International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 07 | July - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

ETL Data Engineer

Chethan Kumar B A [1], Prof. Dr. T. Vijaya Kumar [2]

[1] Student, Department of MCA, Bangalore Institute of Technology, Karnataka, India

[2] Professor, Department of MCA, Bangalore Institute of Technology, Karnataka, India

ABSTRACT

In the analytics age of today, organizations look up to high-speed, scalable, and reliable data pipelines driving business intelligence and strategic decision-making. This project offers an ETL data engineering infrastructure that ingests raw feeds— from enterprise systems such as Salesforce, SAP, Oracle, and POS systems— offering inputs in Excel. Apache Spark with Scala is used to undertake transformation operations including cleaning, aggregation, and reformatting. A PostgreSQL database is the central repository, and containerization with Docker and Docker Compose enables a reproducible and standard configuration, including services of pgAdmin. Dashboards created with Power BI or Tableau provide interactive, real-time visualizations so that decision-makers can extract worthwhile insights. The system is designed to deliver high data quality, consistency, and accessibility at every level. Modularity makes it simple to expand to other sources or cloud-based resources. Leveraging Spark's distributed computation, the pipeline is able to process high-volume workloads with enhanced speed and reduced latency. Overall, the solution demonstrates the power of data engineering tools these days to be brought together for enabling analytics, monitoring of performance, and business growth—making it appropriate for firms seeking to modernize their data capabilities and achieve value in complex data.

1. INTRODUCTION

This ETL Data Engineering exercise is a simulated copy of an enterprise data pipeline in reality, integrating a mix of sophisticated open-source technologies. The overarching goal is to extract raw data from various business platforms—beginning with CRM applications such as Salesforce to ERP applications such as SAP—and restructuring that data in a formatted way that allows for smart analysis and strategic reporting. By simulating a complete ETL cycle, the exercise provides real-world experience in managing sophisticated data flows from start to finish. The pipeline starts with raw data files, typically Excel spreadsheets imported from databases like Oracle or Point-of-Sale (POS) databases. The files are ingested as raw input to the pipeline. Apache Spark is utilized as the chosen processing engine, where the data is processed and transformed at scale. Scala is utilized as the chosen programming language, where native compatibility with Spark features is enabled. With SparkSession and case class constructs, a schema is applied upfront to ensure data consistency and integrity while processing. Data transformation is focused on data preprocessing and cleaning. The raw numeric columns are normalized— rounded, if needed—and additional synthesized information such as city names are added by user-defined functions (UDFs). The additions are used to approximate actual geo-tagged records and prepare the data for longer-term analytical usage. Using the Spark SQL functions, the data is normalized using universal format rules enforced, i.e., it is easier to browse, aggregate, and visualize. Once the data is reshaped into form, aggregation comes next. The most elementary groupBy operations and more complex cube functions are utilized by the pipeline. The cube function enables multi-dimensional examination by constructing all possible combinations of grouping fields. The analysis type enables the stakeholders to analyze housing data in different dimensions, such as city-level incidence, time-based trends, and other aggregations that facilitate decision-making. Once the data has been cleaned and aggregated, it is in the load stage. There are two output targets. The first is having cleaned data loaded into a PostgreSQL database through a JDBC connection. The central store allows analysts and downstream tools to draw upon ready-to-use data. The PostgreSQL server and client front-end PgAdmin are executed in Docker containers defined in a docker-compose definition. The containerized environment provides reproducibility alongside easy environment management. Concurrently, the aggregated output is streamed to an Excel file with the help of the spark-excel connector. Export is beneficial for those who need static spreadsheet computation or a format that can be consumed by

visualization tools such as Power BI and Tableau. These tools are also utilized to generate interactive dashboards where users can recognize trends, anomalies, and patterns in the housing dataset. . It is built based on a Windows 11 OS using IntelliJ IDEA Community Edition as the Integrated Development Environment. Project compilation and build dependencies are handled by Scala Build Tool (SBT). Project libraries like spark-core, spark-sql, PostgreSQL JDBC drivers, the spark-excel module, and utility libraries like Amazon's Deequ for data quality validation are added to build.sbt. In terms of infrastructure, Docker is at the core of service management. The PostgreSQL of database container is to exposed to the port of 5432 and Pg-Admin on port of 8888. Volume mapping and env variables preserve data even when containers are being restarted and also allow easy communication between the containers. This makes the migration from an machine to machine & from environment to the environment a seamless and hassle-free affair without needing to reconfigure it. To enable enhanced processing power, the RDD-level map-Partitions API of Spark has been utilized for processing partitioned data to enable efficient computation across distributed nodes. MongoDB and Hive support has been added as optional dependencies in case future integration is needed with NoSQL data stores or data warehouses. In brief, this project covers the entire life cycle of ETL from

raw data extraction to transformation, loading, and visualization. It illustrates the integration of cutting-edge tools and techniques like Spark for distributed processing, Scala for functional programming, Docker for containerization, and PostgreSQL for relational storage. In this project, the data engineers are able to the acquire hands-on experience in building scalable & production-grade data pipelines on the basis of cutting-edge an industry standards.

2. RELATED WORK

• Vassiliadis provides the thorough taxonomy for — ETL architecture, comparing batch to the incremental loading & transformation styles, also metadata management approaches. He describes an automation methods & design best practices, also quality controls critical to the constructing scalable and maintainable data warehousing systems [1]. • Zaharia describe the Spark's fundamental innovations—its Resilient Distributed an Dataset abstraction and directed acyclic graph scheduler—which allow for the high-performance, in- memory computation. Their benchmarks demonstrate huge speedups over Map-Reduce for an interactive and iterative ETL workloads & supporting Spark's large-scale data processing capability [2, 6]. • Stonebraker points out the PostgreSQL's — extensibility, including functionality like multi-version concurrency control [MVCC], complex indexing, and support for an user-— efined data types. His points out its SQL standards compliance & transactional robustness, also performance optimization, which make it as the robust analytical ETL backend [3]. • Sharma & Jain compare the top BI platforms— Power BI or Tableau, & QlikView—on parameters such as usability, data connectors & visualization capabilities, also scalability. Their findings assist organizations in the selecting an appropriate BI tool to the complement ETL outputs [4]. • Gupta & Nair map the evolution of — ETL practices, recognizing common issues like error detection & latency, also schema changes. They promote metadata — driven pipeline design and test automation, also an incremental loading—a blueprint for the robust ETL systems [5]. • Rao & Prasad thoroughly analyze — Power BI, explaining its ingestion processes through Power Query & dashboarding in real-time, also the potential of DAX. They explain how Power BI is a dynamic interface for ETL-created reports [7]. • Boettiger advocates the containerizing data — workflows using Docker so it can be reproduced across an platforms with the same computational environment. His also offers hands-on advice on the bundling pipelines & managing dependencies, also exchanging standardized container images [8]. • Odersky, Spoon & Venners present Scala's — combination of an object-oriented and functional programming that enables the concise and type-safe, also parallelizable code—traits ideally suited for the large-scale ETL processes [9]. • Reddy & Kumar discuss performance — the tuning strategies within Spark through Scala-based UDFs as well as Data- Frame optimizations. They analyze the various partitioning strategies & caching strategies, also well as the transformation pipelines to minimize runtime and resource consumption in the big data ETL projects [10]. • Singh & Sharma survey the current visualization aids like — D3.js, Vega, and Plotly, discussing sound charting tactics & storytelling methods that maximize the user an interaction with ETL-based dashboard content [11]. • Demchenko his

examine architecture issues in the big data systems—such as ingestion rate & metadata management, also scalability requirements—and suggest a layered

reference framework that bridges data lakes and message queues, also batch / stream processing modules [12]. • Jones & Patel compare the PostgreSQL — to columnar and No-SQL databases & measuring indexing strategies and concurrency, also query performance. Their findings confirm that with the appropriate tuning—partitioning and materialized views—PostgreSQL is extremely competitive for the analytics workloads [13]. • Kulkarni & Deshpande — highlight the Tableau's simple drag-and-drop features and testing its data blending and calculated fields, also mapping capabilities. They illustrate its effectiveness in quickly creating the executive-level dashboards [14]. • Brown & Wilson follow — the evolution of an ETL towards cloud- native and real-time designs comparing batch and micro-batch, also streaming paradigms. They highlight the increasing significance of the event- driven design and change-data-capture in the contemporary pipelines [15]. • Agarwal & Mehta discuss an ETL system using the Kafka and Spark Streaming combination, measuring throughput, latency, also fault tolerance to demonstrate its resilience at the peak loads [16]. • Kim & Lee suggest automated — quality assurance frameworks for the ETL processes that include the schema checks and anomaly detection & automated error-correction routines [e.g., by using Deequ] to the enforce data integrity prior to loading [17]. • Silva & Costa contrast ETL — platforms like Talend and Pentaho, also Apache NiFi on connectivity, transformation capabilities, also community ecosystem—providing selection guidance geared to the specific project requirements [18]. • Nguyen & Zhang compare managed the ETL — offerings likes [AWS Glue and Azure Data Factory also Google Cloud Dataflow] , investigating server-less operation trade-offs , cost and scalability, also customization [19]. • Golfarelli & Rizzi suggest — an hybrid architecture model that integrates the streaming data ingestion with the traditional data- warehouse storage. Their proposed model an accommodates near-real-time and also an analytics without compromising [20].

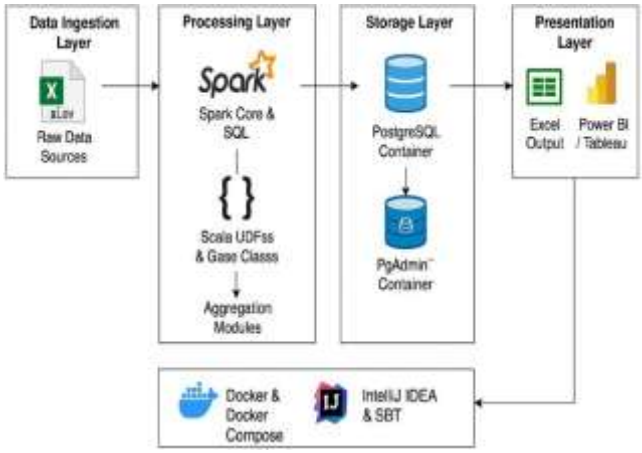
3. PROBLEM STATEMENT

This method addresses these issues by developing an end-to-end self-automated CSV & Excel ETL pipeline in a containerized form. This works on the Apache Spark's distributed computing to eliminate bottlenecks and process — humongous amounts simultaneously. Schema inference and Spark enforcement make sure data types are uniform, and default operations also custom user—defined functions normalize numeric columns and enrich categorical columns consistently. Jobs like cube and grouping are run in the same pipeline and analysis is done without individual jobs. Docker container deployment of PostgreSQL and Pg-Admin guarantees each step occurs in isolated, repeatable environments—eliminating configuration drift between dev and production. Final data is written to a relational database for simple lookup and to Excel files for enterprise business analysis. The power BI or Tableau this integration rounds for out the loop and facilitating interactive also real- time dashboards. In the end result is an end-to-end, integrated data

4. PROPOSED SYSTEM

An architecture, as proposed and leverages Apache Spark's distributed of processing capability to the parallelize data extraction and transformation, also loading like an (ETL). It is supports efficient and scalable processing of the big and heterogeneous data sets. Utilizing Scala's concise syntax and static typing maximizes the robustness of ETL code. This allows developers to express lucid transformation logic without runtime bugs and increases the durability of pipelines. Automatic data aggregation using Spark reduces manual effort through aggregation of large data into meaningful metrics and summary tables. Speedy data delivery allows timely reporting and analytics. Exporting aggregated results to Excel allows offline analysis and sharing. The user can take advantage of utilizing familiar spreadsheet tools for ad-hoc exploration, data verification, and preparation of summaries. Use

of Docker containerization enables uniform environment using dependency isolation and configuration for things like Spark jobs, PostgreSQL, and PgAdmin. Streamlining deployment processes allows quick scaling and reduces environment- related problems. Ordered transformation processes in conjunction with validation checks enhance data quality. Enforcement of schema compliance, type checking, and formatting rules implemented prior to loading data into target systems reduce inconsistencies and errors. Stack Overflow+38Sogolytics-Online Survey Tool+38cx-reports.com+38 Development in the Windows 11 environment using IntelliJ IDEA offers a stable environment along with debugging features, code completion, and integration capabilities. This setting optimizes Scala and Spark- based ETL process development.



5. METHODOLOGY

COMPONENT INTERACTION SEQUENCE IN ETL FRAMEWORK

The ETL process starts when a user triggers a scheduler, who runs autonomously and handles the sequencing and launch of the Spark-based ETL job. When it runs, the scheduler initiates the Spark job that reads raw data from a CSV file and intelligently deduces its schema. In the Spark job, a number of transformations are executed: data rows are converted to specific case classes, numeric values are rounded off, random city assignments are done, and record counts are aggregated. The cleaned and enriched data is then written out to a PostgreSQL database through a JDBC connection. In addition, summarized city-level counts are exported to an Excel file for reporting. Power BI then brings in both the Excel report and database data, which creates interactive dashboards that show up-to-date figures for informed business decision-making.

SYSTEM DATA ANALYSIS

The ETL process begins once the user triggers the Scheduler, which automatically handles the entire data extraction, transformation, and loading process. Raw data is streamed into the Spark ETL engine from sources like CSV files, staging environments, or other external systems, awaiting further processing. Scheduler sends metadata concerning the manipulated records to Spark ETL so that execution can be initiated through time schedules or event triggers. Inside Spark ETL, raw input goes through a set of operations such as data cleansing, enrichment, and aggregation. The final output is then committed to a PostgreSQL database for long- term access and analysis. To close the loop, Power BI integrates with PostgreSQL to bring back the aggregated results and filtered datasets, displaying them in interactive dashboards and visual reports that enable real-time business intelligence.

ETL ACTOR INTERACTION OVERVIEW

1. Secure Login

The user starts by choosing the "Log In" option, providing legitimate credentials to access the features and tools of the platform on an authorized basis.

2. Data Upload

Through the "Upload Data" feature, the user uploads files either in CSV or Excel format. This process triggers the data ingestion mechanism, where the content is loaded for subsequent processing and transformation.

3. Data Visualization

On selecting "Visualize Data," the user initiates the creation of dynamic dashboards. In the background, BI tools are used to transform processed data into interactive charts and visual insights.

4. Report Download

The "Download Report" link enables the user to export summary outputs—such as aggregated results—in Excel or PDF format for offline viewing or distribution.

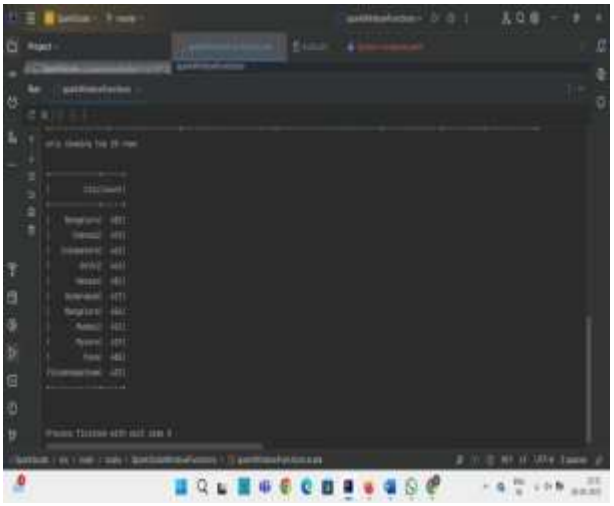
5. Secure Logout

In order to preserve system integrity and information security, the user logs off by clicking on "Log Out," essentially terminating access and warding off

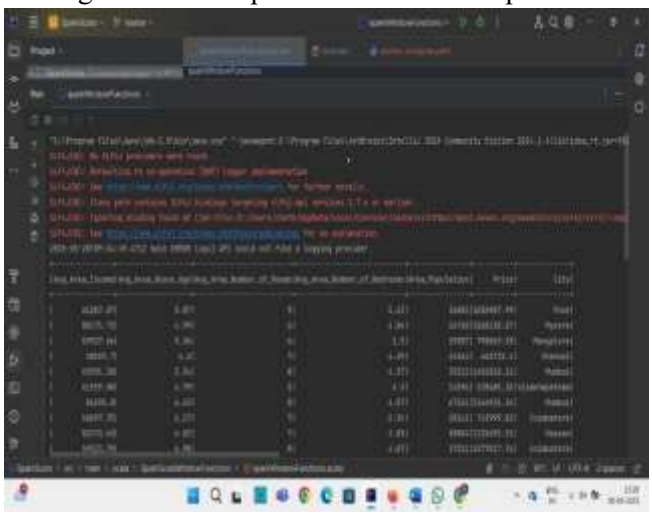
unauthorized use.

DATA PIPELINE PROCESS

The process begins with users authenticating to enable them to access the ETL functionality. Having logged in, they are presented with an option: whether to upload or not to upload. If they choose to upload, the system processes and verifies the files. In the case of not uploading, it proceeds without uploading. Secondly, the system cleans, structures, and formats the data to make it useful. This is done to prepare everything for analysis.



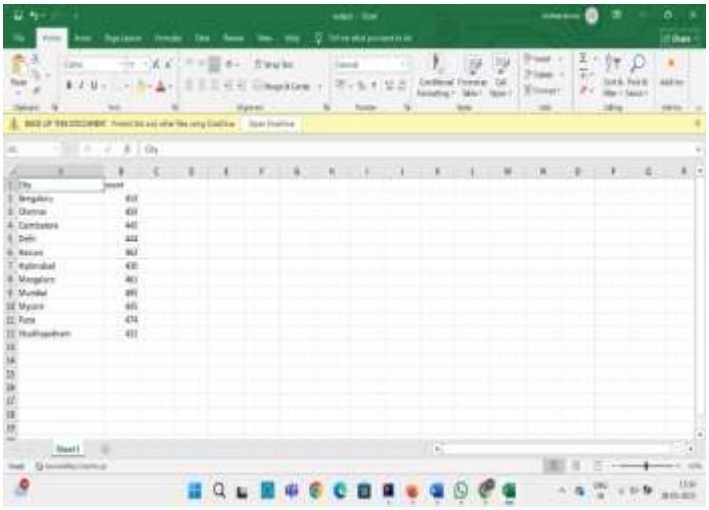
Eventually, users can choose whether to download a report. If they choose to, the system generates files they can utilize outside of the platform. The session concludes with the user signing out, securing all and making sure that the process has been completed.



6. RESULTS AND EVALUATION

SAMPLE RECORDS

The screenshot shows example entries from the Raw Data, which were initially taken from the "USA_Housing.csv" file and are now imported into the housingData DataFrame. The records highlight a number of key fields: Avg_Area_Income , Avg_Area_House_Age, Avg_Area_Number_of_Rooms , Avg_Area_Number_of_Bedrooms , Area_Population , Price , City. These columns are the basic attributes pulled during the ETL process. They get transformed—like formating and enrichment—and are utilized for aggregation and analysis in the pipeline later on.



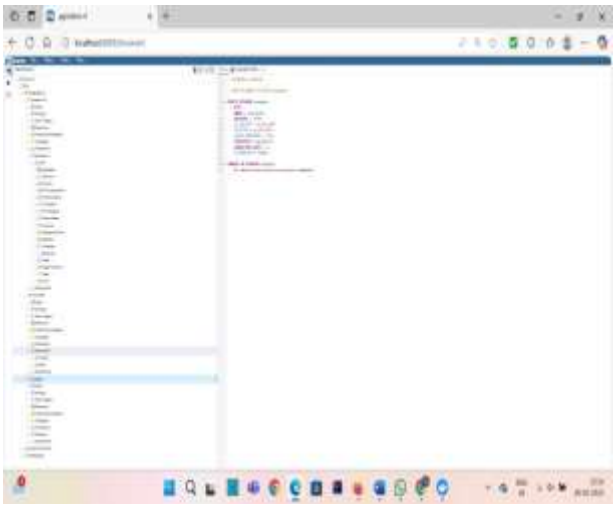
Aggregated data [city] -wise count of records

The screenshot shows example entries from the Raw Data, which were initially taken from the "USA_Housing.csv" file and are now imported into the housingData DataFrame. The records highlight a number of key fields: Avg_Area_Income, Avg_Area_House_Age, Avg_Area_Number_of_Rooms , Avg_Area_Number_of_Bedrooms , Area_Population , Price , City. These columns are the basic attributes pulled during the ETL process. They get transformed—like formating and enrichment—and are utilized for aggregation and analysis in the pipeline .

BACKEND DATABASE COMMUNICATION

In the pgAdmin UI, the left Object Explorer shows the current connection to the housingdb server. In this server, two primary schemas—public and housing—are

shown, serving to keep the database objects in a neat and organized layout. By clicking on the housing schema, users can proceed to access fundamental object types such as Tables, Views, Sequences, and Functions. This layout allows comprehensive investigation and administration of individual elements within the database. Under the Tables section, records like housingdata and housing_cube_aggregate are shown. These are the data processed and stored by Apache Spark throughout the ETL pipeline, available for querying and analysis. The SQL Editor pane on the right-hand side provides a template for CREATE DATABASE statements as a dynamic environment to execute data definition language (DDL) commands, create new schemas, or execute custom SQL queries. The top toolbar of the UI offers easy access to fundamental operations like opening new SQL tabs, object tree refreshes, object properties viewing, and performance dashboard navigation. These instruments make administrative tasks easier and improve the productivity of users when handling the PostgreSQL instance.



AFTER AGGREGATION DATA

After aggregation is done, the ETL pipeline outputs the record counts on a city-by- city basis into an Excel sheet called output.xlsx via the Spark-Excel connector. The file is saved in the directory src/main/resources/csvOutput/, where it can be accessed, opened, or downloaded for viewing and analysis purposes. Business analysts then load the Excel file into Power BI so that the visual reports are made from the latest aggregated data. Inside Power BI, analysts create interactive visualizations—like bar graphs and pie charts—to clearly show city-level pattern of distribution. These visuals help easily spot trends and outliers within different geographic locations. Decision-makers and stakeholders use these interactive dashboards to monitor trends in housing data and make informed, data-driven strategic decisions.

VISUALIZATION ON POWER BI

On Power BI Desktop, the very top ribbon contains data import icons such as "Get Data," "Excel Workbook," and "SQL Server" that enable easy integration with data sources such as PostgreSQL or Excel files—perfect for importing datasets such as housingdata for analysis. The report canvas clearly has a pie chart front and center that illustrates the city distribution of records. Each wedge is explicitly marked with total as well as with corresponding percentage, providing a compact, easy-to-digest overview of the composition of the dataset. On the right, the Visualizations pane illuminates the active chart type here being the pie chart, while below in the Values and Fields areas, they are free to assign data attributes to determine how the visual will behave. To the right of this is the Filters pane, which contains settings for "Filters on this page" and "Filters on all pages." Users can drag and drop particular fields, such as city or numerical metrics, to dynamically filter the report and customize dashboard views. The Data pane verifies that the housingdata table is imported successfully. It gives a formatted representation of the dataset, set for transformation, modeling, and the generation of impactful visual insights in Power BI.

7. CONCLUSION

ETL Data Engineering Project provides an end-to-end data pipeline that fetches unstructured data from a wide range of enterprise-level sources, processes it through advanced frameworks, and stores it in a consistent database system to support analytical as well as visualization procedures. It smoothly combines technologies like Apache Spark to support big-scale data processing, Scala to serve as a programming language, PostgreSQL to store data, and Docker to containerize the process to build a scalable, sustainable, and high-performance workflow. At the heart of the project lies Spark's ability to perform distributed processing, which makes quick and dynamic conversions of large datasets possible. The system can read data from Excel files, execute complex operations such as data aggregation, formatting, and enrichment, and save preprocessed data to PostgreSQL so that data collection is smoothed out into information- dense reporting. It offers integration with big enterprise systems like Salesforce, SAP, and POS systems, consolidating disparate data sources into a single dataset for analysis. Containerizing PostgreSQL and PgAdmin using Docker makes installation, deployment, and management of environments easy, providing consistency between production and development. In addition, its integration with Power BI and Tableau increases its utility by offering dynamic dashboards to facilitate business decision-making.

There existed a systematic testing plan—integration, acceptance, and performance—employed throughout development to facilitate stability, correctness, and performance. Integration tests ensured uniform data flow among modules; acceptance tests ensured business requirement compliance and data integrity; and performance tests enabled removing the bottlenecks, facilitating scalability with increasing data volume and user load. By implementing ETL automation, the project overcomes common problems of manual data management and isolated analytics processes, reducing errors and time consumption. It also aids in on-demand data summarization and Excel reporting for flexible reporting needs.

This project offers a firm base for scalable business data solutions. Certain potential areas of future improvement could be the addition of real-time data streams, the integration of automated quality checks, and expansion to machine learning-based analytics. Overall, the project offers a strong and versatile architecture allowing organizations to utilize data for strategic advantage effectively.

REFERENCES

1. P. Vassiliadis, "A Survey of ETL Technologies," 2009. A technology survey of ETL systems, describing their designs, operation, and best practices.
2. M. Zaharia et al., "Big Data Processing with Apache Spark," 2016. A technical survey of Apache Spark's design, quoting its performance parameters and scalability features.
3. M. Stonebraker, "PostgreSQL: The World's Most Advanced Open Source RDBMS," 2018. A listing of PostgreSQL's state-of-the-art features, including its extensibility and performance features.
4. S. Sharma & P. Jain, "Comparative Study of Business Intelligence Tools," 2020. Comparative study of top business intelligence tools based on usability, feature, and performance.
5. H. Gupta & R. Nair, "ETL Processes in Data Warehousing: A Review," 2017. Overview of new as well as recent ETL techniques, along with problems and best practices encountered.
6. M. Zaharia et al., "Apache Spark: Lightning-Fast Cluster Computing," 2012. Overview of the RDD paradigm of Spark and the in-memory computing capability that it provides.
7. V. Rao & S. Prasad, "Power BI for Data Analysis," 2021. Overview of Power BI's set of features of data connection, transformation, and visualization capability.
8. C. Boettiger, "An Introduction to Docker for Reproducible Research," 2015. Discussion of the use of Docker containers in making reproducible computational research workflows possible.
9. M. Odersky, L. Spoon & B. Venners, "Scala: A Scalable Language," 2010. Analysis of Scala features in combining object-oriented and functional programming paradigms.
10. P. Reddy & R. Kumar, "Enhancing ETL Efficiency with Spark and Scala," 2019. Improvement of efficiency in ETL using Spark and Scala optimization.
11. A. Singh & R. Sharma, "Review of Data Visualization Techniques," 2021. Review of new data visualization techniques, technologies, and best practices in analytics.
12. Y. Demchenko et al., "Data Engineering Challenges in Big Data Analytics," 2014. Summary of architectural, processing, and management challenges in big data systems.
13. M. Jones & D. Patel, "Evaluating PostgreSQL for

Data Analytics," 2020. PostgreSQL performance, scalability, and analytical workload support side-by-side comparison.

14. S. Kulkarni & R. Deshpande, "Business Intelligence Using Tableau," 2019. Summary of Tableau support for linking features, dashboard building, and interactive visualizations.

15. T. Brown & K. Wilson, "The Role of ETL in Modern Data Platforms," 2022. Summary of the changing role of ETL in cloud-native and real-time data architecture.

16. P. Agarwal & V. Mehta, "Streaming Data Ingestion with Kafka and Spark," 2020. Summary of the use of Kafka and Spark Streaming in data ingestion of big data.

17. D. Kim & Y. Lee, "Data Quality Management in ETL Processes," 2018. Discussion of techniques for validating, cleaning, and warranting data quality in ETL processing.

18. J. Silva & F. Costa, "Data Integration Using Open-Source Tools," 2017. Comparison of open-source ETL tools for low-cost integration of heterogeneous data sources.

19. T. Nguyen & Y. Zhang, "Cloud-Based ETL Solutions for Scalable Analytics," 2021. Comparative survey of cloud ETL services providing scalability, dependability, and low- cost analytics solutions.

20. M. Golfarelli & S. Rizzi, "Framework for Real-Time Data Warehousing," 2011. Suggestion of an architecture that merges real-time data streams and data warehouse.