Heman Koreri Israel Mnsen[1]
Master in Information Technology, Faculty of Information Technology,
Universitas Teknologi Digital Indonesia Yogyakarta,
Yogyakarta, Indonesia
email: hemanmnsen476@gmail.com

Bambang Purnomosidi D.P
Master in Information Technology, Faculty of Information Technology,
Universitas Teknologi Digital Indonesia Yogyakarta,
Yogyakarta, Indonesia
email: bpdp@utdi.ac.id

Rikie Kartadie
Department of Computer Engineering,
Faculty of Information Technology
Universitas Teknologi Digital Indonesia,
Yogyakarta, Indonesia
email: rikie@utdi.ac.id

Didi Kurnaedi
Department, Faculty of Information Systems
Information Systems,
STMIK PGRI,
Tanggerang, Indonesia
email: ddk@pgri.id

Data Pipeline Architecture for Academic Information System at Akademi Teknik Biak

In development a information system Intergrated, Architecture planning is the first step must be established. The planning of development in a information system is needed in order to a system can be running according to necessity. The data is used for this research, that is internal data of Akademi Teknik Biak College and external data of Institution of high education service at IV area in Biak Papua. The main goal of this research is design architecture pipelines data of ATB college. The architecture of pipelines is used for carrying resources of big data from one area to the other area in far distance to be efficiency. The method is used for this research, that is Estract – Transform-Load (ETL). The process of estract data is needed a special supporting library on apache spark in using library spark session. This spark session is established in order to call data of Akademi Teknik Biak college with csv extension can be run on apache spark. After the process of estract is established, apache spark will read data with csv extension and establish transform data. The process of transform data csv extension will be loaded in to a frame data as a output of processing ETL The result of research is apache spark technology can be easy for writers in design process information system of Akademi Teknik Biak and to be one of the best solution in processing Estract Load Transform (ETL) data with the big scale and real-time.

***KeyWords**: Apache Spark, Pipeline Architecture, ETL, Information Systems, DataFrame.*

This Article was:
submited: 11-06-24
accepted: 20-04-23
publish on: 20-07-24

How to Cite:
H. K. I. Mnsen, et al, "Data Pipeline Architecture for Academic Information System at Akademi Teknik Biak", Journal of Intelligent Software Systems, Vol.3, No.1, 2024, pp.1–6, 10.26798/jiss.v3i1.1335

1 Introduction

Previously, the Higher Education Service Institution Region XIV Papua only managed universities in two provinces, namely Papua and West Papua. However, in an effort to accelerate development in the provinces of Papua and West Papua through the Ministry of Home Affairs, four new provincial regions were established: Southwest Papua, Central Papua, Highland Papua, and South Papua. This expansion undoubtedly presents a significant challenge and responsibility for the Higher Education Service Institution Region XIV in accommodating and managing all the universities spread across these six provinces. To ensure the creation of accurate, non-overlapping, and consistent data, an integrated system is necessary. The planning for the development of a management information system is essential to ensure that the developed system operates according to the needs. A management information system is a collection of interactions between information systems that function to process data to provide information used by all levels of management. A management information system allows an organization to better control all activities to be more organized, thereby yielding the best results for the organization's continuity. The implementation of a management information system has become commonplace as it guarantees the success of a company. The management information system includes not only computer-based activities but also all activities, whether computer-based or not, because the information system must bind all elements within a company to streamline business processes[1].

In recent years, data engineering has become increasingly important due to the explosion of data generated by businesses, governments, individuals, and education sectors. When building an integrated information system, planning the architecture of data pipelines is the first step that must be taken. A data pipeline architecture is a series of data processing elements connected in sequence, such that the output of one element becomes the input for the next [2]. Conceptually, data pipelines are pathways from a source to a destination, passing through various transformations for different analytical applications. The first step involves connecting to raw data sources and fetching the data load using REST API services. The system processes the load, addressing data integrity issues such as redundant data, skipped data, updated and deleted data, and data type changes for specific fields based on the available schema at the source. The final step is loading the cleaned data into a cloud-based database [3].

The three interdependent data integration processes known as "Extract-Transform-Load" (ETL) are used to retrieve data from one database and perform specific processes to achieve the desired outcomes. With the expansion into four new provinces, the role of technology is crucial in building a data infrastructure between the Akademi Teknik Biak campus and the Higher Education Service Institution Region XIV. Given the existing challenges, the author proposes the development of an integrated data infrastructure system between the Higher Education Service Institution Region XIV and the Akademi Teknik Biak campus. Given the expansion into four new provinces, the data management challenges

[1]Corresponding Author.

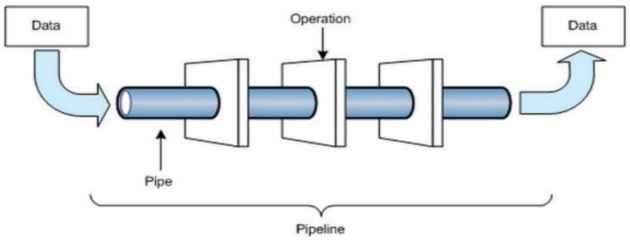

Fig. 1 Pipeline architecture

have intensified, requiring a robust solution that can handle large-scale, real-time data processing efficiently. Apache Spark, with its proven speed and real-time processing capabilities, presents an ideal solution. The proposed data pipeline architecture, leveraging Apache Spark, ensures seamless data integration and analytics across all universities in the region. Specifically, Spark's scalable nature allows it to manage the increasing volume of data while maintaining processing efficiency. Furthermore, its fault-tolerance, achieved through Resilient Distributed Datasets (RDDs), guarantees data integrity and system reliability, a critical aspect given the geographical and administrative complexities of Papua's new provincial structure.

1.1 Conceptual Theory.

1.1.1 Pipeline Architecture. Data pipeline architecture consists of a series of data processing elements connected in a sequence, where the output of one element becomes the input for the next. This architecture is designed to facilitate the efficient flow of data through various processing stages, ensuring that each component builds upon the previous one to enhance the overall effectiveness of data management and analysis [2]. Processing and designing data pipelines is highly efficient, scalable, and cost-effective. This becomes particularly crucial in real-time analytics for decision-making [4]. Data engineering is a broad field, but not every data engineer needs to master the entire range of skills. In this section, a data engineer focuses more specifically on how to build a framework or outline for data engineering or pipeline architecture, and then can proceed through more detailed descriptions that depict specific roles within data engineering.

Several phases in pipeline architecture include:

a. **Ingestion** This is the stage where necessary data is collected.
b. **Processing** This involves the manipulation and transformation of data to achieve the desired outcomes.
c. **Storage** This is the phase where the final results are stored for quick retrieval.
d. **Access** This provides the tools or users with the capability to access the processed and stored final outcomes.

Figure 1 illustrates the components required for software engineering in the ETL/ELT processes, stream processing, and Complex Event Processing. In the pipeline portion, data is processed within database engineering where it is managed in data lakes, warehouses, or sandboxes. The ETL/ELT process can handle various sources including relational databases, NoSQL, or files in formats such as XML, JSON, CSV, etc. After the ETL/ELT process is completed in Data Lakes or Warehouses, the results can be further processed using software engineering into roles like data scientist, data analyst, or even applications. The pipeline represents the entire application definition, including data inputs, transformations, and outputs [5].

1.1.2 Apache Spark. Apache Spark is a project developed by Matei Zaharia in 2009 at UC Berkeley's AMPLab. Several features of Spark make it a favorable choice, such as its ease of use

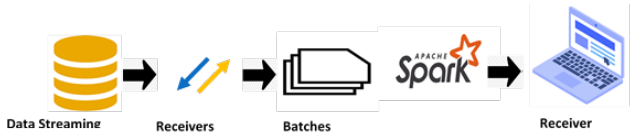

Fig. 2 Apache spark architecture

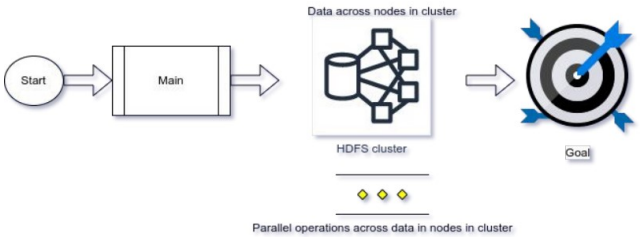

Fig. 3 Parallel operation across data in nodes in cluster

Apache Spark is simple to install. Speed is another significant advantage; Apache Spark operates at incredible speeds compared to Hadoop MapReduce. Apache Spark can run anywhere it operates on Hadoop, in the cloud, or even in a standalone configuration. Additionally, Apache Spark simplifies map and reduce operations. [6] It is an open-source analytics engine and data processing engine used for handling large-scale distributed datasets. Apache Spark's architecture consists of a master and slaves (referred to as driver and workers) mediated by a cluster manager. The cluster manager can use standalone, YARN, or Mesos setups. [7] Building a data pipeline to handle distributed streaming data processing using Apache Spark involves using APIs like Spark Streaming. Apache Spark is built within a cluster, meaning that data is processed in a distributed mode, which helps to address issues when there is an increase in the volume of data processed. The processing within Apache Spark is linear, meaning the running time depends on the amount of data and the number of nodes used; more nodes result in less processing time, and larger data volumes require more running time [8]. The overall architecture of Apache Spark can be seen in Figure 2.

Apache Spark also supports APIs accessible from various programming languages such as Java, Scala, R, and Python. It is used by data scientists and developers to perform ETL processes on data warehouses and data lakes from IoT devices, sensors, and other data sources. The extraction process can handle a wide variety of data, including both streaming and non-streaming data. Spark Streaming is a library provided in Apache Spark for scalable and fault-tolerant stream processing. Data can be ingested from many streaming services (such as Apache Kafka or Amazon Kinesis) or TCP sources, and processed using advanced algorithms (including machine learning and graph processing).

1.2 Resilient Distributed Dataset (RDD).

The Resilient Distributed Dataset (RDD) is a fundamental data structure in Apache Spark. RDDs are immutable, fault-tolerant, and distributed across various nodes in the Apache Spark cluster. Spark Streaming incorporates a high-level abstraction for continuous data streams, called DStream (Discretized Stream), which is internally represented as a series of RDDs. Each RDD in a DStream collects data from a specific time window, and all operations performed on a DStream are passed on to the underlying RDDs [9]. Spark was created using a concept called Resilient Distributed Dataset (RDD). Resilience is demonstrated by the ability of an RDD to roll back to its original state in the event of potential data loss due to unforeseen problems at a node in the Apache Spark cluster. Data written in an RDD is partitioned and distributed across various nodes. Therefore, if one node crashes or fails, the next node will provide a backup [6].

From Figure 3, it is explained how input data forms RDDs which

```
# spark is an existing SparkSession
df =
spark.read.json("examples/src/main/resources/people.json")
# Displays the content of the DataFrame to stdout
df.show()
# +----+-------+
# | age|   name|
# +----+-------+
# |null|Michael|
# |  30|   Andy|
# |  19| Justin|
# +----+-------+
```

Fig. 4 Data Frame

are partitioned into chunks and distributed across all nodes in the Spark cluster, with each node then performing calculations in parallel. RDDs are essential in data science and artificial intelligence, which require the processing of data in very large volumes. The MapReduce model is often considered too slow, hence the need for a fault-tolerant big data framework. MapReduce is a programming model released by Google that can be used to process large-scale data in a distributed and parallel manner across a cluster of thousands of computers [10]. Resilient Distributed Dataset (RDD) is used when low-level access involving transformations and actions, and full control over the dataset are desired. Additionally, RDDs are utilized when dealing with unstructured data such as streaming tasks. RDDs are also preferred when data manipulation using functional programming paradigms is necessary. In RDDs, schema is not as critical. RDDs are employed if optimization and performance benefits available along with DataFrames and Datasets for structured and semi-structured data are sought.

There are two categories of operations that can be performed on RDDs: transformations and actions. Transformations process filters, access, and modify an RDD to produce a new RDD. If the processed RDD is only on one partition, it is called narrow transformations; if it spans more than one partition, it is called wide transformations. Actions perform a computation or aggregation process to produce a specific value or generate an exception if a problem occurs. For managing structured data (table format), Spark SQL, Datasets, and DataFrames are used. A Dataset is a distributed collection of data. The API for Dataset is only available in Scala/Java, while Python and R do not have Datasets because they are already accommodated by DataFrame. Datasets in Scala/Java are necessary for DataFrame operations, where a DataFrame is a Dataset organized into named columns. DataFrames can be generated from various data sources

2 METHODS

2.1 Literature Study. The literature study was conducted by reading and reviewing several national and international journals and books relevant to the research topic. This research employs approaches from various studies [2] noting that raw data cannot be easily interpreted in its unprocessed form. Consequently, there is an increased recognition of the existential properties of business entities, including organizational management, market capabilities, and consumer feedback. This study utilized data pipeline technology for the extract-transform-load (ETL) process using client, consumer, and employee data. The results demonstrated that data pipeline technology is capable of addressing existing business challenges and obtaining directly processed data at the end of the data synchronization cycle. Raw data is difficult to consolidate or unite, so various aspects of the existential characteristics of raw data cannot be directly queried because they are not available to the company, such as market competence. Commodity data, customer data, and employee data are used in the extract-transform-load (ETL) process. Using the extract-transform-load (ETL) method can overcome corporate challenges and utilize intermediary cloud facilities known as a data lake, where data pipeline technology can receive and then load data into a database to be processed at the end of the data synchronization cycle [3].

Constructing a data lake architecture to monitor circulating online news [11]. The results from the architecture built involved combining and standardizing the data structure of online news from several online news channels and then streaming it in real-time to populate the data lake. The results of using a data lake architecture for online news will be stored in MongoDB, which serves as a database to store all data, both short-term and long-term. Ultimately, this data lake will serve as a facility to accommodate, explore, and analyze circulating online news data.

From the above studies, the author observes that many colleges, universities, and large companies still lack a good system for managing data effectively, from data storage to speed and convenience of system use. Some systems have been built but are not yet optimally functional. In this research, the author will develop an integrated data infrastructure at the Akademi Teknik Biak campus using Apache Spark technology for the extract-transform-load (ETL) process due to its speed in processing data in real-time and its capability to handle large volumes of data.

To ensure that other researchers can replicate our findings, we provided detailed information on the setup and configuration of Apache Spark, including the version used (3.3.2) and the specific settings for the SparkContext and SparkSession. Additionally, the versions of supporting software like JDK (Java Development Kit) version 20.0.1 and Anaconda version 4.14.0 were mentioned to provide a complete environment setup. The ETL process, crucial for data handling in our study, was detailed with the exact commands used for data extraction, transformation, and loading. This includes the paths to the CSV files used and the specific DataFrame operations performed in Apache Spark, allowing for clear steps that can be followed or adapted by others in similar or different contexts.

2.2 Designing the Data Pipeline Architecture for Akademi Teknik Biak. The design of this architecture is based on analysis and direct observation by the author, derived from two locations: the Akademi Teknik Biak campus and the Higher Education Service Institution Region XIV.

(1) **Data from Akademi Teknik Biak The campus data includes:**
   (a) **Faculty Data** - Information about faculty members, including qualifications, departments, and contact details.
   (b) **Student Data** - Demographic and academic details of students, such as enrollment numbers, course registrations, and academic performance.
   (c) **Course Data** - Details on courses offered, including course descriptions, credits, schedules, and prerequisites.
   (d) **Study Records (KRS)** - Academic records detailing students' course registrations for each semester.
   (e) **Scholarship Data** - Information regarding scholarship offerings and recipients.

(2) **Data from the Higher Education Service Institution Region XIV The data includes:**
   (a) **PPDIKTI** - National database for higher education data in Indonesia.
   (b) **SISTER** - Information system for education data management.
   (c) **SINTA** - Science and Technology Index for tracking publications and citations.
   (d) **SIMCITABMAS** - Management Information System for Research and Community Service.
   (e) **ARJUNA** - Accreditation ranking system for academic journals.

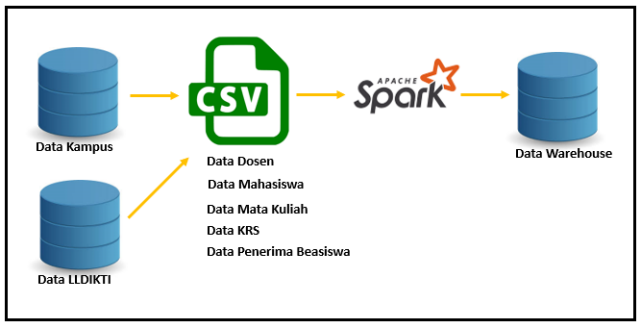

Fig. 5 Data Pipeline Architecture of akademi teknik biak

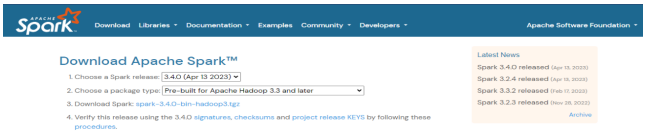

Fig. 6 Apache Spark Website Address

(f) **REG SDM** - Human resources registration system for academia

Based on the data listed above, an integrated data architecture is developed as illustrated in Figure 5. This architecture aims to streamline the flow and processing of data between the Akademi Teknik Biak and the Higher Education Service Institution Region XIV, ensuring efficient management and retrieval of information critical to both educational and administrative functions. The proposed architecture will facilitate seamless data integration and analytics capabilities, supporting decision-making and operational improvements across both institutions.

3 Results and Discussion

3.1 Data The creation. Data The creation of the data pipeline architecture utilizes two key sets of data: the internal data from Akademi Teknik Biak and the data from the Higher Education Service Institution Region XIV.

Data from Akademi Teknik Biak: (1.) **Faculty Data** - Information on faculty members. (2.) **Student Data** - Information on student demographics and academic progress. (3.) **Course Data** - Details of courses offered. (4.) **Study Records (KRS)** - Records of student course registrations. (5.) **Scholarship Data** - Information about scholarships available and awarded.

Data from the Higher Education Service Institution Region XIV; (1.) **PPDIKTI** - National higher education data database. (2.) **SISTER** - Education data management system. (3.) **SINTA** - Index for research and publication tracking. (4.) **SIMCITABMAS** - System for managing research and community services. (5.) **ARJUNA** - Journal accreditation system. (6.) **REG SDM** - Staff registration system.

3.2 Installation and Configuration of Apache Spark. Installation and Configuration of Apache Spark. The steps involved in setting up the ETL process using Apache Spark include:

3.2.1 Installation of Apache Spark. The version of Apache Spark used in this research is 3.3.2, which the author directly downloaded from its official website https://spark.apache.org/ To run Apache Spark, it is mandatory to install the JDK first. The version of JDK used is Java version 20.0.1, released on April 18, 2023

Table 1 A simple table

System Specifications		
1	Laptop	1 pc
2	Processor	Intel(R) Core(TM) i3-3217U CPU @ 1.80GHz 1.80 GHz
3	RAM	4GB
4	OS	Windows 10 64 Bit
5	Apache spark	Versi 3.3.2
6	JDK	Versi 20.0.1 2023-04-18
7	Anaconda spark	Versi 4.14.0
8	Scala	Versi 2.13.6

Fig. 7 Running Apache Spark

(1) Anaconda spark.
To manage and distribute packages in Python programming, the version of Anaconda used in this research is 4.14.0

(2) Scala
For integrating object-oriented and functional programming in a single high-level, concise language, the author uses Scala version 2.13.6. After gaining an overview of the software used in the system, an analysis is conducted to meet the system's requirements, starting from the hardware specifications that will be used as well as its software. The list of system specifications is as Tabel 1 follow:

3.2.2 Extract-Transform-Load (ETL). In this study, the data source selected for the ETL process is from Biak Technical Academy. The files from Biak Technical Academy are converted to CSV format before the extraction process in Apache Spark. To extract the data, supporting libraries in Apache Spark are utilized, specifically using the SparkContext. The SparkContext is essential as it enables the retrieval of student data in CSV format to be executed on Apache Spark. After the extraction process, Apache Spark reads the student.csv data and proceeds with data transformation. From the transformation process of student.csv, the data will be loaded into a Data Warehouse as the output of the ETL process. The ETL process conducted can be viewed in Figure 7.

3.2.3 Spark Extract-Transform-Load (ETL) Process in Apache Spark. For the ETL process in this study, the author uses data from Biak Technical Academy, which includes faculty data, student data, course data, study records (KRS), and scholarship data. This data, in raw CSV file format, is stored in a directory and then subjected to the ETL process in Apache Spark. The explanation in Figure 8 describes the stages and the ETL process in Apache Spark as follows:

3.2.4 Running Apache Spark.

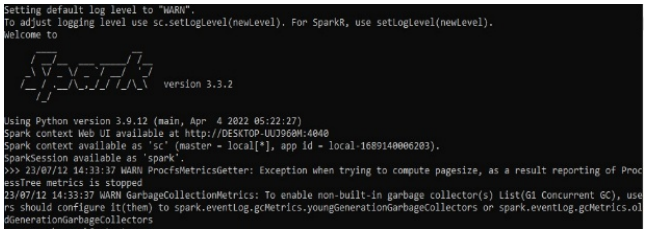

Fig. 8 Apache Spark Interface

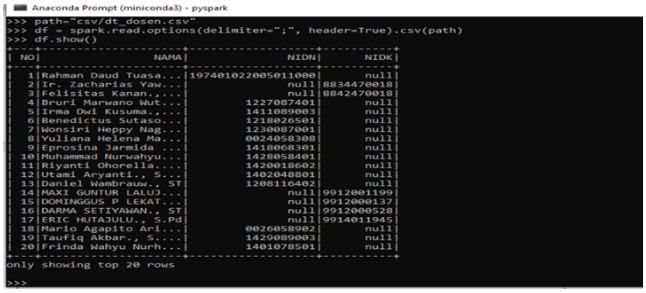

Fig. 9 Faculty DataFrame

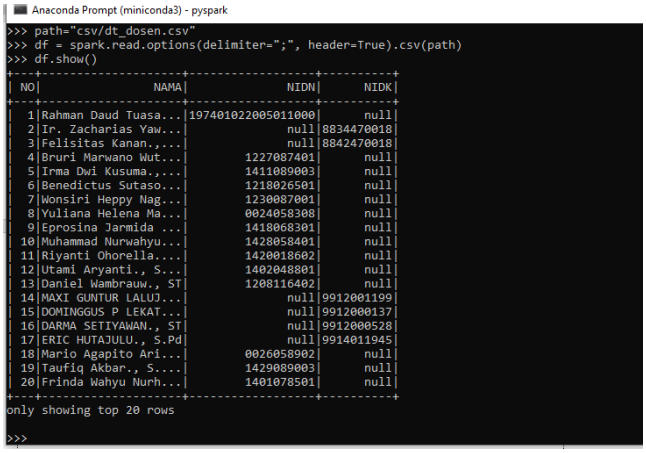

Fig. 10 Student DataFrame

ETL Process for Faculty Data. After successfully launching Apache Spark, the ETL process for faculty data begins with executing the following commands, show on Figure 9.

- sc = spark.sparkContext
- path="csv/dt_dosen.csv"
- df = spark.read.csv(path)
- df = spark.read.options(delimiter=";", header=True).csv(path)
- df.show()

ETL Process for Student Data. The ETL process for student data starts with executing the following commands, end show on Figure 10

- sc = spark.sparkContext
- path="csv/mhs.csv"
- df = spark.read.csv(path)
- df = spark.read.options(delimiter=";", header=True).csv(path)
- df.show()

ETL Process for KRS Data. The ETL process for KRS (study records) data starts with executing the following commands and show on Figure 11.

- sc = spark.sparkContext
- path="csv/dt_krs.csv"
- df = spark.read.csv(path)
- df = spark.read.options(delimiter=";", header=True).csv(path)
- df.show()

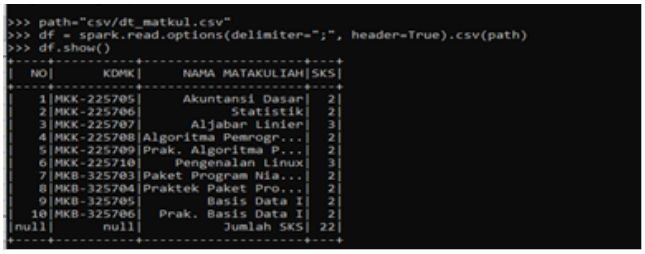

Fig. 11 KRS DataFrame

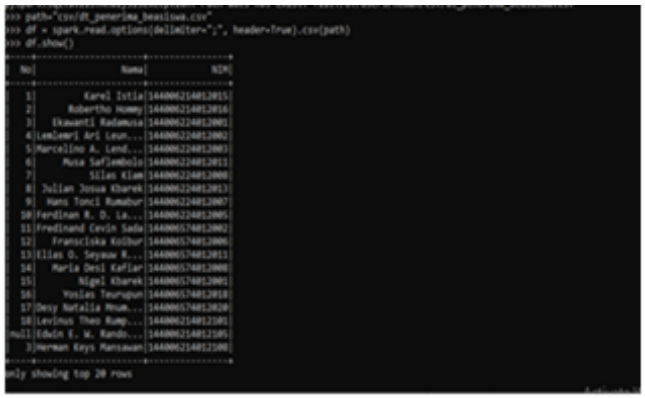

Fig. 12 Scholarship DataFrame

ETL Process for Scholarship Data. The ETL process for scholarship data begins with executing the following commands and show on Figure 12.

- sc = spark.sparkContext
- path="csv/dt_penerima_beasiswa.csv"
- df = spark.read.csv(path)
- df = spark.read.options(delimiter=";", header=True).csv(path)
- df.show()

4 Conclusions

Based on the results and discussions in this thesis, the following conclusions can be drawn, *Efficiency in System Design*, the integrated data system architecture developed using Apache Spark technology has facilitated the author in the process of designing the information system for Akademi Teknik Biak. This architecture has streamlined the handling of various data sources and enhanced the efficiency of data integration and processing.

Effectiveness of Apache Spark for Data Processing, Apache Spark has proven to be an effective solution for processing Akademi Teknik Biak campus data, particularly in handling large-scale and real-time ETL processes. Its robust processing capabilities ensure quick and reliable data transformation, loading, and analysis, which are crucial for maintaining up-to-date and accurate educational and administrative information.

References

[1] Nawawi, M. and Rubedo, H. (2021) 'Sistem Informasi Pengelolaan Data Aktivitas Penelitian dan PKM Dosen Universitas Wanita Internasional', Jurnal Manajemen Informatika (JAMIKA), 11(1), pp. 37–46. Available at: https://doi.org/10.34010/jamika.v11i1.3963.

[2] Science, C. and Science, C. (2023) 'Developing an ETL Pipeline for Data Analysis', International Journal of Computer Applications Technology and Research, 11(08), pp. 315–319. Available at: https://doi.org/10.7753/ijcatr1108.1004.

[3] Cottur, K. and Gadad, V. (2020) 'Design and Development of Data Pipelines', International Research Journal of Engineering and Technology, (May), pp. 2715–2718.

[4] Pogiatzis, A. and Samakovitis, G. (2021) 'An event-driven serverless etl pipeline on aws', Applied Sciences (Switzerland), 11(1), pp. 1–13. Available at: https://doi.org/10.3390/app11010191.

[5] Hesse, G. et al. (2019) 'Quantitative impact evaluation of an abstraction layer for data stream processing systems', Proceedings - International Conference on Distributed Computing Systems, 2019-July, pp. 1381–1392. Available at: https://doi.org/10.1109/ICDCS.2019.00138.

[6] Fauzi, R.A., Cholissodin, I. and Rahayudi, B. (2021) 'Pemanfaatan Spark untuk Analisis Sentimen Mengenai Netralitas Berita dalam Membahas Pemilu Presiden 2019 Menggunakan Metode Naïve Bayes Classifier', Jurnal PengembaSyarifuddin, M. (2020). Analisis Sentimen Opini Publik Mengenai Covid-19 Pada Twitter Menggunakan Metode Naïve Bayes Dan Knn. Inti Nusa Mandiri, 15(1), 23–28.ngan Teknologi Informasi dan Ilmu Komputer, 5(3), pp. 1070–1077.

[7] Rosianti, F., Bhawiyuga, A. and Amron, K. (2020) 'Pengembangan Platform Pengolahan Data Sensor Internet of Things berjenis Streaming dengan Komputasi Terdistribusi menggunakan Spark Streaming', 4(7), pp. 2102–2110.

[8] Aminudin, A. and Cahyono, E.B. (2019) 'Pengukuran Performa Apache Spark dengan Library H2O Menggunakan Benchmark Hibench Berbasis Cloud Computing', Jurnal Teknologi Informasi dan Ilmu Komputer, 6(5), p. 519. Available at: https://doi.org/10.25126/jtiik.2019651520.

[9] Belcastro, L. et al. (2022) Programming big data analysis: principles and solutions, Journal of Big Data. Springer International Publishing. Available at: https://doi.org/10.1186/s40537-021-00555-2.

[10] Oliviandi, S., Osmond, A.B. and Latuconsina, R. (2018) 'Implementasi Apache Spark Pada Big Data Berbasis Hadoop Distributed File System', e-Proceeding of Engineering, 5(1 Maret), pp. 1005–1012.

[11] Thenata, A.P. (2020) 'Data Pipeline Architecture with Near Real-Time Streaming Multiple Source Indonesian Online News Data Lake', JISA(Jurnal Informatika dan Sains), 3(1), pp. 32–37. Available at:https://doi.org/10.31326/jisa.v3i1.657.