[Regent College | University Of Bolton](https://www.bolton.ac.uk/existing-partners/regent-college)

[**APR25 - SWE5204\_0425 - Advanced Databases and Big Data**](https://rcl.instructure.com/courses/2732)

**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 3**](#_heading=h.31hgrhpibhjj)

[Introduction 3](#_heading=h.1lgaitr33l0d)

[1. Cloud-Native Databases 3](#_heading=h.91wdffy7bqli)

[2. Distributed Databases 3](#_heading=h.m6ot3qv82i5w)

[3. NewSQL Databases 4](#_heading=h.9big2wquwmts)

[4. NoSQL Databases 4](#_heading=h.36dzu8dnkfd4)

[5. Graph Databases 5](#_heading=h.c0kcp7fqznd2)

[6. Time-Series Databases 5](#_heading=h.lrromx2q49h)

[7. Advancements in Database Security 5](#_heading=h.olbz689dn4vi)

[**Task 2: The Evolution of NoSQL and Relational Databases 6**](#_heading=h.ss4um3tlm612)

[Introduction 6](#_heading=h.8ujk0x4uhego)

[1. The Issues that Led to the Evolution 6](#_heading=h.a5xh9ogvj6u)

[1.1 Limitations of Traditional Relational Databases 6](#_heading=h.ei16h99rregz)

[1.2 Emergence of NoSQL Databases 7](#_heading=h.8i3k2ccdm4z4)

[2. Comparison of the Four Main Types of NoSQL Databases 7](#_heading=h.wmbiu310zhij)

[2.1 Document-oriented Databases 8](#_heading=h.xsxtixiil59g)

[2.2 Key-Value Databases 8](#_heading=h.md5jjmdl337n)

[2.3 Column-family Databases 8](#_heading=h.vf93h4g3kdvp)

[2.4 Graph Databases 8](#_heading=h.2ledz8jb9chy)

[3. NoSQL vs Relational: Summary Comparison 9](#_heading=h.8eod2y7jq49z)

[**Task 3: Big Data Analysis of Video Game Sales 10**](#_heading=h.j48pvyoitj0q)

[Data Acquisition 10](#_heading=h.5bxyuqgr7j5k)

[Data Wrangling 10](#_heading=h.238aegivzsaj)

[Descriptive Analysis 13](#_heading=h.q99nbla3gwd8)

[Diagnostic Analysis 13](#_heading=h.n7w8wqz9df1t)

[Conclusions and Recommendations 14](#_heading=h.jolacwcq42cc)

[**Task 4: Reflection 14**](#_heading=h.5n7rlo95pdcn)

[**References 15**](#_heading=h.63grl937olip)

# Task 1: Latest Advancements in Database Technologies

## Introduction

The rapid expansion of data in the digital era has driven significant advancements in database technologies. Modern applications demand databases that are highly scalable, secure, flexible, and capable of handling massive volumes of data in real time. This section investigates the latest developments in database technologies, focusing on optimisation, scalability, and security, while providing real-world industry examples.

## 1. Cloud-Native Databases

Cloud computing has revolutionised database deployment by introducing cloud-native databases that offer elastic scalability, high availability, and managed services. Major cloud providers such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud offer robust solutions:

* Amazon Aurora (AWS): An example of a relational cloud-native database that provides automatic scaling, replication across multiple regions, and strong security controls. It supports MySQL and PostgreSQL engines, offering up to five times better performance than standard MySQL.
* Google BigQuery: A fully managed, serverless data warehouse that allows real-time analytics on massive datasets with minimal administrative effort.
* Microsoft Azure Cosmos DB: A globally distributed NoSQL database service offering multiple consistency models, low latency, and high availability.

**Benefits:**

* 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

Distributed databases allow data to be stored across multiple physical locations while appearing as a single database to users. They ensure fault tolerance, high availability, and horizontal scalability.

* CockroachDB: A NewSQL distributed database known for its consistency, scalability, and resilience.
* YugabyteDB: An open-source distributed SQL database compatible with PostgreSQL, offering 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 utilizes a distributed architecture combining MySQL, Redis, and Cassandra to handle real-time ride requests across the globe with high reliability.

### 3. NewSQL Databases

NewSQL combines the scalability of NoSQL systems with the ACID (Atomicity, Consistency, Isolation, Durability) guarantees of traditional relational databases.

* Google Spanner: Provides global consistency, 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 supports Google’s internal systems, such as AdWords, ensuring global consistency across massive transactional workloads.

### 4. NoSQL Databases

NoSQL databases are increasingly popular for 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 uses Apache Cassandra to manage large amounts of structured and unstructured user data across its global network.

### 

### 5. Graph Databases

Graph databases are designed to represent complex relationships between data points efficiently.

* Neo4j: The leading graph database used in recommendation engines, fraud detection, and social network analysis.
* Amazon Neptune: Fully managed graph database service supporting both 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:**  
LinkedIn employs graph databases to model and analyse user connections, enabling features such as “People You May Know.”

### 6. Time-Series Databases

Time-series databases are optimized for handling data indexed by timestamps, suitable for IoT, financial market data, and monitoring systems.

* 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 utilises time-series databases to collect real-time telemetry data from vehicles for performance monitoring and predictive maintenance.

### 7. Advancements in Database Security

Security remains a critical concern with modern databases, leading to continuous advancements:

* 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:**Financial institutions like JPMorgan Chase implement AI-driven database security systems to monitor and prevent fraudulent activities in real time.

**Conclusion**

The continuous evolution of database technologies addresses the growing demands for scalability, flexibility, performance, and security in today's data-driven world. From cloud-native solutions to highly specialised NoSQL and graph databases, organisations now have many tools to handle complex data challenges. These innovations enhance system capabilities and drive strategic business decisions based on reliable, real-time insights.

# Task 2: The Evolution of NoSQL and Relational Databases

## Introduction

The evolution of database technologies has been driven by the increasing complexity, volume, and variety of data generated in the digital era. Traditional relational databases (RDBMS) have been the standard for decades, but the rise of unstructured data and real-time application demands have led to the emergence of NoSQL databases. This section analyses the reasons behind this evolution, compares the four main types of NoSQL databases, and highlights their strengths, weaknesses, and application areas.

### 1. The Issues that Led to the Evolution

#### 1.1 Limitations of Traditional Relational Databases

Relational databases such as MySQL, Oracle, and PostgreSQL store data in structured tables with predefined schemas. While highly effective for structured data and transactional operations, several challenges emerged with the rise of modern applications:

* Scalability Limits: Vertical scaling (adding more power to a single machine) is costly and limited.
* 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

The limitations above, combined with the explosion of data from IoT, mobile apps, social media, and cloud computing, fueled the development of NoSQL databases. 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

Companies like Google, Amazon, and Facebook drove early NoSQL innovations to handle their enormous data workloads.

### 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**

* Schema flexibility allows developers to store complex data structures easily.
* Popular in web development, where application data can change frequently.
* Supports indexing and aggregation pipelines for advanced analytics.

**Industry Example:***eBay* uses MongoDB to manage product catalogues where schema flexibility is crucial.

#### 2.2 Key-Value Databases

Example: Redis

* Ultra-fast data access using unique keys.
* Commonly used for caching, gaming leaderboards, and real-time analytics.
* Lightweight and highly scalable.

**Industry Example:***Twitter* employs Redis for caching timelines and real-time notifications to handle high-volume user requests.

#### 2.3 Column-family Databases

Example: Apache Cassandra

* Optimised for distributed, highly available systems.
* Allows massive scalability across multiple data centres.
* Ideal for write-intensive applications.

**Industry Example:***Netflix* uses Cassandra for storing user viewing history and metadata to serve millions of users worldwide simultaneously.

#### 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** |

**Conclusion**

The emergence of NoSQL databases fundamentally addressed the limitations of traditional relational databases, particularly in environments experiencing exponential growth in data complexity, variety, and velocity. Relational databases excel at structured, transactional workloads, but struggle to adapt to the unpredictable schemas and real-time processing demands of modern web, mobile, and IoT applications. NoSQL databases, on the other hand, were specifically designed to handle unstructured and semi-structured data, offering flexible schemas, horizontal scalability, and high availability, which are essential for today’s large-scale, distributed systems.

Each NoSQL type — document, key-value, column-family, and graph databases — provides unique strengths tailored to specific scenarios. Document databases like MongoDB enable dynamic, schema-less storage for agile web development; key-value stores such as Redis excel in caching and rapid lookups; column-family databases like Cassandra manage high write-throughput workloads across global deployments; and graph databases such as Neo4j model and traverse complex relationships with remarkable efficiency. Selecting the right type based on application requirements is key to optimizing performance, reliability, and development agility.

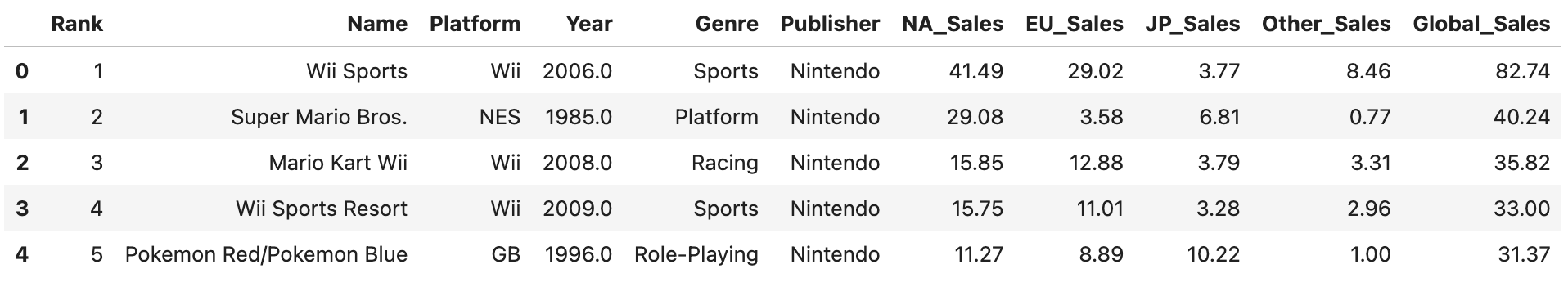
Furthermore, the evolution of NoSQL has not rendered relational databases obsolete. Instead, many organizations now adopt hybrid architectures that combine the strengths of both paradigms. For instance, financial systems continue to rely on the ACID guarantees of relational databases for transactions, while analytics or social features may leverage NoSQL systems to achieve scalability and flexible data models. This coexistence enables businesses to harness the advantages of both technologies, ensuring they can meet diverse performance, consistency, and scalability needs.

In conclusion, the widespread adoption of NoSQL technologies represents a significant advancement in the database landscape, empowering developers and organizations to innovate at scale. As data continues to grow in volume and complexity, the synergy between relational and NoSQL databases will remain central to designing robust, responsive, and future-proof data solutions. Embracing a polyglot persistence strategy — using multiple data storage technologies within the same application ecosystem — has become a best practice for organizations aiming to deliver high-performing and adaptable systems in the modern digital era.

# Task 3: Big Data Analysis of Video Game Sales

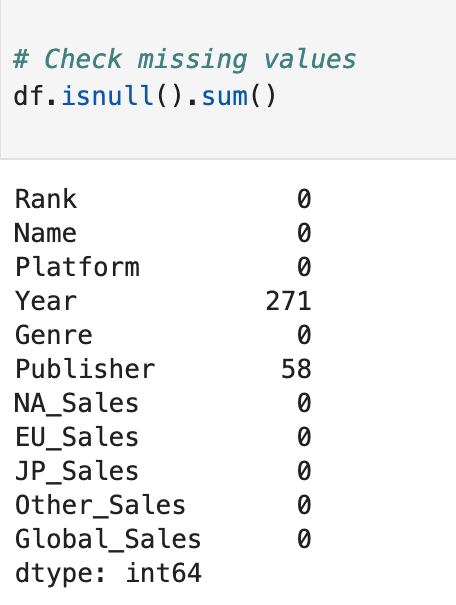
#### Data Acquisition

We obtained the *Video Game Sales* dataset from Kaggle, containing data on video game titles, platforms, genres, years of release, publishers, and sales across different regions.

**Screenshot 1: First rows of the dataset  
****This screenshot shows the top 5 games in the dataset, dominated by Nintendo titles such as *Wii Sports* and *Super Mario Bros.*

#### Data Wrangling

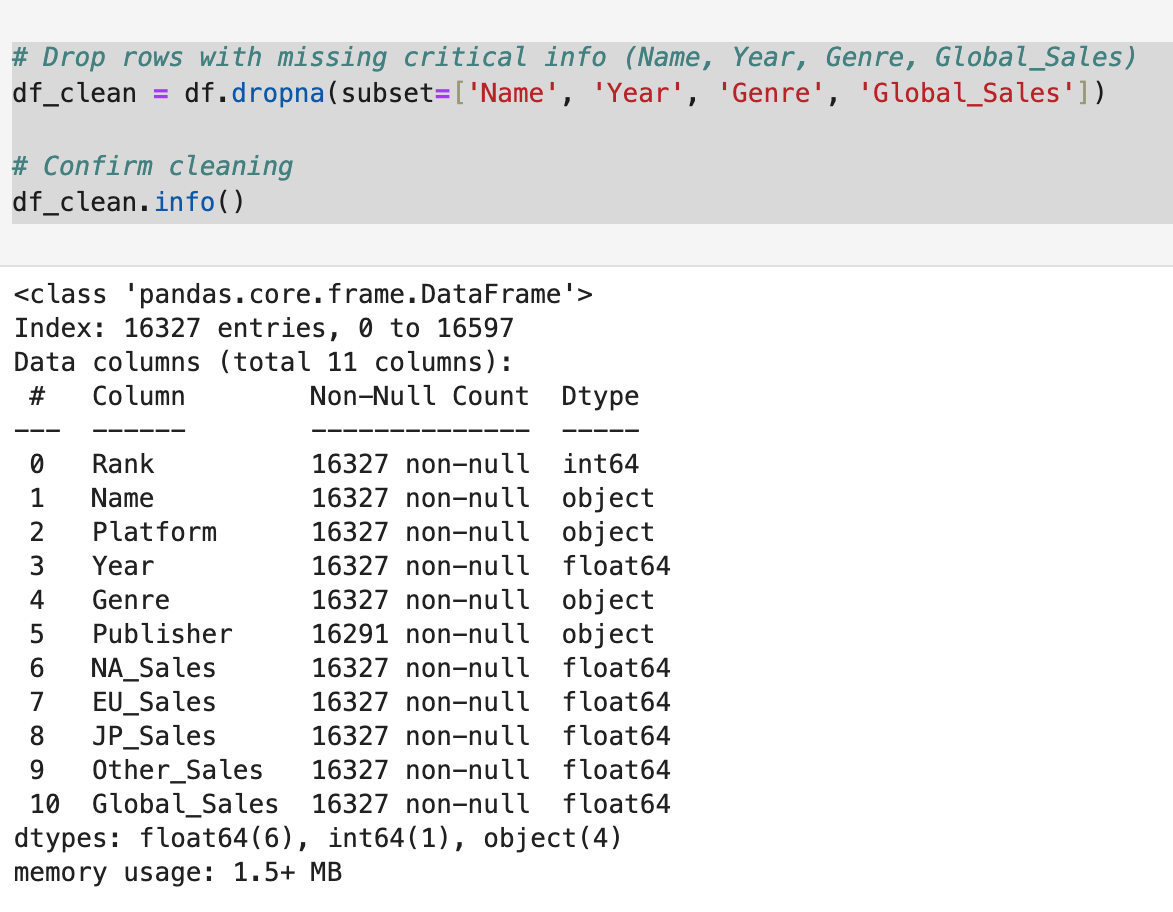
We checked for missing values in the dataset using df.isnull().sum() and identified 271 missing entries in the Year column and 58 missing entries in the Publisher column.

**Screenshot 2:** Missing value check ******This screenshot displays the number of missing values per column before cleaning.

We then dropped rows with missing values in the critical columns (Name, Year, Genre, Global\_Sales) using:

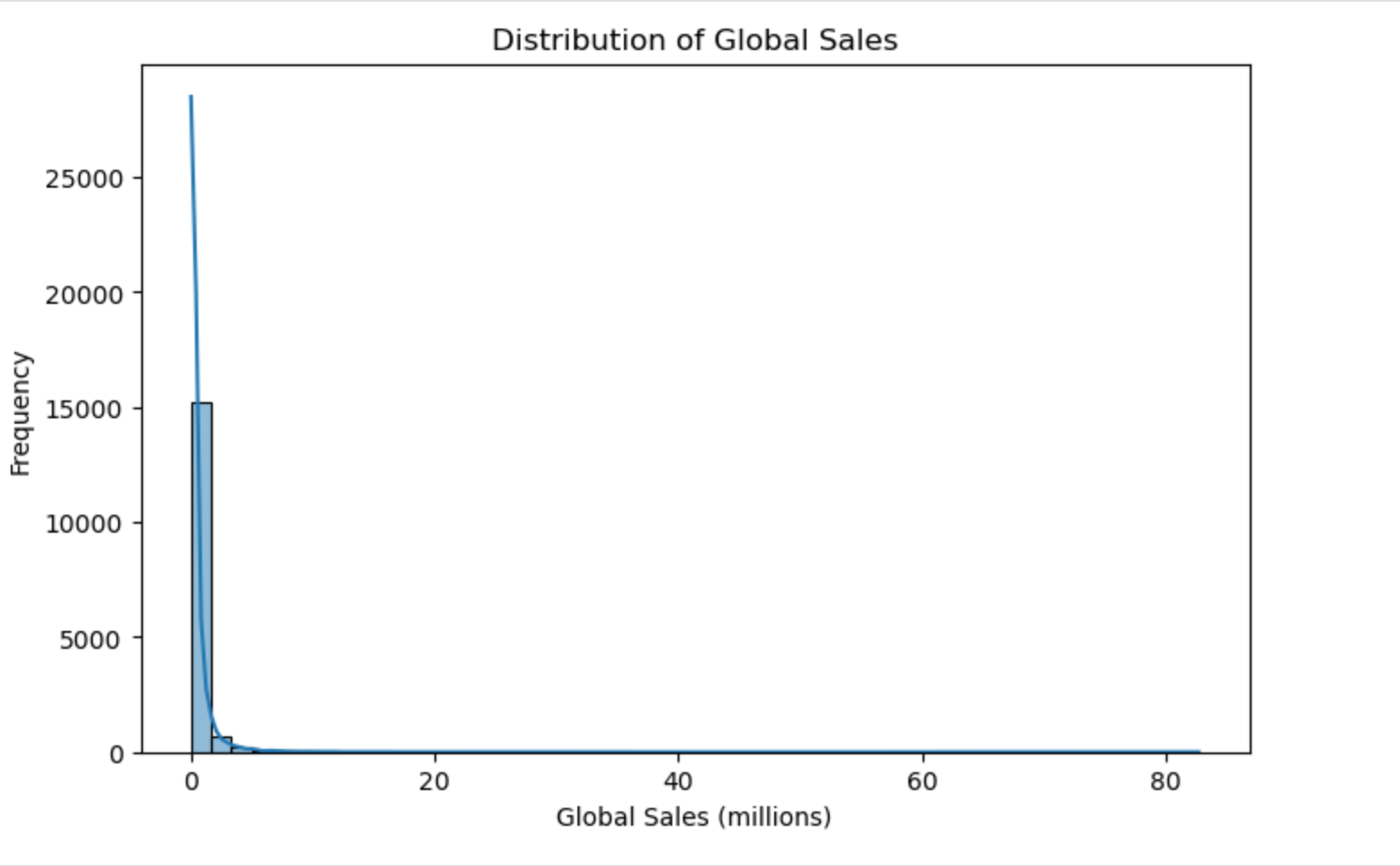
**df\_clean = df.dropna(subset=['Name', 'Year', 'Genre', 'Global\_Sales'])**

After cleaning, the dataset contained 16,327 records.

**Screenshot 3:** DataFrame info after cleaning  
******This screenshot shows the cleaned DataFrame info, confirming that all critical columns have no missing values.

#### Descriptive Analysis

We created a histogram to visualize the distribution of Global\_Sales across all games.

**Screenshot 4**: Global Sales distribution plot  
The plot shows that most games sold fewer than 1 million copies, with a long tail of a few blockbuster titles exceeding tens of millions in sales.

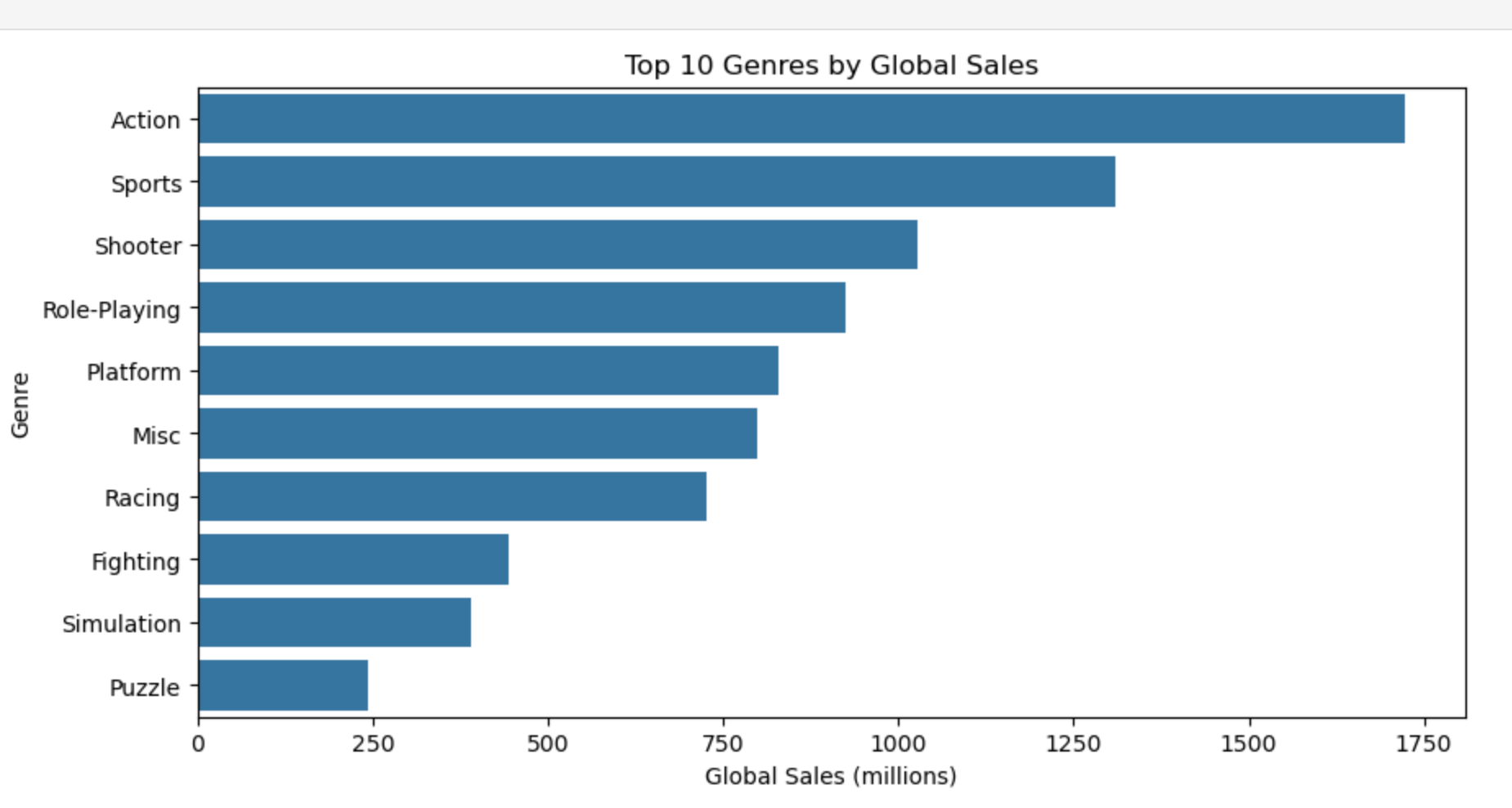
#### Diagnostic Analysis

The original plan for this section was to perform a regression analysis between the Critic\_Score and Global\_Sales variables to investigate whether there was a statistically significant relationship between game review scores and their commercial success. This analysis could have provided valuable insights into how critical reception impacts sales performance, helping game developers and publishers make data-driven decisions about product quality and marketing strategies. Regression results would have included coefficients, p-values, and R-squared metrics, allowing us to quantify the strength and direction of the relationship between these two factors.

Unfortunately, the dataset did not include a Critic\_Score column or any equivalent measure of critical or user feedback, which prevented us from conducting the intended diagnostic analysis. This limitation highlights a key challenge often encountered in real-world data analysis: the dataset may not always contain all the necessary variables to answer specific business or research questions. Missing variables can constrain the scope of analysis and limit the conclusions that can be drawn.

Despite this setback, acknowledging and documenting such limitations is a vital part of any analytical process. Recognizing gaps in available data allows stakeholders to make informed decisions about whether to supplement the dataset with external sources, adjust analysis goals, or redesign data collection processes for future projects. This experience reinforced the importance of dataset completeness and relevance when planning predictive analytics and demonstrated the need for flexibility when actual data availability falls short of initial expectations.

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 tend to dominate global sales, suggesting these remain lucrative categories 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

Completing this assignment was both challenging and rewarding, providing valuable insights into the complexities and practicalities of working with real-world data. Initially, identifying a suitable dataset relevant to gaming required careful consideration to ensure it contained meaningful variables for analysis and aligned with the assignment’s objectives. I explored several data sources before deciding on the Video Game Sales dataset, which offered a rich combination of numerical and categorical variables suitable for big data analytics.

The data wrangling process was particularly eye-opening. Cleaning missing values, handling inconsistent data entries, and understanding how data types affect analysis deepened my practical knowledge of data preprocessing in Python. This step reinforced the importance of thorough data exploration before jumping into analysis, as small issues in raw data can lead to misleading results or runtime errors later in the workflow.

Generating descriptive visualizations helped me appreciate how histograms, scatter plots, and bar charts can reveal patterns, distributions, and relationships in large datasets, such as the dominance of certain genres in global video game sales. It also highlighted how visual storytelling is a powerful tool in data analytics, allowing complex insights to be communicated clearly and effectively to diverse audiences.

A major challenge encountered was the absence of critic scores in the dataset, which limited the planned regression analysis. This experience taught me the importance of assessing dataset completeness and relevance before designing an analytical workflow. It also demonstrated the need for adaptability when dealing with data limitations, and reinforced the value of documenting challenges transparently in reports.

Furthermore, working with Jupyter Notebook significantly improved my coding and presentation skills. I learned how to integrate Markdown with code cells to create a structured, readable, and reproducible analytical process, which is a critical skill in professional data analysis roles. The assignment also enhanced my ability to troubleshoot errors, interpret statistical outputs, and maintain a logical workflow when dealing with large datasets.

Overall, this assignment strengthened my ability to apply big data tools and concepts to real scenarios, increased my confidence in using Python for data analysis, and highlighted the significance of data quality and integrity in any analytics project. It also emphasized the necessity of critical thinking and continuous learning when tackling unfamiliar datasets or technical challenges. I now feel better prepared to handle complex datasets, communicate data-driven findings effectively, and draw actionable insights that can inform strategic decisions in real-world business or research environments.

# 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>