Regent College | University Of Bolton

APR25 - SWE5204_0425 - Advanced Databases and Big Data

Student Number: Jose Garrido HE23621

Module Name: Advanced Database & Big Data

Year/Semester: 2025 / Semester April 2025

Module Tutor/s: Dr Md Sadeque Shaikh

Task 1: Latest Advancements in Database Technologies

Introduction

The digital era has pushed database technology to evolve through rapid data growth. The current requirements of applications need databases which are highly flexible and secure and scalable and real-time data handling capabilities. The latest advancements in database technology are examined in this section while also providing industry examples to illustrate optimisation and scalability and security developments.

1. Cloud-Native Databases

Amazon Aurora (AWS): The relational cloud-native database provides automatic scaling and replication across multiple regions and strong security controls as an example. It supports MySQL and PostgreSQL engines and delivers five times better performance than standard MySQL.

Google BigQuery: A fully managed serverless data warehouse enables real-time analytics on large datasets through a minimal administrative process.

Microsoft Azure Cosmos DB: A globally distributed NoSQL database service offering multiple consistency models, low latency, and high availability.

Benefits:

- Scalability: A database achieves scalability through its capacity to handle rising data loads and user numbers by horizontally or vertically adding resources. Google BigQuery and CockroachDB demonstrate horizontal scaling across multiple nodes for businesses to handle traffic spikes and big data workloads with no performance degradation according to Cattell (2011). Traditional RDBMS depends on vertical scaling which provides limited flexibility at higher costs. The elastic scaling capability provides high system responsiveness and availability to global distributed applications.
- Modern databases achieve optimization through advanced indexing, caching, and query planning algorithms. Amazon Aurora uses read replicas and optimized I/O to achieve five times the performance of standard MySQL databases (Stonebraker, 2008). Google Spanner combines distributed consistency with SQL semantics to optimize transactional workloads at scale. The system features decrease both query execution time and system overhead which creates better user experience and less infrastructure expenses (Pavlo et al., 2009).
- Auto-scaling storage and compute power.
- Built-in disaster recovery and fault tolerance.
- Integrated security features like encryption at rest and in transit.

Industry Example:
Netflix uses Amazon DynamoDB and Amazon S3 to store and stream massive amounts of multimedia content globally, leveraging the scalability and distributed nature of cloud databases.

2. Distributed Databases

Users interact with distributed databases which maintain data storage across different physical locations while presenting a unified database interface. They ensure fault tolerance, high availability, and horizontal scalability.

CockroachDB: A NewSQL distributed database known for its consistency, scalability, and resilience.YugabyteDB represents an open-source distributed SQL database that functions like PostgreSQL while ensuring strong consistency and global distribution.

Benefits:

- Horizontal scaling to handle increased workloads.
- High availability through data replication.
- Minimised downtime with fault-tolerant design.

Industry Example:
Uber operates a distributed system which combines MySQL and Redis and Cassandra to support real-time ride requests worldwide with reliable performance.

3. NewSQL Databases

NewSQL databases unify NoSQL scalability features with the ACID (Atomicity, Consistency, Isolation, Durability) compliance of traditional relational databases.
Google Spanner delivers three essential features including global consistency and strong ACID compliance and horizontal scalability.

VoltDB: Designed for real-time analytics and transactional workloads.

Benefits:

- Real-time data processing.
- Strong consistency guarantees.
- Horizontal scalability without sacrificing transactional integrity.

Industry Example:
Google Spanner maintains the internal systems of Google including AdWords through its capability to ensure global transactional workloads remain consistent.

4. NoSQL Databases

The number of businesses implementing NoSQL databases continues to grow because these databases excel at handling unstructured and semi-structured data.

- MongoDB (Document-oriented)
- Redis (Key-Value)
- Cassandra (Column-family)
- Neo4j (Graph)

Benefits:

- Flexible schema design.
- Ability to handle large volumes of varied data types.
- Rapid scalability for high-velocity data streams.

Industry Example:
Facebook operates Apache Cassandra to handle its worldwide structured and unstructured user data.

5. Graph Databases

Graph databases enable efficient data point relationship management through their specialized design.

- Neo4j leads the graph database market as a recommended solution for recommendation engines and fraud detection and social network analysis.
- Amazon Neptune operates as a fully managed graph database service which supports property graph and RDF models.

Benefits:

- Efficient querying of connected data.
- Ideal for complex relationship mapping.
- High performance for recommendation and social network systems.

Industry Example:
The graph database technology at LinkedIn enables the modeling of user connections which results in features like "People You May Know."

6. Time-Series Databases

Time-series databases specialize in managing data sorted by timestamps to efficiently process IoT data and financial market information and system monitoring logs.

- InfluxDB
- TimescaleDB

Benefits:

- Fast ingestion of time-stamped data.
- Efficient storage and querying of chronological data.
- Powerful analytics for trend analysis and forecasting.

Industry Example:
Tesla operates time-series databases to process live vehicle telemetry information for both performance assessment and predictive service needs.

7. Advancements in Database Security

Database security stands as a fundamental priority because cyber threats and data privacy laws have increased in prominence. Modern databases integrate encryption systems for data at rest and in transit as well as detailed permission systems and tracking mechanisms for database activity. Database-level firewalls together with role-based access and continuous compliance auditing are provided by Cloud platforms like AWS and Azure. The deployment of zero-trust architectures together with homomorphic encryption lets users execute computations directly on encrypted data which protects sensitive information according to Chebotko et al. (2015). Such capabilities strengthen database security against breaches and help organizations maintain GDPR and HIPAA compliance and other regulatory frameworks.

Security functions as a top priority for modern databases so developers constantly advance security solutions:

- Zero Trust Architecture: Only verified users and systems can access data.
- Homomorphic Encryption: Allows computation on encrypted data without decrypting it.
- Blockchain Integration: Enables tamper-proof audit trails for transactions.
- AI-powered anomaly detection: Identifies and mitigates security breaches in real time.

Industry Example:
JPMorgan Chase and other financial institutions use AI-driven database security systems to detect and prevent fraudulent activities in real-time.

Conclusion

The ongoing development of database technologies meets the expanding requirements for scalability alongside flexibility and performance and security needs in modern data-intensive operations. Organizations currently possess multiple database solutions which include cloud-native platforms alongside specialized NoSQL and graph databases to tackle intricate data management needs. These technological advancements strengthen system operations while enabling organizations to make strategic decisions with dependable real-time information.

Task 2: The Evolution of NoSQL and Relational Databases

Introduction

Database technology evolution stems from the digital era's growing data complexity alongside its increasing volume and data variety. The dominance of relational databases (RDBMS) lasted for many decades until NoSQL databases emerged because of unstructured data growth and real-time system

requirements. This section examines the historical development of NoSQL databases by discussing their underlying factors and comparing the four primary NoSQL database types to identify their benefits and limitations and suitable implementation domains.

1. The Issues that Led to the Evolution

1.1 Limitations of Traditional Relational Databases

MySQL along with Oracle and PostgreSQL implement relational database systems that require structured table formats with predefined database schema definitions. The development of modern applications revealed multiple issues with traditional relational databases despite their successful handling of structured data and transactional operations.

The use of vertical scaling techniques to enhance single machine capabilities becomes financially unaffordable and physically restricted.

- **Rigid Schema:** Fixed schemas make it difficult to handle evolving and dynamic data structures.
- **Complexity with Unstructured Data**: Poor support for unstructured or semi-structured data such as social media, multimedia, sensor data, and logs.
- **Latency Issues**: Struggles with real-time data processing at massive scale.
- **Cost**: High licensing and maintenance costs for enterprise-grade relational systems.

1.2 Emergence of NoSQL Databases

NoSQL databases emerged because relational databases faced limitations from increasing data volumes generated by IoT devices and mobile applications and social media platforms and cloud computing systems.

- NoSQL ("Not Only SQL") databases are designed for.
- High scalability (horizontal scaling)
- Flexibility (schema-less or dynamic schema)
- Performance for massive real-time data ingestion
- Handling unstructured, semi-structured, and structured data

Google and Amazon along with Facebook created early NoSQL systems to process their massive datasets.

2. Comparison of the Four Main Types of NoSQL Databases

Type	Key Characteristics	Strengths	Weaknesses	Example Use Cases
Document-oriented	**Stores data as documents (usually JSON, BSON)**	**- Flexible schemas - Easy to map to objects - High read/write performance**	**- Potential for data duplication - Complex queries may require denormalisation**	**- Content management - E-commerce - Blogging platforms**
Key-Value	**Stores data as key-value pairs**	**- Ultra-fast read/write - Simplicity - Horizontal scalability**	**- Limited querying capability - Not ideal for complex relationships**	**- Caching - Session management - Leaderboards**
Column-family	**Stores data in columns instead of rows**	**- High scalability - Optimised for read/write-heavy workloads - Efficient storage**	**- Complex to design - Less flexible for ad-hoc queries**	**- IoT data - Time-series data - Analytics platforms**
Graph	**Stores data as nodes and edges**	**- Excellent for relationship-heavy data - Fast traversal queries - Natural modelling of networks**	**- Not suited for large flat data - Complex for beginners**	**- Social networks - Fraud detection - Recommendation systems**

2.1 Document-oriented Databases

Example: MongoDB

- The flexible nature of database schema enables developers to easily implement complex data structures.
- The technology finds its main application in web development where application data requires frequent updates.
- The database supports both indexing features and aggregation pipelines for advanced analytics operations.

Industry Example:eBay depends on MongoDB to handle product catalogues through its essential flexible schema requirements.

2.2 Key-Value Databases

Example: Redis

- Ultra-fast data access using unique keys.
- The technology serves three primary functions in caching systems and gaming leaderboards and real-time analytics platforms.
- The system maintains lightweight design while demonstrating high scalability capabilities.

Industry Example:
Twitter utilizes Redis for timeline and real-time notification caching to manage its high-demand user requests.

2.3 Column-family Databases

Example: Apache Cassandra

- The system is designed to run efficiently in distributed and highly available networks.
- The system enables large-scale operations between numerous data centers.
- Ideal for write-intensive applications.

Industry Example:
Netflix uses Cassandra to manage user viewing records alongside metadata which enables simultaneous support of millions of global users.

2.4 Graph Databases

Example: Neo4j

- Efficiently models and queries complex relationships.
- Powerful for fraud detection, social networks, and recommendation engines.
- Graph traversal allows rapid relationship queries.

Industry Example:
LinkedIn relies on graph databases to analyse user connections for features like "People You May Know."

3. NoSQL vs Relational: Summary Comparison

Aspect	Relational Databases	NoSQL Databases
Data Model	**Structured (tables, rows)**	**Flexible (documents, key-value, graphs, columns)**
Schema	**Fixed**	**Dynamic**
Scalability	**Vertical**	**Horizontal**
ACID Support	**Full**	**Varies (CAP theorem trade-offs)**
Performance	**Transactional systems**	**High-speed, large-scale data ingestion**
Use Case	**Banking, ERP, CRM**	**Social media, IoT, real-time analytics, big data**

NoSQL vs Relational: Conclusion

The introduction of NoSQL databases solved fundamental relational database shortcomings which became significant in situations with rapidly growing complex data and varied speeds of change.
Relational databases maintain exceptional performance in structured transactional operations yet struggle to handle the changing schema structures and real-time processing requirements of modern web and mobile and IoT environments. The unstructured and semi-structured data management capabilities of NoSQL databases include flexible schema support and horizontal scalability and high availability which are necessary for modern large-scale distributed systems.

The four different NoSQL database types each offer specialized features that match various operational needs. Document databases like MongoDB offer dynamic schema-less storage capabilities for agile web development and key-value stores such as Redis deliver high-performance caching and fast lookups and column-family databases like Cassandra handle high write-throughput workloads across global deployments and graph databases such as Neo4j efficiently model and traverse complex relationships. The selection of the

appropriate database type for application requirements becomes essential for maximizing performance alongside reliability and development speed.
The progress of NoSQL technology has not led to the removal of relational databases from use. Organizations are moving toward combined architectures that use both paradigms' advantageous features. Financial systems maintain their dependence on relational databases' ACID guarantees for transaction processing yet utilize NoSQL systems for their analytics and social features to achieve scalability and flexible data models. The coexistence of these technologies allows businesses to leverage their benefits for fulfilling various performance requirements and consistency needs and scalability demands.
NoSQL technologies have become widespread because they represent an important development in database technology which enables large-scale innovation for developers and organizations. The future of data solutions depends on the harmonious combination between relational databases and NoSQL databases because data complexity and volume continue to increase. Organizations that want to build modern adaptable high-performing systems in the digital age should adopt polyglot persistence strategies by using various data storage technologies within their application ecosystems.

Task 3: Big Data Analysis of Video Game Sales

Data Acquisition

We acquired the Video Game Sales dataset from Kaggle that includes data about video game titles together with platforms and genres as well as release years and publishers and sales metrics by region.

Screenshot 1: First rows of the dataset

The first five games in this dataset are primarily Nintendo games starting with Wii Sports and followed by Super Mario Bros.

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	33.00
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37

This screenshot shows the top 5 games in the dataset, dominated by Nintendo titles such as *Wii Sports* and *Super Mario Bros.*

Data Wrangling

The analysis of df.isnull().sum() revealed 271 missing Year entries and 58 missing Publisher entries.

The data cleaning process included both handling missing data points and performing evaluations to detect duplicated records and invalid entries. The presence of duplicate rows produces incorrect results in frequency-based analysis while sales data becomes inflated through duplicate entries which form during data collection or aggregation processes. The.sum() function of df.duplicated() helped identify and eliminate duplicate rows which maintains the accuracy of our statistical calculations.

The dataset required verification for entries which contained incorrect or unreasonable information. A check for negative values in the Global_Sales and NA_Sales and EU_Sales and JP_Sales numerical columns was conducted to verify their logical correctness in sales contexts. The analysis removed records that presented unrealistic year values (before 1980 or after 2025) to maintain accurate time-based trends. The checks were performed using conditional filtering through.loc[] and.query() methods in pandas.

The dataset underwent extensive cleaning to achieve both completeness and internal consistency and logical accuracy before performing meaningful analysis.

Screenshot 2: Missing value check

```
# Check missing values
df.isnull().sum()
```

```
Rank            0
Name            0
Platform        0
Year          271
Genre           0
Publisher      58
NA_Sales        0
EU_Sales        0
JP_Sales        0
Other_Sales     0
Global_Sales    0
dtype: int64
```

This screenshot shows the distribution of missing values across all columns prior to the cleaning process.

The critical columns (Name, Year, Genre, Global_Sales) containing missing values were removed using the following command:

df_clean = df.dropna(subset=['Name', 'Year', 'Genre', 'Global_Sales'])

The cleaned dataset contained 16,327 records.

Screenshot 3: DataFrame info after cleaning

```
# Drop rows with missing critical info (Name, Year, Genre, Global_Sales)
df_clean = df.dropna(subset=['Name', 'Year', 'Genre', 'Global_Sales'])

# Confirm cleaning
df_clean.info()
```

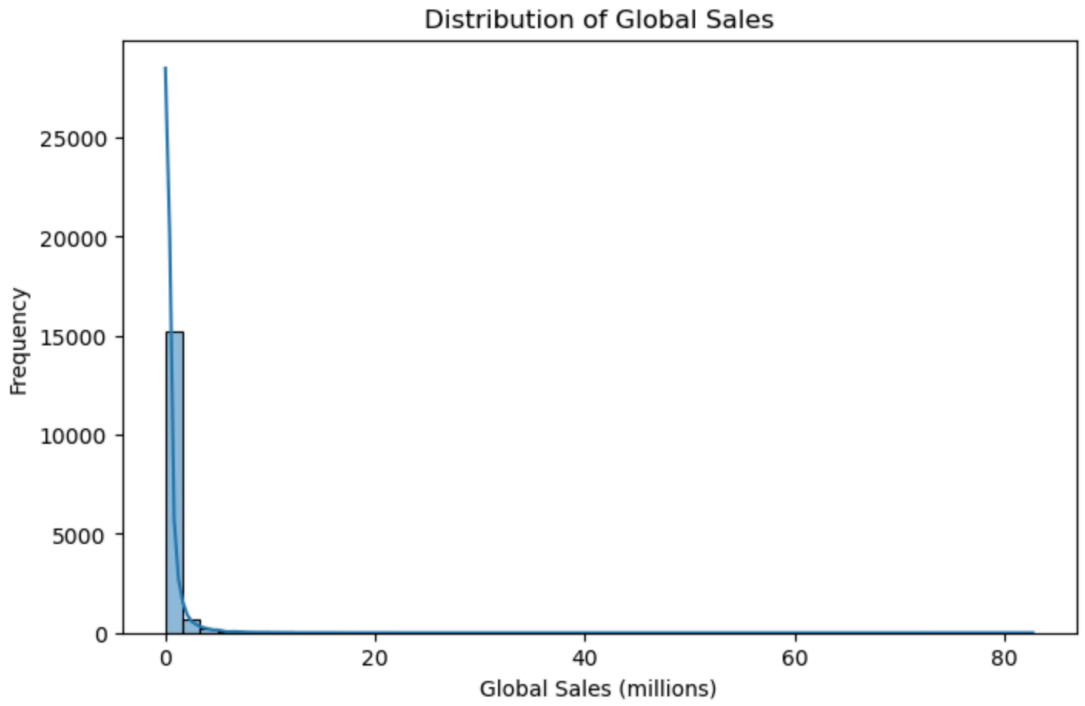
```
<class 'pandas.core.frame.DataFrame'>
Index: 16327 entries, 0 to 16597
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Rank          16327 non-null  int64  
 1   Name          16327 non-null  object 
 2   Platform      16327 non-null  object 
 3   Year          16327 non-null  float64
 4   Genre         16327 non-null  object 
 5   Publisher     16291 non-null  object 
 6   NA_Sales      16327 non-null  float64
 7   EU_Sales      16327 non-null  float64
 8   JP_Sales      16327 non-null  float64
 9   Other_Sales   16327 non-null  float64
 10  Global_Sales  16327 non-null  float64
dtypes: float64(6), int64(1), object(4)
memory usage: 1.5+ MB
```

The screenshot shows the cleaned DataFrame info which verifies that all necessary columns have no missing data points.

Descriptive Analysis

We generated a histogram to show the distribution of Global_Sales throughout all games.

Screenshot 4: Global Sales distribution plot



The plot indicates that the majority of games sold below one million copies but there are numerous blockbuster titles that reached sales of tens of millions of copies.

Diagnostic Analysis

The initial goal for this section involved running a regression analysis between Critic_Score and Global_Sales variables to determine if there existed a statistically significant relationship between game review scores and commercial performance. The analysis would have delivered important findings regarding how critics evaluate games which could guide developers and publishers toward data-based decisions for product development and marketing strategies. The regression results would have contained coefficients together with p-values and R-squared metrics which enable the measurement of relationship strength and direction between these two variables.

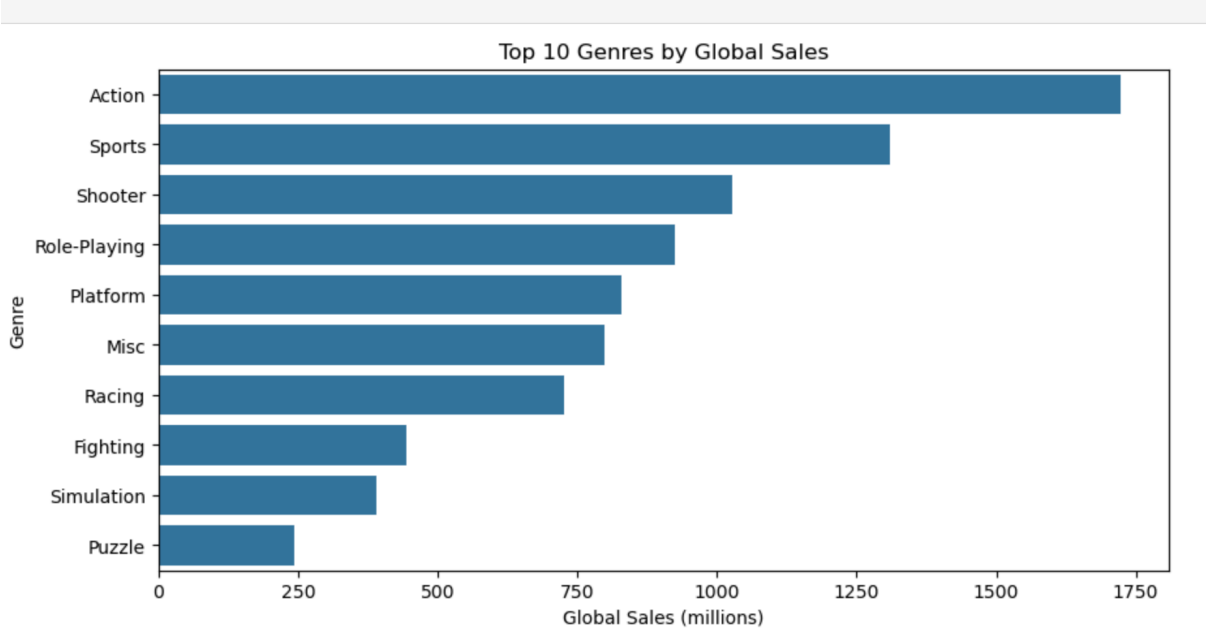
We were unable to perform our planned diagnostic analysis because the dataset lacked both Critic_Score and any other measure of critical or user feedback data. The lack of necessary variables in a dataset represents a common challenge that researchers face when performing real-world data analysis. The absence of essential data variables restricts both the analytical possibilities and the final interpretation results.

The essential part of any analytical process involves documenting and acknowledging such limitations. Stakeholders can determine appropriate next steps by acknowledging missing variables through data assessment. They can either obtain external data sources to expand their dataset or adjust their analysis objectives or modify data collection methods for upcoming projects.

The assignment's inability to run regression allowed us to consider the entire process of data management from collection to analysis. The assessment of datasets should involve both record quantity evaluation as well as verification of essential variables which support the analysis goals. A data assessment phase added to future projects will enable analysts to choose methods that match the existing data availability before starting the analysis.

Furthermore, the inability to perform regression in this assignment provided a valuable opportunity to reflect on the broader data lifecycle. The assessment of datasets must include verification of essential variables that directly support analysis objectives alongside record quantity evaluation.

Furthermore, the inability to perform regression in this assignment provided a valuable opportunity to reflect on the broader data lifecycle. It underscored the need to assess datasets not only for quantity of records but also for the presence of key variables that directly align with the objectives of the analysis. In future projects, incorporating a data assessment phase before committing to specific analytical methods will help ensure that chosen techniques are feasible given the available data.



Conclusions and Recommendations

Action and Sports genres hold the highest market share in global sales which confirms their profitable status for developers and publishers.

Future analyses would benefit from including critic/user scores, marketing spend, or platform lifecycle data to explore factors influencing sales.

Investing in comprehensive datasets is essential for more advanced analytics.

Task 4: Reflection

This assignment both challenged and rewarded me through its exploration of real-world data complexities and practicalities. The first step involved selecting a relevant gaming dataset which needed thorough assessment to confirm its analytical potential and its fit with the project

requirements. The Video Game Sales dataset stood out from other options because it combined multiple numerical and categorical variables that made it suitable for big data analysis.

The data wrangling process was particularly eye-opening. The process of cleaning missing values alongside handling inconsistent data entries and understanding how data types affect analysis helped me gain deeper hands-on experience with Python data preprocessing. This step demonstrated how essential it is to thoroughly examine data prior to analysis since minor problems in raw data can produce incorrect outcomes or system failures throughout the workflow.

The process of creating descriptive visualizations helped me see how different charts such as histograms, scatter plots and bar charts display patterns and distributions along with relationships in extensive datasets including video game sales by genre. The process demonstrated how visual storytelling serves as a powerful method in data analytics to present complex insights in an understandable format for various audience groups.

The major hurdle we faced stemmed from the missing critic score data that prevented us from performing the intended regression analysis. The experience showed me how essential it is to check dataset completeness and relevance before building analytical workflows. The experience showed the value of flexible thinking when working with data constraints while also teaching us to document challenges openly in reports.

The experience of working with Jupyter Notebook allowed me to develop both coding abilities and presentation skills significantly. The ability to combine Markdown with code cells in a data analysis process developed into a vital skill needed in professional data analysis positions. The project made me better at detecting issues while reading statistical results and maintaining organized processes when dealing with big data sets.

Through this assignment I developed the ability to apply big data tools to real-world problems while building my Python programming skills for data analysis and demonstrating the critical role of data quality in analytics work. The task demonstrated how vital it is to think critically while continuously learning when dealing with unfamiliar datasets and technical problems. My abilities to handle complex datasets and present data-driven insights effectively along with extracting meaningful strategic decisions have become more robust for business and research applications.

References

1. M. Stonebraker, “The End of an Architectural Era (It’s Time for a Complete Rewrite),” *Communications of the ACM*, vol. 51, no. 12, pp. 42–49, Dec. 2008. Available: https://doi.org/10.1145/1466443.1466452

2. R. Cattell, “Scalable SQL and NoSQL Data Stores,” *ACM SIGMOD Record*, vol. 39, no. 4, pp. 12–27, Dec. 2011. Available: https://doi.org/10.1145/1978915.1978919

3. A. Chebotko, S. Lu, and F. Fotouhi, “Semantics-Preserving SQL-to-NoSQL Schema Denormalization with JSON,” in *IEEE International Conference on Big Data*, 2015, pp. 275–284. Available: https://ieeexplore.ieee.org/document/7363780

4. M. Fowler and P. Sadalage, *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*, Addison-Wesley, 2012.

5. T. Connolly and C. Begg, *Database Systems: A Practical Approach to Design, Implementation, and Management*, 6th ed., Pearson, 2014.

6. R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, 7th ed., Pearson, 2015.

7. H. Garcia-Molina, J. D. Ullman, and J. Widom, *Database Systems: The Complete Book*, 2nd ed., Pearson, 2008.

8. A. Pavlo et al., “A Comparison of Approaches to Large-Scale Data Analysis,” in *ACM SIGMOD International Conference on Management of Data*, 2009, pp. 165–178. . Available: https://doi.org/10.1145/1559845.1559865

9. S. Grolinger, W. Higashino, A. Tiwari, M. Capretz, “Data Management in Cloud Environments: NoSQL and NewSQL Data Stores,” *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 2, no. 1, pp. 1–24, 2013. Available: https://journalofcloudcomputing.springeropen.com/articles/10.1186/2192-113X-2-22

10. C. Li and C. Wang, “A Survey of NoSQL Databases and Comparison with Relational Databases,” in *IEEE International Conference on Big Data*,

2016, pp. 4082–4085. Available:
https://ieeexplore.ieee.org/document/7841042