

Enhanced query performance for stored streaming data through structured streaming within spark SQL

Benymol Jose¹, Rajesh N.², Lumy Joseph¹

¹Department of Computer Applications, Marian College Kuttikkanam Autonomous, Idukki, Kerala, India

²Department of Computer Applications, SAS SNDP Yogam College, Pathanamthitta, India

Article Info

Article history:

Received Jan 16, 2024

Revised Apr 26, 2024

Accepted May 7, 2024

Keywords:

Bigdata

MongoDB

NoSQL databases

Spark SQL

Streaming data

ABSTRACT

Traditional database systems like relational databases can store data which are structured with predefined schema, but in the case of bigdata, the data comes in different formats or are collected from diverse sources. The distributed databases like not only spark querying language (NoSQL) repositories are often used in relation to bigdata analytics, but a continual updating is required in business because of the streaming data that comes from stock trading, online activities of website visitors, and from the mobile applications in real time. It will not have to delay, for some report to show up, to assess and analyse the current situation, to move forward with the next business choice. Apache Spark's structured streaming offer capabilities for handling streaming data in a batch processing mode with faster responses compared to MongoDB which is a document-based NoSQL database. This study completes similar queries to evaluate Spark SQL and NoSQL database performance, focusing on the upsides of Spark SQL over NoSQL databases in streaming data exploration. The queries are completed with streaming data stored in a batch mode.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Benymol Jose

Department of Computer Applications, Marian College Kuttikkanam Autonomous

Kuttikkanam, Idukki, Kerala, India

Email: benymol.jose@mariancollege.org

1. INTRODUCTION

Bigdata computing is a new trend in computing and based on processing requirements, it may be roughly divided into two categories: processing stream data in real-time and batch processing of stored data [1]–[3]. Large volumes and high velocity of data have been made conceivable by advances in information technology, but the necessity to hold data continually leads to a number of computing issues [1]. Continuously created data is referred to as streaming data, and with batch processing, stream data can be stored and converted into a corpus which can be processed later. Identifying pertinent and appropriate content from these streams is necessary to make timely judgements using this massive volume of unstructured data [4]. Using the information gathered from this type of analysis, businesses can monitor a variety of company and consumer activities including service consumption, server activity, website clicks, and device and person geolocation. And allows them to respond quickly to new circumstances.

As a result of the recent explosion in the amount of unstructured data, non-traditional databases like not only spark querying language (NoSQL) are emerging to address the problems associated with conventional databases [5], [6]. NoSQL databases like HBase, MongoDB, CouchDB, and Neo4j. Offer the scalability feature [7] which allows distributed computing possibilities. This feature allows storage of huge data in a distributed fashion in different commodity machines and hence these types of databases are mainly

used for bigdata and real-time web applications [8], [9]. Wei-ping *et al.* [10], in their study, tries to present NoSQL as a new technology which can be used effectively, when querying in massive datasets is to be performed. Here a performance comparison of the two technologies has been done with the purpose to replace the relational databases with the NoSQL implementations. Nayak *et al.* [11] conducted a discussion and study concerning the uses and problems of NoSQL databases and proposed it as a possible replacement to relational databases, which are still the most used type of databases. But a real mechanism is missing to get the operational flexibilities of querying with stream data or batch data, to get a timely response or to perform a real time mining [12] in a NoSQL database that houses unstructured data [13].

Singh *et al.* [14] suggest that the existence of growing unstructured data requires management, other than storage. According to them, after the data has been cleaned and restructured, we must use some data processing tools to analyse and visualize it. In their work, the data processing and analysis frameworks employed are Apache Spark and Hadoop MapReduce. Memory, CPU latency, and query performance are some of the data processing parameters used to make a comparison between these two frameworks. Kumar *et al.* [15] describes the MapReduce method, which was shaped and effectively deployed by Google, as the most successful algorithm for effective bigdata analytics (BDA). They suggested the Apache Hadoop distributed file system, which works with bigdata across the clusters of computers. Hadoop is known for its batch processing capabilities [16] and the processing done in it is based on the MapReduce strategy. The major consequence is that it can initiate a memory operation only by a two-pass operation, which are the 'map' and the 'reduce' algorithms [17]. But, with this method the complexity increases exponentially with each iteration which makes it incompetent to handle the velocity aspect of bigdata [18].

In another approach, Shoro and Soomro [19] explores the concept of bigdata analysis and the significance of real time processing by retrieving some meaningful information from twitter data. Unlike earlier Hadoop tests, they suggested Apache Spark, an open-source cluster computing platform that can complete tasks up to 100 times quicker in memory and 10 times faster when operating in secondary memory [20]. Apache Spark can handle workloads involving frequent access to datasets such as machine learning, interactive processing, graph processing, in-memory processing, and SQL [21]. The authors propose to use this potent open-source engine in processing bigdata, which is proficient of both stream and batch processing [22], [23]. Kolajo *et al.* [24] conducted a thorough assessment of bigdata stream analysis and discovered that there is growing interest in analysing bigdata in motion. A search of the literature revealed that limited studies have focused on processing of streaming data in a batch mode up to now. Hence, the authors steered an exhaustive examination of the state of stream processing today and recommend that more emphasis be placed on the empirical analysis of bigdata streaming technologies and approaches.

This research developed a way for performing faster querying with stored streaming data using Spark SQL. With the proposed approach, processing of streaming data in a batch mode is done using a unified bigdata analytics platform, Apache Spark, which can handle both stream and batch data. To make a performance test, the queries are applied with Spark SQL and MongoDB, which is a document-based NoSQL database. To measure performance, the process of comparing the execution times of similar queries that are utilized with Spark SQL and MongoDB is used [25], [26] on the same the dataset of size 2 GB. The added advantages of the stream processing competence, which supports the velocity component of Bigdata is better achieved with Apache Spark SQL querying strategy. This document is organized as follows: section 1 provides an overview of Apache Spark, the Hadoop ecosystem, and the review of literature. Section 2 goes on to detail the methodology, the dataset that is used, and the experimental setup. All the experimental evaluations and result discussions are followed in section 3, and finally the summary and the conclusion are arranged in section 4.

2. METHOD

With the new method, in-memory support can be used for querying either streaming data or batch data stored in the distributed file system [27], [28]. A resilient distributed dataset, an immutable collection of objects that conducts computations across several cluster nodes, is the fundamental data structure of Apache Spark. In Spark resilient distributed dataset (RDD), each dataset is logically divided among numerous servers to enable computation on various cluster nodes. They are resilient because they are fault tolerant and distributed since data resides on multiple nodes [28]. With the ability to compute in-memory, Spark RDDs keep intermediate results in primary memory rather than on a disk or other stable storage medium. Both the batch and stream processing in Spark are enabled by the pipelining architecture, which is a core component of computer organizational aspects. Typically, pipelines are separated into stages that connect to each other to produce a structure similar to a pipe [28]. It consists of a sequence of interconnected data processing elements where each element's output serves as its subsequent element's input [29], [30] and it increases the total instruction throughput. The new method opted for performing bigdata analytics using Spark is more

useful to the administrative sections of the organizations rather than the end users. The following are the different operations executed to achieve this in a unified environment with the unstructured data.

2.1. Querying strategy for processing stored streaming data

The querying is performed on batch data with SQL. The entire stream of data, reaching to the system is stored in the distributed file system and later it is considered as batch data and queries are applied on this. The query processing is postponed to a later part and streams are stored as they arrive. When the queries are made, the entire stream of data will be considered as a static data frame and the data can be studied with the querying of the information which helps in understanding the situation. To validate the faster query performance achieved here, a performance comparison is done with the execution time taken by the same category of queries used with MongoDB which is a NoSQL database. Both the query operations with Spark SQL and MongoDB, which is document-based NoSQL database, are performed with the same the dataset of size 2 GB.

2.2. Dataset used and experimental setup

The dataset utilized comes from https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Sports_v1_00.tsv.gz and is called Amazon-reviews of sports and outdoors. It consists of 4,833,093 records of around 2 GB size and a sample of the JSON dataset is shown in Figure 1 and it includes information about sports products, customer reviews, and a summary of reviews. The query processing is done with the specified dataset, considering the unstructured textual reviews, 'review_body' made by customers along with the 'star rating' of different sports products. The spark distributed file system stores the data by dividing it into several chunks.

The experiments are conducted with the databricks community edition for Apache Spark ecosystem consisting of the following configurations, 6 GB RAM, 0.88 cores, databricks runtime (DBR) 6.4 with Spark 2.4.5 and Scala 2.11, which is mainly used for executing the Spark SQL operations. Another interesting option used was the Google Colab, the free cloud platform provided by Google to motivate the research to opt google cloud platform (GCP) for computations and evaluation, which gives a wide range of environmental benefits such as integrating reporting tools and ETL tools.

```
{
  "_id" : ObjectId("5f8471f59de7409940e3191f"),
  "marketplace" : "US",
  "customer_id" : NumberInt(27260960),
  "review_id" : "R380VB8600X3H4",
  "product_id" : "B00L70AMM1",
  "product_parent" : NumberInt(883962979),
  "product_title" : "Yes4All Deep Tissue Massage AccuPoint Roller (Clearance Sale)",
  "product_category" : "Sports",
  "star_rating" : NumberInt(5),
  "helpful_votes" : 0.0,
  "total_votes" : 0.0,
  "vine" : "N",
  "verified_purchase" : "Y",
  "review_headline" : "Five Stars",
  "review_body" : "Perfect for home use. As Described. Good Price",
  "review_date" : ISODate("2015-08-31T00:00:00.000+0000")
}
```

Figure 1. The sample JSON document in the dataset

By using Colab, Google offers us better opportunities to move forward the research to a more powerful environment on GCP, which can help to integrate various resources along with horizontal scalability with the help of Google compute engine. The pipelining executions in Spark are executed using PySpark with 12.72 GB RAM and 107.77 GB disk space, which is supported by the GCP compute engine backend.

3. RESULTS AND DISCUSSION

3.1. Querying strategy for processing stored streaming data with spark SQL

To execute queries, the entire data is considered as a batch and the queries are applied to mine information, following the strategies in Spark SQL. The entire data which are kept in the distributed file system is accessed with the data frame named 'staticInputDF' which is shown below.

```
staticInputDF: pyspark.sql.dataframe.DataFrame = [_c0: string, _c1: string ... 7 more fields]
```

To filter the whole data and to get the count of each ‘star_rating’ on each ‘review_date’, an output data frame, ‘staticCountsDF’ is constructed from it with one day windows. To do this, grouping is done by the ‘star_rating’ column and one day windows over the ‘review_date’ column. Then this data frame is registered as a table ‘static_counts’ with fields ‘star_rating’, ‘counts’ and ‘window’, which corresponds to per day review_date as shown in Figure 2.

```

from pyspark.sql.functions import *
staticCountsDF = (
    staticInputDF
    .groupBy(
        staticInputDF.star_rating,
        window(staticInputDF.review_date, "1 day"))
    .count()
)
staticCountsDF.cache()
staticCountsDF.createOrReplaceTempView("static_counts")

```

Figure 2. The data frame ‘staticCountsDF’

The query executed with SQL for retrieving the sum of count of each ‘star_rating’ from the ‘static_counts’ table and the result obtained are as shown in Figure 3. Since the data is so big, the results can be easily understood by using a graphical representation as shown in Figure 4. Here, an area chart or area graph is used, in which the region between the axis and the line is typically highlighted with colors, textures, and hatchings that displays quantitative data graphically. Another graphical representation for the same query with the legacy line chart is as shown in Figure 5, which typically display data points connected by straight lines, making them accessible to a wide range of users without specialized training in data visualization.

This query displays the total count of each star_rating, and since the queries are applied without applying any pre-processing to the data, we have 20 undefined star_ratings as evident from Figure 4. It is evident from the aforementioned Figures 3 to 5 that there are 2,956,533 cases of “5” star ratings, 805,605 occurrences of “4” star ratings, 365,912 cases of “3” star ratings, “2” star ratings with a count of 220,084, and “1” star ratings with a total count of 348,847. We have optimized query execution performance using this method, taking only 5.75 seconds to complete.

```

spark.sql("select star_rating, sum(count) as total_count from static_counts group by star_rating").show()

```

star_rating	total_count
null	20
1	348847
3	365912
5	2956533
4	805605
2	220084

Figure 3. The query used to retrieve the ‘total_count’ of each ‘star_rating’

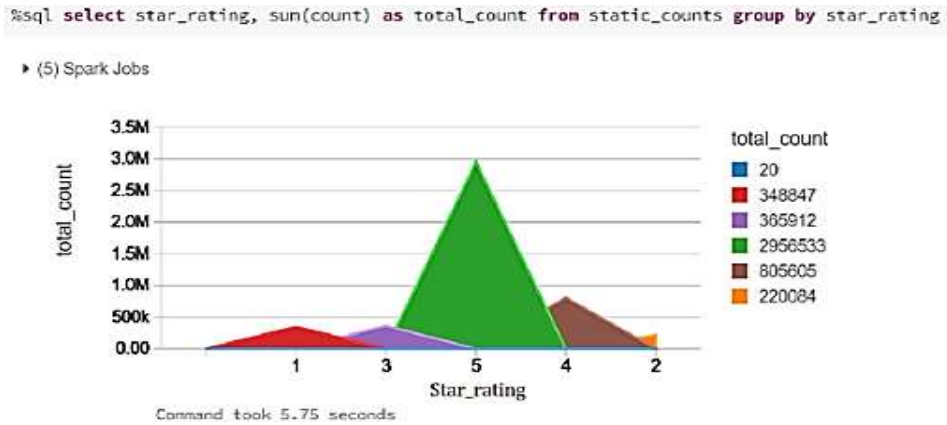


Figure 4. Area plot representation of query in Figure 3



Figure 5. Legacy line chart representation of query in Figure 3

3.2. Querying strategy for processing stored streaming data with MongoDB

The queries are completed on a data with a size of around 2 GB. When the same dataset was used for the SELECT query with aggregation function ‘sum()’ in MongoDB, which is a document based NoSQL database, it took 12,372 milliseconds. With the proposed method of using Spark SQL, the query execution took 5.75 seconds i.e., 5,750 milliseconds as shown in Figure 4, which is 2.15 times faster than using MongoDB. The performance comparison of both the queries based on their execution time and size of data is shown in Figure 6.

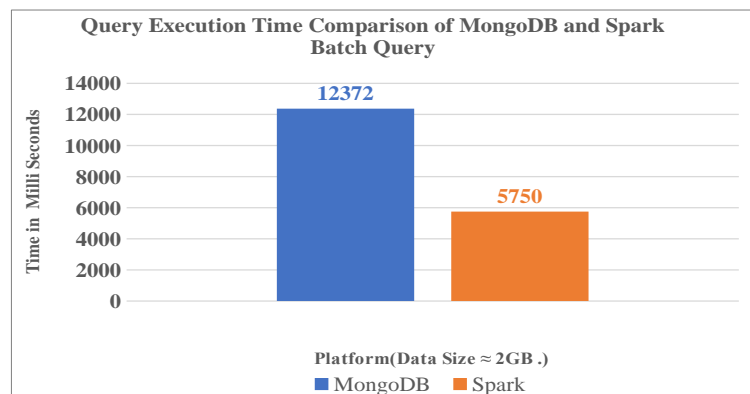


Figure 6. Query execution time comparison of MongoDB and Spark SQL

The time expended to explore the data with some constraining parameters cost us around 0.63 seconds as shown in Figure 2, which is a tremendous leap since dealing with a bigdata environment is dealt with. Retrieving or aggregating data over a distributed computing environment is far more complex than of operations in inhouse data repositories. It is found that the performance of the query with Spark SQL is very satisfactory as we consider the time taken to conclude it, with the distributed nature and map reduce computational complexities. And with the structured streaming concept used, queries were executed on stored streaming data, which can help in real time decision making.

4. CONCLUSION

The objective of this study was to accomplish the new concept of implementing faster querying with stored streaming data with Apache Spark. The queries are applied, and the performance is compared to that of equivalent queries in MongoDB, which is a document-based NoSQL database. Spark SQL outshines the general query language interfaces such as NoSQL databases by providing faster responses. With the structured streaming concept, queries are executed on streaming data, which can help in real time decision making. The paper came up with a solution for performing faster querying with stored streaming data with Spark SQL and it was found that querying on stored stream data is conceivable with the Spark distributed systems. All the experiments are done using pseudo clusters and the performance can be further improved by integrating high performance computing units with parallel processing capabilities with the nodes in the distributed platform.





REFERENCES

- [1] A. Mavragani, G. Ochoa, and K. P. Tsagarakis, "Assessing the methods, tools, and statistical approaches in Google trends research: systematic review," *Journal of Medical Internet Research*, vol. 20, no. 11, p. e270, Nov. 2018, doi: 10.2196/jmir.9366.
- [2] M. T. Özsu and P. Valduriez, "Big data processing," in *Principles of Distributed Database Systems*, Cham: Springer International Publishing, 2020, pp. 449–518.
- [3] D. Sun, G. Zhang, W. Zheng, and K. Li, "Key technologies for big data stream computing," in *Big Data*, Chapman and Hall/CRC, 2015, pp. 230–251.
- [4] B. P. Ranjitha, "Streaming analytics over real-time big data," *Global Journal of Computer Science and Technology*, vol. 15, no. 5, 2015.
- [5] D. Damodaran B, S. Salim, and S. M. Vargese, "Performance evaluation of MySQL and MongoDB databases," *International Journal on Cybernetics & Informatics*, vol. 5, no. 2, pp. 387–394, Apr. 2016, doi: 10.5121/ijci.2016.5241.
- [6] B. Jose and S. Abraham, "Exploring the merits of nosql: a study based on MongoDB," in *2017 International Conference on Networks and Advances in Computational Technologies, NetACT 2017*, Jul. 2017, pp. 266–271, doi: 10.1109/NETACT.2017.8076778.
- [7] J. R. Lourenço, B. Cabral, P. Carreiro, M. Vieira, and J. Bernardino, "Choosing the right NoSQL database for the job: a quality attribute evaluation," *Journal of Big Data*, vol. 2, no. 1, p. 18, Dec. 2015, doi: 10.1186/s40537-015-0025-0.
- [8] J. Pokorny, "NoSQL databases: a step to database scalability in web environment," in *ACM International Conference Proceeding Series*, Dec. 2011, pp. 278–283, doi: 10.1145/2095536.2095583.
- [9] P. Sun and Y. Wen, "Scalable architectures for big data analysis," in *Encyclopedia of Big Data Technologies*, Cham: Springer International Publishing, 2018, pp. 1–9.
- [10] Z. Wei-ping, L. Ming-xin, and C. Huan, "Using MongoDB to implement textbook management system instead of MySQL," in *2011 IEEE 3rd International Conference on Communication Software and Networks*, May 2011, pp. 303–305, doi: 10.1109/ICCSN.2011.6013720.
- [11] A. Nayak, A. Poriya, and D. Poojary, "Type of NOSQL databases and its comparison with relational databases," *International Journal of Applied Information Systems*, vol. 5, no. 4, pp. 16–19, 2013.
- [12] T. Pay, "Totally automated keyword extraction," in *2016 IEEE International Conference on Big Data (Big Data)*, Dec. 2016, pp. 3859–3863, doi: 10.1109/BigData.2016.7841059.
- [13] K. Gutfreund, "Big data techniques for predictive business intelligence," *Journal of Advanced Management Science*, vol. 5, no. 2, pp. 158–163, Mar. 2017, doi: 10.18178/joams.5.2.158-163.
- [14] A. Singh, M. Mittal, and N. Kapoor, "Data processing framework using apache and spark technologies in big data," in *Studies in Big Data*, vol. 43, 2019, pp. 107–122.
- [15] M. Kumar, G. S. Baluja, and D. P. Sahu, "Conceptualizing big data analytics through Hadoop," *COMPUSOFT, An international journal of advanced computer technology*, vol. 6, no. V, pp. 2335–2340, 2017.
- [16] P. Raj, "A detailed analysis of NoSQL and NewSQL databases for bigdata analytics and distributed computing," in *Advances in Computers*, vol. 109, 2018, pp. 1–48.
- [17] I. A. T. Hashem, N. B. Anuar, A. Gani, I. Yaqoob, F. Xia, and S. U. Khan, "MapReduce: review and open challenges," *Scientometrics*, vol. 109, no. 1, pp. 389–422, Oct. 2016, doi: 10.1007/s11192-016-1945-y.
- [18] S. N. Khezr and N. J. Navimipour, "MapReduce and its applications, challenges, and architecture: a comprehensive review and directions for future research," *Journal of Grid Computing*, vol. 15, no. 3, pp. 295–321, 2017, doi: 10.1007/s10723-017-9408-0.
- [19] A. G. Shoro and T. R. Soomro, "Big data analysis: Apache Spark perspective," *Global Journal of Computer Science and Technology*, vol. 15, no. C1 SE-Articles, pp. 7–14, 2015, [Online]. Available: <https://computerresearch.org/index.php/computer/article/view/1137>.
- [20] U. Suthakar, L. Magnoni, D. R. Smith, and A. Khan, "Optimised lambda architecture for monitoring scientific infrastructure," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 6, pp. 1395–1408, Jun. 2021, doi: 10.1109/TPDS.2017.2772241.
- [21] Apache Software Foundation, "Unified engine for large-scale data analytics," *Apache Spark*, 2023. <https://spark.apache.org/> (accessed Nov. 15, 2023).





- [22] E. Shaikh, I. Mohiuddin, Y. Alufaisan, and I. Nahvi, "Apache Spark: a big data processing engine," in *2019 2nd IEEE Middle East and North Africa COMMUNICATIONS Conference, MENACOMM 2019*, Nov. 2019, pp. 1–6, doi: 10.1109/MENACOMM46666.2019.8988541.
- [23] L. R. Nair and S. D. Shetty, "Streaming twitter data analysis using spark for effective job search," *Journal of Theoretical and Applied Information Technology*, vol. 80, no. 2, pp. 349–353, 2015.
- [24] T. Kolajo, O. Daramola, and A. Adebiyi, "Big data stream analysis: a systematic literature review," *Journal of Big Data*, vol. 6, no. 1, p. 47, Dec. 2019, doi: 10.1186/s40537-019-0210-7.
- [25] S. Sankarapandi, M. Sai Baba, S. Jayanthi, and E. Soundararajan, "Storing of unstructured data into MongoDB using consistent hashing algorithm," *International Journal of Emerging Technologies in Engineering Research (IJETER)*, vol. 3, no. January, 2015, doi: 10.13140/RG.2.1.3749.8961.
- [26] G. Wang and J. Tang, "The NoSQL principles and basic application of cassandra model," in *Proceedings - 2012 International Conference on Computer Science and Service System, CSSS 2012*, Aug. 2012, pp. 1332–1335, doi: 10.1109/CSSS.2012.336.
- [27] A. Svyatkovskiy, K. Imai, M. Kroeger and Y. Shiraito, "Large-scale text processing pipeline with Apache Spark," *2016 IEEE International Conference on Big Data (Big Data)*, Washington, DC, USA, 2016, pp. 3928–3935, doi: 10.1109/BigData.2016.7841068.
- [28] M. V. Kamal, P. Dileep, and D. Vasumati, "Spark streaming for predictive business intelligence," in *Advances in Intelligent Systems and Computing*, vol. 898, 2019, pp. 289–298.
- [29] "Data Pipeline development | Deductive | Data science | Data analytics," 2020. <https://deductive.com/data-pipelines/> (accessed Sep. 17, 2020).
- [30] Amazon Web services, "What is streaming data?," *Amazon Web Services (AWS)*, 2017. https://aws.amazon.com/streaming-data/?nc1=h_ls (accessed Oct. 21, 2023).

BIOGRAPHIES OF AUTHORS







Dr. Benymol Jose     pursued Masters in Computer Science from Bharathidasan University, Thiruchirappally, Tamil Nadu in 1999, M. Phil in Computer Science from Madurai Kamaraj University, Madurai, Tamil Nadu in the year 2014 and Ph.D. in computer science from Mahatma Gandhi University, Kottayam, Kerala, India in 2021 in the topic "Unstructured data mining in Bigdata: a NoSQL perspective". She is currently working as Associate Professor in Department of Computer Applications, Marian College, Kuttikkanam Autonomous, Idukki, Kerala, India. She had published many papers in journals and conference proceedings including IEEE, Elsevier, and ACM and most of them are indexed by Scopus. Her main research work focuses on unstructured data mining, machine learning, NoSQL databases and bigdata analytics. She has 25 years of teaching experience and 7 years of research experience. She can be contacted at email: benymol.jose@mariancollege.org.



Dr. Rajesh N     is currently working as an Associate Professor in the Department of Computer Applications, S A S S N D P Yogam College, Konni, Pathanamthitta, Kerala, under the affiliation of Mahatma Gandhi University, Kottayam, Kerala, India. He did his MCA from University of Madras, Tamilnadu, in 1998 and Ph.D. in Computer Science from Mahatma Gandhi University, Kottayam, Kerala in 2021. He has 24 years of undergraduate and 14 years of postgraduate teaching experience till date. He has now 7 years of research experience especially in the field of privacy preserved spatio-temporal trajectory data mining and Publication. His research interests include bigdata management, spatio-temporal data mining, e-learning, data analytics, and machine learning. He has published 17 papers in the various reputed International, National and State Journals and Conference proceedings and most of them were indexed by Scopus / ESCI / WoS. He can be contacted at email: nrjesh1121@gmail.com.



Dr. Lumy Joseph     is an Associate Professor in the Department of Computer Applications at Marian College Kuttikkanam Autonomous, in Kerala, India. She received her doctorate in "an intelligent e-learning environment for enhancing learner performance" from Mahatma Gandhi University, Kottayam, Kerala, India in 2021. Learning analytics, machine learning, educational data mining, e-learning and bigdata analytics are some of her research areas. She has articles in international journals and conference proceedings. She has 26 years of teaching experience and 7 years of research experience. She can be contacted at email: lumy.joseph@mariancollege.org.