

Implementation of Apache Airflow for Pipeline Automation: Case Study of Mental Health Issues on Social Media X

Putri Indah Cahyani^{1*}, Danur Wijayanto²

^{1,2}Information Technology Study Program, Faculty of Science and Technology, Aisyiyah University, Sleman, DI Yogyakarta.

E-mail: ^{1*}putriindah2343@gmail.com, ²danurwijayanto@unisayogya.ac.id

(Received: 14 Mar 2026, revised: 4 May 2026, 24 May 2026, accepted: 25 May 2026)

Abstract

The increasing use of social media as a platform for discussing mental health issues presents significant opportunities for psychological and social media research. However, the lack of automated, fail-resilient, and clean data provisioning remains a major infrastructure challenge. This study developed an automated ETL (Extract, Transform, Load) pipeline using Apache Airflow and Docker Compose to collect and process bilingual social media data from the X platform (Twitter). The system integrates a container-based architecture with persistent volumes to ensure infrastructure resilience and data continuity. A bilingual text preprocessing pipeline was implemented to support automated cleaning and normalization of English and Indonesian social media content. Experimental results show that the system successfully collected 1,768 unique tweets with a preprocessing throughput of 75 tweets per second and achieved a 39% reduction in average text length as part of preprocessing efficiency measurement. These measurements are intended to evaluate ETL preprocessing efficiency and throughput rather than analytical or linguistic quality. Resilience testing confirmed 100% data integrity with zero data loss across multiple failure scenarios, including container restarts and process suspensions. In addition, the implementation of ShortCircuitOperator successfully mitigated API credit exhaustion (HTTP 402) through a graceful degradation mechanism. This study contributes a reproducible and reliable data engineering framework for automated social media data collection and preprocessing, resulting in structured bilingual datasets ready for downstream analytical tasks. However, mental health in this study is positioned as a domain-specific case study, and the resulting dataset is not intended to serve as a clinical diagnostic tool.

Keywords: ETL Pipeline, Apache Airflow, Data Engineering, Mental Health, Docker

I. INTRODUCTION

The increasingly rapid development of technology has driven the increasing use of social media in everyday life [1]. One of the most widely used is X (Twitter), a place where users express themselves, interact, and share experiences in real-time [2]. Beyond communication, X has also become a space for self-expression, social support, and discussion of sensitive topics such as mental health, because users often feel more comfortable sharing personal experiences in an open but less formal environment [3][4][5].

Social media data, especially from X, has great potential for understanding user-generated text patterns and social phenomena. However, to obtain this data in a form ready for analysis, a systematic collection and processing process is required through the Extract, Transform, Load (ETL) stages. Although several previous studies have discussed social media-based mental health analysis, most have focused on sentiment or text classification tasks and have not emphasized

the automation of large-scale data collection and preprocessing [6].

Previous studies have explored social media data analysis and ETL automation from different perspectives. Hidayatullah and Maharani (2022) applied the Decision Tree algorithm to Twitter data for mental health-related detection [7]. Ramadhan et al. (2025) implemented an Apache Airflow-based ETL pipeline to automate data extraction and processing in a recommendation system, which improved the efficiency of data collection from various sources [8]. Fanani Mz et al. (2025) showed that optimizing database architecture and data load management in an ETL pipeline can accelerate large-scale data integration [9]. However, research specifically focusing on the automation of continuous social media text collection and preprocessing using ETL pipelines remains limited. Therefore, this study focuses on the development of an Apache Airflow-based ETL framework to produce clean, structured, and analysis-ready datasets from bilingual social media data. Unlike previous studies that primarily emphasized mental



health classification or recommendation systems, this research highlights the engineering aspect of automated data acquisition, preprocessing, and infrastructure resilience using Apache Airflow and Docker Compose. The main contribution of this study lies in building a fault-tolerant and reproducible ETL framework for bilingual social media text processing, while positioning mental health as a domain-specific case study rather than the primary analytical objective. Therefore, this study does not focus on sentiment analysis, mental health classification, or advanced feature engineering methods, but rather on the automation and reliability of the ETL infrastructure for social media text processing.

In data research, the ETL process plays an important role in preparing data for analysis. ETL is used to extract data from various sources, clean it, and organize it into a structured format [10]. In practice, the ETL process is still often carried out manually using simple scripts, especially when the data volume is relatively small, because it is easier to control and does not require many resources [11]. This manual approach has several limitations, such as long processing times, the risk of failure when there is a network disruption, and the difficulty of replicating the process on a large scale [9]. The manual approach is also less efficient and difficult to scale because any change in data structure or analytical requirements must be adjusted manually. Automating transformation logic has been shown to save up to 47% of development time and improve code quality by 23% compared to manual practices, making the implementation of an ETL automation system more efficient [12].

To address these issues, ETL pipeline automation is needed to organize, schedule, and monitor workflows efficiently. One solution that can be used to automate the ETL process is Apache Airflow, an open-source platform designed to manage workflow orchestration in a scheduled, monitored, and scalable manner [13]. Apache Airflow allows users to define process flows in the form of a Directed Acyclic Graph (DAG), monitor the status of each process in real-time, and access logs to trace the cause of a task failure [14]. By implementing Apache Airflow, the collection and processing of social media data can be carried out more efficiently, in a structured manner, and in a reproducible way [15]. The implementation of this automation also helps produce clean datasets that are ready to be used for further analysis, thereby improving the efficiency and reliability of social media data processing.

In addition to workflow automation, infrastructure reliability is also a critical factor in a production-ready ETL system. A containerized architecture using Docker Compose allows ETL processes to run portably, consistently, and be easily replicated across environments. However, containers are ephemeral by default, meaning that data stored in container storage can be lost when the container is restarted, removed, or when system failure occurs. This characteristic is one of the main challenges in container-based architectures, especially when storage is tied directly to the container lifecycle and does not guarantee data continuity [16]. This condition becomes a serious problem for ETL pipelines that process large amounts of data on a schedule, because data loss can lead to inconsistent results and process failure.

To address the volatile storage characteristics of container-based architectures, a persistent storage system is required to ensure data availability and consistency in stateful applications. Previous research confirms that deploying stateful applications in container-based environments requires a persistent storage mechanism to maintain data continuity and system stability in the event of restarts or component failures [17]. In implementations using Docker Compose, persistent storage can be achieved through named volumes or bind mounts that allow data to remain stored on the host even if the container is stopped or recreated.

Furthermore, data sustainability, structured capacity management, and proper volume configuration are also important. Studies on volume management in stateful applications show that adaptive storage capacity management contributes to better availability, efficiency, and operational stability of the system [18].

II. RESEARCH METHODOLOGY

This research was conducted by involving a series of ETL stages. ETL is a series of processes used to retrieve data from various sources, then filter, process, and combine it so that it is ready to be stored in a data warehouse [19]. This process also aims to maintain data consistency and quality by applying standard rules, so that the resulting data can support analysis and decision-making [20]. ETL works by processing data from one or more sources and combining it into a new, structured data source [21]. In general, ETL includes three main stages: extract, transform, and load, where the processed data is then loaded into a data warehouse for further analysis [22]. The ETL pipeline architecture in this study consists of three main stages: extract, transform, and load, which are orchestrated using Apache Airflow. The pipeline structure is shown in Figure 1.

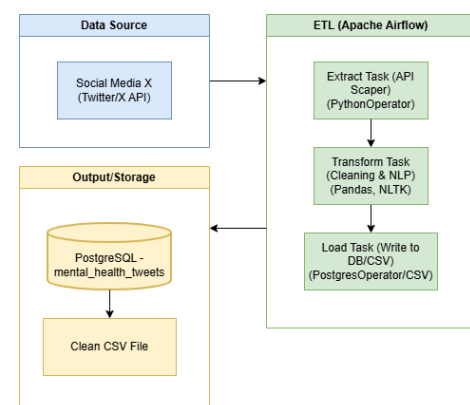


Figure 1. ETL Pipeline Using Apache Airflow

This ETL method is applied to build a workflow (pipeline) [23] that is able to extract data from certain sources, perform transformations to make the data clean and ready to use, and load the results into structured storage [24][25]. The entire process is run automatically and scheduled using Apache Airflow, which functions as a workflow orchestrator. Apache

Airflow allows for modular workflow setup, automatically executes tasks, monitors the duration of DAG execution, and ensures coordination between tasks so that the pipeline runs consistently [25].

To ensure reproducibility and portability, the entire system is deployed in a containerized environment using Docker Compose. The container architecture consists of four main services: (1) postgres as the Airflow metadata database to store DAG state and logs, (2) postgres-data as the mental health tweet data storage database, (3) airflow-webserver as the monitoring and pipeline management interface, and (4) airflow-scheduler as the DAG execution engine that runs all tasks on a schedule. Each container is configured with restart: always to ensure high availability and automatic recovery after system failure. This containerization approach enables consistent deployment across multiple environments without dependency conflicts, and facilitates system replication for further research.

Apache Airflow functions as a workflow orchestrator that manages, executes, and monitors the entire ETL pipeline process automatically and on a schedule. In this study, the pipeline is divided into four main tasks: (1) Extract Task that retrieves data from the X API, (2) Transform Task that cleans and normalizes data using NLP techniques with bilingual support (English and Indonesian), (3) Load Task that saves the final results into a PostgreSQL database with indexing optimization, and (4) Export Task that exports data to CSV format for portability. With this design, the ETL system is able to automatically collect and update data, maintain data consistency and integrity, and simplify the social media-based mental health analysis stage.

The ETL method consists of three main steps:

1. Extract

Extract (Data Extraction) is the initial stage of ETL, which is responsible for collecting data from various sources deemed relevant [26][27][28]. At this stage, the required tables or data are selected from the operational system, and then the data is collected so that it is ready to be processed in the next step [29]. This stage ensures that only data that meets the analysis needs is carried over to the next process [26].

In this study, the primary data source comes from the social media platform X (formerly Twitter) through the X API. The data extracted consists of tweet text, timestamps, user information, and metadata relevant to mental health analysis. Examples of extracted data include: tweet text content (e.g., posts containing emotional expressions or mental health indicators), posting time information (timestamps for temporal pattern analysis), user information (user ID, location, number of followers), and tweet metadata such as the number of retweets, likes, replies, and whether the tweet is a reply or a retweet. A similar data extraction approach, encompassing text content features, temporal features, user profile metadata, and engagement metrics, has been commonly used in mental health prediction research on Twitter [30]. In Indonesia, previous research adopted the X Search API as a means of extracting social media data, which was then processed through filtering and validation stages to identify information related to mental

health [31]. Mental health keywords are used merely to filter relevant content for the case study, not to infer individual mental health status. The resulting dataset is intended as a resource for further analysis rather than as a diagnostic tool.

The extraction process is carried out using Apache Airflow with PythonOperator to run data retrieval from the X API on a scheduled basis every day at 00:00 UTC (07:00 WIB) using the cron expression `0 0 * * *`. The system implements rate limiting handling to overcome API limitations (450 requests per 15 minutes) and a pagination mechanism to retrieve data from more than 100 tweets per request. Raw data is stored in JSON format in the `/data/raw/` directory for traceability and audit purposes. Thus, the system can automatically update the latest data without the need for manual intervention, so that the data collection flow remains consistent and supports continuous collection and preparation of social media text data related to mental health discussions.

2. Transform

Transform data (Data Transformation) is a crucial stage in the ETL (Extract, Transform, Load) process that functions to change raw, unstructured, or semi-structured data into a more organized and clean form so that it is ready to be used for analysis and reporting purposes [32]. At this stage, structuring, formatting, and providing the necessary attributes [33], and cleaning the data is carried out to maintain consistency and quality [26]. The transformation process ensures that data from various sources can be adjusted and changed into a uniform form so that it is ready to be loaded into the Data Warehouse [27].

In this study, the transformation phase follows general text preprocessing approaches commonly used in ETL-based data processing as described in [34], [35]. The preprocessing includes text cleaning and normalization processes such as case folding, removal of noise, normalization of whitespace, deduplication, and language filtering for English and Indonesian tweets. In practical implementation for social media data, additional handling is applied to remove or clean noisy elements such as URLs, emojis, mentions, hashtags, and special characters to improve data consistency and structure.

The transformation stage in this research is designed for automated data preparation within an ETL pipeline rather than advanced NLP feature engineering or semantic analysis. Therefore, the process is limited to data cleaning and standardization to ensure consistency and readiness of the dataset for downstream analytical tasks. Advanced linguistic processing such as translation, stemming, or lemmatization is not applied, as the focus of this stage is on ETL-oriented text standardization.

3. Load

Loading (Loading Data) is the final stage in the ETL process, namely entering data that has gone through the extraction and transformation process into the Data Warehouse or Data Mart [26]. In this stage, the integrated and standardized data is loaded into storage tables so that it can be accessed by applications or analysis needs [33]. The loading process can be carried out in stages or in bulk [33][27], depending on the

system needs and data volume. This stage ensures that the data is ready to be used for reporting, visualization, and decision-making [29].

In this research, the loading process is carried out by saving the cleaned data into two main storage forms:

a) PostgreSQL Database

PostgreSQL uses the bulk insert technique with executemany() to improve data loading performance. The mental_health_tweets table schema is structured with 14 columns that include tweet and user identities, original and preprocessed text content, time and language metadata, interaction metrics (likes, retweets, replies, and quotes), and account verification information.

b) CSV Export

In addition to the database, the data is also exported as a CSV file (mental_health_tweets_latest.csv), which can be used for exploration, visualization, or machine learning model training. The CSV export is performed after the database load is complete, ensuring data consistency between the two storage formats.

This load process is run automatically every day after the transformation is complete via the load_to_storage and export_database_to_csv tasks in the Airflow DAG. This mechanism enables the ETL system to consistently and continuously generate datasets ready for use in social media mental health analysis.

4. Containerization and Data Persistence

To ensure reproducibility, portability, and fault tolerance of the ETL system, the entire infrastructure is deployed using a containerized architecture with Docker Compose. The containerization approach enables the system to run consistently across multiple environments without dependency conflicts, and simplifies deployment and scaling. Figure 2 illustrates the container architecture and the implementation of persistent volumes, showing how each service is

interconnected to support the ETL pipeline and maintain data integrity.

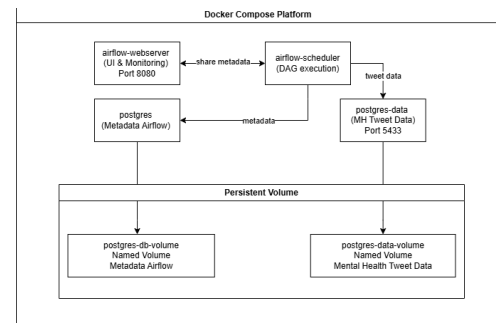


Figure 2. Container Architecture and Persistent Volumes System

The Docker Compose-based system architecture consists of four main containers: airflow-webserver, airflow-scheduler, postgres, and postgres-data. The system implements a persistent volume mechanism to separate data storage from the container lifecycle, ensuring data integrity even in the event of a container failure or restart.

III. RESULT AND DISCUSSION

This section explains the results of the implementation of the developed ETL system, as well as a discussion of the performance of each stage, starting from the data extraction process, transformation, and loading into structured storage, along with the constraints and adjustments made during the research.

A. ETL Pipeline System Infrastructure

The system infrastructure is built using a containerized architecture with Docker Compose to ensure portability and reproducibility as shown in Figure 3. The architecture consists of four main services running in isolated containers, as detailed in Section II.

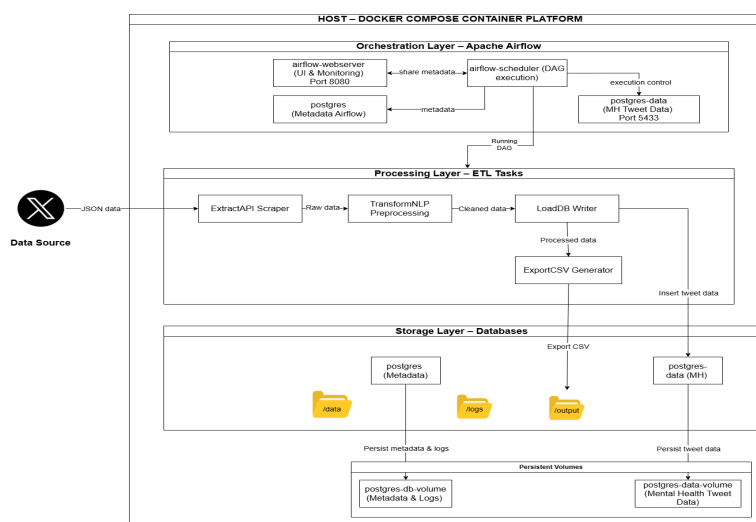


Figure 3. High-Level Architecture of ETL Pipeline System

JSON data from the X API is processed through extraction, NLP preprocessing, saving to PostgreSQL, and CSV export. Airflow manages workflow execution and metadata, while data and logs are stored persistently using volumes to maintain data continuity.

1. Resource Utilization

Monitoring resource usage indicates system efficiency in a production environment. Table 1 shows the CPU and memory usage of each container running the main service in a Docker Compose-based ETL system.

Table 1. Resource Utilization per Container

Container	CPU Usage	Memory
airflow-webserver	0.1 – 0.4%	626 MB
airflow-scheduler	3 – 28%	510–580 MB
postgres (airflow)	1-7%	54 MB
postgres (data)	0.01 – 0.1%	27 MB
Total (idle)	5 – 8%	1,217 MB
Total (active)	35%	1,285 MB

The idle state indicates the system is running without active DAG execution, while the active state occurs when the ETL pipeline is fully running. Total memory usage is approximately 1.2 GB, indicating the system can run on devices with mid-range specifications (at least 4 GB of RAM and 2 CPU cores).

From an analytical perspective, the relatively low resource usage in the idle state indicates that the container-based architecture is able to optimize resource utilization through service isolation. The 35% increase in CPU usage in the active state is primarily due to the computationally intensive NLP-based transformation process. This indicates that the system's primary bottleneck lies in the transformation phase, not in orchestration or data storage.

2. Persistent Volumes for Data Protection

The system implements persistent volumes as a fault tolerance mechanism at the infrastructure layer to prevent data loss in the event of a container failure. This configuration is shown in the code below.

```
volumes:
  postgres-db-volume: # Airflow metadata
  storage
  postgres-data-volume: # Mental health
  data storage
services:
  postgres-data:
    image: postgres:13
    volumes:
      - postgres-data-
    volume:/var/lib/postgresql/data
      - ./data:/opt/airflow/data
      - ./output:/opt/airflow/output
    restart: always
```

By separating the storage medium from the container's lifecycle, data persists despite system restarts, deletions, or

disruptions. This approach ensures data integrity is maintained regardless of the container's survival.

B. Orchestration with Apache Airflow

Apache Airflow is used as a workflow orchestrator to manage scheduling, dependencies between tasks, and ETL pipeline monitoring.

1. DAG Configuration and Data Redundancy Handling

The ETL pipeline runs daily at 00:00 UTC. The search_recent_tweets endpoint of API X retrieves data from the previous seven days, resulting in a six-day overlap between consecutive executions. The daily schedule was intentionally selected to enable near real-time collection of social media discussions related to mental health topics and to reduce the risk of data loss caused by temporary API limitations or system interruptions.

This overlap is intentionally maintained as a fault-tolerance strategy. To handle the resulting data redundancy, a database-level deduplication mechanism (ON CONFLICT DO NOTHING) is applied, ensuring that each tweet is stored only once without requiring additional validation at the application level. This configuration is shown in the code below.

```
ON CONFLICT (tweet_id) DO NOTHING
```

This approach ensures that each unique piece of data is stored only once without requiring additional validation at the application level, thereby increasing pipeline efficiency. This configuration is shown in the code below.

```
with DAG(
    dag_id='mental_health_etl_pipeline',
    default_args=default_args,
    description='ETL Pipeline for Mental
    Health Data from X (Recent 7 days)',
    schedule='0 0 * * *', # Daily execution
    at_midnight (00:00 UTC)
    start_date=datetime(2025, 12, 1),
    catchup=False,
    max_active_runs=1,
) as dag:
```

From an analytical perspective, the data overlap strategy combined with deduplication has proven effective in improving data reliability. Furthermore, using a database-level deduplication mechanism provides a simple yet efficient solution for maintaining data consistency.

2. Measurement Results and Performance Analysis

Testing was conducted between November 28, 2025, and March 2, 2026. Early on, challenges were encountered in the extraction process due to API credit limitations. Although the DAG was scheduled once per day, multiple retry attempts and pagination requests increased the total number of API calls per execution. Additionally, concurrent API usage during testing contributed to faster quota depletion. These factors

explain the relatively low success rate observed during the initial testing phase. To address this issue, a `ShortCircuitOperator` was used, which allows the pipeline to gracefully terminate when the API quota is exhausted, without causing a complete system failure. The results of these sessions are summarized in Table 2.

Table 2. DAG Execution Summary

Session	Date	Tweet	Duration
1	Nov 28, 2025	44	22 seconds
2	Nov 29, 2025	38	22 seconds
3	Dec 1, 2025	39	22 seconds
4	Jan 27, 2026	100	22 seconds
5	Jan 28, 2026	78	22 seconds
6	Mar 2, 2026	1,469	22 seconds
Total		1,768	22 seconds

Duration analysis was performed to identify execution time contributions and potential bottlenecks in the ETL pipeline. Table 3 shows the average execution duration of each task.

Table 3. Average Duration per Task (Successful Execution)

Task	Average	Contribution (%)
extract tweets	2.5 seconds	11.4%

Task	Average	Contribution (%)
transform tweets	19.5 seconds	88.6%
load to storage	0 seconds	0.0%
export database to csv	0 seconds	0.0%

The analysis shows that the transformation process is the main bottleneck, accounting for 88.6% of the total execution time. This is due to the multi-stage text preprocessing operations involved in cleaning and normalizing bilingual social media content.

Implementing `ShortCircuitOperator` provides a solution for automatically handling external system disruptions. This feature acts like a switch at the `check_api_quota` stage, ensuring the API is still usable before executing more demanding processes.

Figure 4 shows the implementation pattern of the `ShortCircuitOperator`. When the API quota is exhausted, the system is no longer forced to run until it encounters an error, but instead "jumps" or skips the next process (marked by the pink block). The system will only run to completion (marked by the green block) if the quota is available. This logic saves computing power because the heavy data cleanup process is not triggered if the data is not retrieved successfully.

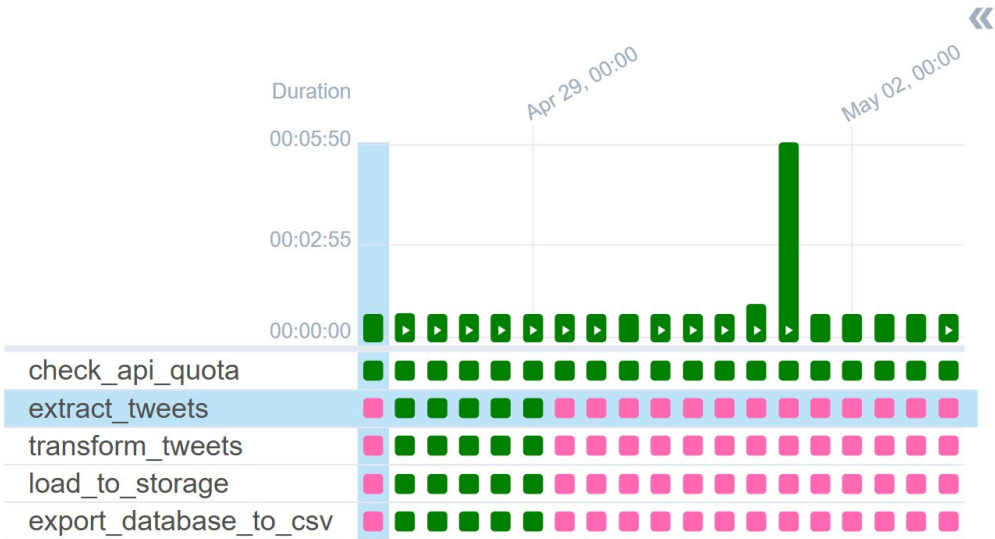


Figure 4. Grid View - DAG Execution

C. Data Persistence and Infrastructure Resilience Testing

Resilience testing scenarios are designed to represent common container failures in production environments, such as restarts, stop/starts, and container rebuilds. This approach adheres to the principles of failure and recovery in container-

based systems, emphasizing the separation of container lifecycles and persistent storage to maintain data consistency and accelerate service recovery [36].

Testing is performed to evaluate the system's ability to maintain data integrity under various failure scenarios. The detailed results of these tests are summarized in Table 4.

Table 4. Resilience Test Results

No	Failure Scenario	Frequency	Data Loss	Recovery Time	Success Rate
1	Restart container (postgres-data)	1x	0 rows	1.58 seconds	100%
2	Stop & start container (postgres-data)	1x	0 rows	21.44 seconds	100%
3	Suspend scheduler process (in-process)	1x	0 rows	30.36 seconds	100%
4	API credits exhausted (graceful skip)	4x	0 rows	Instant	100%
Total		7x	0 lines	Average: 13.35 seconds	100%

Resilience Testing Scenarios:

- Scenario 1, Container Restart: The 'postgres-data' container was restarted manually while no DAG was executing. The database reattached to the named volume in 1.58 seconds, with all 1,768 tweet records intact and verified immediately after recovery.
- Scenario 2, Container Stop & Start: The 'postgres-data' container was stopped completely for 15 seconds to simulate fullservice outage, then restarted. Total recovery time was 21.44 seconds. Post-recovery verification confirmed zero data loss (1,768 rows intact), demonstrating that named volumes preserve data independently of container lifecycle.
- Scenario 3, In-Process Orchestrator Suspension: To simulate in-process operational stress, the 'airflow-scheduler' container was suspended (docker pause) while a DAG run was queued and actively being processed. The suspension lasted 30.36 seconds. Upon resumption ('docker unpause'), the scheduler automatically recovered, resumed DAG processing, and restored healthy status within 20 seconds with zero data corruption or loss. This scenario demonstrates the system's ability to recover from mid-operation process-level failures.
- Scenario 4, API Credit Depletion (Graceful Degradation): When all API tokens returned HTTP 402 CreditsDepleted, the 'check_api_quota' ShortCircuitOperator immediately halted pipeline execution before any extraction was attempted. This occurred across 4 consecutive runs, with zero data loss and immediate recovery demonstrating fault-tolerant graceful degradation under external dependency failure. All four scenarios confirmed 100% data integrity (1,768 tweets before and after all tests) and 100% system recovery success rate, validating the effectiveness of Docker named volumes and Airflow's process recovery mechanisms.

D. ETL Pipeline Process

The ETL (Extract-Transform-Load) process is the system's core functionality, consisting of four sequential phases: Extract for data acquisition from the Twitter API, transform for text cleaning and preprocessing, Load for bulk insertion into a PostgreSQL database, and Export for generating portable CSV files. Each phase is implemented as a PythonOperator task in an Airflow DAG with error handling

and data validation to ensure data quality and pipeline reliability.

1. Extract

The extraction phase employed a keyword-based search strategy encompassing over 60 mental health-related terms. The resulting dataset was bilingual, consisting of English and Indonesian tweets without any automatic translation process, thus preserving the linguistic context and psychological meaning.

The query uses Boolean operators (OR) to combine multiple mental health-related keywords, as well as the -is:retweet and -is:reply filters to ensure only original content is collected. Additionally, a language filter (lang:en OR lang:id) is used to limit the data to two languages. An example implementation is shown in code below.

```
keywords = ['depression', 'anxiety', 'mental health', 'stress', 'depresi', 'stres', 'cemas', ...]
# 13 keywords total
query = f"({' OR '.join(keywords)}) -is:retweet -is:reply (lang:en OR lang:id)"
for tweet in tweepy.Paginator(client.search_recent_tweets, query=query, max_results=100).flatten(limit=500):
    tweets.append(tweet)
```

The code shows that the extraction process was performed by combining multiple keywords using the logical OR operator, as well as filtering the data to exclude retweets and replies to ensure higher quality and relevance. Furthermore, a language filter (lang:en OR lang:id) was used to limit the data to English and Indonesian. The profile of the extracted dataset is shown in Table 5.

Table 5. Dataset Characteristics

Metric	Value
Total tweets (after deduplication)	1,768
Number of unique users	1,601
Data collection period	Nov 28, 2025 – Mar 2, 2026
Successful extraction sessions	6 sessions

Metric	Value
Average tweets per session	295 tweets
Verified accounts	10 (0.57%)
Tweets with engagement	396 (22.4%)
Maximum likes	145
Maximum retweets	52
Valid data from cleaning	1,766 (99.9%)

The development of the amount of data in each extraction session is shown in Table 6, which shows a significant increase after the API limitations were successfully overcome.

Table 6. Data Acquisition per Extraction Session

Session	Date	Tweet	Cumulative Total
1	Nov 28, 2025	44	44
2	Nov 29, 2025	38	82
3	Dec 1, 2025	39	121
4	Jan 27, 2026	100	221
5	Jan 28, 2026	78	299
6	Mar 2, 2026	1.469	1.768

The monthly distribution in Table 7 shows that the majority of data was acquired in March 2026, indicating the success of the pipeline in efficiently retrieving historical data.

Table 7. Data Distribution per Month

Month	Tweets	Unique Users
November 2025	82	82
December 2025	39	39
January 2026	178	175
March 2026	1.469	1.307
Total	1.768	1.601

The temporal analysis in Table 8 shows a pattern of user activity, with the peak of the discussion occurring at 10.00 UTC.

Table 8. Activity Tracking Based on Time (UTC)

Time	Number of Tweets	Description
01:00	38	-
02:00	44	-
09:00	514	-
10:00	955	Peak activity
11:00	78	-
16:00	139	-

2. Transform

Transform implements a bilingual text preprocessing pipeline to standardize English and Indonesian social media content before storage. This code below shows the implementation of the text cleaning pipeline used in the data transformation process.

```
class TextCleaner:
    def clean_text(self, text):
        text = self.remove_urls(text)
        text = self.remove_emojis(text)
        text = self.normalize_text(text)
        text = self.clean_special_characters(text)
        text = self.normalize_whitespace(text)
        cleaned = text.strip()
        return cleaned
```

The preprocessing pipeline applies text cleaning, normalization, deduplication, and language filtering to improve data consistency and reduce text noise before storage. The implemented cleaning processes include URL removal, emoji removal, mention and hashtag cleaning, special character removal, case folding, and whitespace normalization. The implementation uses a modular TextCleaner class to support maintainability and reproducibility within the ETL workflow. Experimental results show a 39% reduction in average text length and a processing throughput of 75 tweets per second, indicating efficient preprocessing performance for automated ETL operations. These measurements are intended to evaluate preprocessing efficiency and ETL throughput rather than analytical quality or predictive capability.

3. Load

The load phase uses a bulk insert strategy with PostgreSQL for performance optimization and duplicate prevention. Code below shows the implementation of a data storage process using the bulk insert method with a duplicate prevention mechanism.

```
def load_to_database(tweets, **context):
    # Database connection
    conn = psycopg2.connect(
        host=os.getenv('POSTGRES_HOST'),
        database=os.getenv('POSTGRES_DB'),
        user=os.getenv('POSTGRES_USER'),
        password=os.getenv('POSTGRES_PASSWORD')
    )
    cursor = conn.cursor()

    # Bulk insert dengan batch size 1000
    execute_batch(cursor, """
        INSERT INTO mental_health_tweets
        (id, text, created_at, author_id,
        lang, cleaned_text)
        VALUES (%s, %s, %s, %s, %s, %s)
        ON CONFLICT (id) DO NOTHING
    """, tweets, page_size=1000)
    conn.commit()
    logging.info(f"Loaded {cursor.rowcount} tweets to database")
```

The use of ON CONFLICT DO NOTHING serves to prevent data duplication due to overlapping extraction processes, thereby maintaining dataset consistency.

4. Export CSV

In addition to being stored in the PostgreSQL database, the transformed data is also exported into CSV format to support advanced analysis and interoperability with various analytical tools. This export process is implemented in the code below.

```
df.to_csv("tweets.csv",          index=False,
encoding='utf-8')
```

This process generates structured datasets that are easy to use for machine learning, visualization, and statistical analysis purposes without relying directly on databases.

IV. CONCLUSION

This research successfully implemented a robust and fault-tolerant automated ETL pipeline using Apache Airflow and Docker Compose for collecting and processing social media data related to mental health discussions. The system architecture demonstrated efficient resource utilization and portability, operating with a modest memory footprint of approximately 1.2 GB and maintaining stability even during computationally intensive NLP transformations.

The integration of persistent volumes and a database-level deduplication mechanism (ON CONFLICT DO NOTHING) proved highly effective, achieving a 100% recovery success rate and zero data loss across various infrastructure failure scenarios, including container restarts and process-level suspensions.

Furthermore, the implementation of `ShortCircuitOperator` provided a proactive solution to external dependency failures, specifically addressing the HTTP 402 CreditsDepleted issue by gradually terminating execution without affecting system integrity.

From a data preprocessing perspective, the implemented bilingual text cleaning and normalization pipeline achieved a 39% reduction in average text length with a processing throughput of 75 tweets per second. The preprocessing workflow improved dataset consistency and reduced text noise, resulting in a structured dataset consisting of 1,768 unique tweets ready for downstream analytical tasks. The extracted dataset demonstrates that the implemented query strategy was effective in collecting relevant bilingual social media discussions associated with mental health topics.

This study contributes to the data engineering layer of mental health case studies on Social Media X by providing a reproducible, robust, and fault-tolerant infrastructure for automated data acquisition and ETL pipeline orchestration. However, the extracted data was limited to publicly available tweets obtained through the X API and depended on keyword-based filtering, which may not fully represent broader mental health discussions on social media. In addition, this research focused on the engineering and automation aspects of ETL pipelines rather than predictive modeling or clinical interpretation of mental health conditions. Therefore, mental health in this study is positioned as a domain-specific case study, and the resulting dataset is not intended to serve as a clinical diagnostic tool.

Future work may focus on integrating advanced NLP feature engineering and machine learning methods, such as TF-IDF, word embeddings, or Decision Tree-based classification, to enable comparative analysis with prior mental health detection studies and to extend the analytical applicability of the collected dataset.

REFERENCES

- [1] M. K. D. Jaya, I. G. A. G. A. Kadnyanan, and D. L. G. Astuti, "ETL Pipeline Di Pt Sirkadian Indonesia Sehat," *Jurnal Pengabdian Informatika*, vol. 4, no. 2, pp. 589–594, 2024.
- [2] Z. Aurellia, E. Sahara, M. Ilham, Z. Welerubun, and O. D. Ramadhani, "Pengalaman Pengguna X Sebagai Platform Curhat (Studi Fenomenologi)," *Seminar Nasional Universitas Negeri Surabaya*, vol. 3, pp. 616–623, 2024.
- [3] Kukuh Wijayanti and Qoniah Nur Wijayani, "Peranan Aplikasi Twitter (X) dalam Interaksi Komunikasi untuk Membantu Menyeimbangkan Kesehatan Mental pada Remaja Saat Ini," *Jurnal Sains Student Research*, vol. 2, no. 1, pp. 07–15, Dec. 2023, doi: 10.61722/jssr.v2i1.469.
- [4] D. K. N. Manullang, F. Veronica, E. Anastasia, M. Akailupa, D. Chrisnawandi, and R. Gunawan, "Dampak Penggunaan Twitter Mempengaruhi Kesehatan Mental Generasi Z," *TERAPUTIK: Jurnal Bimbingan dan Konseling*, vol. 8, no. 3, pp. 91–103, Feb. 2025, doi: 10.26539/teraputik.833609.
- [5] M. Rahman Nayla, "Memahami Dampak Media Sosial terhadap Kesehatan Mental Mahasiswa," *Jimad: Jurnal Ilmiah Mutiara Pendidikan*, vol. 2, no. 1, 2024, doi: 10.31004/jptam.v7i3.11993.
- [6] Rizqa Annisa Dwiatmoko and Imelda, "Analisis Sentimen Kesehatan Mental Di Twitter Menggunakan Naive Bayes Classifier Dan K-Nearest Neighbor," *JIFOSI*, vol. 6, no. 1, pp. 1–9, Apr. 2025, doi: 10.33005/jifosi.v6i1.466.
- [7] M. R. Hidayatullah and W. Maharani, "Depression Detection on Twitter Social Media Using Decision Tree," *Jurnal Resti*, vol. 6, no. 4, pp. 677–683, Aug. 2022, doi: 10.29207/resti.v6i4.4275.
- [8] H. Ramadhan, Y. Hendriyani, and T. Sriwahyuni, "Penerapan Pipeline ETL Apache Airflow Menggunakan Algoritma Collaborative Filtering untuk Pemberian Rekomendasi Gastrodiplomasi Indonesia," *Jurnal Pendidikan Tambusai*, vol. 9, no. 2, 2025, doi: <https://doi.org/10.31004/jptam.v9i2.27485>.
- [9] L. Fanani Mz *et al.*, "Rekonstruksi Arsitektur DataBase untuk Peningkatan Proses Load Data," *Jurnal Media Informatika (JUMIN)*, vol. 6, no. 2, pp. 1455–1460, 2025, doi: <https://doi.org/10.55338/jumin.v6i2.5585>.
- [10] D. Andriansyah, "Implementasi Extract-Transform-Load (ETL) Data Warehouse Laporan Harian Pool," *Jurnal Teknik Informatika Stimik Antar Bangsa*, vol.

- VIII, no. 2, pp. 45–49, Dec. 2022, doi: <https://doi.org/10.51998/jti.v8i2.486>.
- [11] S. Kumar Singu, “ETL Process Automation: Tools and Techniques,” *ESP Journal of Engineering & Technology Advancements*, vol. II, no. 1, pp. 74–85, Feb. 2022, doi: 10.56472/25832646/JETA-V2I1P110.
- [12] D. Chanda, “Automated ETL Pipelines for Modern Data Warehousing: Architectures, Challenges, and Emerging Solutions,” *The Eastasouth Journal of Information System and Computer Science*, vol. 1, no. 03, pp. 209–212, 2024, doi: 10.58812/esiscs.v1i03.
- [13] U. Nayak, “Comparing AWS Glue vs. Apache Airflow for Data Orchestration: A Comprehensive Performance and Cost Analysis,” *International Journal of Emerging Trends in Computer Science and Information Technology*, vol. 6, no. 3, pp. 51–55, 2025, doi: 10.63282/3050-9246.ijetsit-v6i3p109.
- [14] E. Eko Wahyudi *et al.*, “Akuisisi Data Prediksi Curah Hujan Secara Periodik Menggunakan Apache Airflow,” *Badan Meteorologi, Klimatologi, dan Geofisika Jl. Angkasa I*, vol. 4, no. 2, pp. 1–012, doi: 10.20895/INISTA.V4I2.
- [15] I. R. Wibowo *et al.*, “Big Data Pipeline Infrastructure Design in MSME E-Commerce Systems with a Focus on Data Source Processing Using Orchestration Tools,” *Transmisi: Jurnal Ilmiah Teknik Elektro*, vol. 26, no. 1, pp. 48–54, Jan. 2024, doi: 10.14710/transmisi.26.1.48-54.
- [16] S. Harsha and V. Sanne, “Addressing Persistent Storage Challenges in Kubernetes Environments,” *International Journal on Recent and Innovation Trends in Computing and Communication*, [Online]. Available: <http://www.ijritcc.org>
- [17] Z. Bakhshi, G. Rodriguez-Navas, and H. Hansson, “Analyzing the performance of persistent storage for fault-tolerant stateful fog applications,” *Journal of Systems Architecture*, vol. 144, Nov. 2023, doi: 10.1016/j.sysarc.2023.103004.
- [18] J. H. Na *et al.*, “PVA: The Persistent Volume Autoscaler for Stateful Applications in Kubernetes,” *IEEE Access*, vol. 12, pp. 179130–179143, 2024, doi: 10.1109/ACCESS.2024.3507194.
- [19] M. Musrini, N. Fitrianti, F. Reysandi, A. Rifky, N. Fitrianti F, and R. A. Rifky, “Membangun Data Warehouse untuk Menganalisis Pola Siswa yang Mendaftar di ITENAS (Studi Kasus Institut Teknologi Nasional Bandung),” *Jurnal Ilmiah Teknologi Informasi Terapan*, vol. 8, no. 1, pp. 45–56, Dec. 2021, doi: <https://doi.org/10.33197/jitter.vol8.iss1.2021.715>.
- [20] A. W. Iswara, H. Setiadi, and A. Wijayanto, “Implementation of Business Intelligence for Quality Support of RSUD Ir. Soekarno Sukoharjo with Data Warehouse,” *ITSMArt: Jurnal Ilmiah Teknologi dan Informasi*, vol. 9, no. 1, Jun. 2020, doi: <https://doi.org/10.20961/itsmart.v9i1.42915>.
- [21] I. Jubaidah, D. Pratiwi, and T. Siswanto, “Forecasting sales data on e-commerce using single exponential smoothing methods,” *Jurnal Teknologi Informasi : Jurnal Keilmuan dan Aplikasi Bidang Teknik Informatika*, vol. 17, no. 2, Aug. 2023, doi: 10.47111/JTI.
- [22] I. Putu, W. Prasetya, I. Nyoman, and H. Kurniawan, “Implementasi ETL (Extract, Transform, Load) pada Data warehouse Penjualan Menggunakan Tools Pentaho,” *TIERS Information Technology Journal*, vol. 2, no. 1, pp. 39–47, 2021.
- [23] D. Wijayanto and A. Firdonsyah, “Implementation of Continous Delivery using Jenkins And Kubernetes with Docker Local Images,” *sinkron*, vol. 8, no. 4, pp. 2226–2235, Oct. 2023, doi: 10.33395/sinkron.v8i4.12624.
- [24] N. Reddy Rachamala, “Automated Orchestration for Distributed ETL: Practical Approaches with Airflow,” *Journal of Information Systems Engineering and Management*, vol. 2025, no. 59s, pp. 2468–4376, 2025.
- [25] A. Satya Vivek Vardhan Akisetty *et al.*, “Automating ETL Workflows with CI/CD Pipelines for Machine Learning Applications,” *Iconic Research and Engineering Journals (IRE Journals)*, vol. 7, no. 3, Feb. 2025.
- [26] R. Tahir, G. Surya Mahendra, and R. Sandra Yofa Zebua, *Bisnis Intelegent (Pengantar Business Intelligence dalam Bisnis)*. Cilegon: PT. Sonpedia Publishing Indonesia, 2023.
- [27] R. W. P. G. T. Kusumah, “Perancangan Data Warehouse Pada Bagian Akademik Universitas di Bandung,” *Jurnal Strategi (Sarana Tugas Akhir Mahasiswa Teknologi Informasi)*, vol. 3, no. 1, May 2021.
- [28] I. Zaelani, “Implementasi Data Mart Terhadap Sistem Penjualan Pada Perusahaan Bidang Distributor Di Pt. Eigen Trimathema Implementation Of Data Mart On Sales System In Distributor Companies In Pt. Eigen Trimathema,” *JUPITER: Jurnal Penelitian Mahasiswa Teknik Dan Ilmu Komputer*, vol. 1, no. 2, 2021.
- [29] I. Gusti Ngurah Agung Trisna Putra *et al.*, “Implementasi ETL Data Warehouse Dengan Konsep Fitur Metadata Dan Cleansing Data Pada Toko Kue,” *Jurnal Sistem Informasi*, vol. 9, no. 2, pp. 274–289, 2020.
- [30] K. Rahayu, V. Fitria, D. Septhya, R. Rahmaddeni, and L. Efrizoni, “Klasifikasi Teks untuk Mendeteksi Depresi dan Kecemasan pada Pengguna Twitter Berbasis Machine Learning,” *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, vol. 3, no. 2, pp. 108–114, Sep. 2023, doi: 10.57152/malcom.v3i2.780.
- [31] S. Shevira, I. Made, A. Dwi Suarjaya, W. Buana, J. Raya, and K. Udayana, “Lexicon and Naive Bayes Algorithms to Detect Mental Health Situations from Twitter Data,” *Journal of Information Systems Engineering and Business Intelligence*, vol. 8, no. 2, 2022, doi: 10.20473/jisebi.8.2.
- [32] Nithish, Ravi, and David, “Data Transformation Techniques in ETL,” *International Journal of Multidisciplinary on Science and Management*, vol. 1, pp. 1–16, doi: 10.71141/30485037/V1I2P101.

- [33] I. Zaelani, "Implementasi Data Mart terhadap Sistem Penjualan pada Perusahaan Distributor di PT. Eigen Trimathema," *Jupiter: Jurnal Penelitian Mahasiswa Teknik Dan Ilmu Komputer*, vol. 1, no. 2, 2021.
- [34] R. Wijaya and B. Pudjoatmodjo, "Penerapan Extraction-Transformation-Loading (ETL) Dalam Data Warehouse (Studi Kasus: Departemen Pertanian)," *Jurnal Nasional Pendidikan Teknik Informatika (JANAPATI)*, vol. 5, no. 2, 2016.
- [35] Nithish, Ravi, and David, "Data Transformation Techniques in ETL," *International Journal of Multidisciplinary on Science and Management*, vol. 1, pp. 1–16, doi: 10.71141/30485037/V1I2P101.
- [36] J. Lee *et al.*, "MDB-KCP: persistence framework of in-memory database with CRIU-based container checkpoint in Kubernetes," *Journal of Cloud Computing*, vol. 13, no. 1, Dec. 2024, doi: 10.1186/s13677-024-00687-9.

