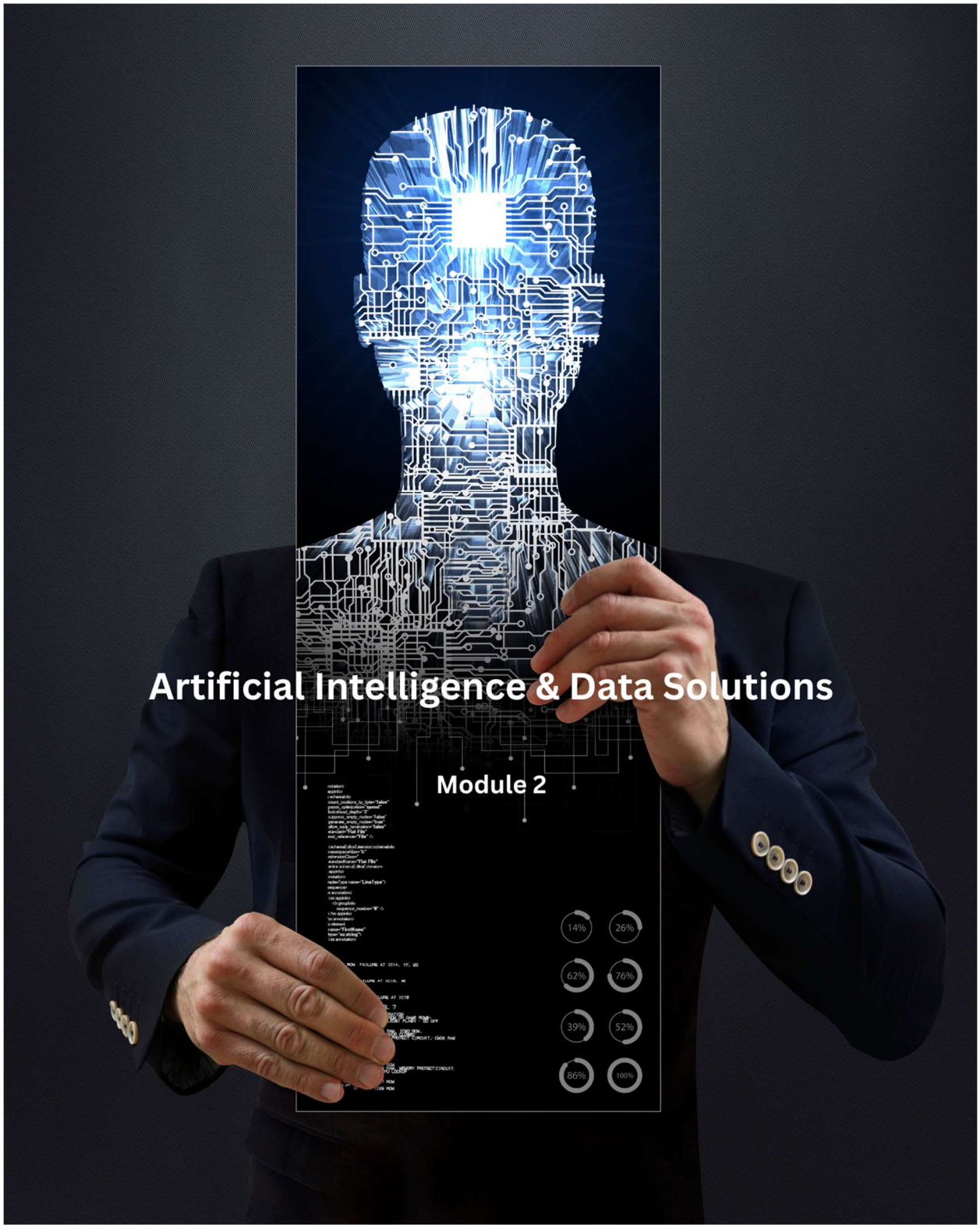

Artificial Intelligence & Data Solutions
Module 2

# Module 2: Data Architecture, Big Data Engineering, and SQL

## 2.1 Relational Database Systems and Advanced SQL Engineering

## Relational Database Management Systems (RDBMS)

At the core of structured corporate data solutions sits the Relational Database Management System (RDBMS). Based on relational algebra, an RDBMS organizes data into strict, two-dimensional tables consisting of rows (tuples) and columns (attributes). Data integrity, consistency, and reliability across transactions are enforced through strict adherence to the **ACID** framework:

- **Atomicity:** Transactions are treated as atomic units. Either the entire transaction executes successfully, or every single operation is completely rolled back.
- **Consistency:** A transaction can only transition the database from one valid state to another, maintaining all predefined schema rules and constraints.
- **Isolation:** Concurrent execution of transactions yields the exact same database state as if they were executed sequentially.
- **Durability:** Once a transaction is committed, its changes are permanently written to non-volatile storage and will survive subsequent system crashes.

## Database Normalisation Frameworks

To eliminate data redundancy and prevent data anomalies (insertion, update, and deletion issues), schemas follow systematic structural normalisation stages:

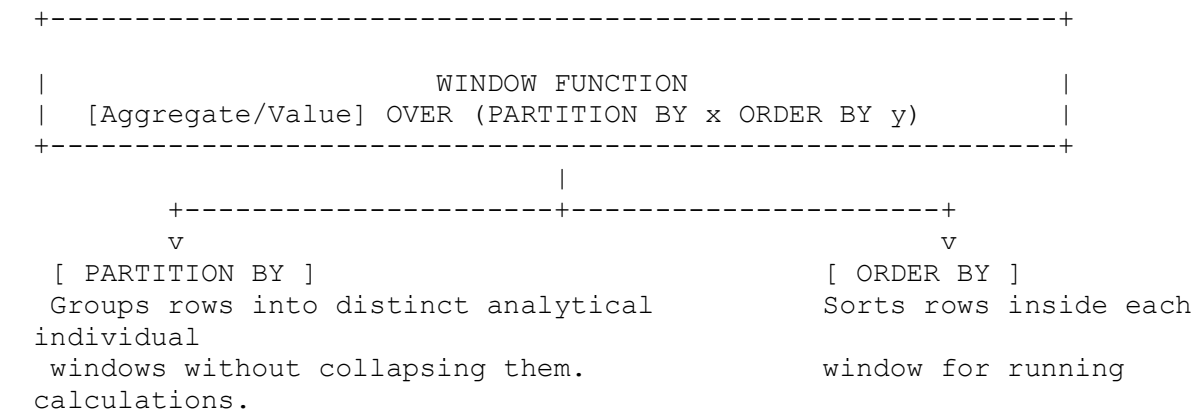
- **First Normal Form (1NF):** Table attributes must contain only atomic (indivisible) values, and there must be no repeating groups or columns.
- **Second Normal Form (2NF):** Meets 1NF requirements, and all non-key columns must be fully dependent on the entire Primary Key. This eliminates partial dependencies in tables with composite keys.
- **Third Normal Form (3NF):** Meets 2NF requirements, and no non-key column can be transitively dependent on another non-key column. All fields must depend directly on "the key, the whole key, and nothing but the key".

## Advanced SQL for Enterprise Data Solutions

Structured Query Language (SQL) remains the primary tool for manipulating and extracting data from corporate databases. Enterprise data engineering requires moving beyond simple queries to complex, high-performance data transformations.

### Analytic Window Functions

Window functions perform calculations across a set of table rows that are split into distinct partitions, while maintaining the unique identity of each individual row in the output.




The general syntactic structure is formulated as:

sql

```
SELECT
    employee_id,
    department_id,
    salary,
    AVG(salary) OVER(PARTITION BY department_id) as dept_avg_salary,
    RANK() OVER(PARTITION BY department_id ORDER BY salary DESC) as salary_rank
FROM corporate_payroll;
```

Use code with caution.

### Query Optimisation Strategies

As datasets scale into millions of rows, inefficient SQL operations can stall production systems. Data engineers use several optimisation strategies to maintain performance:

- **Indexing Architecture:** Implementing B-Tree or Hash indexes on columns frequently used in `WHERE` clauses, `JOIN` conditions, and `ORDER BY` operations. Indexes bypass slow sequential table scans by pointing directly to physical storage addresses.
- **Avoiding Correlated Subqueries:** Replacing subqueries that execute once for every single row in the outer query with high-performance `INNER JOIN` or `LEFT JOIN` structures.
- **Common Table Expressions (CTEs):** Using `WITH` clauses to break complex, nested queries into readable, sequential blocks. This improves query planning and code maintainability.

# 2.2 NoSQL Systems, Distributed Storage, and CAP Theorem

## The Limits of Relational Databases

While relational databases excel at maintaining structured transactions, they face significant scaling bottlenecks when confronted with the velocity, volume, and variety of modern big data. RDBMS systems are primarily designed for vertical scaling (upgrading a single server's CPU, RAM, and SSD capacity), which quickly hits hardware performance ceilings and cost barriers.

NoSQL (Not Only SQL) systems are engineered to scale horizontally by spreading workloads across distributed clusters of commodity servers.

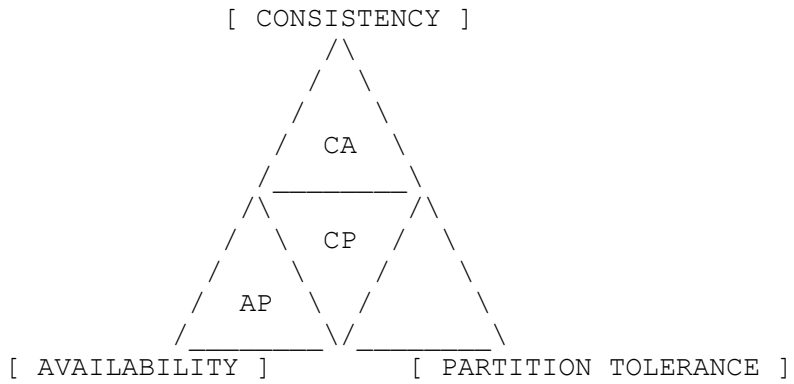
## NoSQL Architectural Typologies

NoSQL systems drop strict relational tables and schemas in favour of flexible data models optimized for specific big data use cases:

| NoSQL Category | Operational Mechanism | Primary Enterprise Use Cases | Example Technologies |
|---|---|---|---|
| **Document Stores** | Stores semi-structured data as JSON or BSON documents. Every document tracks its own internal schema. | Content management, user profiles, e-commerce product catalogs. | MongoDB, CouchDB |
| **Key-Value Databases** | Stores data as simple, high-speed pairs of unique keys and matching data values. | Real-time session caching, user shopping carts, leaderboards. | Redis, Amazon DynamoDB |
| **Wide-Column Stores** | Organizes data into dynamic rows containing an arbitrary number of flexible columns. | High-volume time-series logging, IoT sensor networks, web analytics. | Apache Cassandra, HBase |
| **Graph Databases** | Structures data as discrete nodes (entities) linked by explicit edges (relationships). | Social networks, fraud detection, recommendation engines. | Neo4j, Amazon Neptune |

## Distributed Systems Framework: The CAP Theorem

Formulated by Eric Brewer, the CAP Theorem states that a distributed data system can simultaneously provide only two of the following three core guarantees:




1. **Consistency (C):** Every read operation across the distributed network returns the most recent write or an error. All nodes hold identical data simultaneously.
2. **Availability (A):** Every non-failing node returns a non-error response for every incoming request, though it cannot guarantee the data is the absolute latest write.
3. **Partition Tolerance (P):** The system continues to operate despite an arbitrary number of communication dropouts or message delays between nodes in the network.
   **The Reality of Distributed Networks**

   Because physical hardware networks will inevitably experience packet delays and connectivity drops, **Partition Tolerance (P) is mandatory**. Therefore, a distributed system must choose between two operational trade-offs during a network partition:

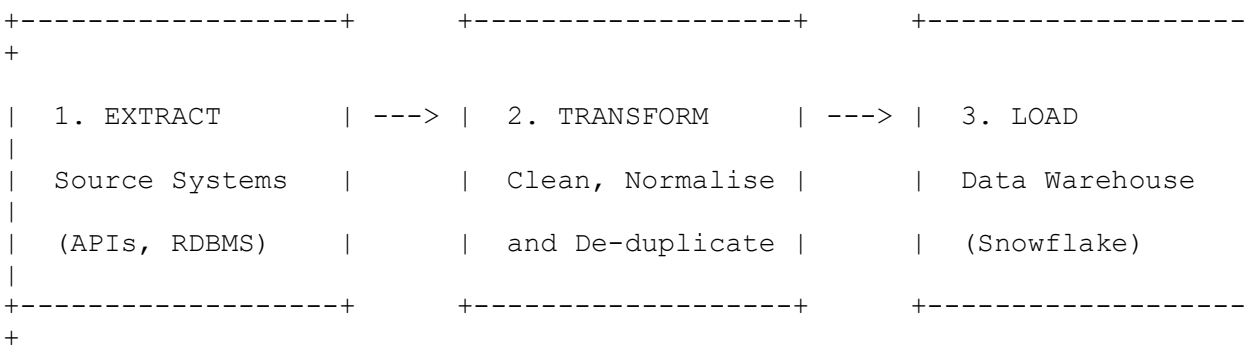
- **CP Systems (Consistency + Partition Tolerance):** The system rejects or stalls incoming requests on isolated nodes to prevent reading out-of-date data. It prioritizes absolute data precision over system availability.
- **AP Systems (Availability + Partition Tolerance):** Nodes remain fully available and accept local reads and writes, but risk serving inconsistent data across the network. The system prioritizes operational uptime, relying on **Eventual Consistency** (e.g., using conflict-free replicated data types) to synchronize data across nodes once the network heals.

---

## 2.3 Big Data Frameworks, ETL Pipelines, and Cloud Warehousing

## The Architecture of Modern Data Pipelines

Enterprise data engineering converts raw data from fragmented, chaotic operational systems into clean, structured datasets that can power business intelligence dashboards and machine learning models. This journey is managed through robust data pipelines using the traditional **ETL (Extract, Transform, Load)** workflow or the modern cloud-native **ELT (Extract, Load, Transform)** workflow.




- **Extract:** Programmatically pulling raw datasets from diverse source systems, including transactional database logs, third-party enterprise APIs, and unstructured cloud storage buckets.
- **Transform:** Running validation checks, stripping out duplicate entries, handling missing values, changing data types, and restructuring schemas to prepare the data for downstream analytics.
- **Load:** Writing the transformed, analytical datasets into an enterprise storage environment.

## Distributed Parallel Processing: Apache Spark

When datasets grow into terabytes or petabytes, processing them on a single machine becomes impossible. Apache Spark resolves this by using an in-memory, distributed data processing framework that parallelizes computation across a cluster of servers.

- **In-Memory Computation:** Unlike legacy frameworks like Hadoop MapReduce that write intermediate states back to slow physical disks, Spark keeps data in RAM across computation steps. This delivers up to a 100x speed increase for iterative processing algorithms.
- **Resilient Distributed Datasets (RDDs):** Spark's core abstraction tool. RDDs represent an immutable, fault-tolerant collection of data elements partitioned across cluster nodes. If a worker node fails, Spark uses the dataset's logical lineage graph to automatically rebuild the missing data partitions on another node.

## Cloud Data Warehouses and Data Lakes

Modern data architecture separates storage and computation to scale efficiently in cloud environments:

- **Data Lakes:** Mass repositories designed to store vast quantities of raw, unstructured, and semi-structured data files in their native formats (e.g., AWS S3, Azure ADLS). Data lakes prioritize low-cost storage, with schemas defined later when the data is queried (Schema-on-Read).
- **Data Warehouses:** Highly structured, centralized cloud data platforms (e.g., Snowflake, Google BigQuery, Amazon Redshift). Warehouses store highly cleaned, curated datasets optimized for fast SQL queries, corporate compliance reporting, and enterprise business intelligence (Schema-on-Write).

# Legal Notice & Terms of Use

## 1. Copyright and Intellectual Property Notice

### *© 2026 Avenbury College. All Rights Reserved.*

All course materials, including but not limited to text, structures, curriculum design, digital assets, diagrams, and methodologies contained within this Free Course Initiative, are the exclusive intellectual property of **Avenbury College** and are protected under the United Kingdom Copyright, Designs and Patents Act 1988 and international intellectual property treaties.

## 2. Free Education Initiative & Access Terms

This course is published strictly as a Free-to-Access Educational Resource for the public. It is designed to support aspiring students, professionals, and functional leaders worldwide by removing financial barriers to high-quality education. There are absolutely no fees, subscriptions, or hidden charges required to access, read, or study this material directly on our official platform. Please note that while access is free, the content remains the proprietary property of the College and is not released under an open-source or public-domain license.

## 3. Permitted and Restricted Usage (Terms of Distribution)

While we encourage the widespread use of this material for personal development and academic growth, strict conditions apply to protect institutional integrity:

- **Personal and Educational Use:** You are permitted to read, study, and reference this text solely for your personal, non-commercial education. Downloading or printing is limited to individual, private study use only.
- **Mandatory Attribution:** If you quote, reference, or summarize parts of this course on external websites, blogs, academic papers, or social platforms, you must provide explicit and visible attribution to **Avenbury College**, alongside a direct hyper-link back to our official course page.
- **Strict Prohibition on Plagiarism and Re-publishing:** You are strictly prohibited from copying, replicating, scraping, or re-publishing this content in its entirety or in substantial parts on any other website, learning management system (LMS), or digital platform.
- **AI Scraping Restriction:** Data mining, robots, or similar data gathering and extraction tools are strictly prohibited from scraping this content for the purpose of training artificial intelligence (AI) models without express written consent.
- **Commercial Restriction:** Selling, white-labeling, or using this text to generate commercial revenue under another brand name is strictly illegal and will result in immediate legal action under UK copyright enforcement acts.
- **Permissions Contact:** For any requests regarding licensing, external reuse, or distribution permissions, please contact our administrative team at: **info@avnc.co.uk**.

## 4. Educational Disclaimer

The information contained in this foundational course is for general educational and informational purposes only. While we strive for academic excellence, **Avenbury College** makes no representations or warranties of any kind regarding specific career outcomes, financial success, or professional performance resulting from the study of these free materials.