Journal of Information System Research (JOSH)
Volume 7, No. 1, Oktober 2025, pp 85−100
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i1.8468

In-Situ Database Machine Learning: Evaluating SQL-Based K-Means for E-Commerce Sales Analysis

Joanne Polama Putri Sembiring, Rajif Agung Yunmar*

Fakultas Teknologi Industri, Teknik Informatika, Institut Teknologi Sumatera, South Lampung
Jl. Terusan Ryacudu, Way Huwi, Kec. Jati Agung, Kabupaten Lampung Selatan, Lampung, Indonesia
Email: [1]joanne.121140128@student.itera.ac.id, [2,*]rajif@if.itera.ac.id
Correspondence Author Email: rajif@if.itera.ac.id
Submitted: **06/10/2025**; Accepted: **29/10/2025**; Published: **29/10/2025**

Abstract−Conventional machine learning techniques, such as K-Means clustering, often necessitate transferring data outside the database for analysis, which introduces inefficiencies, potential data inconsistencies, or security and privacy concerns. This research proposes an in-situ database machine learning approach by implementing the K-Means clustering algorithm directly within the database management system through using stored procedure. The methodology comprises five main stages: collection of public datasets (from Kaggle), data preparation and cleaning, transformation of data through cyclical feature encoding for temporal context, in-database K-Means implementation, and performance evaluation. The evaluation utilized the Silhouette Score metric and execution time to compare the proposed in-situ approach with a conventional off-database implementation. The in-situ database clustering achieved an optimal Silhouette Score of $S \approx 0.914$ in a remarkably short time of 0.0121 seconds. In comparison, the conventional off-database clustering achieved an identical quality score, but required a significantly longer execution time of 1.2956 seconds. This means that, to achieve the exact same cluster quality, the in-situ method is approximately 107.07 times faster than the off-database method. The identical score confirms the mathematical correctness of the SQL-based implementation and indicates excellent cluster quality. The findings of this study demonstrate that the in-situ database clustering approach is a superior methodology. This exceptional efficiency, validated by the successful categorization of e-commerce sales data into distinct demand patterns, lays a strong foundation for developing more effective and efficient predictive analytical strategies and data-driven decision-making, particularly for inventory planning.

Keywords: K-Means; In-Situ Database Machine Learning; Stored Procedure; Silhouette Score; E-Commerce.

1. INTRODUCTION

The continuous advancement of digital technology has precipitated a significant proliferation of data across multiple sectors of society. The extensive integration of smart devices, digital applications, and cloud-based infrastructures has resulted in the generation of vast quantities of information, characterized by a high rate of growth and originating from a myriad of sources. This phenomenon delineates the era of big data, which is defined not only by the sheer volume of data generated but also by the variety of its types and the velocity at which it flows.

The e-commerce sector has undergone significant transformation, emerging as a crucial catalyst for global economic change. Projections indicate that its growth rate will attain 8.63% by 2025 compared to the preceding year [1]. The proliferation of online commercial activities generates extensive datasets encompassing sales metrics, consumer demographics, behavioral activity logs, product evaluations, and financial transactions. In Indonesia, the volume of e-commerce enterprises surpassed 3.8 million units in 2023, with forecasts suggesting continued growth [2]. The data generated in this context possesses considerable strategic potential, provided it is effectively managed and analyzed. This potential is particularly salient in the realms of marketing strategy development, service personalization, and market demand forecasting.

The proliferation of data, characterized by its increasing volume and complexity, presents considerable challenges in terms of management, storage, and processing, particularly within large-scale database environments. A system's inefficiency in processing data can obstruct the decision-making continuum and adversely affect service quality [3]. Thus, the application of analytical methodologies is essential to mitigate data complexity and transform raw data into actionable insights for further analysis [4]. Among the diverse analytical techniques, clustering emerges as a significant method for unlabelled data grouping. Within the realm of e-commerce, this technique can be specifically applied to product segmentation, sales pattern analysis, and the categorization of customer behavior, thereby enhancing strategic decision-making and operational efficiency. Consequently, in order to conduct analysis and clustering on e-commerce or sales data stored within a database, it is imperative to extract the relevant data prior to any analytical processes [5].

The rising demand for precise and effective data clustering methodologies has catalyzed the development of various approaches within the field. Hierarchical clustering is particularly distinguished for its capability to reveal the inherent hierarchical structure present within datasets. However, a notable limitation of this method is the considerably increased computational time required for processing large datasets, especially when compared to the K-Means clustering algorithm [6]. In contrast, the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm demonstrates a capacity to generate more accurate clustering results without necessitating prior knowledge of the cluster count. Nonetheless, it is often less effective when applied to datasets characterized by varying densities or high dimensionality [7]. The K-Medoids method, while benefiting from its

Journal of Information System Research (JOSH)
Volume 7, No. 1, Oktober 2025, pp 85−100
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i1.8468

ability to identify representative objects, similarly experiences a decline in efficiency as the size of the dataset increases [8].

Conversely, the K-Means clustering technique is widely acknowledged for its efficiency in managing large datasets, coupled with its straightforward implementation and rapid convergence. These attributes render K-Means highly applicable across a range of diverse domains [9]. Numerous studies have successfully examined the application of the K-Means algorithm across a variety of fields. For instance, Rivaldo et al. [10] employed K-Means to discern productive fish species, demonstrating its effectiveness in ecological research. In the domain of public health, Munawar and Purnamasari [11] applied the algorithm to cluster HIV case data, showcasing its utility in analyzing health-related information. Furthermore, Handayanna [12] utilized K-Means to categorize impoverished populations, illustrating the algorithm's relevance in socio-economic studies. Within the e-commerce sector, Anjani et al. [13] applied K-Means to analyze cosmetic sales data, elucidating product clusters that demonstrate optimal performance. Collectively, these extensive applications affirm the versatility and effectiveness of the K-Means algorithm in handling diverse data sets across multiple domains.

The K-Means algorithm has garnered substantial popularity within the realm of data clustering; however, its implementation in the referenced studies primarily occurs externally to database systems. This is exemplified in the work of Anjani et al. [13], who utilized RapidMiner for their clustering tasks. Such an external execution introduces a disconnection between the clustering process and the data management system, raising significant concerns regarding data consistency and potentially leading to increased processing times—especially under the constraints of large-scale datasets [14]. Furthermore, the extraction of data from database management systems heightens concerns related to data privacy, thereby increasing the risks of data leakage. These considerations illuminate the pressing need for addressing inefficiencies and risks, particularly with sales data that predominantly reside within database systems, engendering a focus on in-situ data clustering methodologies. Although Arwani [5] previously investigated the notion of in-situ data clustering through a comparative analysis of time complexity, the study was limited in its scope, lacking a thorough performance evaluation and a quantitative assessment of cluster quality using metrics such as the Silhouette Score.

This research proposes for the implementation of the K-Means clustering algorithm directly within the database management system, a methodology termed SQL-based in-situ clustering. The system utilizes MariaDB, which is compatible with MySQL, both of which are highly popular among developers [15]. This innovative approach aims to facilitate the clustering process in a manner that is fully integrated at the database level, thereby obviating the necessity for data extraction to external platforms. As a result, this method significantly mitigates the risk of data inconsistencies, enhances processing efficiency, and bolsters the overall integrity of the information system. Through this integration, the research underscores the potential for improved data management practices and effective analytical capabilities within database environments.

This research investigates two primary objectives. Firstly, it examines the implementation of the K-Means clustering algorithm directly within a database environment, utilizing it to analyze e-commerce sales data for the purpose of generating product groupings based on sales patterns. This classification framework offers valuable insights that can inform inventory procurement strategies, capitalizing on the distinct groupings established by the clustering process. Secondly, the study rigorously evaluates the performance of the implemented approach, focusing on two critical dimensions: the quality and validity of the resultant clusters, measured using the Silhouette Score metric; and the computational efficiency, quantified by the execution time. Collectively, this research aims to enhance the development of database-centric analytical methodologies, thereby facilitating more informed and efficient decision-making in the realm of e-commerce sales data management.

2. RESEARCH METHODOLOGY

2.1 Research Stages

The research methodology encompasses several structured and critical stages, each integral to ensuring the integrity and validity of the findings. The initial stage involves data collection, which is paramount for securing the essential datasets required for subsequent analysis. Once the data has been accumulated, it progresses to a rigorous data cleaning phase, aimed at eliminating extraneous noise and anomalies that could compromise the accuracy of the analysis. Following this, a data transformation process is undertaken to adequately prepare the datasets to meet the specific requirements of the analytical procedures.

The subsequent stage forms the crux of the research endeavor: the implementation of K-Means Clustering. This particular step is executed directly within the database through the utilization of SQL queries, thereby maximizing computational efficiency and ensuring the robustness of the clustering process. Finally, the methodology culminates in a quantitative evaluation of the clustering outcomes, employing the Silhouette Score metric as a means of assessment. A comprehensive workflow illustrating the entirety of this research methodology is presented in Figure 1.

Journal of Information System Research (JOSH)
Volume 7, No. 1, Oktober 2025, pp 85−100
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i1.8468

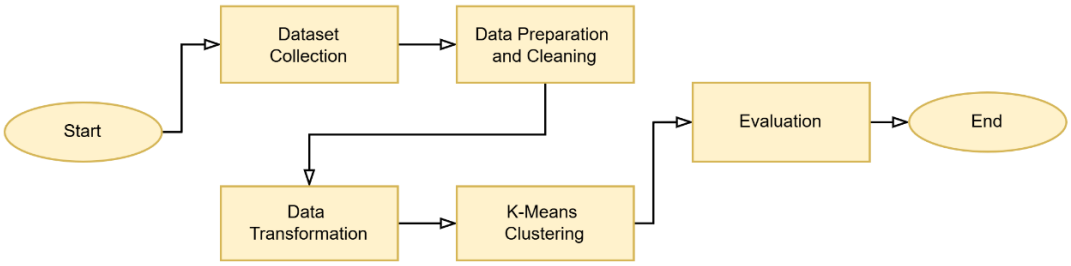

Figure 1. Research stage in general.

2.2 Research Dataset

The data utilized in this research is secondary data obtained from a public e-commerce dataset [16], which encompasses a comprehensive compilation of 27,764 transaction records collected throughout the year 2022. This dataset is systematically organized into three principal tables: `pelanggan`, `produk`, and `transaksi`. Table 1 provide the information about dataset used in this research. While the structural relationships among these data tables are illustrated in Figure 2, providing a clear visual representation of their interconnections.

Table 1. Dataset used in this research.

Information	Value
Source	Kaggle [16]
Number of Sample	27,764 transaction
Years	2022
Important Data	pelanggan, produk, transaksi

To facilitate the in-database implementation and subsequent computations related to clustering, several derivative tables were generated within the database environment. These tables serve to retain intermediate results and critical components requisite for the K-Means Algorithm analysis, feature engineering, and evaluation of the Silhouette Score. The derivative tables established include: `a_intra`, `b_nearest`, `centroid`, `new_centroid`, `pairwise_distance`, `produk_cluster`, `produk_distance`, `produk_features`, `produk_final`, `silhouette`, and `transformasi`.

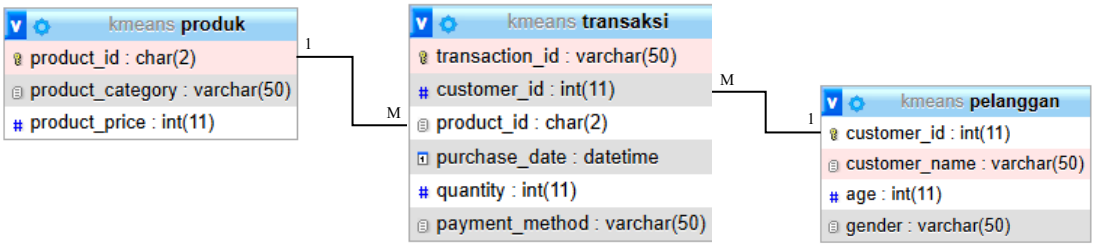

Figure 2. E-commerce table relation for data clusterisation.

2.3 Data Preparation and Cleaning

This stage represents a critical juncture in the application of the K-Means Algorithm, which is particularly susceptible to variations in scale, noise, and the presence of data outliers. The primary aim of this phase is to ensure the dataset is not only accurate and devoid of errors but also free from unrepresentative values that may hinder the clustering process. The data cleaning procedure encompasses several essential steps: first, it involves assessing the dataset for missing values, thereby addressing any incompleteness that could impact the integrity of the analysis. Additionally, it entails the identification and examination of data outliers, which are extreme values with the potential to significantly influence the location of the clustering centroid. Notably, this phase deliberately excludes the evaluation for duplicate data, as the principles of relational database management systems (RDBMS) inherently mitigate the occurrence of data duplication through the implementation of primary key constraints.

2.3.1 Removing Missing Value

This study addresses the critical issue of data validation, specifically targeting the detection and management of NULL values (i.e., missing values) within the key attributes of the 'transactions' table. The analysis scrutinizes several essential columns, including customer_id, product_id, purchase_date, quantity, and payment_method. To uphold data integrity prior to analytical procedures, a robust handling strategy is employed: any data row identified as containing a NULL value in any of these significant columns will be systematically removed (row deletion).

Journal of Information System Research (JOSH)
Volume 7, No. 1, Oktober 2025, pp 85−100
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i1.8468

This approach is essential for ensuring the reliability and quality of the transaction data under investigation. The procedure for inspection and subsequent deletion is elucidated through SQL code as presented in.

```
DELETE FROM transaksi
WHERE transaction_id IN (
    SELECT transaction_id FROM transaksi
    WHERE customer_id IS NULL
       OR product_id IS NULL
       OR purchase_date IS NULL
       OR quantity IS NULL
       OR payment_method IS NULL
);
```

2.3.2 Removing Data Outlier

The outlier detection methodology employed in this study relies on the Z-Score, a robust statistical technique that quantifies the extent to which a given data point deviates from the mean, expressed in terms of standard deviations. The implementation of this method includes the automated calculation of the mean and standard deviation through SQL subqueries directly within the database framework. This design ensures that the Z-Score computation remains responsive to the most current data characteristics within the 'transactions' table.

The criterion established for the identification of outliers is defined as |Z-Score| > 3. This threshold is informed by the Empirical Rule (often referred to as the 68-95-99.7 Rule), which posits that approximately 99.7% of observations in a normally distributed dataset fall within ±3 standard deviations from the mean [17]. Thus, any observation that lies beyond this prescribed boundary is deemed exceptionally rare and is consequently classified as an outlier, necessitating further examination and potential intervention. The SQL code presented in exemplifies the methodology for identifying and subsequently removing the outliers detected through this process.

```
DELETE FROM transaksi
WHERE transaction_id IN (
  SELECT t.transaction_id
  FROM transaksi t
  CROSS JOIN (
     SELECT AVG(quantity) AS mean_quantity,
        STDDEV(Quantity) AS std_quantity
     FROM transaksi
  ) stats
  WHERE ABS((t.quantity - stats.mean_quantity) / stats.std_quantity) > 3
);
```

2.4 Data Transformation

The objective of this stage is to refine and prepare the dataset, ensuring that the employed features are optimal for the K-Means clustering process. A significant challenge in clustering seasonal products arises from the dependence on temporal data, particularly with the representation of months as integers ranging from 1 to 12. This numerical representation presents a considerable issue, as K-Means clustering calculates the distance between Month 12 (December) and Month 1 (January) as extensive, despite the close proximity of these months in the annual cycle [18].

To address this challenge, the present study employs Cyclical Feature Encoding through the use of sine and cosine trigonometric functions. This methodological transformation effectively represents periodic time data within Cartesian coordinates, positioning each month as a point on a unit circle. Consequently, this representation ensures that the geometric distance between December and January (12 and 1) is minimized, thereby maintaining the inherent cyclical nature of the temporal data.

This approach proves particularly advantageous for distance-based algorithms such as K-Means, as it enables the analysis of seasonal patterns without compromising the contextual inter-period connectivity. This is exemplified by the illustrative depiction in

Figure 3. The encoding process, in a mathematical context, involves the mapping of each month value (m), which is defined within the range of 1 to 12, to a corresponding angle (θ) on the unit circle. The calculation of this angle (θ) is executed through the application of Equation (1), as delineated below:

$$\theta = \frac{2\pi \times m}{12} \quad (1)$$

In this context, let θ (theta) denote the angular measurement in radians that reflects the position of a particular month (m) on the unit circle. The mathematical constant π (Pi), approximately equal to 3.14159, signifies half the circumference of a circle, corresponding to an angle of 180 degrees. Consequently, the value of 2π represents a complete rotation of the circle (360 degrees), which is analogous to one full cycle encompassing the

Journal of Information System Research (JOSH)
Volume 7, No. 1, Oktober 2025, pp 85−100
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i1.8468

twelve months of the year. Within this framework, the variable m represents the specific month (e.g., m=1 for January, m=2 for February, and so forth, up to m=12 for December), while the total number of cyclical periods is represented by the integer 12, corresponding to the total number of months in a year.

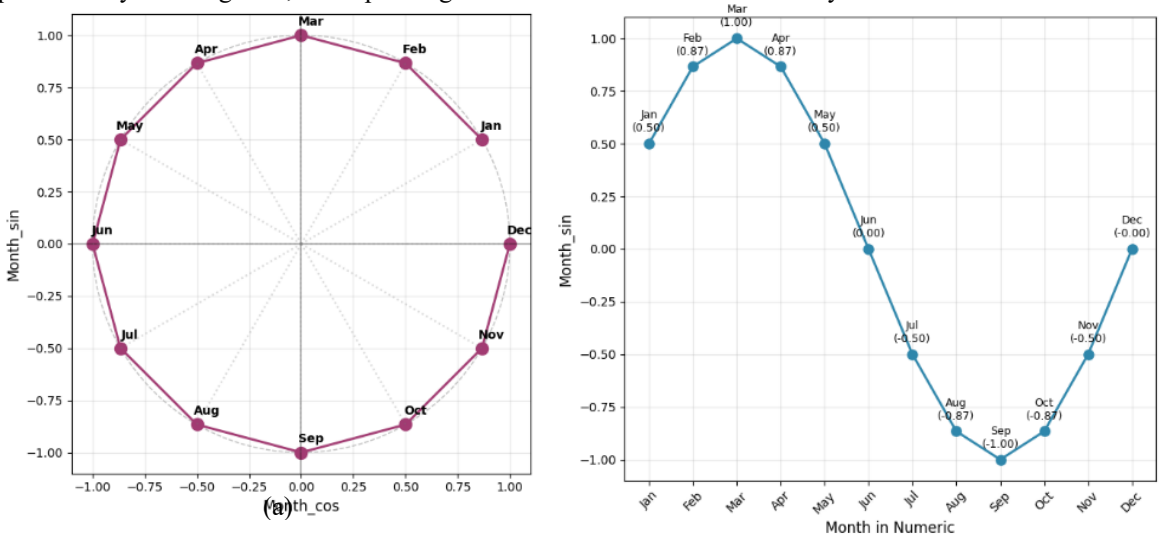

Figure 3. Cyclical feature encoding for the month utilizes sine and cosine trigonometric functions. This method requires two visual components: (a) Representation of the month on a unit circle; and (b) Sinusoidal encoding for monthly data.

Upon deriving θ, the subsequent step involves computing the sine and cosine values, which are intended to serve as new features for further clustering analyses. This methodological approach guarantees that the angular representations for Month 12 and Month 1 are closely aligned, thereby minimizing the geometric distance between them. Such a design is inherently consistent with the cyclical nature of temporal events. The specific calculations for the sine and cosine features can be executed using the formulas denoted in Equations (2) and (3) below:

Sine Feature (Y-Axis): month_sin=sin(θ) (2)

Cosine Feature (X-Axis): month_cos=cos(θ) (3)

Following the transformation process, the dataset was augmented from its initial state, which comprised solely two principal attributes, to incorporate four numerical features suitable for clustering analysis. These features include `quantity`, `freq_bulan` (monthly purchase frequency), `month_sin` (the sine encoding result of the month), and `month_cos` (the cosine encoding result of the month). The primary attributes selected for clustering were `Purchase Date` and `Quantity`, as they were identified as critical for delineating transaction characteristics that are essential for informed inventory procurement strategies. The forthcoming clustering analysis will be conducted using both the quantity and temporal purchase features, with the objective of delineating three distinct cluster classes.

The SQL code presented in details the procedural steps undertaken during this data transformation process. The initial phase of the SQL implementation involves the creation of a new table, designated as produk_features. This table is intended to house the critical features utilized for clustering, derived from the aggregation of transactions per product. The features included are: `product_id`, `total_quantity` (representing the cumulative quantity of products sold), `freq_bulan` (indicating the frequency of distinct sales months), `month_sin` (the mean sine value corresponding to the purchase month), and `month_cos` (the mean cosine value correlating to the purchase month).

```
1.  CREATE TABLE produk_features AS
2.  SELECT
3.      product_id,
4.      SUM(quantity) AS total_quantity,
5.      COUNT(DISTINCT MONTH(purchase_date)) AS freq_bulan,
6.      -- Cyclical encoding
7.      AVG(SIN(2 * PI() * MONTH(purchase_date) / 12)) AS month_sin,
8.      AVG(COS(2 * PI() * MONTH(purchase_date) / 12)) AS month_cos
9.  FROM transaksi
10. GROUP BY product_id;
```

2.5 K-Means Clustering

Following the data transformation phase, the subsequent process entails data clustering utilizing the K-Means Algorithm. The selection of the K-Means Algorithm is attributed to its inherent simplicity, computational

Journal of Information System Research (JOSH)
Volume 7, No. 1, Oktober 2025, pp 85−100
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i1.8468

efficiency, and ease of implementation, facilitating the partitioning of the dataset into (k) groups (clusters) based on the principle of similarity [5]. At the core of this algorithm lies the determination of (k) centroids, which serve as representative center points for each cluster. The algorithm assigns each data point to the nearest centroid, thereby establishing clusters. This clustering process is executed iteratively, as presented in Figure 4, with the detailed procedural steps outlined below:

1. Initial Centroid Initialization: This step involves the determination of the initial positions of the centroids for (k=3) clusters.
2. Data Classification: Here, each data point is classified into the nearest cluster through the computation of the Euclidean distance to each centroid.
3. Centroid Update: In this phase, the position of each centroid is recalibrated based on the mean value of all data points assigned to that particular cluster.
4. Convergence: The process is repeated through steps 2 and 3 until the centroids exhibit negligible movement (convergence) or until the maximum iteration limit is achieved.

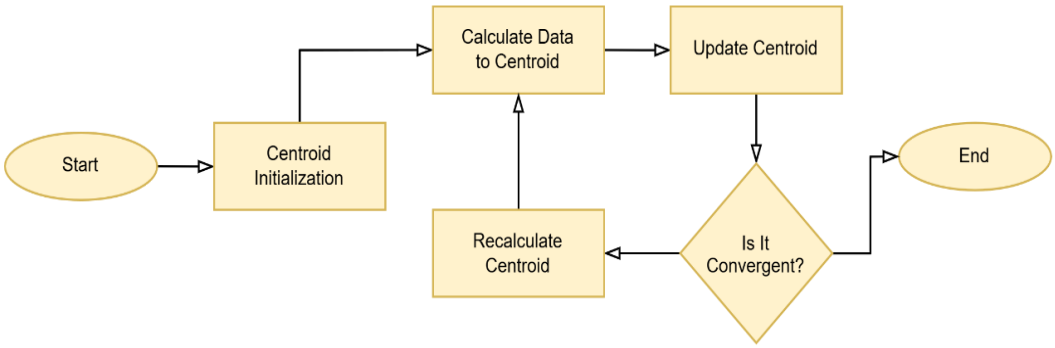

Figure 4. The stages of the clustering process using the K-Means algorithm.

The K-Means Algorithm employs the Euclidean Distance formula to determine the classification of data points into the nearest cluster, as articulated in Equation (4) [19]. In this context, the variable $d_{(x,y)}$ denotes the Euclidean Distance measured between two points. Here, the first point, denoted as x, refers to the data point (or data record) that is to be classified, while y signifies the centroid point (or cluster center) that encapsulates the respective cluster. Specifically, x_i and y_i represent the values of the i -th attribute (feature) of the data point x and the centroid point y, respectively. Moreover, n indicates the number of attributes (feature dimensions) utilized during the clustering process. This distance calculation facilitates the determination of the cluster to which data point x will be assigned, specifically identifying the cluster that exhibits the minimal distance $d_{(x,y)}$.

$$d_{(x,y)} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (4)$$

Upon the completion of classifying all data points into their respective clusters, the subsequent phase in the K-Means Algorithm involves the updating of the centroid positions. This update is executed by determining the mean value of all objects belonging to a given cluster. The mathematical formulation utilized for calculating the new centroid is delineated in Equation (5) below [20]:

$$C_i = \frac{\sum_{i=1}^{n} x_i \in S_i}{n} \quad (5)$$

In the context of centroid computation within cluster analysis, let C_i represent the new centroid point being determined for cluster i, which is derived from the calculation of the mean. The symbol S_i denotes the cluster i itself, encompassing the complete set of data objects that are aggregated within this grouping. The notation $\sum_{x \in S\_i} x$ signifies the total summation of values for each data object x included in cluster i. Furthermore, the variable n refers to the total number of objects (or data points) contained in cluster i. By dividing the aggregate sum of object values by the total number of objects n, one can accurately determine the new centroid, thereby positioning it precisely at the geometric center of the cluster.

2.6 K-Means Implementation

Upon the successful completion of the data preparation phase, the subsequent stage entails the execution of the core implementation of the research, specifically the K-Means Clustering Algorithm. This algorithm is implemented directly within the MariaDB environment utilizing structured query language (SQL) queries, leveraging the features that have been meticulously organized in the `produk_features` table. The implementation of the K-Means algorithm is conducted through a systematic series of iterative steps, continuing until convergence is achieved. The methodological approach is designed to yield three distinct clusters (k=3), aligning with the primary objectives of the research.

Journal of Information System Research (JOSH)
Volume 7, No. 1, Oktober 2025, pp 85−100
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i1.8468

2.6.1 Initialization of Initial Centroids

The initial phase of the K-Means clustering implementation involves the determination of three centroids (k=3) which will function as the initial cluster centers. This process employs random initialization, wherein three data points (products) are selected at random from the `produk_features` table to serve as the starting centroids. This initial selection is a critical step, as the positioning of the centroids significantly influences the convergence trajectory of the algorithm. The coordinates of these initial centroids are subsequently recorded in a newly created table designated as "`centroid`." delineates the SQL code utilized to execute this initial centroid assignment.

```
CREATE TABLE centroid (
    cluster_id INT,
    total_quantity FLOAT,
    freq_bulan FLOAT,
    month_sin FLOAT,
    month_cos FLOAT
);

-- Take the first 3 products as the initial centroids
INSERT INTO centroid
SELECT ROW_NUMBER() OVER() - 1 AS cluster_id,
   total_quantity, freq_bulan, month_sin, month_cos
FROM produk_features
ORDER BY RAND()
LIMIT 3;
```

2.6.2 Calculating Data to Centroid

This section represents the initial phase of each iteration within the K-Means clustering algorithm, focusing on the computation of distances between individual products and all designated centroids. Following this computation, each product is assigned to the cluster corresponding to the nearest centroid.

2.6.2.1 Calculating Distance to Centroid

The process commences with the establishment of a new relational table, designated as `produk_distance`. This table serves the purpose of storing the computed distances between each product in the `produk_features` dataset and the existing centroids, as delineated in the `centroid` table. The calculation of distance is executed utilizing the Euclidean Distance formula, as articulated in Equation (4). The methodology for this distance computation is further elucidated in which provides the SQL code necessary to ascertain the distance from each individual product data point to the corresponding cluster centroid.

```
CREATE TABLE produk_distance AS
SELECT
    f.product_id,
    c.cluster_id,
    SQRT(
        POWER(f.total_quantity - c.total_quantity, 2) +
        POWER(f.freq_bulan - c.freq_bulan, 2) +
        POWER(f.month_sin - c.month_sin, 2) +
        POWER(f.month_cos - c.month_cos, 2)
    ) AS distance
FROM produk_features f
CROSS JOIN centroid c;
```

2.6.2.2 Determining Clusters Based on Nearest Distance

Subsequently, the next phase involves the identification of clusters for each product. Each product is allocated to the cluster characterized by the minimum distance, as determined from the earlier distance calculations. The resultant classifications are recorded in a new table, referred to as `produk_cluster`. delineates the SQL code employed in this classification process, ensuring a systematic assignment of products to their nearest clusters based on computed distances.

```
CREATE TABLE produk_cluster AS
SELECT product_id, cluster_id
FROM (
   SELECT
      product_id,
      cluster_id,
      distance,
```

Journal of Information System Research (JOSH)
Volume 7, No. 1, Oktober 2025, pp 85−100
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i1.8468

```
        ROW_NUMBER() OVER (
            PARTITION BY product_id ORDER BY distance ASC
        ) AS rn
    FROM produk_distance
) t
WHERE rn = 1;
```

2.6.3 Centroid Update

Upon the classification of each product into its respective nearest cluster, it becomes imperative to update the position of the cluster center, or centroid. This update is vital, as the new centroid must accurately represent the mean values of all members currently assigned to that particular cluster, in accordance with Equation (5). The completion of this step signifies the conclusion of one full iteration of the K-Means algorithm, thus setting the stage for the subsequent cycle of classification.

The process initiates with the establishment of a new table, designated as `new_centroid`. This table is tasked with storing the coordinates of the updated centroid, which are derived by aggregating data based on the `cluster_id` and computing the average (AVG) values of each pertinent feature—namely, `total_quantity`, `freq_bulan`, `month_sin`, and `month_cos`—within that cluster. Following the creation of the `new_centroid` table, a comparison is conducted with the previous `centroid` table to assess convergence. Should the disparity between the two tables remain beneath the predetermined error threshold, the iterative process is deemed complete. The methodology for the centroid update is enumerated in presented in SQL code.

2.6.4 Iteration

The Iteration Phase represents the fundamental component of the K-Means Algorithm, wherein the processes of data clustering and centroid recalibration are executed iteratively to achieve an optimal clustering configuration. This iterative mechanism is pivotal, as it allows the centroids' positions to progressively align more closely with the inherent distribution of the data. To enhance efficiency and structure, the complete iteration process can be encapsulated within a MariaDB stored procedure, thereby facilitating the automated execution of the algorithm.

During each iteration cycle, the stored procedure systematically carries out the essential steps of the K-Means algorithm, encompassing the procedures delineated in Algorithms 4 through 7, which include distance calculation, data classification, and centroid updating. The iterative process continues until one of two predetermined stopping criteria is satisfied:

1. Convergence: This criterion is met when the variation in centroid positions, quantified as the error threshold, falls below an established limit (specifically, when the error is less than 0.001).
2. Maximum Iterations: Alternatively, the process may be halted when the total number of iterations attains a specified maximum limit (set at 20 iterations) to mitigate the risk of the algorithm running indefinitely.

```
-- Calculate new centroid
CREATE TABLE new_centroid AS
SELECT pc.cluster_id,
    AVG(f.total_quantity) AS total_quantity,
    AVG(f.freq_bulan) AS freq_bulan,
    AVG(f.month_sin) AS month_sin,
    AVG(f.month_cos) AS month_cos
FROM produk_cluster pc
JOIN produk_features f ON pc.product_id = f.product_id
GROUP BY pc.cluster_id;

-- Calculate the difference (old vs new centroid)
SELECT SUM(
   SQRT(
      POWER(c.total_quantity - n.total_quantity, 2) +
      POWER(c.freq_bulan - n.freq_bulan, 2) +
      POWER(c.month_sin - n.month_sin, 2) +
      POWER(c.month_cos - n.month_cos, 2)
   )
) INTO diff FROM centroid c
JOIN new_centroid n ON c.cluster_id = n.cluster_id;

-- Update centroid with new centroid
TRUNCATE TABLE centroid;
INSERT INTO centroid AS SELECT * FROM new_centroid;
```

```
CREATE PROCEDURE kmeans_produk_optimized(
```

Journal of Information System Research (JOSH)
Volume 7, No. 1, Oktober 2025, pp 85−100
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i1.8468

```
    IN p_k INT, IN p_max_iterations INT, IN p_stop_threshold DOUBLE
)
BEGIN
    DECLARE iter INT DEFAULT 0;
    DECLARE converged INT DEFAULT 0;
    DECLARE max_change DOUBLE DEFAULT 999999;
    DECLARE n INT;
    WHILE iter < p_max_iterations AND converged = 0 DO
        SET iter = iter + 1;
        -- SQL CODE ALGORITHM 4 TO ALGORITHM 7 --
    END WHILE;
END
```

The overall implementation of K-Means clustering on the MariaDB server follows. To initiate the clustering process with specific parameters, the stored procedure in is called using the CALL command. For example: `CALL kmeans_produk_optimized(3, 20, 0.001)`. This call will attempt to group the data into 3 clusters and will stop if it reaches 20 iterations or if the centroid movement is less than 0.001, indicating that convergence has been achieved.

2.7 Evaluation

The evaluation phase is critical for assessing both the efficacy and efficiency of the K-Means algorithm implementation. This evaluation focuses on two primary metrics. First, cluster quality is assessed using the Silhouette Score (SC), which measures the internal cohesion and separation of the resultant clusters. Second, computational performance is measured through the Execution Time, where the time required to complete the clustering process is recorded. This dual approach enables a direct comparison between the proposed in-database implementation and the conventional out-of-database method.

2.7.1 Evaluation with Silhouette Score

Specifically, the Silhouette Score is computed based on the distance of an object to other objects within the same cluster contrasted with its distance to the nearest objects in adjacent clusters [21]. The SC value ranges from 0 to 1, with higher scores (approaching 1) indicative of well-defined, dense clusters characterized by clear separation. Conversely, scores nearing zero signal the presence of overlapping clusters. Interpretative benchmarks for quality assessment of the resulting clusters will be guided by the categorizations delineated in Table 2.

Table 2. Silhouette score and the interpretation [21].

SC Value	Quality	Interpretation
0,71 – 1,00	Very Good	The optimal cluster structure has been found.
0,51 – 0,70	Good	The cluster placement is reasonable and satisfactory.
0,26 – 0,50	Poor	The structure is weak; additional methods may be necessary.
<=0,25	No structure	No discernible cluster structure has been found.

To calculate the Silhouette Score, the Equation (5) is employed:

$$S = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{b(i)-a(i)}{max(a(i),b(i))}\right) \quad (5)$$

In this equation, S represents the average Silhouette Score for the entire dataset, while $\frac{b(i)-a(i)}{max(a(i),b(i))}$ signifies the silhouette index for the i -th sample. Here, $a(i)$ denotes the average distance between the i-th sample and all other samples within the same cluster, known as cohesion, which reflects the degree of internal proximity within that cluster. Conversely, $b(i)$ indicates the minimum average distance between the i-th sample and all samples in the nearest different cluster, referred to as separation, which illustrates the proximity between clusters. It is also important to note that N represents the total number of samples or data points considered in the analysis. The resultant value of S, which is constrained within the range of 0 to 1, serves as a metric for assessing the effectiveness of clustering with respect to the grouping of each individual sample.

Understanding the contributions of each variable—specifically, $a(i)$ representing internal cohesion and $b(i)$ denoting external separation—facilitates a comprehensive evaluation of the Silhouette Score through a systematic series of computational processes. The methodology for calculating the Silhouette Score unfolds in the following structured stages, i.e., calculating pairwise distance, calculating intra-cluster distance ($a(i)$), calculating nearest-cluster distance ($b(i)$), calculating Silhouette Score per product, and finally calculating average Silhouette Score (S).

```
CREATE TABLE pairwise_distance AS
SELECT
    f1.product_id AS id1,
```

Journal of Information System Research (JOSH)
Volume 7, No. 1, Oktober 2025, pp 85−100
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i1.8468

```
    f2.product_id AS id2,
    SQRT(
        POWER(f1.total_quantity - f2.total_quantity, 2) +
        POWER(f1.freq_bulan - f2.freq_bulan, 2) +
        POWER(f1.avg_quantity - f2.avg_quantity, 2) +
        POWER(f1.month_sin - f2.month_sin, 2) +
        POWER(f1.month_cos - f2.month_cos, 2)
    ) AS distance
FROM produk_features f1
JOIN produk_features f2 ON f1.product_id <> f2.product_id;
```

2.7.1.1 Calculating Pairwise Distance

Initially, the Euclidean Distance formula is employed to compute the distances between every pair of distinct products, with the results systematically recorded in a pairwise_distance table. This distance metric serves as a foundational measure of proximity, essential for the subsequent calculations of both internal cohesion (denoted as $a(i)$) and external separation (denoted as $b(i)$). This process commences with the establishment of a new data structure referred to as the pairwise_distance table, which encapsulates the Euclidean distances between each distinct product within the produk_features table. delineates the SQL code implementation of this computational procedure.

2.7.1.2 Calculating Intra-cluster Distance ($a(i)$)

Subsequently, the average proximity of each product to all other members within the same cluster is assessed. This metric, denoted as $a(i)$, is calculated and documented in the a_intra table. Specifically, $a(i)$ represents the mean distance between a given product and all other products that are situated within the same cluster. The process initiates with the establishment of a new dataset, referred to as a_intra, which is employed to encapsulate the average intra-cluster distance $a(i)$ corresponding to each product.

In a technical context, this calculation utilizes the pairwise_distance table in conjunction with the produk_cluster table to aggregate the distances between all products that share an identical cluster_id. Following this aggregation, the summed distances are averaged to yield the value $a(i)$. The procedural implementation of this calculation is depicted

```
CREATE TABLE a_intra AS
SELECT
    c1.product_id, c1.cluster_id,COALESCE(AVG(d.distance), 0) AS a_value
FROM produk_cluster c1
LEFT JOIN produk_cluster c2
    ON c1.cluster_id = c2.cluster_id AND c1.product_id <> c2.product_id
LEFT JOIN pairwise_distance d
    ON (c1.product_id = d.id1 AND c2.product_id = d.id2)
    OR (c1.product_id = d.id2 AND c2.product_id = d.id1)
GROUP BY c1.product_id, c1.cluster_id;
```

2.7.1.3 Calculating Nearest-cluster Distance ($b(i)$)

This step involves determining the minimum average distance from a given product to all members of the nearest different cluster, referred to as $b(i)$. These results are meticulously stored in the b_nearest table. This metric signifies the minimum average distance between a given product and all members of the nearest distinct cluster, referred to as the nearest-cluster. The determination of $b(i)$ is paramount for evaluating the clarity of a cluster's separation from its neighboring clusters.

To initiate this process, a new data structure, labeled `b_nearest`, is established. This table serves to encapsulate the average distances of each product to the constituents of the nearest cluster. The underlying procedure involves the systematic grouping of products based on clusters to which they do not belong, followed by the calculation of the average distances. Subsequently, the minimum average distance is extracted, representing the closest competing cluster—often referred to as the 'neighboring cluster.'

A higher $b(i)$ value indicates a more pronounced separation between clusters, thereby contributing significantly to the overall integrity and delineation of the cluster structure. Moreover, it is essential to note that the disparity between $b(i)$ and $a(i)$ will ultimately influence the final Silhouette Index for the sample under consideration. The implementation details for calculating this Nearest-Cluster Distance are encapsulated in

```
CREATE TABLE b_nearest AS
SELECT
    t.id1 AS product_id,MIN(t.avg_dist) AS b_value
FROM (
    SELECT d.id1,c2.cluster_id,AVG(d.distance) AS avg_dist
    FROM pairwise_distance d
```

Journal of Information System Research (JOSH)
Volume 7, No. 1, Oktober 2025, pp 85−100
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i1.8468

```
    JOIN produk_cluster c1 ON d.id1 = c1.product_id
    JOIN produk_cluster c2 ON d.id2 = c2.product_id
    WHERE c1.cluster_id <> c2.cluster_id
    GROUP BY d.id1, c2.cluster_id
) AS t
GROUP BY t.id1;
```

2.7.1.4 Calculating Silhouette Score per Product

The fouth stage is calculating Silhouette Score per product. Utilizing the $a(i)$ values from the a_intra table in conjunction with the $b(i)$ values from the b_nearest table, the silhouette index for each product is computed utilizing the established Silhouette Score formula. This procedure facilitates the integration of the results derived from the computations of internal cohesion (denoted as $a(i)$) and external separation (denoted as $b(i)$), thereby enabling the determination of the Silhouette Index for each product. The Silhouette Index serves as a metric to evaluate the effectiveness of the clustering process for individual products. The initial step involves the creation of a new table, referred to as 'silhouette', which serves to catalog the Silhouette Score values corresponding to each product.

The computation is executed through a SQL query that joins the a_intra and b_nearest tables, effectively applying the Silhouette Index formula, expressed as $\frac{b(i)-a(i)}{max(a(i),b(i))}$, directly within this query. The detailed implementation of this calculation is outlined in the SQL code provided in which systematically illustrates the methodology employed in deriving the Silhouette Scores.

```
CREATE TABLE silhouette AS
SELECT
  a.product_id,
  -- Calculates Silhouette Index: (b - a) / max(a, b)
  (b.b_value - a.a_value) / NULLIF(GREATEST(a.a_value, b.b_value), 0)
  AS silhouette_value
FROM a_intra a
JOIN b_nearest b ON a.product_id = b.product_id;
```

2.7.1.5 Calculating Average Silhouette Score (S)

The mean of all silhouette indices across the individual products is calculated to yield a singular value, S. This value represents the extent to which the dataset is effectively organized into coherent clusters. This section elucidates the methodology employed to derive the final Silhouette Score (S), which serves as an overarching metric for assessing clustering quality across the entire dataset. The average Silhouette Score is computed by aggregating the silhouette indices of individual products, meticulously stored within the `silhouette` table.

To achieve this, the `AVG()` aggregation function is applied to the `silhouette_value` column, yielding a singular metric that encapsulates the degree of effective clustering present within the dataset. A resultant value approaching 1 indicates superior clustering quality, signifying high internal cohesion among the clusters while illustrating a notable external separation. delineates the SQL implementation of the average Silhouette Score calculation.

```
SELECT
   AVG(silhouette_value) AS silhouette_score
FROM silhouette;
```

2.7.2 Evaluation of Computational Performance

This sub-section aims to analyze the computational efficiency of the in-situ K-Means implementation by comparing its execution time against the conventional off-database clustering method. The execution time measurement is a key metric for quantifying the performance gains offered by the database-centric approach. The execution time (T) is defined as the total duration required for the algorithm to complete the clustering process. This measurement is performed by recording the start time (T_{start}) and the end time (T_{end}) of the algorithm execution. The formula used to calculate the execution time is provide by Equation (6).

$$T = T_{end} - T_{start} \tag{6}$$

The $T_{in-situ}$ measures the duration of the entire execution of the SQL K-Means queries run directly within the database management system. Conversely, T_{off-db} measures the total time required for the conventional workflow executed outside the database environment. The off-database method is implemented using Python with the Scikit-learn (sklearn) library. Crucially, the T_{off-db} is segmented into three sequential components, as represented by Equation (7): $T_{data\_retrieval}$ represent the time needed to retrieve data from MariaDB and save it into a CSV file, T_{load_csv} represent the time Python takes to load the data from the CSV file, and $T_{processing}$ represent the time required by Python to execute the K-Means clustering and Silhouette Score calculation. To

Journal of Information System Research (JOSH)
Volume 7, No. 1, Oktober 2025, pp 85−100
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i1.8468

quantitatively measure the increase in efficiency, the Speedup Factor (S) is calculated by comparing the off-database execution time to the in-situ execution time by Equation (8).

$$T_{off-db} = T_{data\_retrieval} + T_{load\_data} + T_{processing} \quad (7)$$

$$S = \frac{T_{off-db}}{T_{in-situ}} \quad (8)$$

3. RESULTS AND DISCUSSION

3.1 Clustering Results

Following the successful completion of all stages of implementation for the K-Means Clustering Algorithm, with a specified target of three clusters (k=3), we have achieved notable clustering outcomes through the utilization of SQL queries within the database environment. The grouped categorization of products was derived based on the computed features encompassing sales quantity, frequency, and temporal patterns, which were represented through sine and cosine encoding. The allocation of products to their corresponding clusters is delineated in Table 3.

Table 3. Final results of product clustering using K-Means algorithm.

Product ID	Product Category	Cluster ID
CD	Compact disc	0
CT	Correction tape	1
NB	Notebook	1
PN	Pen	1
SN	Sticky notes	1
MK	Mechanical keyboard	2
MP	Monitor portable	2
OC	Office chair	2

The clustering analysis elucidates the existence of three distinct groups, each characterized by divergent member attributes, which underscore substantial variations in product demand patterns. As illustrated in Figure 5, the visualization delineates the distribution of products within each cluster, while Figure 6 presents the cluster distribution alongside their respective centroids.

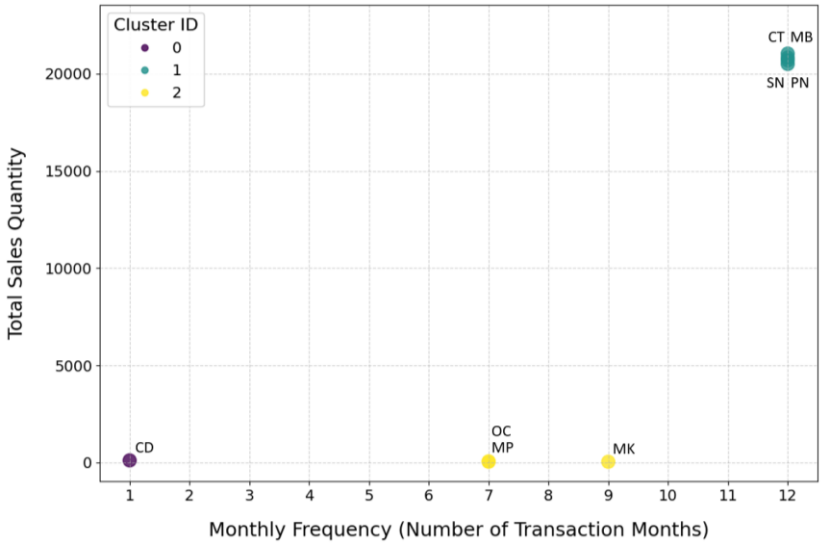

Figure 5. The visualization of product distribution pre cluster.

Cluster 0 emerges as a solitary entity, comprising exclusively one product: the Compact Disc (CD). The singularity of this cluster implies that the CD exhibits sales characteristics—including both quantity and temporal trends—that are markedly distinct from those of all other products in the dataset. In contrast, Cluster 1 represents the most populous group, encompassing four products: Correction Tape (CT), Notebook (NB), Pen (PN), and Sticky Notes (SN). Products within this cluster exhibit analogous patterns with respect to sales quantity, frequency, and/or temporal distribution. Collectively, this cluster is emblematic of stationary and office supply products that demonstrate either stable demand patterns or analogous seasonal fluctuations.

Journal of Information System Research (JOSH)
Volume 7, No. 1, Oktober 2025, pp 85−100
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i1.8468

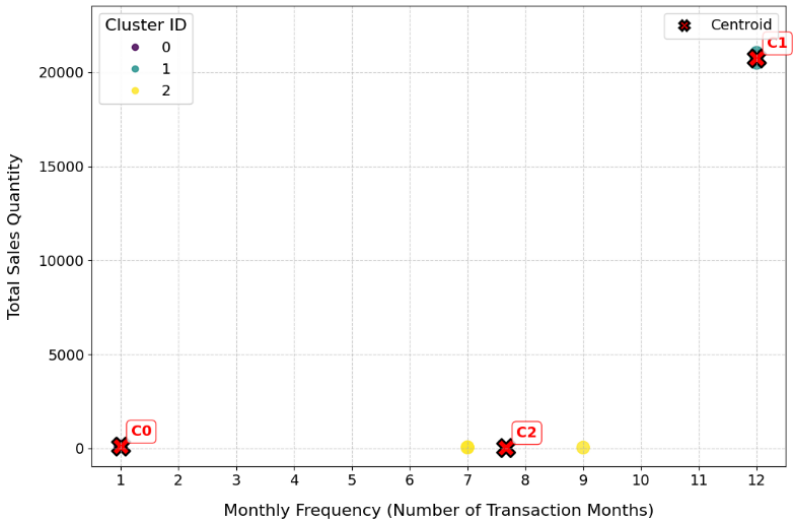

Figure 6. The visualization of product distribution per cluster with centroids.

Lastly, Cluster 2 is composed of three products: Mechanical Keyboard (MK), Portable Monitor (MP), and Office Chair (OC). The strong correlation in characteristics among the members of this cluster suggests shared sales quantity values and/or comparable demand patterns, indicative of items associated with less frequent yet potentially larger purchase transactions.

3.2 The Strategic Implication of Clustering

The K-Means Algorithm was effectively utilized to segment products into three distinct clusters, each representing unique demand patterns that exert a significant influence on inventory procurement strategies. These clusters were systematically classified into three stock management categories: Fast Moving, characterized by a high and stable movement rate; Slow Moving, defined by a low to moderate movement rate, typically associated with high-value items; and Seasonal/Unique, which encompasses products with sporadic sales or those that are concentrated around specific temporal periods. The classification of products based on sales frequency and cluster membership is presented in Table 4.

Table 4. Classification of products according to sales frequency and cluster membership.

Product ID	Product Category	Product Price	Freq. (Month)	Category	Cluster ID
CD	Compact disc	12000	1	Seasonal	0
CT	Correction tape	7000	12	Fast Moving	1
NB	Notebook	25000	12	Fast Moving	1
PN	Pen	5000	12	Fast Moving	1
SN	Sticky notes	8000	12	Fast Moving	1
MK	Mechanical keyboard	950000	9	Slow Moving	2
MP	Monitor portable	1950000	7	Slow Moving	2
OC	Office chair	450000	7	Slow Moving	2

Cluster 1, which includes Correction Tapes, Notebooks, Pens, and Sticky Notes, is identified as the Fast Moving Product category. This cluster is characterized by a maximum purchase frequency of 12 months and a relatively high total quantity, bolstered by an affordable price range of 5,000 to 25,000. The consistent and stable sales patterns observed throughout the year suggest a high demand and rapid inventory turnover. Consequently, products within Cluster 1 necessitate a robust stocking strategy, requiring regular replenishment to mitigate the risk of stockouts and associated losses.

In contrast, Cluster 2, which encompasses Mechanical Keyboards, Portable Monitors, and Office Chairs, is categorized as Slow Moving Products. The products in this cluster are distinguished by their relatively high prices, ranging from 450,000 to 1,950,000, and a low total quantity, alongside a purchase frequency that, while routine, varies between 7 to 9 months. This indicates a lower sales rotation compared to Cluster 1. Therefore, the inventory procurement strategy for Cluster 2 should prioritize maintaining sufficient stock levels to avoid excess inventory, as overstocking could lead to capital being tied up unnecessarily.

Lastly, Cluster 0 pertains to the singular product Compact Disc (CD), which exhibits a purchase frequency of only 1 month. This cluster is indicative of Seasonal or highly unique product traits, with notable sales occurring predominantly during specific timeframes, such as December. The stocking strategy for this cluster must be adaptive in nature, emphasizing stock increases during seasonal peaks, while maintaining minimal inventory levels during off-peak periods.

Journal of Information System Research (JOSH)
Volume 7, No. 1, Oktober 2025, pp 85−100
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i1.8468

3.3 Clustering Quality Results and Analysis With Silhouette Score

The evaluation phase validated the performance of the clustering algorithm that was implemented, focusing specifically on the resulting cluster quality. Crucially, the overall average Silhouette Score (S) calculated by the in-database SQL implementation and the off-database Python/scikit-learn implementation yielded the exact same value: $S \approx 0.914$. This score similarity extends to the individual level, where the Silhouette Score for every single Product ID was identical between the two methods. This perfect agreement confirms the correctness and mathematical accuracy of the K-Means implementation within the MariaDB database using SQL queries.

According to the quality assessment criteria delineated in Table 2, this score is classified within the "Very Good" category ($0.71 - 1.00$), thereby substantiating the efficacy of the in-database K-Means implementation. The results that shown in Table 5 indicate that the clusters formed exhibit exceptionally high internal cohesion alongside pronounced external differentiation.

A more granular analysis of individual cluster members further corroborates these findings. Cluster 1, designated as Fast Moving, displays the highest Silhouette Scores, approximately 0.98, signifying that its constituents are characterized by strong cohesiveness and distinct separation from other clusters. Cluster 0, identified as Seasonal, attained a perfect score of 1.00, a result anticipated due to its status as a singleton cluster, thereby affirming its unique and isolated position within the clustering framework.

Table 5. The Silhouette Score, which measures the quality of placement for each cluster member.

Cluster ID	Product ID	Silhouette Score	Cluster Quality
0	CD	1	Very Good
1	CT	0.989033	Very Good
1	NB	0.982612	Very Good
1	PN	0.982676	Very Good
1	SN	0.988956	Very Good
2	MK	0.843244	Very Good
2	MP	0.858458	Very Good
2	OC	0.670915	Good
Average		**0.914486**	Very Good

In contrast, Cluster 2, referred to as Slow Moving, reveals a broader range of score variability, spanning from 0.67 to 0.85. While the majority of its members, namely MK and MP, uphold high Silhouette Scores, the Office Chair (OC) product registered the lowest score at approximately 0.67. Although this result still aligns with the "Good" category, it intimates that OC represents the least strongly affiliated member of Cluster 2, thereby positioning it nearest to the cluster's decision boundary. In summation, the high Silhouette Score serves as a compelling validation of the accuracy and robustness of the developed clustering solution.

3.4 Computational Performance Analysis

The second key objective of this evaluation was to assess the computational efficiency of the in-situ approach by juxtaposing its execution time with that of the conventional off-database method developed in Python. The findings indicate a substantial performance enhancement, with the in-situ clustering methodology exhibiting a remarkable reduction in total execution time, achieving a speed increase of 107.09% compared to the traditional workflow. In order to pinpoint the factors contributing to this improvement, the overall off-database clustering time was methodically disaggregated into distinct sequential stages: data retrieval from the MariaDB database and aving of retrieved data into a CSV file, loading of the CSV file, and the subsequent processing in Python employing sklearn's K-Means algorithm and silhouette_score for evaluation. The comprehensive analysis and the resultant comparison are encapsulated in Table 6.

The findings presented in Table 6 underscore that the predominant computational bottleneck within the off-database workflow resides in the data retrieval phase, which consumes 0.7481 seconds, thereby comprising over fifty percent of the total execution duration. In stark contrast, this stage is completely circumvented in the in-situ methodology, as all computational tasks are conducted directly within the database engine. Moreover, the clustering process revealed a remarkable acceleration, decreasing from 0.5277 seconds in the off-database approach to a mere 0.0121 seconds in the in-situ execution, which corresponds to a phenomenal speedup ratio of 43.61×. This significant enhancement emphasizes the efficacy of reducing data transfer overhead while capitalizing on database-native computational capabilities.

Table 6. Detailed comparison of execution time: off-database vs. in-situ database clustering.

No	Execution Step	Off-Database (in second)	In-Situ Clustering (in second)	Overhead/Speedup
1	Featch data from database	0.7481	N/A	Major Bottleneck
2	Time to Load the CSV file	0.0198	N/A	External Overhead
3	Clustering Process	0.5277	0.0121	43.61 times Faster

Journal of Information System Research (JOSH)
Volume 7, No. 1, Oktober 2025, pp 85−100
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i1.8468

Overall	**1.2956**	**0.0121**	**107.09 times Faster**

The results indicate a significant reduction in total execution time, decreasing from 1.2956 seconds in the off-database approach to 0.0121 seconds when employing in-situ clustering techniques. This stark contrast underscores the considerable benefits of integrating machine learning computations directly within the database environment, thereby enhancing computational efficiency and performance.

4. CONCLUSION

This research provides robust validation regarding the efficacy of implementing the K-Means Clustering Algorithm executed in situ within a database management system environment, pioneering an in-situ database machine learning approach. A comprehensive evaluation conducted across two critical dimensions substantiates the superiority of this database-centric approach. Firstly, in terms of cluster quality, the in situ method achieved an optimal Silhouette Score ($S \approx 0.914$), which was found to be identical to the score attained by the conventional off-database Python implementation. This result affirms the mathematical accuracy and reliability of the in situ approach. The high-quality clustering facilitated the effective categorization of products into distinct demand patterns, specifically Fast Moving, Slow Moving, and Seasonal categories. Secondly, the computational performance exhibited a remarkable enhancement, demonstrating an improvement of 107.09% in speed compared to the off-database method. This acceleration is primarily attributed to the elimination of data transfer overhead, thereby underscoring the efficiency of the embedded processing model. In conclusion, this study posits that the integration of clustering algorithms directly within a database environment represents a highly efficient and equally effective methodology for large-scale data processing. It is recommended that future research explore more sophisticated centroid initialization techniques, such as K-Means++, and apply this implementation to significantly larger datasets in order to further investigate the scalability and efficacy of this approach.

REFERENCES

[1] R. Geuens, “Analisis Pasar E-Commerce Global: Tren dan Prediksi 202410,” 2025. https://soax.com/research/growth-ecommerce (accessed Sep. 28, 2025).
[2] B. P. S. "Direktorat Statistik Keuangan, Teknologi Informasi, dan Pariwisata, “Statistik E-Commerce 2023,” 2023. [Online]. Available: https://www.bps.go.id/id/publication/2025/01/30/d52af11843aee401403ecfa6/statistik-e-commerce-2023.html.
[3] R. A. Alifia, N. R. Safitri, D. M. Irhami, R. Hidayat N, and I. R. Kusumasari, “Challenges and Solutions for Decision Making in the Era of Big Data,” J. Bisnis dan Komun. Digit., vol. 2, no. 2, p. 13, Dec. 2024, doi: 10.47134/jbkd.v2i2.3498.
[4] D. T. Warianta, P. Astagina, R. Julianto, and F. Y. Arini, “Optimalisasi K-Means Menggunakan Algoritma Firefly Untuk Segmentasi Pelanggan pada E-commerce,” J. FASILKOM, vol. 14, no. 3, pp. 775–785, Jan. 2025, doi: 10.37859/jf.v14i3.8287.
[5] I. Arwani, “Integrasi Algoritma K-Means Dengan Bahasa SQL Untuk Klasterisasi IPK Mahasiswa (Studi Kasus: Fakultas Ilmu Komputer Universitas Brawijaya),” J. Teknol. Inf. dan Ilmu Komput., vol. 2, no. 2, p. 143, 2015, doi: 10.25126/jtiik.201522148.
[6] I. Indra, N. Nur, M. Iqram, and N. Inayah, “Perbandingan K-Means dan Hierarchical Clustering dalam Pengelompokan Daerah Beresiko Stunting,” INOVTEK Polbeng - Seri Inform., vol. 8, no. 2, p. 356, 2023, doi: 10.35314/isi.v8i2.3612.
[7] F. M. Pranata, S. H. Wijoyo, and N. Y. Setiawan, “Analisis Performa Algoritma K-Means dan DBSCAN Dalam Segmentasi Pelanggan Dengan Pendekatan Model RFM,” J. Pengemb. Teknol. Inf. dan Ilmu Komput., vol. 8, no. 7, pp. 2548–964, 2024.
[8] S. Sindi, W. R. O. Ningse, I. A. Sihombing, F. I. R.H.Zer, and D. Hartama, “Analisis Algoritma K-Medoids Clustering dalam Pengelompokan Penyebaran Covid-19 di Indonesia,” J. Teknol. Inf., vol. 4, no. 1, pp. 166–173, Jun. 2020, doi: 10.36294/jurti.v4i1.1296.
[9] A. Ikhwan and N. Aslami, “Implementasi Data Mining untuk Manajemen Bantuan Sosial Menggunakan Algoritma K-Means,” J. Teknol. Inf., vol. 4, no. 2, pp. 208–217, Dec. 2020, doi: 10.36294/jurti.v4i2.2103.
[10] M. D. Rivaldo, G. W. N. Wibowo, and H. Mulyo, “Implementasi Algoritma K-Means untuk Klasterisasi Data Hasil Tangkapan Ikan di Karimunjawa,” J. Minfo Polgan, vol. 13, no. 1, pp. 1045–1056, Jul. 2024, doi: 10.33395/jmp.v13i1.13928.
[11] K. Kodratul Munawar and A. Irma Purnamasari, “Implementasi Algoritma K-Means Clustering pada Klasterisasi Kasus HIV di Jawa Barat,” JATI (Jurnal Mhs. Tek. Inform., vol. 7, no. 2, pp. 1092–1099, Aug. 2023, doi: 10.36040/jati.v7i2.6372.
[12] F. Handayanna, “Penerapan Algoritma K-Means untuk Klasterisasi Penduduk Miskin di Provinsi Banten,” INTI Nusa Mandiri, vol. 18, no. 1, pp. 93–99, Aug. 2023, doi: 10.33480/inti.v18i1.4399.
[13] A. Anjani, “Klasterisasi Data Penjualan Terlaris Produk Kosmetik You Menggunakan Algoritma K-Means,” J. Tika, vol. 9, no. 1, pp. 17–25, Apr. 2024, doi: 10.51179/tika.v9i1.2531.
[14] J. M. Hellerstein et al., “The MADlib analytics library,” Proc. VLDB Endow., vol. 5, no. 12, pp. 1700–1711, 2012, doi: 10.14778/2367502.2367510.
[15] “Most popular technologies - Stackoverflow Developer Survey,” 2025. https://survey.stackoverflow.co/2024/technology#most-popular-technologies-database-learn (accessed Oct. 22, 2025).
[16] S. Jagtap, “E-commerce Customer Data For Behavior Analysis,” 2023. https://www.kaggle.com/datasets/shriyashjagtap/e-commerce-customer-for-behavior-analysis?resource=download

Journal of Information System Research (JOSH)
Volume 7, No. 1, Oktober 2025, pp 85−100
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v7i1.8468

(accessed Sep. 29, 2025).
[17] P. K. Dunn, "Scientific Research and Methodology," Sci. Res. Methodol., 2025, doi: 10.1201/9781003394938.
[18] T. Mahajan, G. Singh, and G. Bruns, "An Experimental Assessment of Treatments for Cyclical Data," pp. 1–6, 2021, [Online]. Available: https://scholarworks.calstate.edu/downloads/pv63g5147.
[19] M. Anjelita, A. P. Windarto, A. Wanto, and S. Saifullah, "Analisis Metode K-Means pada Kasus Ekspor Barang Perhiasan dan Barang Berharga Berdasarkan Negara Tujuan," Semin. Nas. Sains Teknol. Inf., pp. 476–482, 2019, [Online]. Available: http://prosiding.seminar-id.com/index.php/sensasi/issue/archivePage%7C476.
[20] M. Muhtadin Billah, D. Rasyid Al-Hadi, D. Zatusiva Haq, and D. C. R. Novitasari, "Analisis Cluster Negara di Asia Berdasarkan Tingkat Kenyamanan Hidup Menggunakan Metode K-Means," JATI (Jurnal Mhs. Tek. Inform., vol. 8, no. 5, pp. 10551–10557, Sep. 2024, doi: 10.36040/jati.v8i5.10753.
[21] P. S. Rosiana, A. A. Mohsa, and Y. Umaidah, "Implementasi K-Means dalam Pengelompokan Penyebaran Penyakit DBD di Jawa Barat," J. Inform. dan Tek. Elektro Terap., vol. 11, no. 3, Aug. 2023, doi: 10.23960/jitet.v11i3.3344.