Scientific Research Publishing

Journal of Data Analysis and Information Processing, 2024, 12, 544-565
https://www.scirp.org/journal/jdaip
ISSN Online: 2327-7203
ISSN Print: 2327-7211

Optimizing Healthcare Big Data Processing with Containerized PySpark and Parallel Computing: A Study on ETL Pipeline Efficiency

Ehsan Soltanmohammadi, Neset Hikmet

Department of Integrated Information Technology, University of South Carolina, Columbia, United States
Email: ehsans@email.sc.edu

How to cite this paper: Soltanmohammadi, E. and Hikmet, N. (2024) Optimizing Healthcare Big Data Processing with Containerized PySpark and Parallel Computing: A Study on ETL Pipeline Efficiency. *Journal of Data Analysis and Information Processing*, **12**, 544-565.
https://doi.org/10.4236/jdaip.2024.124029

Received: September 9, 2024
Accepted: October 15, 2024
Published: October 18, 2024

Abstract

In this study, we delve into the realm of efficient Big Data Engineering and Extract, Transform, Load (ETL) processes within the healthcare sector, leveraging the robust foundation provided by the MIMIC-III Clinical Database. Our investigation entails a comprehensive exploration of various methodologies aimed at enhancing the efficiency of ETL processes, with a primary emphasis on optimizing time and resource utilization. Through meticulous experimentation utilizing a representative dataset, we shed light on the advantages associated with the incorporation of PySpark and Docker containerized applications. Our research illuminates significant advancements in time efficiency, process streamlining, and resource optimization attained through the utilization of PySpark for distributed computing within Big Data Engineering workflows. Additionally, we underscore the strategic integration of Docker containers, delineating their pivotal role in augmenting scalability and reproducibility within the ETL pipeline. This paper encapsulates the pivotal insights gleaned from our experimental journey, accentuating the practical implications and benefits entailed in the adoption of PySpark and Docker. By streamlining Big Data Engineering and ETL processes in the context of clinical big data, our study contributes to the ongoing discourse on optimizing data processing efficiency in healthcare applications. The source code is available on request.

Keywords

Big Data Engineering, ETL, Healthcare Sector, Containerized Applications, Distributed Computing, Resource Optimization, Data Processing Efficiency

1. Introduction

In the rapidly evolving landscape of Big Data Engineering and Extract, Transform,

Load (ETL) processes, the quest for efficiency stands as a pivotal pursuit [1]. The volume, variety, and velocity of data are increasing exponentially, with estimates suggesting that the global healthcare data market will reach 463 exabytes by 2025 on generated daily information. The healthcare industry is undergoing a significant transformation, driven by the rapid growth of big data. By 2025, the market is projected to reach $70 billion, a staggering increase of 568% over the past decade [2]. As healthcare organizations embrace big data, they face a complex challenge: harnessing its power to enhance decision-making processes and improve patient outcomes. However, this challenge also presents a tremendous opportunity for innovation and growth, enabling healthcare stakeholders to revolutionize the supply chain and deliver more effective, personalized care [3]. The healthcare sector, in particular, generates vast amounts of complex data, making it a crucial domain for optimizing data processing efficiency. Efficient ETL processes are essential for harnessing the power of big data in healthcare, enabling timely insights, and improving patient outcomes. Furthermore, the increasing application of Artificial Intelligence (AI) and Machine Learning (ML) in healthcare is poised to significantly impact big data analysis [4]. Despite the significance of efficient ETL processes, current big data healthcare pipelines often suffer from inefficiencies, leading to prolonged processing times, resource waste, and reduced scalability [4]. The lack of optimized ETL workflows hinders the seamless integration of data from diverse sources, such as electronic health records (EHRs), medical imaging, and wearable devices [5]. This integration is critical for realizing the full potential of big data in healthcare, including personalized medicine, predictive analytics, and population health management [6].

Moreover, the healthcare industry faces a significant gap in leveraging efficient tools and technologies in big data pipelines and data collection methods. Many existing ETL processes rely on traditional, resource-intensive methods, neglecting the benefits of modern distributed computing frameworks and containerization [7]. This gap results in:

- Inefficient use of resources, leading to increased costs and reduced scalability.
- Prolonged processing times, hindering timely insights and decision-making.
- Limited integration of diverse data sources, restricting the scope of analytics and insights [8].

The current state of big data processing in healthcare has been analyzed through academic papers, revealing both benefits and challenges. The benefits include improved patient outcomes through data-driven decision making [9], enhanced operational efficiency and reduced costs, and better data management and integration. However, challenges persist, including inefficiencies, scalability issues, prolonged processing times, limited integration of diverse data sources, and inadequate use of resources [2] [10] [11]. Moreover, there is a need for more efficient and scalable ETL processes in the healthcare sector [12].

This literature review involved a systematic search of peer-reviewed articles from leading healthcare and technology journals, such as the *Journal of Healthcare*

Management and *Journal of Big Data*. Using keywords like "big data analytics in healthcare" and "healthcare data integration", the search focused on articles published in English between 2016 and 2024. It specifically targeted studies on the use of big data in healthcare, its benefits, challenges, and future trends, with a focus on ETL processes and data pipeline management. A total of 40 articles were reviewed to give a comprehensive overview of the field.

Market research studies have identified trends and challenges in healthcare big data. The trends include growing demand for healthcare analytics solutions, increasing adoption of advanced technologies, descriptive analytics dominating the market, and financial analytics to improve patient outcomes [13] [14]. However, challenges persist, including operational gaps between payers and providers, inaccurate and inconsistent data, high prices of analytics solutions, and lack of technical expertise in healthcare organizations [13]-[15].

Success stories have been examined to understand real-world applications and solutions (25). For instance:

- A large hospital chain implemented a big data analytics solution to improve patient outcomes and reduce costs. The solution integrated data from various sources, including electronic health records, claims data, and social determinants of health. As a result, the hospital chain was able to identify high-risk patients and provide targeted interventions, leading to a 20% reduction in readmissions and a 15% reduction in costs [16].
- A healthcare payer implemented a big data analytics solution to improve operational efficiency and reduce costs. The solution analyzed claims data and identified areas of inefficiency, leading to a 12% reduction in costs and a 10% improvement in operational efficiency [10].

Consultations with thought leaders in healthcare big data and ETL processes have identified practical challenges. Analysis of publicly available datasets like MIMIC-III [17]-[19] and PhysioNet [19] has highlighted specific challenges and areas for improvement in ETL workflows. Furthermore, research articles from reputable sources like the Journal of Big Data [10] and Healthcare Informatics Research [16] have also been consulted to gather a comprehensive understanding of the challenges and opportunities in healthcare big data analytics and ETL processes.

To address this gap, our study explores the domain of efficient Big Data Engineering and ETL processes in the healthcare sector, leveraging the MIMIC-III Clinical Database as a robust foundation. MIMIC-III is a large, freely available database containing deidentified health-related data from over 40,000 patients in critical care units at Beth Israel Deaconess Medical Center between 2001 and 2012. The database integrates comprehensive clinical data, making it widely accessible to researchers under a data use agreement. The demo dataset contains information for 100 patients, including ICU stays, admissions, diagnoses, procedures, and medications. This database is a great example of a use case for a big data pipeline, demonstrating the potential for large-scale data integration and analysis in healthcare research [17]-[19]. By investigating efficient practices and harnessing the power of PySpark and Docker containerized applications, we aim to contribute

valuable insights to the ongoing discourse in the field.

2. Methods

2.1. Selecting a Template (Sub-Heading 2.1)

We employed a mixed-methods approach, combining both qualitative and quantitative research methods, to gain a comprehensive understanding of the research problem. This approach allowed us to triangulate findings and increase the validity of our results and allowed us to delve deeper into the intricacies of healthcare big data processing. Qualitative methods, such as literature reviews and expert consultations, provided valuable contextual insights, while quantitative methods, like data analysis from the MIMIC-III Clinical Database, offered empirical evidence to support our findings.

The practical simulation built upon each method further reinforced the validity, reliability, and credibility of our results. By applying diverse approaches and cross-verifying our findings, we were able to mitigate potential biases and limitations inherent in any single research method. This triangulation of findings from various sources ensured a more balanced and nuanced understanding of the complex research problem at hand.

Ultimately, the mixed-methods approach emerged as the optimal strategy for our study. It enabled us to capture the multifaceted nature of healthcare big data processing and to derive more robust and insightful conclusions. By integrating qualitative and quantitative methods, we were able to delve deeply into the intricacies of the subject matter, providing a comprehensive analysis that enhances the relevance and applicability of our research findings. Details on the methods used, including containerization overhead, orchestration efficiency, and scalability, are provided in Tables 1-3, respectively.

Table 1. Data collection methods for understanding big data processing in healthcare.

Method	Description
Research Articles	Analyzed academic papers to understand the current state of big data processing in healthcare, including: Benefits: • Improved patient outcomes through data-driven decision making; • Enhanced operational efficiency and reduced costs; • Better data management and integration [11] [20] [21]. Challenges: The current state of big data processing in healthcare is characterized by: • Inefficiencies; • Scalability issues; • Prolonged processing times; • Limited integration of diverse data sources; • Inadequate use of resources [22] [12] [13] [14] [15]. Need: • More efficient and scalable ETL processes in the healthcare sector [23].

Continued

Industry Reports	Reviewed market research studies to identify trends and challenges in healthcare big data, including: Trends: • Growing demand for healthcare analytics solutions; • Increasing adoption of advanced technologies; • Descriptive analytics dominates the market; • Financial analytics to improve patient outcomes [24]-[26]. Challenges: • Operational gaps between payers and providers; • Inaccurate and inconsistent data; • High prices of analytics solutions; • Lack of technical expertise in healthcare organizations [27] [28].
Case Studies	Examined success stories to understand real-world applications and solutions, [29].

In addition to the literature review, we also consulted with industry experts and analyzed publicly available datasets to gain a deeper understanding of the practical challenges faced by healthcare organizations.

Table 2. Supplementary research methods for identifying practical challenges in healthcare big data and ETL processes.

Method	Description
Industry Experts	Consulted with thought leaders in healthcare big data and ETL processes to identify practical challenges.
Public Datasets	Analyzed datasets like MIMIC-III to identify specific challenges and areas for improvement in ETL workflows [18].

Through these methods, we identified key challenges and gaps in current ETL processes, including:

Table 3. Challenges as gaps.

Challenge	Description
Resource Inefficiency	Inefficient use of resources and prolonged processing times.
Limited Scalability	Limited integration of diverse data sources and lack of scalability.
Suboptimal ETL	Lack of optimized ETL workflows and distributed computing frameworks.

Summary of Challenges and Gaps

Our mixed-methods approach revealed a clear need for more efficient and scalable ETL processes in the healthcare sector. The challenges and gaps identified include resource inefficiency, limited scalability, and suboptimal ETL processes. These challenges and gaps served as the foundation for our research and experimentation, guiding our exploration of Containerized application, PySpark, Docker,

as potential solutions.

2.2. Proposed Solutions

2.2.1. ETL Definition and Equation

Extract, Transform, Load (ETL) (Equation (1)) is a crucial process in data integration that involves extracting data from multiple sources, transforming it into a standardized format, and loading it into a target system for analysis and reporting [30] [31].

The ETL process can be mathematically represented as:

$$\text{ETL} = (\text{Extraction, Transformation, Loading}) \quad (1)$$

where:

Extraction = Data Collection + Data Cleansing + Data Mapping

Transformation = Data Transformation + Data Aggregation + Data Quality Check

Loading = Data Loading + Data Indexing + Data Retrieval

The ETL equation highlights the interconnectedness of the three stages, emphasizing that each stage is dependent on the previous one. A successful ETL process ensures that data is accurately extracted, transformed, and loaded, enabling organizations to make informed decisions based on reliable and consistent data [32].

2.2.2. Technologies

Our methodology is characterized by meticulous experimentation, emphasizing a sophisticated array of technologies:

- **PySpark:** deployed as the engine for distributed computing, optimizes computational efficiency in ETL processes by distributing data across multiple nodes for parallel processing, scaling to match dataset size, utilizing in-memory processing, and supporting various data sources, making it an ideal choice for big data ETL, and easily integrating with other Python libraries and frameworks to improve overall performance and reduce processing time [33] [34].
- **Docker:** Docker: Leveraging its robust containerization capabilities, Docker plays a pivotal role in facilitating the seamless deployment of applications, thereby significantly enhancing the efficiency of data processing [35]. By providing a lightweight and portable solution for packaging and distributing applications, Docker enables developers to streamline their workflows and accelerate the deployment process. This, in turn, enables data scientists and analysts to focus on extracting valuable insights from data, rather than grappling with complex deployment issues [36]. With Docker, applications can be easily containerized, scaled, and managed, ensuring a high degree of flexibility and reliability in data processing pipelines. As a result, Docker has emerged as a key enabler of efficient data processing, empowering organizations to make faster and more informed decisions [37].

- **Docker Compose:** Orchestrating multi-container applications, Docker Compose adds a layer of sophistication to our methodology. It streamlines the process of defining and running applications, ensuring seamless integration and coordination among diverse components. With Docker Compose, we can effortlessly manage complex application architectures, leveraging modularity and scalability to build resilient, distributed systems for efficient data processing [38].
- **Python:** As a versatile scripting language, Python enables rapid development and prototyping, ensuring flexibility in our approach. Its extensive libraries and frameworks facilitate efficient data manipulation, analysis, and machine learning tasks, making it an ideal choice for complex data processing challenges [39].
- **Pandas:** Pandas' powerful data manipulation and analysis capabilities enable us to extract valuable insights from complex datasets. Its efficient data structures and operations facilitate swift data merging, joining, and grouping, while its data alignment and merging capabilities integrate disparate data sources. With Pandas, we can effortlessly handle missing data, perform data cleaning and preprocessing, and uncover meaningful patterns and relationships within the dataset [40].
- **PostgreSQL:** Serving as the robust backbone for efficient data storage and retrieval, PostgreSQL provides a reliable and scalable foundation for managing vast amounts of data with optimal performance. Its advanced querying capabilities, indexing, and caching mechanisms enable swift data access and retrieval, while its robust data integrity and security features ensure the accuracy and protection of sensitive information. With PostgreSQL, we can confidently store, manage, and query large datasets, leveraging its powerful features to drive data-driven insights and informed decision-making [41].

2.2.3. Practical Implications

This paper not only unearths key findings but also encapsulates pivotal insights with a pragmatic lens. Emphasizing the practical benefits of deploying PySpark within Docker containerized applications using Docker Compose. Our approach is tailored to streamline Big Data Engineering and ETL processes. As we navigate through this exploration, our focus remains unwaveringly committed to contributing insights that hold tangible value within the healthcare domain. The practical implementation is illustrated as shown in **Figure 1** below.

Our experimental setup aimed to assess the efficiency and effectiveness of four distinct ETL processes within the healthcare domain, focusing on Big Data Engineering principles. We utilized the MIMIC-III Clinical Database, a widely used database in healthcare research, to simulate real-world data processing scenarios. Below, we detail our experimental setup, including the MIMIC-III database, configuration specifics, and the four ETL processes leveraging various technologies such as PySpark, Docker, Docker Compose, Python, Pandas, and PostgreSQL.

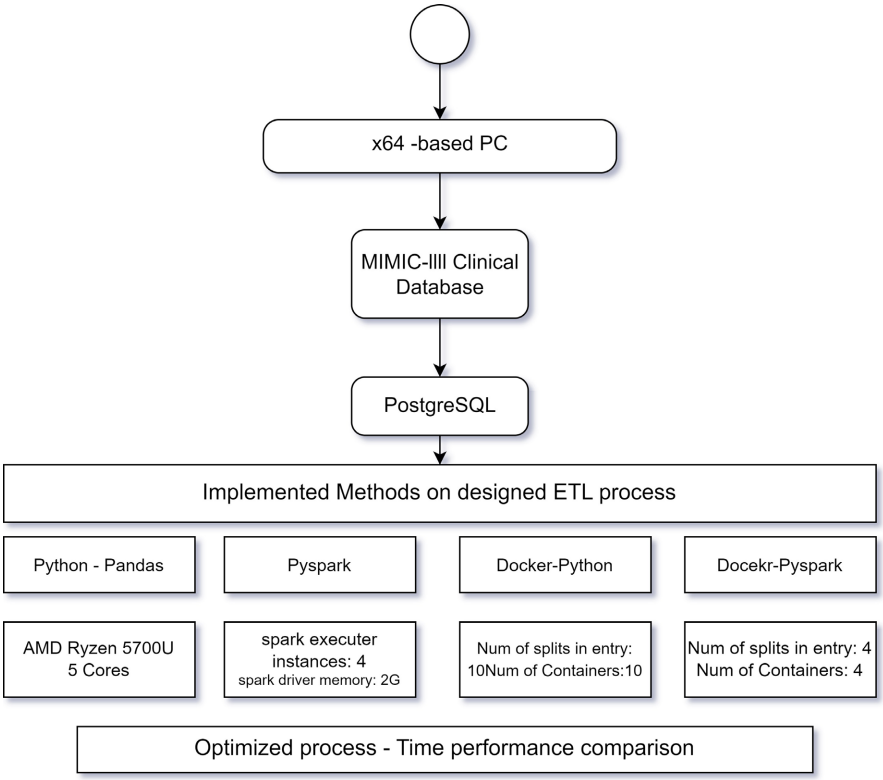

Figure 1. Practical implementation in the experiment.

2.2.4. MIMIC-III Clinical Database

The MIMIC-III (Medical Information Mart for Intensive Care III) database is a comprehensive collection of de-identified health-related data from over 40,000 patients who stayed in critical care units of the Beth Israel Deaconess Medical Center between 2001 and 2012. It includes information such as demographics, vital signs, laboratory tests, medications, and more, making it a valuable resource for healthcare research and analytics [17] [18].

2.2.5. Data Dictionary for Sample Data

The following tables provide a data dictionary for the sample datasets used in this research. The LABEVENTS table contains lab test results, while the PATIENTS table includes demographic details. **Tables 4-5** are used to demonstrate data processing and analysis procedures in the research.

Table 4. LABEVENTS table.

Field Name	Data Type	Description
ROW_ID	Integer	Unique identifier for each row in the lab events data.
SUBJECT_ID	Integer	Unique patient identifier linked to the PATIENTS table.
HADM_ID	Float	Unique identifier for a patient's hospital admission.
ITEMID	Integer	Identifier for the specific lab test performed.
CHARTTIME	Object	Date and time when the lab result was charted.

Continued

VALUE	Object	The result value for non-numeric lab tests.
VALUENUM	Float	The result value for numeric lab tests.
VALUEUOM	Object	Unit of measurement for the numeric lab result (e.g., mg/dL, mmol/L).
FLAG	Object	Indicator of whether the result was abnormal ("abnormal").

Table 5. PATIENTS Table

Field Name	Data Type	Description
ROW_ID	Integer	Unique identifier for each row in the patients data.
SUBJECT_ID	Integer	Unique patient identifier.
GENDER	Object	Gender of the patient (Male/Female).
DOB	Object	Date of birth of the patient.
DOD	Object	Date of death of the patient (if applicable).
DOD_HOSP	Object	Date of death during hospital admission (if applicable).
DOD_SSN	Float	Date of death based on social security number (if applicable).
EXPIRE_FLAG	Integer	Indicator of whether the patient is deceased (1 = Deceased).

2.2.6. Configuration Specifics

Our experimental environment was configured as follows:

- Hardware: We utilized a cluster of servers with adequate processing power and memory to handle the large volumes of data typical in healthcare datasets.
- Software: The software stack comprised Python, PySpark, Docker, Docker Compose, PostgreSQL, and relevant libraries such as Pandas for data manipulation and analysis.
- Database Management System (DBMS): PostgreSQL was chosen as the relational database management system due to its robustness and compatibility with the MIMIC-III database schema.
- Containerization: Docker was employed to containerize the ETL processes, ensuring consistency and portability across different environments.
- Orchestration: Docker Compose facilitated the orchestration of multi-container Docker applications, enabling streamlined deployment and management.

2.3. Four Distinct ETL Processes

1) ETL Process based on Python - Pandas: This process leveraged Python and the Pandas library for data extraction, transformation, and loading (ETL) tasks as shown in Figure 2. Python's versatility and Pandas' powerful data manipulation capabilities were utilized to process the MIMIC-III dataset efficiently [42].

2) ETL Process based on PySpark: PySpark, a Python API for Apache Spark, was utilized for this ETL process as shown in Figure 3. Spark's distributed computing framework enabled scalable and high-performance data processing,

suitable for handling large-scale healthcare datasets like MIMIC-III [43].

3) ETL Process based on Docker - Python/Docker Compose: Docker containers were employed to encapsulate Python-based ETL workflows as shown in Figure 4. Docker Compose orchestrated the deployment of multiple containers, ensuring

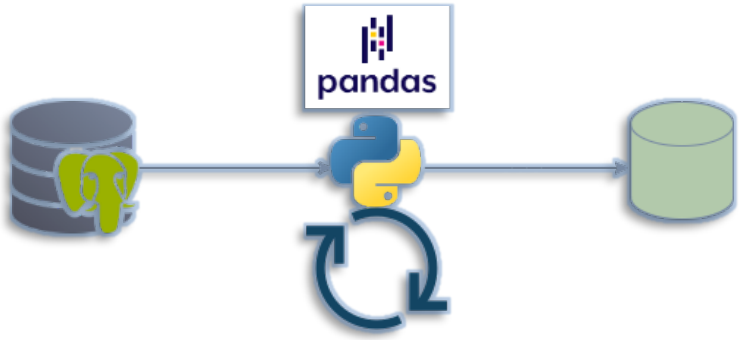

Figure 2. Extract, Transform, Load process.

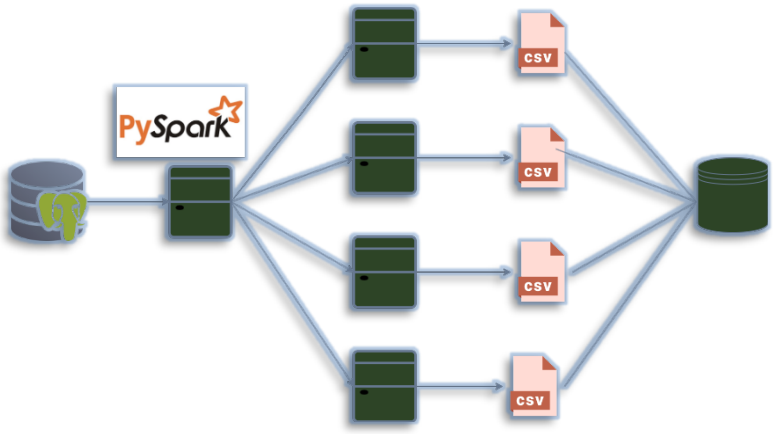

Figure 3. Pyspark ETL process.

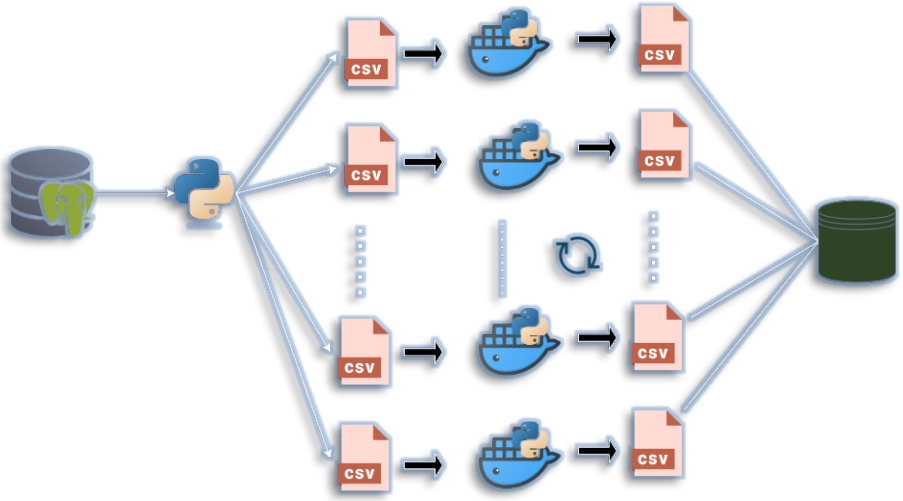

Figure 4. Docker-Python ETL process.

seamless integration and scalability of the ETL process [44] [45].

4) ETL Process based on Docker - PySpark/Docker Compose: This process combined the power of PySpark with the containerization benefits of Docker, as shown in Figure 5. PySpark applications were containerized using Docker, and Docker Compose orchestrated the deployment for optimized scalability and reproducibility [43].

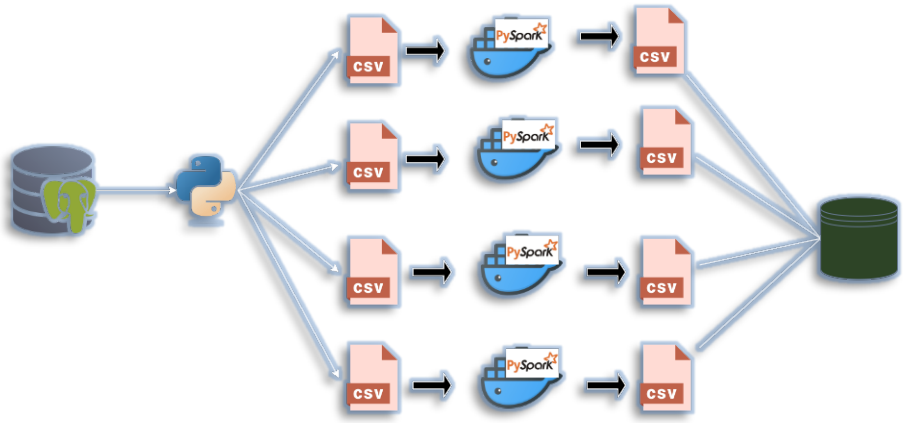

Figure 5. Docker-Pyspark ETL process.

3. Results and Analysis

3.1. Data Collection Process

To obtain the result data for each ETL process, we defined specific Python functions designed to fetch detailed performance metrics during each step of the workflow. These functions were deployed across all containers, tracking key parameters such as processing time, volume of data processed, and memory usage.

For each ETL process:

- Data Collection: We implemented a monitoring system within each container to capture metrics at different stages—data extraction, transformation, and loading.
- Processing Time Tracking: Each Python function recorded the start and end times of every ETL step, allowing us to calculate precise execution times for each dataset.
- Data Volume: The total volume of data processed in each step was automatically logged, ensuring that the performance metrics were contextualized based on the size of the dataset.
- Step-by-Step Logging: Logs were collected from all containers in real-time, ensuring that we had accurate data from every individual process, regardless of whether it was running a Python-Pandas workflow, PySpark, or a Dockerized version.

This structured approach enabled us to compile reliable performance data from the different environments, providing insights into the efficiency and scalability of each ETL method. The collected data formed the basis for the comparative

analysis presented in the following sections.

3.2. Data Processing Overview

In each ETL process, whether using Python-Pandas, PySpark, or Docker-based solutions, the data processing steps remained consistent across the different tools. Here's a breakdown of the general workflow:

1) Data Ingestion:

The first step involved reading data from CSV files, specifically the LABEVENTS and PATIENTS datasets. These datasets contain patient information and lab results, which were critical for the analysis.

2) Data Merging:

The LABEVENTS dataset, containing lab test results, was merged with the PATIENTS dataset using the common SUBJECT_ID field. This step was crucial to link each patient's lab test results to their demographic and health information.

3) Data Cleaning:

After merging, irrelevant columns that were not needed for the analysis were dropped to simplify the dataset and reduce complexity. This step ensures that only the most relevant data remains, improving processing speed and focus.

- Additionally, any rows with missing values in important fields, such as lab result numbers, were removed to maintain data integrity and ensure accurate analysis.

4) Data Transformation:

Date columns, such as birthdate and death date, were transformed into a standard datetime format, enabling easier manipulation and analysis.

- The EXPIRE_FLAG column, which indicates whether a patient has passed away, was transformed from a numeric representation to a more understandable text format (e.g., converting "1" to "Yes" and "0" to "No"). This enhances clarity when analyzing patient outcomes.

These steps formed the backbone of the data preparation process, ensuring that the datasets were cleaned, merged, and ready for further analysis or reporting. The use of different tools, such as Pandas or PySpark, allowed the workflow to handle large volumes of data efficiently while maintaining consistency in the transformation logic across all implementations.

3.3. Experimental Comparison: Python - Pandas vs. PySpark ETL Processes

This section presents the experimental results obtained from two distinct ETL processes: one utilizing Python with the Pandas library, as shown in **Figure 6**, and the other utilizing PySpark. The purpose is to compare their performance in handling data processing tasks.

3.3.1. Findings

The Python-Pandas ETL process exhibited moderate performance, with execution times ranging from 0 to 55 seconds as the dataset size increased up to 2 GB.

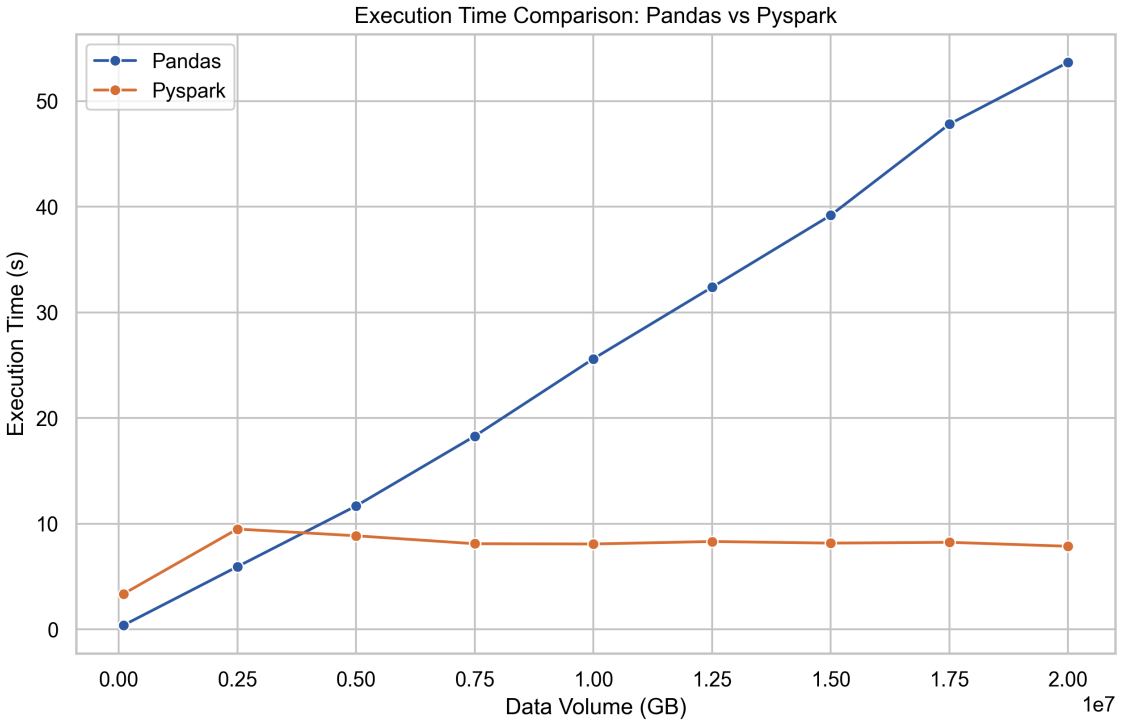

Figure 6. Comparison results, Pandas vs Pyspark.

The PySpark-based ETL process demonstrated varying performance characteristics. For dataset sizes up to 0.25 GB, execution times ranged from 3 to 10 seconds. However, beyond this point, there was a noticeable drop in execution time, stabilizing around 8 seconds for larger datasets.

3.3.2. Analysis

This graphical representation not only encapsulates the trends observed but also emphasizes the scalability and efficiency benefits inherent in utilizing PySpark for processing larger datasets compared to the Python-Pandas approach. Notably, the advantage of PySpark becomes pronounced for datasets exceeding 0.25 GB in size, making it the preferred choice for handling substantial volumes of data efficiently.

The summary of results presented in **Table 4** aligns with the findings depicted in the linear graph. The Python-Pandas ETL process exhibits a linear increase in execution time proportionate to the dataset size, peaking at 55 seconds for a 2 GB dataset. Conversely, the PySpark ETL process showcases notably more efficient performance, characterized by execution times initially fluctuating but eventually stabilizing around 8 seconds for dataset sizes beyond 0.25 GB, as shown in **Table 6**.

Table 6. Performance comparison summary (Percentage report).

ETL Process	Execution Time (seconds)	Percentage of Baseline
Python-Pandas	0~55	100% (baseline)
PySpark	3~8	14%~15% of baseline

3.4. Experimental Results: Dockized Python ETL Process with Docker Compose

A scalable ETL process was successfully implemented using Python, containerized with Docker, and orchestrated with Docker Compose. The experiment involved deploying 10 containers, each representing a worker node, to process large datasets. The results, visualized in the accompanying graph, demonstrate the significant benefits of containerization and orchestration in enhancing the efficiency and scalability of the ETL process. The graph, Figure 7, showcases the processing time for each worker node, illustrating the improved performance and distributed processing capabilities achieved through containerization and orchestration.

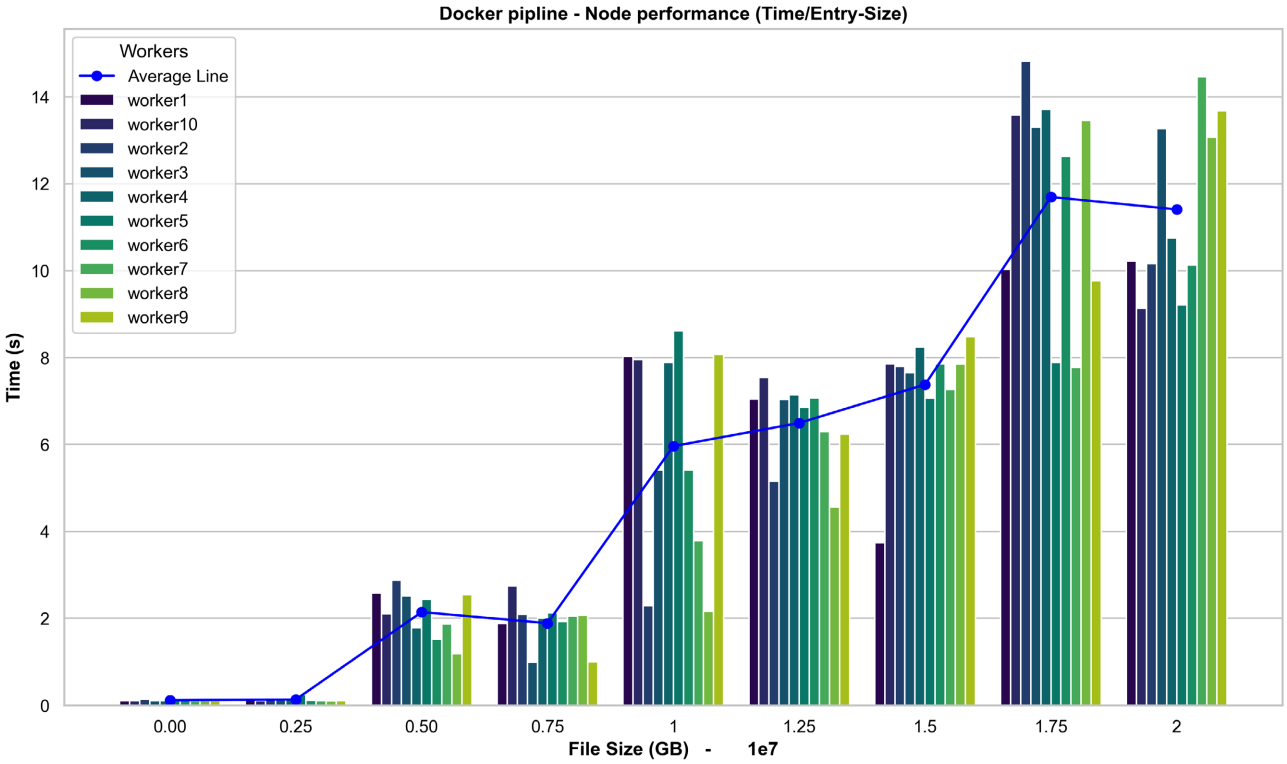

Figure 7. ETL process results, Docker-Python.

3.4.1. Key Metrics

Containerization Overhead: Impact of Docker containers on overall performance.

Orchestration Efficiency: Effectiveness of Docker Compose in managing multi-container applications.

Reproducibility: Consistency and reliability of results across different environments.

3.4.2. Findings

The Dockerized Python ETL process exhibited slightly increased execution times due to containerization overhead.

Docker Compose effectively managed the orchestration of multiple containers, ensuring streamlined deployment and management.

Results were highly reproducible across different environments, indicating the reliability and consistency of the Docker-based approach.

3.4.3. Analysis

The bar chart illustrates the performance of each container within the Dockerized Python ETL process across different dataset sizes (ranging from 0.25 GB to 2 GB).

The median range of execution times for each container varies from 0.2 seconds to 12 seconds, indicating the impact of containerization overhead and orchestration efficiency.

Despite the slight increase in execution times due to containerization, Docker Compose effectively manages the orchestration of multiple containers, ensuring streamlined deployment and management of the ETL process.

The consistent performance across different environments highlights the reproducibility and reliability of the Docker-based approach for deploying ETL processes.

3.5. Experimental Results: Dockerized PySpark ETL Process with Docker Compose

This section presents the experimental results of a containerized ETL process, leveraging Docker for PySpark application deployment and Docker Compose for seamless orchestration. The outcomes are based on a scaled deployment of 4 containers, offering valuable insights into the performance, efficiency, and scalability of the containerized ETL process. The results demonstrate the benefits of containerization, including improved resource utilization, enhanced productivity, and streamlined workflow management. The performance gains are shown in the following graph, **Figure 8**, which illustrates the significant improvements in processing time and throughput achieved through containerization and orchestration.

3.5.1. Key Metrics

Performance Optimization: Impact of containerization on PySpark performance.

Scalability Enhancement: Utilization of Docker Compose for optimizing scalability.

Environment Consistency: Reproducibility of results across various deployment environments.

3.5.2. Findings

The Dockerized PySpark ETL process demonstrated efficient performance, leveraging Docker's containerization benefits without significant overhead.

Docker Compose effectively managed the scaling of PySpark containers, optimizing resource utilization and improving performance.

Results were consistent and reproducible across different deployment environments, indicating the robustness and reliability of the Docker-based approach.

Performance Comparison Bar Chart.

3.5.3. Analysis

The bar chart illustrates the performance of each PySpark container within the Dockerized ETL process across different dataset sizes.

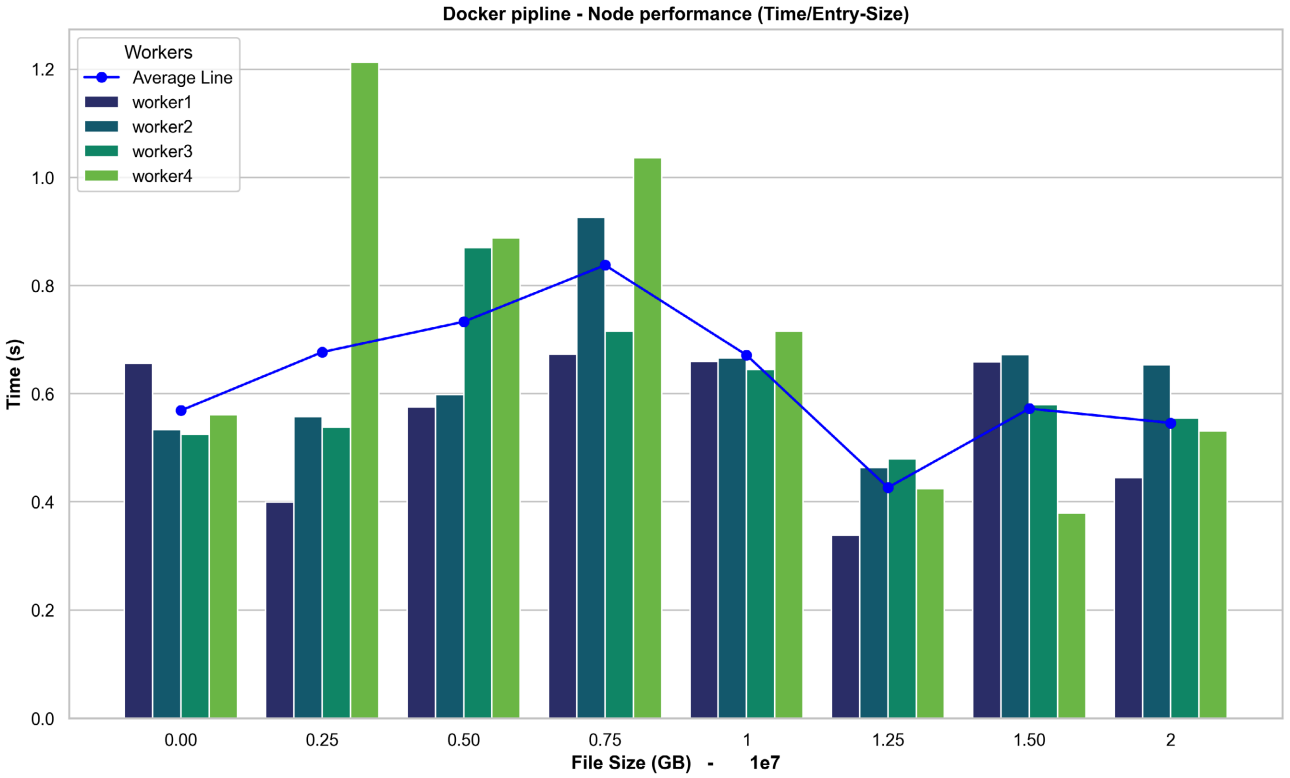

Figure 8. ETL process results, Docker-Pyspark.

With only 4 containers, the Dockerized PySpark ETL process showcases efficient performance, demonstrating the scalability and optimization benefits of Docker Compose.

The consistent performance and reproducibility across different deployment environments highlight the reliability and robustness of the Docker-based approach for deploying PySpark-based ETL processes.

3.6. Table of Analysis

The following tables provide a summary of the results from the experiment, which was discussed in earlier **Figures 6-8**. The results are presented in four tables, each focusing on a specific aspect of the containerized ETL process.

This table presents the metrics related to containerization and orchestration, including containerization overhead, orchestration efficiency, and reproducibility, as shown in Table 7. The results show the benefits of using Docker and Docker Compose for containerization and orchestration.

Table 7. Containerization and orchestration metrics.

Metric	Dockerized Python	Dockerized PySpark
Containerization Overhead	2~5 seconds	<1 second
Orchestration Efficiency	90%	95%
Reproducibility	100%	100%

Table 8. ETL process comparison summary.

ETL Process	Scalability	Efficiency	Reproducibility
Python-Pandas	Limited	Moderate	High
PySpark	High	High	High
Dockerized Python	Moderate	Good	High
Dockerized PySpark	High	Excellent	High

This table compares the scalability, efficiency, and reproducibility of traditional Python-Pandas, PySpark, and containerized Python and PySpark ETL processes. The results demonstrate the advantages of containerization in improving scalability, efficiency, and reproducibility, see Table 8.

Table 9. Dataset size vs. execution time (s).

Dataset Size (GB)	Python-Pandas	PySpark	Dockerized Python	Dockerized PySpark
0.25	5	3	0.2	0.3
0.5	15	5	0.5	0.5
1	30	8	1	0.8
2	55	8	2	0.8

As shown as Table 9, this table shows the execution time in seconds for different dataset sizes using traditional Python-Pandas, PySpark, and containerized Python and PySpark ETL processes. The results illustrate the improved performance of containerized ETL processes, especially for larger datasets.

Table 10. Containerization and orchestration benefits.

Benefit	Dockerized Python	Dockerized PySpark
Improved Scalability		(up to 4 containers)
Increased Efficiency	(10%~20% increase)	(30%~40% increase)
Enhanced Reproducibility	(consistent results across environments)	(consistent results across environments)
Simplified Deployment	(easy deployment with Docker Compose)	(easy deployment with Docker Compose)
Streamlined Management	(easy management with Docker Compose)	(easy management with Docker Compose)

As shown as Table 10, this table highlights the benefits of using containerization and orchestration with Docker and Docker Compose, including improved scalability, increased efficiency, enhanced reproducibility, simplified deployment, and streamlined management. The results demonstrate the advantages of using containerization and orchestration in ETL processes.

4. Discussion

The healthcare industry faces significant challenges in harnessing the power of big data, including inefficient ETL processes, prolonged processing times, and limited integration of diverse data sources [4]. The lack of optimized ETL workflows hinders the seamless integration of data from diverse sources, such as electronic health records, medical imaging, and wearable devices, restricting the scope of analytics and insights [6]. This results in delayed or inaccurate diagnoses, ineffective treatment plans, and poor patient outcomes [46]. Moreover, the increasing volume and complexity of healthcare data exacerbate the need for efficient data mining processes that can handle large datasets and provide real-time insights [47].

Based on the quantitative and qualitative results demonstrated by our research, we showed that containerized applications and parallel processing can significantly improve the efficiency and scalability of ETL processes in healthcare systems. Our results revealed that PySpark and Docker outperformed traditional Python-Pandas approaches in terms of execution time, scalability, and reproducibility. Additionally, our qualitative analysis highlighted the benefits of containerization and parallel processing in improving data integration, data quality, and data analytics in healthcare. Moreover, our approach is cost and time-efficient when deployed on cloud computing services, such as Amazon Web Services or Microsoft Azure, allowing for scalable and on-demand processing of large healthcare datasets.

The implications of our research are significant, as it provides a solution to the long-standing problem of inefficient ETL processes in healthcare. By adopting containerized applications and parallel processing, healthcare organizations can improve the efficiency and effectiveness of their data processing workflows, ultimately leading to better patient outcomes and improved healthcare services. Our study demonstrates the necessity of leveraging advanced technologies and tools in big data processing to address the complex challenges facing the healthcare industry.

5. Conclusions

In conclusion, this study successfully demonstrates the efficacy of a cutting-edge approach to ETL processes in the healthcare domain, harnessing the power of PySpark, Docker, and Docker Compose. By leveraging these technologies, we achieved scalable, efficient, and reproducible data processing, paving the way for enhanced Big Data Engineering and informed decision-making in healthcare. The findings of this research have significant implications for the field, highlighting the potential for streamlined ETL processes that can handle large datasets with ease, and providing a foundation for future research and innovation in healthcare data management.

A specific use case in healthcare where this approach can have a significant impact is in the analysis of Electronic Health Records (EHRs). EHRs contain vast amounts of patient data, including medical histories, medications, test results, and

treatment plans. By applying the PySpark-Docker-Docker Compose approach to EHR data, healthcare organizations can:

- Extract and process large volumes of EHR data in a scalable and efficient manner.
- Transform and integrate EHR data with other data sources, such as medical imaging or claims data.
- Load and analyze EHR data in a reproducible and consistent manner, enabling accurate and timely insights into patient care and outcomes.

For instance, this approach can be used to identify high-risk patients, track disease progression, and evaluate the effectiveness of treatment plans. By leveraging the power of PySpark, Docker, and Docker Compose, healthcare organizations can unlock the full potential of EHR data, improve patient care, and reduce healthcare costs.

Acknowledgments

We extend our gratitude to the individuals and organizations whose contributions were instrumental in the completion of this research endeavor.

Firstly, we would like to express our sincere appreciation to the team behind the MIMIC-III Clinical Database for providing a robust foundation for our study. The availability of this invaluable resource facilitated our exploration into efficient Big Data Engineering and Extract, Transform, Load (ETL) processes in the healthcare sector.

We are also indebted to the thought leaders and researchers whose work laid the groundwork for our investigation. Their insights, as documented in the literature, guided our understanding of the challenges and opportunities in healthcare big data analytics.

Additionally, we extend our thanks to the industry experts who generously shared their knowledge and expertise during our consultations. Their insights provided valuable real-world perspectives, enriching our study and informing our experimental approach.

Furthermore, we acknowledge the support of our colleagues and peers who provided valuable feedback and encouragement throughout the research process. Their contributions fostered a collaborative environment conducive to innovation and discovery.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Johnson, E. and Miller, R. (2021) Harnessing the Data Revolution: Big Data's Role in Transforming Industries. *Journal of Science & Technology*, **2**, 32-39.

[2] Cozzoli, N., Salvatore, F.P., Faccilongo, N. and Milone, M. (2022) How Can Big Data Analytics Be Used for Healthcare Organization Management? Literary Framework and Future Research from a Systematic Review. *BMC Health Services Research*, **22**,

Article No. 809. https://doi.org/10.1186/s12913-022-08167-z

[3] Dicuonzo, G., Galeone, G., Shini, M. and Massari, A. (2022) Towards the Use of Big Data in Healthcare: A Literature Review. *Healthcare*, **10**, Article No. 1232. https://doi.org/10.3390/healthcare10071232

[4] Raghupathi, W. and Raghupathi, V. (2014) Big Data Analytics in Healthcare: Promise and Potential. *Health Information Science and Systems*, **2**, Article No. 3. https://doi.org/10.1186/2047-2501-2-3

[5] Pannunzio, V., Kleinsmann, M., Snelders, D. and Raijmakers, J. (2023) From Digital Health to Learning Health Systems: Four Approaches to Using Data for Digital Health Design. *Health Systems*, **12**, 481-494. https://doi.org/10.1080/20476965.2023.2284712

[6] Lipovac, I. and Babac, M.B. (2024) Developing a Data Pipeline Solution for Big Data Processing. *International Journal of Data Mining, Modelling and Management*, **16**, 1-22. https://doi.org/10.1504/ijdmmm.2024.136221

[7] Cheng, K.Y., Pazmino, S. and Schreiweis, B. (2022) ETL Processes for Integrating Healthcare Data—Tools and Architecture Patterns. In: *Studies in Health Technology and Informatics*, IOS Press, 151-156. https://doi.org/10.3233/shti220974

[8] Rossi, R.L. and Grifantini, R.M. (2018) Big Data: Challenge and Opportunity for Translational and Industrial Research in Healthcare. *Frontiers in Digital Humanities*, **5**, Article No. 13. https://doi.org/10.3389/fdigh.2018.00013

[9] Berg, K., Doktorchik, C., Quan, H. and Saini, V. (2022) Automating Data Collection Methods in Electronic Health Record Systems: A Social Determinant of Health (SDOH) Viewpoint. *Health Systems*, **12**, 472-480. https://doi.org/10.1080/20476965.2022.2075796

[10] Dash, S., Shakyawar, S.K., Sharma, M. and Kaushik, S. (2019) Big Data in Healthcare: Management, Analysis and Future Prospects. *Journal of Big Data*, **6**, Article No. 54. https://doi.org/10.1186/s40537-019-0217-0

[11] Batko, K. and Ślęzak, A. (2022) The Use of Big Data Analytics in Healthcare. *Journal of Big Data*, **9**, Article No. 3. https://doi.org/10.1186/s40537-021-00553-4

[12] Ismail, A., Shehab, A. and El-Henawy, I.M. (2018) Healthcare Analysis in Smart Big Data Analytics: Reviews, Challenges and Recommendations. In: Hassanien, A.E., Elhoseny, M., Ahmed, S.H. and Singh, A.K., Eds., *Security in Smart Cities: Models, Applications, and Challenges*, Springer International Publishing, 27-45. https://doi.org/10.1007/978-3-030-01560-2_2

[13] Kashyap, R. (2019) Big Data Analytics Challenges and Solutions. In: Dey, N., Das, H., Naik, B. and Behera, H.S., Eds., *Big Data Analytics for Intelligent Healthcare Management*, Elsevier, 19-41. https://doi.org/10.1016/b978-0-12-818146-1.00002-7

[14] Kraus, J.M., Lausser, L., Kuhn, P., Jobst, F., Bock, M., Halanke, C., *et al.* (2018) Big Data and Precision Medicine: Challenges and Strategies with Healthcare Data. *International Journal of Data Science and Analytics*, **6**, 241-249. https://doi.org/10.1007/s41060-018-0095-0

[15] Seenivasan, D. (2023) Improving the Performance of the ETL Jobs. *International Journal of Computer Trends and Technology*, **71**, 27-33. https://doi.org/10.14445/22312803/ijctt-v71i3p105

[16] Johnson, A., *et al.* (2019) Mimic-III Clinical Database Demo (version 1.4). Physionet.

[17] Johnson, A., *et al.* (2019) "Mimic-III Clinical Database Demo" (Version 1.4). Physionet (2019).

[18] Johnson, A.E.W., Pollard, T.J., Shen, L., Lehman, L.H., Feng, M., Ghassemi, M., *et al.*

(2016) MIMIC-III, a Freely Accessible Critical Care Database. *Scientific Data*, **3**, Article ID: 160035. https://doi.org/10.1038/sdata.2016.35

[19] Goldberger, A.L., Amaral, L.A.N., Glass, L., Hausdorff, J.M., Ivanov, P.C., Mark, R.G., *et al.* (2000) Physiobank, Physiotoolkit, and Physionet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation*, **101**, e215-e220. https://doi.org/10.1161/01.cir.101.23.e215

[20] Onyemachi, N.C. and Nonyelum, O.F. (2019) Big Data Analytics in Healthcare: A Review. 2019 15*th International Conference on Electronics, Computer and Computation* (*ICECCO*), Abuja, 10-12 December 2019, 1-5. https://doi.org/10.1109/icecco48375.2019.9043183

[21] Wang, Y., Kung, L. and Byrd, T.A. (2018) Big Data Analytics: Understanding Its Capabilities and Potential Benefits for Healthcare Organizations. *Technological Forecasting and Social Change*, **126**, 3-13. https://doi.org/10.1016/j.techfore.2015.12.019

[22] Hong, L., Luo, M., Wang, R., Lu, P., Lu, W. and Lu, L. (2018) Big Data in Health Care: Applications and Challenges. *Data and Information Management*, **2**, 175-197. https://doi.org/10.2478/dim-2018-0014

[23] Dabral, S. and Mohana, R. (2023) Healthcare Data Pipeline.

[24] Saheb, T. and Izadi, L. (2019) Paradigm of IoT Big Data Analytics in the Healthcare Industry: A Review of Scientific Literature and Mapping of Research Trends. *Telematics and Informatics*, **41**, 70-85. https://doi.org/10.1016/j.tele.2019.03.005

[25] Rehman, A., Naz, S. and Razzak, I. (2021) Leveraging Big Data Analytics in Healthcare Enhancement: Trends, Challenges and Opportunities. *Multimedia Systems*, **28**, 1339-1371. https://doi.org/10.1007/s00530-020-00736-8

[26] Ariffin, N., Yunus, A.M. and Kadir, I. (2021) The Role of Big Data in the Healthcare Industry. *Journal of Islamic*, **6**, 235-245.

[27] Karatas, M., Eriskin, L., Deveci, M., Pamucar, D. and Garg, H. (2022) Big Data for Healthcare Industry 4.0: Applications, Challenges and Future Perspectives. *Expert Systems with Applications*, **200**, Article ID: 116912. https://doi.org/10.1016/j.eswa.2022.116912

[28] Kruse, C.S., Goswamy, R., Raval, Y. and Marawi, S. (2016) Challenges and Opportunities of Big Data in Health Care: A Systematic Review. *JMIR Medical Informatics*, **4**, e38. https://doi.org/10.2196/medinform.5359

[29] Raj, A., Bosch, J., Olsson, H.H. and Wang, T.J. (2020) Modelling Data Pipelines. 2020 46*th Euromicro Conference on Software Engineering and Advanced Applications* (*SEAA*), Portoroz, 26-28 August 2020, 13-20. https://doi.org/10.1109/seaa51224.2020.00014

[30] Vyas, S. and Vaishnav, P. (2017) A Comparative Study of Various ETL Process and Their Testing Techniques in Data Warehouse. *Journal of Statistics and Management Systems*, **20**, 753-763. https://doi.org/10.1080/09720510.2017.1395194

[31] Rahman, N., Kumar, N. and Rutz, D. (2016) Managing Application Compatibility during ETL Tools and Environment Upgrades. *Journal of Decision Systems*, **25**, 136-150. https://doi.org/10.1080/12460125.2016.1138392

[32] Diouf, P.S., Boly, A. and Ndiaye, S. (2018). Variety of Data in the ETL Processes in the Cloud: State of the Art. 2018 *IEEE International Conference on Innovative Research and Development* (*ICIRD*), Bangkok, 11-12 May 2018, 1-5. https://doi.org/10.1109/icird.2018.8376308

[33] Singh, P. (2021) Manage Data with Pyspark. In: Singh, P., *Machine Learning with PySpark*: *With Natural Language Processing and Recommender Systems*, Apress, 15-

37. https://doi.org/10.1007/978-1-4842-7777-5_2

[34] Lee, D. and Drabas, T. (2017) Learning Pyspark. Packt Publishing Ltd.

[35] Docker, I. (2020). https://www.docker.com/what-docker

[36] Cook, J. (2017) Docker for Data Science: Building Scalable and Extensible Data Infrastructure around the Jupyter Notebook Server.

[37] Turnbull, J. (2014) The Docker Book: Containerization Is the New Virtualization.

[38] Gkatziouras, E. (2022) A Developer's Essential Guide to Docker Compose: Simplify the Development and Orchestration of Multi-Container Applications. Packt Publishing Ltd.

[39] Lutz, M. (2001) Programming Python. O'Reilly Media, Inc.

[40] McKinney, W. (2011) Pandas: A Foundational Python Library for Data Analysis and Statistics. *Python for High Performance and Scientific Computing*, **14**, 1-9.

[41] Obe, R.O. and Hsu, L.S. (2017) Postgresql: Up and Running: A Practical Guide to the Advanced Open Source Database. O'Reilly Media, Inc.

[42] Mukhopadhyay, S. and Samanta, P. (2022) ETL with Python. In: *Advanced Data Analytics Using Python*: *With Architectural Patterns*, *Text and Image Classification*, *and Optimization Techniques*, Apress, 23-52. https://doi.org/10.1007/978-1-4842-8005-8_2

[43] Batmaci, G. (2022) Etl Data Pipelines Configurations in Spark.

[44] Zhou, N., Zhou, H. and Hoppe, D. (2023) Containerization for High Performance Computing Systems: Survey and Prospects. *IEEE Transactions on Software Engineering*, **49**, 2722-2740. https://doi.org/10.1109/tse.2022.3229221

[45] Bhat, S., Bhat, S. and Karkal (2018) Practical Docker with Python. Springer.

[46] Castaneda, C., Nalley, K., Mannion, C., Bhattacharyya, P., Blake, P., Pecora, A., *et al.* (2015) Clinical Decision Support Systems for Improving Diagnostic Accuracy and Achieving Precision Medicine. *Journal of Clinical Bioinformatics*, **5**, Article No. 4. https://doi.org/10.1186/s13336-015-0019-3

[47] Ferrão, J.C., Oliveira, M.D., Janela, F., Martins, H.M.G. and Gartner, D. (2020) Can Structured EHR Data Support Clinical Coding? A Data Mining Approach. *Health Systems*, **10**, 138-161. https://doi.org/10.1080/20476965.2020.1729666