scalable compute engines to process high-volume claims data in real time.

Ihtisum et al. compare big data frameworks and affirm Spark's superiority over MapReduce due to its in-memory processing and iterative execution model [6]. This supports the choice of Spark for both ETL workloads in this architecture.

III. PROBLEM DEFINITION

The architecture must handle two unique analytical needs, first one being, generation of strategic reports for policy renewals monitoring and second, predictive detection of fraudulent claims.

1. **Descriptive Analytics Use Case:** The Renewal Progress dashboard visualizes Gross Written Premium (GWP) nearing to expiration, allowing teams to engage with policyholders for timely renewal. Users of this dashboard are account managers, underwriting teams, and customer support representatives. Visualizations such as performance KPIs guide renewal actions and area charts. Metrics include renewal rates, expiring GWP, and turnaround time [4].
2. **Prescriptive Analytics Use Case:** A machine learning pipeline identifies fraudulent patterns using historical claims data. The system calculates fraud scores and generates alerts based on anomalies in CPT, HCPCS, ICD codes, billing patterns, and procedure mismatches [5]. Model results help compliance and investigation teams in mitigating financial risks. Metrics include precision, recall, and ROC_AUC.

The dataset includes structured healthcare claims with over 10 million records generated daily in compressed csv.gz format from mainframes system. Files are then transferred securely via SFTP to an edge server and ingested into HDFS. Scalability is essential to ensure the pipeline processes 10-20 TB of data daily within the 1:00 to 7:00 AM IST window, ensuring delivery of policy renewal dashboards and fraud detection model outputs before business hours begin.

IV. METHODOLOGY

A. UNDERSTANDING CLAIMS DATA

Claims data, also referred to as administrative healthcare data, is a structured record of medical services billed to insurers by healthcare providers. It serves as a critical resource for payment processing, cost analysis, fraud detection, and healthcare research. Table 1 represents the granular data of each claim record.

TABLE 1: CLAIMS METADATA

Category	Sub-Category	Attributes	Sample Value
Patient Demographics	Identification	Patient ID, Name (First/Last), DOB	P-88762, John Doe, 1980-05-15
	Contact Details	Address, Phone, Email	123 Main St, Boston, MA, (555) 123-4567
	Insurance	Payer Name, Policy Number, Plan Type	Medicare, PLN-88921, PPO
Diagnosis Codes	Primary/Secondary	ICD-10 Code, Description, Present-On-Admit	J18.9 (Pneumonia), Yes
Procedure Codes	Service Type	CPT Code, HCPCS Code, Modifiers	99213 (Office Visit), J1100 (Injection)
Service Details	Timing	Admission Date, Discharge Date, Service Hours	2024-03-10, 2024-03-12, 2.5 hrs
	Place of Service	Facility Type (POS Code), Provider NPI	11 (Office), 1234567890
Financial Transactions	Billing	Billed Amount, Allowed Amount, Adjustments	$500, $400, $100 (Contractual)
	Patient Responsibility	Coinsurance, Copay, Deductible	$50, $20, $150
	Payer Payment	Paid Amount, Payment Date	$330, 2024-04-01

B. DATA MODEL

A star schema model is adopted with a central "fact_claims" table linked to dimension tables as illustrated in Fig 1. This structure supports efficient querying to create materialized views serving defined use cases.

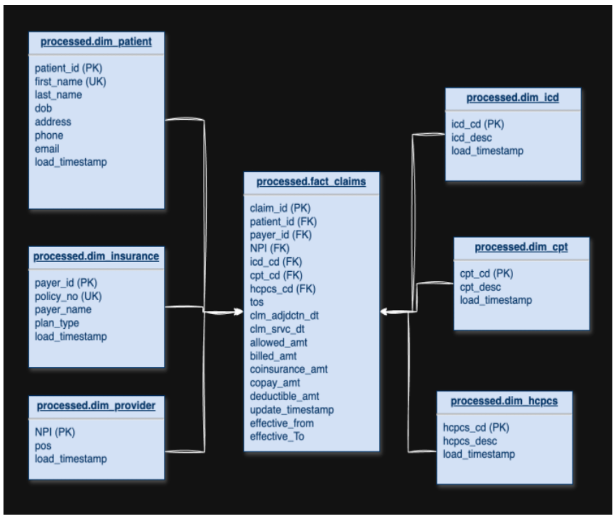


Fig. 1. Claims Data Model

C. ARCHITECTURE

The claims data processing architecture is a multi-layered framework designed to support high-volume, secure, and scalable data ingestion, transformation, and analytics delivery. As illustrated in Fig 2, the architecture is logically divided into three core layers: Source, ETL, and Analytics.

Source Layer: Claims data originates from a mainframe host system, which has traditionally been used to manage terabytes of claims data daily. In this layer, data is exported as compressed CSV file as csv.gz through nightly batch processes and securely transferred to the edge server using SFTP.

ETL Layer: ETL pipeline is fully orchestrated through Apache Airflow and manages the end-to-end ingestion and transformation workflow. It consists of four sub-layers:

Raw layer: Once the claims data files arrive at the edge server, they are transferred into the HDFS "raw layer" bucket. This layer stores the compressed CSV files in their original form, without applying any transformation logic. A strict archival policy ensures that files older than one month are deleted via shell script. This layer also functions as a disaster recovery checkpoint and provides traceability for the original source data.

Stage Layer: This layer is meant for data cleaning where essential preprocessing tasks such as deduplication, data type validation, null checks against unique keys, and masking of patient PHI/PII data are performed. Additionally, audit column "load_timestamp" field is added for tacking purpose. These transformations are executed via PySpark job submitted using spark-submit command, and the results are written in Parquet format to HDFS “stage layer” bucket. Furthermore, SCD Type 1 logic is applied here to ensure that only the latest available data is retained.

Processed Layer: After successful staging, the cleansed data flows into the processed layer, where business transformation logic such as filtration, aggregation, joins are applied. This layer incorporates SCD Type 2 methodology to preserve historical versions of slowly changing attributes over a five-year period. The data is structured into fact and dimension tables and stored as Parquet format, partitioned by “yyyymmdd” to support high-performance analytics. Prior to the data write to HDFS location, surrogate key is generated for each unique record and audit fields such as “insert_timestamp”, “update_timestamp”, and “delete_timestamp” are added. This layer serves as the backbone for analytics and historical reporting.

Consumption Layer: As depicted in Fig 2, the final stage of the ETL pipeline is the consumption layer, where materialized views with two-year rolling window data is created from the processed layer, which is run through Spark SQL job. The final datasets are written in parquet format and exposed through Hive external tables for efficient querying to next layers.

Analytics Layer: Hive external tables generated in the consumption layer serve as the entry point in this layer to generate Policy Renewal Progress Dashboard and Fraud Detection Analytics.

D. TECHNOLOGY

To efficiently handle high-volume claims data processing and analytics, it is essential to consider a robust and scalable tech-stack. Table 2 outlines the key architectural components required to implement the previously described pipeline, along with the chosen technologies and their justifications.

Furthermore, a 10-node compute cluster with each node configured with 16 cores and 128 GB RAM is considered for this workflow.

To execute ETL operations, a custom ETL framework has been developed using PySpark, incorporating modular functions such as “read_csv, drop_duplicates, add_audit, generate_sk, read_sql, and write_parquet”. The framework initializes by creating a Spark Session, then reads data into PySpark data frames and applies transformations based on the specific workflow requirements It is designed to be extensible, allowing for the seamless integration of additional functionality as needed in future stages.

Fig. 2. Architecture

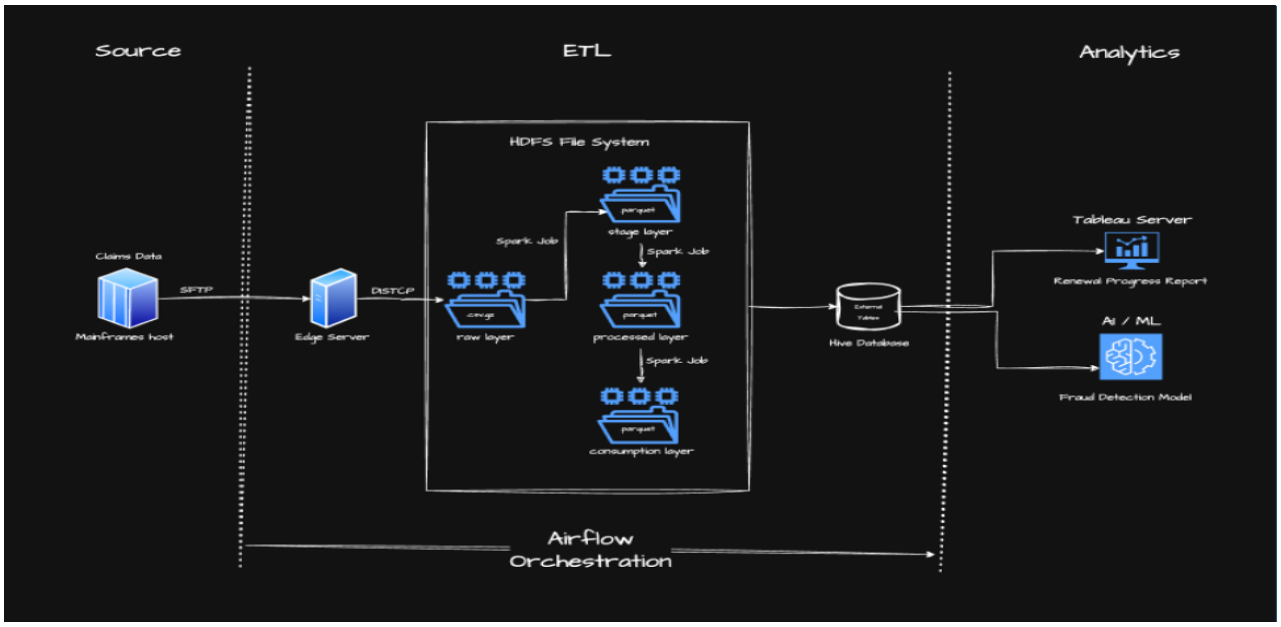


TABLE 2: TECHNOLOGY STACK AND JUSTIFICATION

Component	Technology		Justification
Storage	HDFS	hadoop HDFS	Distributed, fault-tolerant, optimized for large file I/O
Processing Engine	Apache Spark	APACHE Spark	In-memory execution, ideal for iterative and large-scale ETL [6][9]

Orchestration	Apache Airflow		DAG-based workflow management, retry/failure handling
BI Platform	Tableau		Interactive dashboards and support for Hive-backed sources [3][4]
File Transfer	Hadoop DistCp		Parallel, efficient transfer of large volumes
File Format	Parquet		Columnar storage format, highly compressed, fast query performance
Metadata Layer	Apache Hive		Schema abstraction and SQL interface for BI and data science access

E. WORKFLOW EXECUTION

To illustrate the end-to-end claims data processing workflow, this section draws upon Anthem's benchmark workload, which involves processing approximately 10 TB of data and over 10 million records daily [1]. The compute pipeline comprises of the following steps:

Step1: Data Generation & Secure Transfer Claims data generation from the mainframe system starts every night as batch job and gets exported as compressed csv.gz file. This large file is securely unloaded to the edge server via SFTP by 12:30 AM Ireland Standard Time.

Step2: Ingestion to HDFS Raw Layer Upon arrival of the file at edge server, Airflow DAG is scheduled to trigger at 1:00 AM IST, and the first step is to run below DistCp command that copies claim source data file from edge server to HDFS raw layer bucket.

hadoop distcp </local/claim_raw_20250715.csv.gz> </raw/claim_raw/>

Step3: Stage Layer – Data Cleansing Custom PySpark framework triggered via spark-submit firstly reads the csv.gz file into a data frame through "read_csv" operation and later continue with data cleanse operations like "drop_duplicates", "null_check", "mask_phi_pii_data", "add_audit" and write the output as SCD Type 1 logic in Parquet format.

Step4: Processed Layer – Business Transformation Cleansed data from the stage layer is enriched through business logic such as joins, aggregations, filters initially, later add surrogate key to every row of the data frame, finally add audit column, and write data as daily partitions in "yyyymmdd" format to the HDFS processed layer bucket in parquet format maintain history of data through SCD Type 2 logic.

Step 5: Consumption Layer – Materialized View A Spark SQL job aggregates a two-year rolling data from the built facts and dimensions of processed layer and write it to the consumption layer again in Parquet format. Data is exposed through Hive external tables, forming materialized views for analytical layer.

Now, Airflow DAG ends here and estimated to complete by 6:00 AM IST.

Step 6: Analytics Delivery By 7:00 AM IST, all downstream outputs including Tableau dashboards for policy renewals and fraud detection models, are refreshed and ready for consumption before business logs in.

F. PERFORMANCE AND OPTIMIZATION

During the claims pipeline execution processing 10 TB dataset, multiple performance bottlenecks were encountered, primarily due to shuffle overhead and out-of-memory (OOM) errors.

a) **Issue 1: Shuffle Overhead due to Heavy Aggregations:** Due to intensive groupBy operations in claims data, the triggered PySpark job failed unexpectedly. Upon root cause analysis through Spark UI and executor logs, it was understood that the failure was due to excessive shuffle overhead caused by unoptimized aggregations. This was particularly problematic given the data scale of 10 TB input data translating into a large no. of intermediate shuffle partitions.
To resolve this, a spark configuration parameter **spark.sql.shuffle.partitions** was set to 40000 from the default 200, based on the rule of thumb: 10 TB / 256 MB = 40,000 partitions, and to accommodate appropriate parallelism and fault tolerance.
This optimization not only reduced executor shuffle wait time but also resolved small files problems, as output files were coalesced to 40000 Parquet sizes, avoiding excessive task overhead.

b) **Issue 2: Delay in Job completion:** Although the shuffle issue was resolved, the job was still taking 7-8 hours to complete, exceeding the SLA window of 6:00 AM IST.
To address this, cluster parameters were fine-tuned to make better use of the available hardware (10 nodes, 16 cores, 128 GB RAM each). The following Spark job configurations were applied to the spark-submit command:

```
spark-submit \
--deploy-mode cluster \
--executor-cores 5 \
--num-executors 30 \
--executor-memory 40 G\
--driver-memory 15 G \
--conf
spark.sql.shuffle.partitions=40000\
--conf spark.default.parallelism=40000\
custom_pyspark_framework.py
```

This setup launched 30 parallel executors with 5 cores each enabling 5 tasks running in parallel per executor, and 40 GB of executor memory each, enabling 150 concurrent tasks and efficient memory distribution across 10 nodes.

G. SINGLE-THREADED VS PARALLEL PROCESSING ANALYSIS

A comparative study is performed between single-threaded vs. parallel Spark processing. Table 3 demonstrates the comparison with metrics.

This comparison clearly demonstrates that parallel processing using a Spark cluster offers significantly improved performance, scalability, and fault tolerance over single-threaded execution, making it ideal for large-scale claims data processing.

V. FUTURE WORK

As the volume and complexity of healthcare claims data continue to grow, the proposed architecture can be further optimized by migrating to cloud-native platforms. Table 4 presents a comparative study outlining the difference between the current On Premise and potential Cloud architecture.

TABLE 3: SINGLE THREADED VS PARALLEL PROCESSING COMPARISON

Attribute	Single-Threaded	Parallel (Spark Cluster)
Cluster Size	1 server	10 servers
Cores	16	16 cores * 10 servers = 160
RAM	128 GB	128 GB * 10 servers = 1.28 TB
Processing Approach	Sequential	Parallel across nodes
Shuffling	Minimal	Fully supported
Performance	~40-50 hours (10TB)	~4-5 hours (10TB)
Scalability	Limited (vertical)	High (horizontal scaling)
Fault Tolerance	Low	High

TABLE 4: ON PREMISE VS CLOUD COMPARISON

Component	On-Premise Architecture	Cloud-Native Architecture
Storage	HDFS (local Hadoop clusters)	Amazon S3 (object storage, highly durable & scalable)
Compute Engine	Apache Spark (on YARN/standalone)	AWS EMR or Databricks (auto-scaling clusters, pay-as-you-go)
Orchestration	Apache Airflow (manual setup, on-prem)	AWS lambda / Step Functions (serverless, managed orchestration)
BI & Reporting	Tableau + Hive tables	Tableau + Snowflake
Data Warehouse	Hive Metastore	Snowflake (cloud-native, scalable, secure SQL analytics)
File Transfer	Hadoop DistCp (manual job configuration)	AWS DataSync / S3 Transfer Acceleration (automated, optimized)
Scalability	Vertical scaling, hardware-bound	Horizontal scaling, elastic resources
Cost Model	Fixed CapEx for infrastructure	Pay-per-use OpEx, cost-optimized for dynamic workloads
Maintenance	High (manual patching, upgrades, monitoring)	Low (fully managed services, built-in monitoring and alerting)

VI. CONCLUSION

This paper presents a scalable, cost-efficient architecture for large-scale claims data processing in healthcare insurance. By leveraging open-source technologies like Spark, Hadoop, and Airflow, the system delivers robust ETL performance, high scalability, and actionable insights. It supports both descriptive dashboards and predictive fraud models while enabling future migration to cloud-native solutions for enhanced agility and cost savings.

ACKNOWLEDGEMENT

The author would like to thank “Jaswinder Singh” for supporting this research.

REFERENCES

[1] How Anthem Achieves Holistic Healthcare Using Snowflake's Data Cloud (2025) Snowflake.com.
Available at: https://www.snowflake.com/en/blog/how-anthem-achieves-holistic-healthcare-using-snowflakes-data-cloud/

[2] Pareek, T., Sood, K. and Grima, S. (2022), "The Impact of Big Data Technology on the Advancement of the Insurance Industry"
DOI: https://doi.org/10.1108/978-1-80262-637-720221012

[3] Machairidou, S. (2018). Big Data and Tableau.
Available at: https://ikee.lib.auth.gr/record/297918/files/GRI-2018-21585.pdf

[4] 7 Great Examples of Insurance Dashboards in Tableau (2024) Quantize Analytics.
Available at: https://www.quantizeanalytics.co.uk/tableau-insurance-dashboards-examples/

[5] D. Kenyon and J. H. P. Eloff, "Big data science for predicting insurance claims fraud," 2017 doi: 10.1109/ISSA.2017.8251773.
Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8251773

[6] S. Ibtisum et al., "A Comparative Analysis of Big Data Processing Paradigms: MapReduce vs Apache Spark" 2017.

[7] Ebenezer, J. Gemson Andrew, and S. Durga. "Big data analytics in healthcare: A survey." ARPN Journal of Engineering and Applied Sciences 10.8 (2015): 3645-3650.

[8] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51, 1 (January 2008), 107–113.
DOI: https://doi.org/10.1145/1327452.1327492

[9] Matei Zaharia et al., "Apache Spark: a unified engine for big data processing" 2016.
DOI: https://doi.org/10.1145/2934664