International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 3 (2026)

ijerst.editor@gmail.com

editor@ijerst.com

ISSN 2319-5991 www.ijerst.com
© 2026 IJERST. All Rights Reserved
Peer Reviewed, Referred & Indexed Journal

Research Paper

NATURAL LANGUAGE TO SQL QUERY GENERATOR

[1] B Divya, [2] V Sai Vaishnavi, [3] E Hari Charan, [4] M Shruthi, [5] B Dhanush
[1]Assistant Professor, [2345]Students
Department of CSE(AIML)
Siddhartha Institute of Technology & Sciences, Narapally
bdivya.cse@siddhartha.co.in, 23TQ1A6639@siddhartha.co.in, 23TQ1A6649@siddhartha.co.in, 23TQ1A6644@siddhartha.co.in, 23TQ1A6609@siddhartha.co.in

ABSTRACT

The Natural Language to SQL Query Generator is an intelligent system designed to allow users to retrieve information from databases by asking questions in simple natural language. Traditionally, users need knowledge of SQL syntax to interact with relational databases. This creates difficulties for non-technical users who want to access business or organizational data without writing complex SQL queries.

The proposed system uses Natural Language Processing, Generative AI, Large Language Models, database metadata, and SQL generation techniques to convert natural-language questions into executable SQL queries. Users can enter questions such as “Show the total sales for each month” or “Find the employees who joined after 2024,” and the system generates an appropriate SQL query based on the connected database schema.

The system first analyzes the user's question and identifies important elements such as entities, conditions, filters, aggregations, sorting requirements, and requested outputs. It then retrieves relevant database schema information, including table names, column names, data types, and relationships. This context is provided to the SQL-generation engine to produce a query that matches the database structure.

Before execution, the generated SQL can pass through a validation layer that checks syntax, referenced tables and columns, query type, permissions, and other safety constraints. The validated query is then executed against the database using controlled database access. The resulting data can be displayed in tables, charts, or summarized natural-language responses.

Overall, the Natural Language to SQL Query Generator makes database querying more accessible by reducing the need for users to understand SQL syntax. It can be useful for business analysts, managers, students, developers, and other users who need to retrieve data quickly. Future enhancements can include support for multiple database types, conversational follow-up queries, automatic chart generation, query explanation, multilingual input, advanced schema reasoning, and enterprise access controls.

I. INTRODUCTION

Databases are widely used by organizations to store information about customers, employees, products, transactions, sales, inventory, and other business activities. SQL is the standard language commonly used to retrieve and manipulate information from relational databases. However, writing SQL queries requires knowledge of database structures, syntax, joins, filtering, aggregation, and other concepts.

Non-technical users may find it difficult to translate their information requirements into SQL syntax. Even simple questions can require knowledge of table relationships and column names. Complex questions involving multiple joins, grouping, filtering, and aggregation can become particularly difficult for users without database expertise.

Natural Language Processing provides a way for computers to understand human language. Recent advances in Large Language Models and Generative AI have made it possible to translate natural-language instructions into structured representations such as SQL queries. These capabilities can provide a conversational interface for interacting with databases.

The **Natural Language to SQL Query Generator** connects a natural-language interface with a database system. The user enters a question, the system analyzes the request, retrieves relevant schema information, generates SQL, validates the query, and executes it against an authorized database. The result can then be presented in a user-friendly format.

The main objective of the system is to simplify database access while maintaining correctness and security. Generated SQL should be validated before execution, and database permissions should restrict the operations available to the AI system. The system can also show the generated query so that technical users can inspect and understand how the answer was obtained.

II. LITERATURE SURVEY

Traditional database-query systems require users to write SQL commands manually. Database management tools provide query editors, schema browsers, autocomplete features, and result tables. These tools are effective for trained database users but can be difficult for individuals without SQL knowledge.

Natural Language Interfaces to Databases have been studied to allow users to ask questions using everyday language. Earlier approaches often used rule-based systems, predefined templates, and grammar-based techniques to convert natural-language questions into database queries. These approaches can work well for restricted domains but may have difficulty handling varied language and complex database structures.

Machine Learning and deep learning approaches have improved natural-language understanding for database querying. Neural models can learn relationships between user questions and SQL queries from training examples. These systems can generalize better than fixed rule-based approaches but require suitable datasets and careful evaluation.

Large Language Models have significantly expanded text-to-SQL capabilities. LLMs can analyze natural-language questions and generate SQL queries using database schema information provided through prompts or retrieval mechanisms. They can also handle complex operations such as joins, grouping, filtering, subqueries, and aggregations. However, generated queries may contain incorrect table names, columns, joins, or unsupported assumptions.

Retrieval-based approaches can improve SQL generation by providing the model with relevant schema metadata, table descriptions, relationships, sample values, and previous query examples. Query validation and execution feedback can further help identify and correct errors. These techniques are especially useful for databases with many tables and complex relationships.

Recent research is exploring agentic text-to-SQL systems that can inspect schemas, generate queries, execute them, analyze errors, and iteratively improve the query. Despite these developments, important challenges remain in SQL correctness, security, privacy, access control, ambiguity resolution, and query efficiency. The proposed **Natural Language to SQL Query Generator** combines natural-language understanding, schema-aware retrieval, Generative AI, SQL validation, controlled execution, and result visualization to provide a reliable database-querying interface.

III. SYSTEM ANALYSIS

The **Natural Language to SQL Query Generator** is designed to convert user questions written in natural language into executable SQL queries. The system receives a question through a web or conversational interface and analyzes the user's intent and requested information. A database metadata module retrieves relevant table names, column names, data types, relationships, and descriptions from the connected database. An NLP processing layer identifies entities, filters, conditions, aggregations, sorting requirements, and other query components. A schema-retrieval module selects only the relevant database information required for the current question. The Generative AI or SQL-generation engine uses the question and schema context to generate an SQL query. A query-validation layer checks SQL syntax, schema references, permissions, and potentially unsafe operations before execution. The system can optionally explain the generated query in simple language to improve transparency. The validated query is executed through a controlled database connection with appropriate user permissions. The database returns the requested records or aggregate results to the application. A result-processing module formats the returned data into tables, summaries, or visualizations. Users can review the generated SQL and ask follow-up questions to refine the query. Finally, monitoring and audit mechanisms record query activity, execution performance, errors, and access events.

Existing System

Existing database systems generally require users to have knowledge of SQL or depend on technical personnel to create queries. Database administrators and developers commonly write SQL statements manually using database management tools. Business users may need to request reports from technical teams when they cannot construct the required queries themselves. Simple filtering can sometimes be performed through graphical query builders, but complex queries may still require SQL knowledge. Users must understand table names, column names, relationships, joins, conditions, and aggregation functions. Incorrect SQL syntax can result in errors or unexpected results. Manually creating queries can also become repetitive when similar questions are asked frequently. Traditional interfaces generally do not understand natural-language questions directly. Some business-intelligence tools provide natural-language querying, but their capabilities can depend on supported data models and configurations. Security risks can occur if generated or manually entered SQL is executed without proper validation and permissions. These limitations create a need for a schema-aware natural-language interface that can generate, validate, and execute SQL safely.

Disadvantages of Existing System

- Requires SQL knowledge.
- Difficult for non-technical users.
- Manual query writing can take time.
- Complex joins are difficult for beginners.
- Users need to understand database schemas.
- SQL syntax errors can occur.
- Repetitive queries require repeated manual work.
- Limited natural-language interaction.
- Technical teams may be required for custom reports.
- Query mistakes can produce incorrect results.
- Unsafe queries can create security risks.
- Limited conversational follow-up capabilities.

Proposed System

The proposed **Natural Language to SQL Query Generator** provides a conversational interface through which users can ask database questions using natural language. The system first receives the user's request and analyzes its intent, entities, conditions, filters, aggregations, and desired output. A schema-retrieval module retrieves relevant database metadata, including tables, columns, relationships, descriptions, and data types. This schema context is combined with the user's question and provided to the Generative AI-based SQL-generation engine. The model generates an SQL query that is designed to match the available database structure. A validation layer checks the generated query for syntax errors, invalid schema references, unsupported operations, and permission restrictions. The system can also identify ambiguous questions and request clarification when required. After validation, the query is executed through a controlled database connection using appropriate access permissions. The result-processing module formats returned records into tables, summaries, or charts. The system can display the generated SQL and provide a simple explanation of what the query does. Users can ask follow-up questions such as adding filters, changing sorting, or comparing time periods. Query history can be maintained for authorized users to reuse previous queries. Overall, the proposed system makes database interaction easier while maintaining validation, security, and controlled database access.

Advantages of Proposed System

- Enables natural-language database querying.
- Reduces the need for SQL knowledge.
- Provides schema-aware SQL generation.
- Supports complex queries.
- Reduces repetitive query-writing effort.
- Provides generated SQL for transparency.

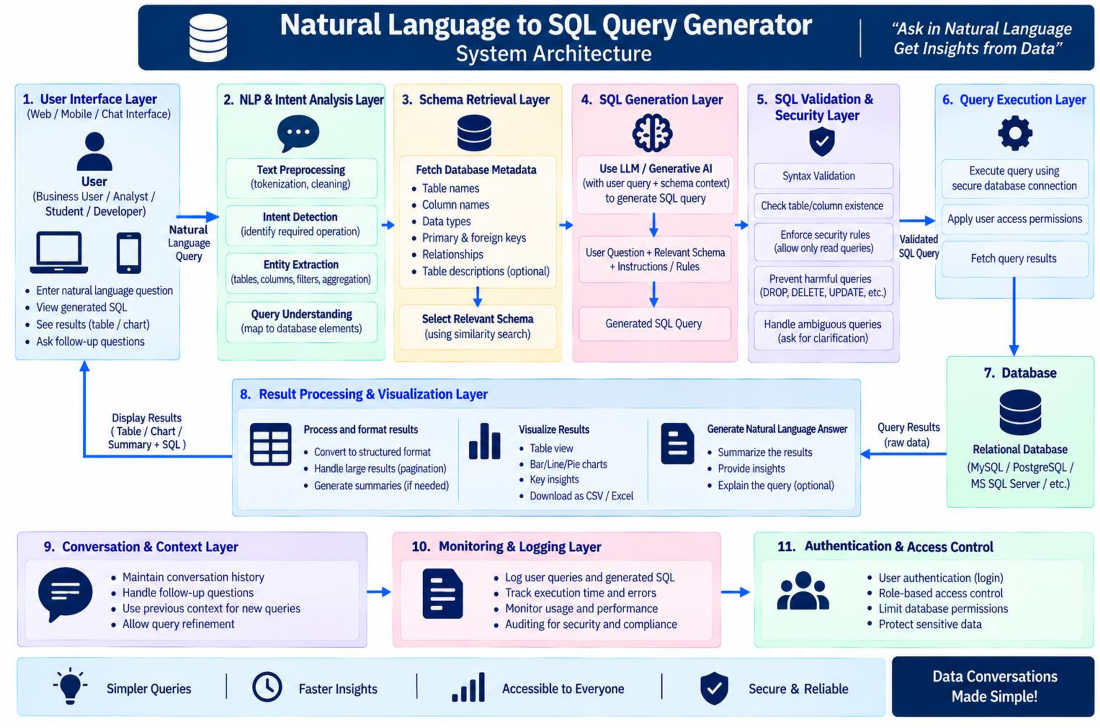
- Supports query validation before execution.
- Helps identify invalid schema references.
- Supports conversational follow-up questions.
- Can present results as tables and charts.
- Improves accessibility for non-technical users.
- Supports controlled database access.

IV. METHODOLOGY

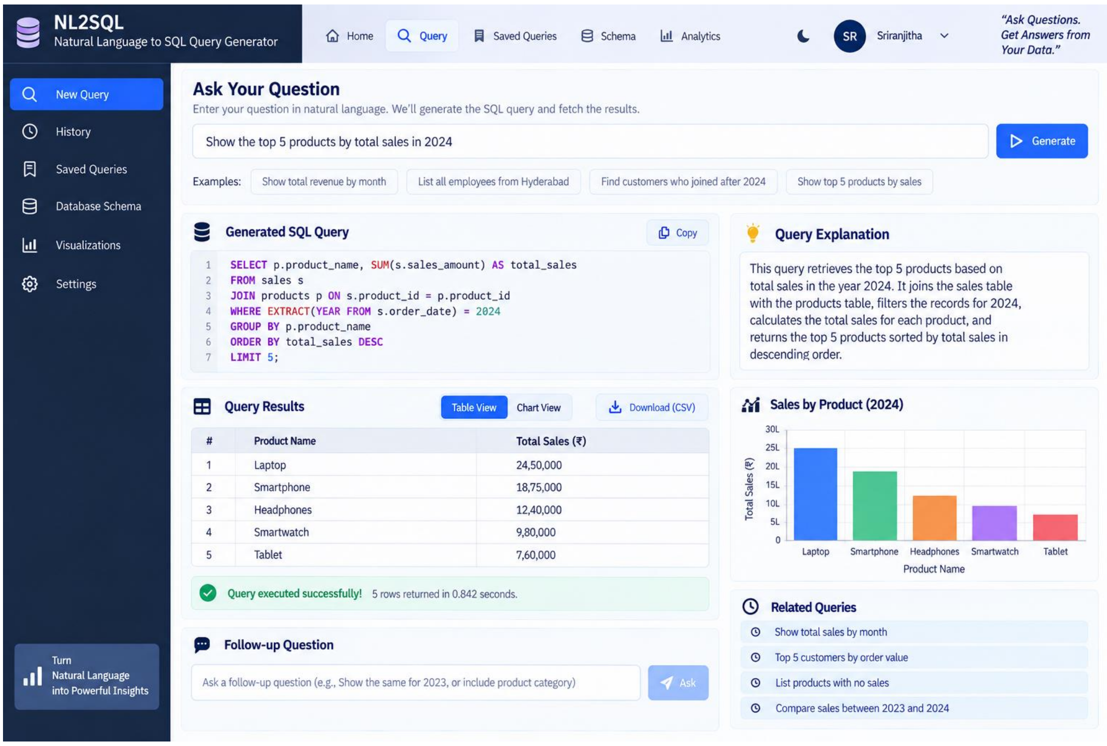
The methodology begins when a user enters a database-related question through the application interface. The NLP module analyzes the natural-language request and identifies the intended operation, entities, filters, conditions, aggregations, and expected output. The schema-retrieval module connects to the database metadata and identifies relevant tables, columns, data types, keys, and relationships. Only the schema information relevant to the user's question is selected and provided as context to the SQL-generation engine. A Generative AI model generates an SQL query based on the user's request and retrieved schema information. The generated query is passed through a validation layer that checks syntax, table and column references, query structure, permissions, and predefined security rules. If the query is invalid or the user's request is ambiguous, the system can request clarification or regenerate the query. Validated read operations are executed through a controlled database connection with appropriate access privileges. The returned data is processed and formatted into tables, summaries, or charts depending on the user's request. The system can display the generated SQL and explain its purpose in simple language. Users can continue the conversation and modify the query using follow-up instructions. Query logs, execution time, errors, and access events can be monitored for auditing and system improvement. Finally, security controls, authentication, authorization, and database-level permissions protect the underlying data from unauthorized access or modification.

SYSTEM ARCHITECTURE

The Natural Language to SQL Query Generator consists of several layers that connect the user's natural-language request with a controlled database-querying system. The User Interface Layer provides a web or conversational interface where users enter questions and view generated queries and results. The NLP and Intent Layer processes the question and identifies the requested operation, entities, filters, conditions, and output requirements. The Database Metadata Layer retrieves schema information such as tables, columns, data types, primary keys, foreign keys, relationships, and descriptions. The Schema Retrieval Layer selects relevant metadata based on the user's question. The Generative AI SQL Engine uses the question and retrieved schema context to generate an SQL query. The SQL Validation and Security Layer checks syntax, schema references, permissions, query type, and potentially unsafe operations. The Query Execution Layer sends validated queries to the authorized database connection. The Result Processing Layer converts returned data into structured tables, summaries, or visualization-ready information. The Visualization Layer can generate charts such as bar charts, line charts, or pie charts when appropriate. The Conversation Layer maintains context so users can ask follow-up questions and refine previous queries. The Monitoring and Audit Layer tracks query execution, errors, performance, and access events.



V. RESULT AND OUTPUT



VI. CONCLUSION

The Natural Language to SQL Query Generator provides an intelligent and user-friendly approach to database querying by allowing users to interact with databases using simple natural-language questions. It reduces the need for users to have detailed knowledge of SQL syntax, database relationships, joins, filtering, and aggregation.

The system combines Natural Language Processing, Generative AI, Large Language Models, database schema retrieval, SQL generation, validation, and controlled query execution to convert user questions into meaningful SQL queries. By understanding the database schema before generating the query, the system can produce queries that are better aligned with the available tables and columns.

A major feature of the proposed system is the SQL validation and security layer. Generated queries can be checked for syntax errors, invalid database references, unauthorized operations, and potentially unsafe commands before execution. This provides an additional layer of protection when AI-generated queries interact with databases.

The system also improves the usability of database information by presenting query results in tables, summaries, and visualizations. Users can view the generated SQL, understand what the query performs, and ask follow-up questions to refine their requests. This creates a conversational approach to data analysis.

Overall, the Natural Language to SQL Query Generator demonstrates how Generative AI can make database access more accessible to both technical and non-technical users. It can be useful in business analytics, reporting, education, customer analysis, inventory management, finance, and enterprise data exploration.

Future enhancements can include support for multiple database systems, multilingual queries, advanced conversational context, automatic chart selection, query optimization, voice-based database interaction, improved schema reasoning, and agentic SQL correction. Strong authentication, authorization, read-only permissions, query validation, and auditing should remain important components when deploying the system with real-world databases.

REFERENCES

[1] Kumar, R. D., Prudhviraj, G., Vijay, K., Kumar, P. S., & Plugmann, P. (2024). Exploring COVID-19 through intensive investigation with supervised machine learning algorithm. In Handbook of Artificial Intelligence and Wearables (pp. 145-158). CRC Press.

[2] Swathi, B., Vijay, K., Sushanth Babu, M., & Dinesh Kumar, R. (2024, November). Machine Learning Techniques in Cloud Based Intrusion Detection. In The International Conference on Artificial Intelligence and Smart Environment (pp. 557-564). Cham: Springer Nature Switzerland.

[3] Sv satyakrishna, shirisha rangu ,bhargavi nalacheruve.(2024) Prospective investigation on colorectal cancer with SMOTE on machine learning Algorithm

[4] Dr.G.Vishnu Murthy, BhargaviNalacheruve 1Professor, Department of computer Science & engineering, Anurag University, TS, India. 2Student, Department of computer Science & engineering, Anurag University, TS, India.

[5] V. N. S. Manaswini, K. K, C. Nigam, S. S. Ali, R. Niranjana, and Suman, “Real-Time Object Detection in Drone Surveillance Using YOLOv5,” in Proc. 2025 3rd Int. Conf. IoT, Communication and Automation Technology (ICICAT), Gorakhpur, India, 2025, pp. 1–6, doi: 10.1109/ICICAT68430.2025.11414670.

[6] B. Soundarya, V. N. S. Manaswini, M. Ayyakrishnan, R. D. Kumar, “Contextual Analysis of Big Data Analytics in Intelligent Transportation Frameworks,” in

Intersection of Artificial Intelligence, Data Science, and Cutting-Edge Technologies: From Concepts to Applications in Smart Environment, Lecture Notes in Networks and Systems, vol. 1353, Cham: Springer, 2025, doi: 10.1007/978-3-031-88304-0_79.

[7] R. D. Kumar, V. N. S.Manaswini, "Applications of blockchain in smart cities: detecting fake documents from land records using blockchain technology," in Blockchain for Smart Cities, Elsevier, 2021, pp. 105–117, doi: 10.1016/B978-0-12-824446-3.00017-X.

[8] Tejavath Veeramma, Badarla Anil, Guguloth Ravinder, "An advanced movie recommender using collaborative filtering and sentiment analysis," International Research Journal of Modernization in Engineering Technology and Science, vol. 7, no. 7, July 2025, doi: 10.56726/IRJMETS81618.

[9] Ravi Kumar Banoth, Ramana Murthy B V, "Automatic crop recommendation system using LightGBM and decision tree machine learning models," Journal of Machine and Computing, vol. 5, no. 1, pp. 343, Jan. 2025, doi: 10.53759/7669/jmc202505026.

[10] Ravi Kumar Banoth, Dr. B.V. Ramana Murthy, "Smart agriculture through IoT and machine learning for analyzing carbon footprints," in Proc. Int. Conf. Computer Science and Communication Engineering (ICCSCE), Apr. 2025.

[11] Ravi Kumar Banoth, B. V. Ramana Murthy, "Soil image classification using transfer learning approach: MobileNetV2 with CNN," SN Computer Science, vol. 5, art. no. 199, 2024, doi: 10.1007/s42979-023-02500-x.

[12] V. N. S. Manaswini, K. K, C. Nigam, S. S. Ali, R. Niranjana, and Suman, "Real-Time Object Detection in Drone Surveillance Using YOLOv5," in Proc. 2025 3rd Int. Conf. IoT, Communication and Automation Technology (ICICAT), Gorakhpur, India, 2025, pp. 1–6, doi: 10.1109/ICICAT68430.2025.11414670.

[13] B. Soundarya, V. N. S. Manaswini, M. Ayyakrishnan, R. D. Kumar, "Contextual Analysis of Big Data Analytics in Intelligent Transportation Frameworks," in Intersection of Artificial Intelligence, Data Science, and Cutting-Edge Technologies: From Concepts to Applications in Smart Environment, Lecture Notes in Networks and Systems, vol. 1353, Cham: Springer, 2025, doi: 10.1007/978-3-031-88304-0_79.

[14] R. D. Kumar, V. N. S. Manaswini, "Applications of blockchain in smart cities: detecting fake documents from land records using blockchain technology," in Blockchain for Smart Cities, Elsevier, 2021, pp. 105–117, doi: 10.1016/B978-0-12-824446-3.00017-X.

[15] M. M. Rao, M. B. Mohiuddin, P. Srinu, A. Satyanarayana, J. Madhavan and R. D. Kumar, "Carbon-Neutral-Agriculture: Renewable-Energy Powered IoT Systems and Resource Optimization," *2026 3rd International Conference on Research Methodologies in Knowledge Management, Artificial Intelligence and Telecommunication Engineering (RMKMATE)*, Chennai, India, 2026, pp. 1-6, doi: 10.1109/RMKMATE69073.2026.11518903.

[16] R. Uma, M. Maggidi, A. R. Ekkati, S. Bandlamudi, C. Madugundi and S. K. Taduri, "Zero-Trust Architecture for Hybrid Cloud Security in Critical Infrastructures," 2026 IEEE International Conference for Convergence in Computing Technology (I3CTCON), Lonavala, India, 2026, pp. 1-6, doi: 10.1109/I3CTCON68242.2026.11507458.

[17] VNS Manaswini, Chithanoor Arjun, Yamini Chouhan, & V. Sumalatha. (2026). MB-EFF: A Multi-Branch Environmental Feature Fusion Deep Learning Model for Crop Classification and Recommendation. *International Journal of Computer Information Systems and Industrial Management Applications*, *18*(22s), 460–468. https://doi.org/10.70917/ijcisim-2026-5453

[18] Yamini Chouhan, SIDDOJU DIVYA, VNS Manaswini, & J. Priyanka. (2026). Vision-Threatening Diabetic Retinopathy Detection Using EfficientNetB4: A Two-Stage Transfer Learning Approach. *International Journal of Computer Information Systems and Industrial Management Applications*, *18*(22s), 469–476. https://doi.org/10.70917/ijcisim-2026-5454
[19] Real Time Vehicle Counting and Classification system, **Dr. A.Satyanarayana,et.al** International Research Journal of Moderrnization in Engineering Technology and Science, Vol-06, Issue-05, 2024 **[ISSN: 2582-5208] doi:** 10.56726/IRJMETS57804
[20] Facial Depth Maps: Detecting Sleep Apnea with Deep Learning, **Dr. A.Satyanarayana,et.al** , International Journal of Scientific Development and Research, Vol 9, Issue-5, 2024 **[ISSN: 2455-2631]**
[21] Moodchef: Mood Based Smart Recipe Recommender with cooking Assistance, **Dr. A.Satyanarayana,et.al** , International Journal of Progressive Research in Engineering Management and Sciences, Vol-05,Issue-7, July 2025 **[ISSN: 2583-5162]** doi: **10.58257/IJPREMS42696**
[22] News Article Text Summarization, **Dr. A.Satyanarayana,et.al,** Journal of Emerging Technologies and Innovative Research, Vol-11, Issue -5, 2024, **[ISSN: 2349-5162]**
[23] Voice Assistant and gesture controlled virtual mouse using Deep learning Techniques, **Dr. A.Satyanarayana,et.al** , International Research Journal of Moderrnization in Engineering Technology and Science Vol-6, Issue-5, 2024 **[ISSN: 2582-5208] 10.56726/IRJMETS57214**