Module-4 Data Literacy for Data Science(Optional)

This optional module focuses on understanding data and data literacy and supplements what you learned in the first three modules. As a data scientist, you will need to understand the ecosystem in which your data lives and how it gets manipulated to analyze it. This module introduces you to some of these fundamentals. In lesson one, you explore how data can be generated, stored, and accessed. In lesson two, you dive deeper into data repositories and processes for handling massive data sets.

Learning Objectives

- Compare and contrast structured, semi-structured, and unstructured data characteristics.
- Describe the purpose of metadata.
- Describe where to look for data.
- Summarize the capabilities of data storage tools and technologies.

Understanding Data

Lesson Overview: Understanding Data

In this lesson, "Understanding Data," you'll explore data basics through videos on data comprehension and sources. The reading covers metadata's role, and quizzes reinforce your understanding. The lesson ends with a summary video, ensuring you grasp essential data concepts.

Asset name and type	Description
“Understanding Data:” video	Gain foundational insights into data comprehension. The video will also help you explore data types, characteristics, and their importance in various fields of study.
“Data Sources” video	Explore the diverse origins of data in this video, uncovering where and how data is generated, collected, and utilized in different contexts.
“Working on Varied Data Sources and Types” video	This video equips you with the skills to effectively manage and analyze data from a wide range of sources and in various formats, ensuring adaptability in data handling.
“Metadata” reading	Read an excerpt on “Metadata” and discover the significance of metadata in data analysis through this reading, which encompasses three primary metadata types: technical, process, and business metadata.
Practice quiz	Test your knowledge of metadata concepts based on the previous reading.
“Lesson Summary” video	Recap the key points from this lesson.
Practice quiz	Take a practice quiz to assess your comprehension of data fundamentals.

Understanding Data

📌 **What is Data?**

- **Definition**:
  Data is unorganized information that must be processed to become meaningful.

- **Constituents of Data**:

  - Facts
  - Observations
  - Perceptions
  - Numbers

- Characters
- Symbols
- Images

- **Purpose**:
  These elements can be interpreted to derive meaning.

📊 Categories of Data by Structure

Data can be categorized into three types based on its structure:

---

1. Structured Data

- **Definition**:
  Data with a well-defined structure or format that follows a specific data model.
- **Storage Format**:
  - Organized in schemas like relational databases
  - Represented in tabular form (rows and columns)
- **Characteristics**:
  - Objective facts and numbers
  - Easily collected, exported, stored, and organized
  - Suitable for standard data analysis tools
- **Examples/Sources**:
  - SQL Databases
  - Online Transaction Processing (OLTP) Systems
  - Spreadsheets (e.g., Excel, Google Sheets)
  - Online forms
  - Sensors (e.g., GPS, RFID tags)
  - Network & Web server logs
- **Storage Systems**:
  - Relational or SQL databases
- **Analysis**:

- Compatible with standard data analysis tools and methods

---

2. Semi-Structured Data

- **Definition**:
  Data that has some organizational properties but lacks a strict schema or format.
- **Structure**:
  - Not in traditional rows/columns
  - Contains **tags**, **elements**, or **metadata** that provide hierarchy and grouping
- **Examples/Sources**:
  - Emails
  - XML and other markup languages
  - JSON
  - Binary executables
  - TCP/IP packets
  - Zipped files
  - Data from multiple integrated sources
- **Formats Used**:
  - XML and JSON are common for storing and exchanging this type of data

---

3. Unstructured Data

- **Definition**:
  Data without a recognizable structure or format
- **Characteristics**:
  - Cannot be stored in rows/columns
  - Lacks a predefined format, sequence, or rules
  - Can come from heterogeneous sources
  - Useful in business intelligence and analytics
- **Examples/Sources**:

    - Web pages
    - Social media feeds
    - Images (e.g., JPEG, PNG, GIF)
    - Videos and audio files
    - PDFs and other documents
    - Presentations (e.g., PowerPoint)
    - Media logs
    - Survey responses
- **Storage Options**:
    - Files (e.g., Word documents)
    - NoSQL databases (support analysis tools for such data)

📌 Summary

This video introduces **data** as unprocessed information made meaningful through analysis. It explains three key types of data based on their **structure**: **Structured, Semi-structured, and Unstructured**. Structured data fits cleanly into databases and is ideal for traditional analysis. Semi-structured data includes elements like tags or metadata but doesn't conform to a strict schema. Unstructured data lacks a consistent format and includes multimedia content, documents, and social media. The video also outlines sources and storage mechanisms for each data type, forming a foundational understanding for further study in data organization and analysis.

📋 Table: What We Learnt in the Video

Topic/Section	What We Learnt
What is Data?	Data is unorganized information like facts, numbers, or images, processed for meaning.
Structured Data	Has a fixed schema, can be stored in databases, analyzed easily with tools.
Sources of Structured Data	SQL databases, spreadsheets, online forms, GPS, RFID, logs.
Semi-Structured Data	It contains tags/metadata but lacks a rigid format; it is stored in XML, JSON, etc.
Sources of Semi-Structured Data	Emails, XML, zipped files, TCP/IP packets, executable files.
Unstructured Data	No predefined format; includes media files, documents, web pages.
Storage for Unstructured Data	Stored in documents or NoSQL databases for manual or tool-based analysis.

Data Sources

1. Introduction: Dynamic and Diverse Data Sources

- Modern data sources are more varied and changing faster than ever.
- The video explores commonly used data sources in analytics:
  - Relational Databases
  - Flat Files and XML Datasets
  - APIs and Web Services
  - Web Scraping
  - Data Streams and Feeds

---

2. Relational Databases

- Used by internal organizational systems to manage operations like:
  - Business transactions
  - HR activities
  - Workflow management
- Examples:
  - **SQL Server**, **Oracle**, **MySQL**, **IBM DB2**
- Store data in **structured tabular format** (tables with rows and columns).
- Can be used in analytics for:
  - **Sales analysis** across regions
  - **Sales projections** using CRM data

---

3. External Datasets

- Public or commercial datasets outside the organization.
- Examples:
  - Government demographic/economic data
  - Purchased datasets: **POS data**, **financial data**, **weather data**
- Uses:

    - Strategic planning
    - Demand prediction
    - Marketing decision-making

---

4. Flat Files and Spreadsheets

Flat Files

- Plain text, tabular data with delimiters (comma, tab, semicolon).
- Each line = one record
- Maps to **single table** (unlike relational DBs with multiple tables)
- **CSV** is the most common format

Spreadsheets

- A type of flat file with added functionality:
    - Multiple worksheets (can map to different tables)
    - Stored in proprietary formats (.XLS, .XLSX)
    - Contains formatting, formulas, metadata
- Examples:
    - Microsoft Excel
    - Google Sheets
    - Apple Numbers
    - LibreOffice

---

5. XML Datasets

- Markup-based format: data enclosed in tags
- Suitable for **hierarchical** or **complex structures**
- Used in:
    - Online surveys
    - Bank statements

- Other semi-structured data

---

6. APIs and Web Services

- Allow programmatic access to external/internal data
- Respond to **web/network requests** with:
  - Text, XML, HTML, JSON, or media
- Use Cases:
  - **Social Media APIs**: Twitter, Facebook for sentiment analysis
  - **Stock Market APIs**: For prices, EPS, historical data
  - **Data Validation APIs**: e.g., postal code to city mapping
  - **Database APIs**: Fetching DB data internally/externally

---

7. Web Scraping

- Extracts data from unstructured web sources
- Also known as:
  - Screen scraping
  - Web harvesting
  - Web data extraction
- Scraping Targets:
  - Text, images, videos, product listings, contact info
- Applications:
  - Price comparison
  - Lead generation
  - Forum data extraction
  - Dataset generation for ML
- Tools:
  - **BeautifulSoup**, **Scrapy**, **Pandas**, **Selenium**

---

8. Data Streams and Feeds

- Real-time continuous data flows from:
  - IoT devices
  - Instruments
  - GPS, websites, apps, social media
- Data often includes **timestamps** and **geo-tags**

Use Cases:

- **Stock tickers** for real-time trading
- **Retail transactions** for demand prediction
- **Surveillance feeds** for threat detection
- **Social media** for sentiment analysis
- **Sensor feeds** for industrial/farming machinery
- **Web clickstreams** for performance analysis
- **Flight events** for airline scheduling

Tools:

- **Apache Kafka**
- **Apache Spark Streaming**
- **Apache Storm**

---

9. RSS Feeds

- **Really Simple Syndication**
- Used for:
  - Capturing frequent updates from websites and forums
- Uses a **feed reader** to convert RSS into readable stream format

---

Summary

This video covers the diverse and evolving landscape of data sources used in data analytics. It categorizes data sources into internal (like relational databases) and external (like flat files, APIs, and streams), explaining their formats, examples, and use cases. Key methods include accessing structured data from relational databases, semi-structured data from XML and spreadsheets, unstructured data via web scraping, and real-time insights through data streams and RSS feeds. The video also introduces tools and technologies that facilitate data extraction, processing, and analysis across all these formats.

Table: What We Learnt in the Video

Topic/Section	What We Learnt
Relational Databases	Store structured internal business data (e.g., MySQL, Oracle); used for analysis.
External Datasets	Include public and paid sources like government or POS data for analytics.
Flat Files	Plain text files like CSV store tabular data using delimiters.
Spreadsheets	Advanced flat files (.XLS, .XLSX) with support for formulas and multiple sheets.
XML Datasets	Marked-up data for hierarchical/semi-structured content; used in surveys, banks.
APIs & Web Services	Enable real-time, programmatic data access (e.g., Twitter API, stock market API).
Web Scraping	Extracts data from websites using tools like BeautifulSoup and Selenium.
Data Streams	Continuous flows from sensors, GPS, apps, processed using Kafka, Spark, Storm.
RSS Feeds	Capture updated content from forums/news via feed readers.

Viewpoints: Working with Varied Data Sources and Types

🧑‍💼 Experiences of Data Professionals with Diverse Data Sources

- **Variety of Data Sources**
    - Data can originate from many types and systems.
    - Flexibility is essential due to differences in source structure and requirements.

🗄️ Working with Relational Databases (RDBMS)

- **Personal Preference**
    - Some professionals favor relational databases.
    - Frequent use of SQL for:
        - Moving data between locations.
        - Structuring and transforming data.

- Managing security aspects.

- **Challenges**
  - Moving data even within relational databases can be tricky.
  - Vendor differences (e.g., Oracle vs. SQL Server) complicate transfers.
  - Versioning issues:
    - Features may vary or be unsupported across versions.
    - Some required functions may only exist in newer versions.
- **Performance Considerations**
  - One-time data movement is manageable if under a terabyte.
  - Continuous and performant data transfer is more complex.

🆚 Relational vs. Non-Relational Databases

- **Limitations of Relational Databases**
  - Not ideal for:
    - Unstructured data (logs, documents, XML, JSON).
    - Write-heavy workloads (e.g., IoT, social media data).
  - Performance bottlenecks due to B-tree structures in write-intensive scenarios.
- **Rise of NoSQL**
  - Triggered by white papers like Google's BigTable (2006).
  - Inspired databases like Cassandra and HBase.
  - Better suited for:
    - Heavy write workloads.
    - Scalable distributed architectures.

🛠️ Formats and Tools a Data Engineer Must Work With

- **Common Data Formats**
  - Standard: CSV, JSON, XML.
  - Proprietary formats may also appear in projects.

- **Types of Data**
    - Data at rest.
    - Streaming data (real-time).
    - Data in motion (in-transit processing).
- **Skill Requirements**
    - Expect to learn new tools and formats continuously.
    - Adaptability is key to handling project-specific requirements.

📄 Comparison of Data Formats

Data Format	Characteristics	Challenges	Usage Context
Log Data	Unstructured	Custom parsing often needed	Monitoring, debugging
XML	Hierarchical, tag-based	Memory-intensive, verbose	Used with SOAP protocols
JSON	Lightweight key-value format	Easier than XML but still requires parsing	RESTful APIs
Apache Avro	Compact binary format	Efficient but less human-readable	Big data pipelines

---

🔄 Real-World Scenario: Data Migration Challenge

- **Scenario**
    - Migrating data from IBM Db2 to SQL Server.
- **Issues Encountered**
    - Import/export expectations differ by database.
    - Special character handling:
        - Data contained many delimiters (commas, special symbols).
        - Custom delimiters were needed per table.
        - Some characters couldn’t be used due to incompatibility (e.g., Bell character).

Summary

This video explores the real-world complexities of working with various data types, formats, and sources through insights shared by data professionals. It underscores the flexibility and adaptability needed when handling structured and unstructured data, moving between database systems, or selecting the right data format. It particularly highlights the challenges of relational databases in modern data scenarios and the emergence of NoSQL databases like Cassandra and HBase as a solution to limitations in scalability and performance. Key issues such as format-specific challenges, system versioning, and the intricacies of data migration are also discussed.

📘 Table: What We Learnt in the Video

Topic/Section	What We Learnt
Working with Data Sources	Requires flexibility; sources vary in format, structure, and expectations.
Relational Databases	Suitable for structured data and security, but faces versioning and performance issues.
NoSQL Evolution	NoSQL emerged to address the high-write and unstructured data limitations of RDBMS.
Common Data Formats	Each format (JSON, XML, etc.) has specific challenges and optimal use cases.
Data Migration	Different systems require different approaches; delimiters can cause complications.
Real-World Engineering	Engineers must constantly adapt and learn to handle new data challenges.

Metadata and Metadata Management

Objectives

After completing this reading, you will be able to:

- Define what metadata is
- Describe what metadata management is
- Explain the importance of metadata management
- List popular tools for metadata management

What is metadata?

Metadata is data that provides information about other data.

This is a very broad definintion. Here we will consider the concept of metadata within the context of databases, data warehousing, business intelligence systems, and all kinds of data repositories and platforms.

We'll consider the following three main types of metadata:

- Technical metadata
- Process metadata, and
- Business metadata

Technical metadata

Technical metadata defines the data structures in data repositories or platforms, primarily from a technical perspective.

For example, technical metadata in a data warehouse includes assets such as:

- Tables that record information about the tables stored in a database, like:
  - Each table's name
  - The number of columns and rows each table has
- A data catalogue, which is an inventory of tables that contain information, like:
  - The name of each database in the enteprise data warehouse
  - The name of each column present in each database
  - The names of every table in which each column is contained
  - The type of data that each column contains

The technical metadata for relational databases is typically stored in specialized tables in the System Catalogue database.

Process metadata

Process metadata describes the processes that operate behind business systems such as data warehouses, accounting systems, or customer relationship management tools.

Many important enterprise systems collect and process data from various sources. Such critical systems need to be monitored for failures and any performance anomalies that arise. Process metadata for such sytems includes tracking things like:

- process start and end times
- disk usage

- where data was moved from and to, and
- how many users access the system at any given time

This sort of data is invaluable for troubleshooting and optimizing workflows and ad hoc queries.

Business metadata

Users who want to explore and analyze data within and outside the enterprise are typically interested in *data discovery*. They need to be able to find data which is meaningful and valuable to them and know where that data can be accessed from. These business-minded users are thus interested in business metadata, which is information about the data described in readily interpretable ways, such as:

- how the data is acquired
- what the data is measuring or describing
- the connection between the data and other data sources

Business metadata also serves as documentation for the entire data warehouse system.

Managing metadata

Managing metadata includes developing and administering policies and processes to ensure information can be accessed and integrated from various sources and appropriately shared across the entire enterprise.

Creation of a reliable, user-friendly data catalog is a primary objective of a metadata management model. The data catalog is a core component of a modern metadata management system, serving as the main asset around which metadata management is administered. It serves as the basis by which companies can inventory and efficiently organize their data systems. A modern metadata managment model will include a web-based user interface that enables engineers and business users to easily search for and find information on key attributes such as CustomerName or ProductType. This kind of model is central to any Data Governance initiative.

Why is metadata management important?

Good metadata management has many valuable benefits. Having access to a well implemented data catalog greatly enhances data discovery, repeatability, governance, and can also facilitate access to data.

Well managed metadata helps you to understand both the business context associated with the enterprise data and the data lineage, which helps to improve data governance. Data lineage provides information about the origin of the data and how it gets transformed and moved, and thus it facilitates tracing of data errors back to their root cause. Data governance is a data management concept concerning the capability that enables an organization to ensure that high data quality exists throughout the complete lifecycle of the data, and data controls are implemented that support business objectives.

The key focus areas of data governance include availability, usability, consistency, data integrity and data security and includes establishing processes to ensure effective data management throughout the enterprise such as accountability for the adverse effects of poor data quality and ensuring that the data which an enterprise has can be used by the entire organization.

Popular tools for metadata management

Popular metadata management tools include:

- IBM InfoSphere Information Server
- CA Erwin Data Modeller
- Oracle Warehouse Builder

- SAS Data Integration Server
- Talend Data Fabric
- Alation Data Catalog
- SAP Information Steward
- Microsoft Azure Data Catalog
- IBM Watson Knowledge Catalog
- Oracle Enterprise Metadata Management (OEMM)
- Adaptive Metadata Manager
- Unifi Data Catalog
- data.world
- Informatica Enterprise Data Catalog

Summary

In this reading, you learned that:

- Metadata is data that provides information about other data, and includes three main types: technical, process, and business metadata
- The technical metadata for relational databases is typically stored in specialized tables in the database called the system catalog
- A primary objective of business metadata management modelling is the creation and maintenance of a reliable, user-friendly data catalog
- Having access to a well-implemented data catalog greatly enhances data discovery, repeatability, and governance and facilitates data access.
- Metadata management tools from IBM include InfoSphere Information Server and Watson Knowledge Catalogue.

Lesson Summary: Understanding Data

🔍 Understanding Data

- **Data is the foundation of data science**
    - To succeed as a data scientist, understanding different forms of data is essential.
    - Data must be interpreted to derive value and insight.

🧱 Types of Data

1. Structured Data

- Has a **well-defined structure** or conforms to a **specified data model**
- Typically stored in **schemas**, e.g., **relational databases**
- Organized into **tables with rows and columns**
- Schema defines **relationships between tables**

2. Semi-Structured Data

- Has **organizational properties** but **lacks a rigid schema**
- Cannot be stored in traditional **row-column format**
- Uses **tags, elements, or metadata** to form a hierarchy

- Examples: **XML, JSON**
- **Metadata** provides essential information:
    - **Technical** (e.g., format, structure)
    - **Process** (e.g., how data is collected)
    - **Business** (e.g., purpose of data)

Metadata Management

- Managed through a **data catalog**
- Enhances:
    - **Data discovery**
    - **Repeatability**
    - **Governance**
    - **Access and sharing across the organization**

3. Unstructured Data

- **Heterogeneous** and comes from **diverse sources**
- No predefined structure
- Examples: **social media posts, videos, emails, sensor data**
- Requires **Artificial Intelligence (AI)** for effective analysis
- Used for:
    - **Business Intelligence**
    - **Analytics applications**

📤 Data Sources & Formats

- Can be generated **automatically or manually**
- Older data might exist in **analog formats** (e.g., paper), requiring **conversion to digital**

Internal Data Sources

- **Applications used within organizations** for day-to-day operations

External Data Sources

- **Public or private datasets** available for analysis

- **Proprietary datasets** can be purchased

Common Formats

- **Flat files** like:
    - CSV
    - Spreadsheets
- **XML**: Stores data structure info (used historically)
- **JSON**:
    - Modern alternative
    - Human and machine readable
    - Schema-less, adaptable over time

🌐 **Accessing Data via APIs**

- Modern cloud-based applications expose data through **APIs (Application Programming Interfaces)**
- Common type: **RESTful APIs**
- Examples:
    - **Twitter**, **Facebook** APIs allow for **social media data access**
    - Used in **opinion mining** and **sentiment analysis**

👷 **Roles in Data Management**

- **Data Engineers**:
    - Handle **gathering and managing** data
- **Data Scientists**:
    - Must transfer and work with data
    - Often deal with **terabytes of data**
    - Need flexibility in handling data

Examples of Large Data Sources

- **IoT applications**

- **Sensor networks**
- **Social media feeds**

🧠 **The Data Scientist's Role**

- Develop an **intimate understanding** of data
- Navigate the **modern data ecosystem**:
    - Organizing
    - Storing
    - Manipulating
    - Retrieving data

✨ **Summary**

This lesson introduces the essential knowledge a data scientist needs to understand various forms of data—structured, semi-structured, and unstructured. It explains how data is generated, stored, accessed, and used in analytics. You learn about metadata, data catalogs, APIs (especially RESTful APIs), and the formats in which data is shared (CSV, XML, JSON). The video also highlights the role of data engineers and scientists in managing large-scale data, sourced from modern systems like IoT and social media. Ultimately, the lesson emphasizes that understanding the data ecosystem is a fundamental step toward effective data analysis.

📋 **Table: What We Learnt in the Video**

Topic/Section	What We Learnt
Types of Data	Structured, semi-structured, and unstructured data, with examples and characteristics
Structured Data	Stored in tables using a defined schema
Semi-Structured Data	Uses metadata and tags; lacks a rigid schema
Unstructured Data	Diverse, large, and AI-required for analysis
Metadata	Categorised as technical, process, and business metadata, managed via data catalogue
Data Catalog	Improves data governance, sharing, and discovery
Data Formats	CSV, XML, JSON; JSON is flexible and widely used
APIs and RESTful APIs	Allow access to cloud-based and web data
External Data Sources	Public, private, and proprietary datasets
Role of Data Engineers	Data gathering and management

Role of Data Scientists	Deep understanding and handling of large-scale, evolving data
Data Sources	IoT, sensors, social media, internal applications

Data Literacy

Lesson Overview: Data Literacy

In the "Data Literacy" lesson, you'll dive into the fundamental aspects of data through a series of instructional videos. You'll gain insights into data collection, organization, and the distinctions between Relational Database Management Systems (RDBMS) and NoSQL databases. Additionally, you'll navigate the world of Data Marts, Data Lakes, ETL processes, and Data Pipelines. The lesson ends with a summary video, ensuring you grasp essential data concepts.

Asset name and type	Description
“Data Collection and Organization” video	Explore the fundamentals of collecting and organizing data in this instructional video.
“Relational Database Management System” video	Gain insights into the workings of Relational Database Management Systems (RDBMS) through this informative video.
“NoSQL” video	Delve into NoSQL databases and their unique characteristics in this video.
“Data Marts, Data Lakes, ETL, and Data Pipelines” video	Learn about Data Marts, Data Lakes, ETL processes, and Data Pipelines in this comprehensive video.
“Considerations for Choice of Data Repository” video	Discover key factors to consider when selecting a data repository in this video.
“Data Integration Platforms” video	Explore the role and significance of Data Integration Platforms through this enlightening video.
Practice quiz	Test your knowledge of Data Integration Platforms with this practice quiz.
“Lesson Summary” video	Recap the key points from this lesson.
Practice quiz	Take a practice quiz to assess your comprehension of data literacy.
Glossary	Access a comprehensive glossary that defines and clarifies key data literacy terms relevant to the field of data science.
“Summary” reading	Summarize the essential concepts and insights enhancing your understanding of data literacy within the context of data science.

Data Collection and Organization

📁 **Introduction to Data Repositories**

- **Definition**:
  A **data repository** is a general term for collected, organized, and isolated data used for:
  - Business operations
  - Reporting
  - Data analysis
- **Scale & Infrastructure**:
  - Can be small or large-scale
  - May involve one or multiple databases
- **Types to Be Covered**:
  - Databases
  - Data Warehouses
  - Big Data Stores

🗃️ **Databases**

- **Definition**:
  A **database** is a structured collection of data designed for:
  - Input
  - Storage
  - Search and retrieval
  - Modification
- **Database Management System (DBMS)**:
  - Software that manages and maintains the database
  - Enables data storage, retrieval, and manipulation through **querying**
  - Example: Find customers inactive for 6+ months via a query
- **Terminology Note**:
  - "Database" and "DBMS" are often used interchangeably but are technically different

📚 **Types of Databases**

- **Factors Influencing Choice**:

- Data type and structure
- Query mechanisms
- Latency requirements
- Transaction speed
- Intended data usage

1. Relational Databases (RDBMS)

- Organize data in **tables** with **rows and columns**
- Rely on **structured schema**
- Ideal for:
  - Complex queries
  - Multi-table relationships
  - Large volumes of structured data
- Use **SQL (Structured Query Language)**

2. Non-Relational Databases (NoSQL)

- Known as **Not Only SQL**
- Schema-less or flexible format
- Built for:
  - High-speed operations
  - Handling **unstructured or semi-structured data**
  - Scaling with large datasets
- Popular due to:
  - Cloud computing
  - Internet of Things (IoT)
  - Social media data

🏢 **Data Warehouses**

- **Definition**:
  A central repository that:
  - Collects data from multiple sources

- Consolidates and transforms it via **ETL (Extract, Transform, Load)** for analytics

- **ETL Process**:
    - **Extract**: Pull data from various sources
    - **Transform**: Clean and format data
    - **Load**: Move it into a unified database
- **Purpose**:
    - Supports **analytics and business intelligence**
    - Ensures clean, consolidated, and ready-to-analyze data
- **Additional Concepts** (to be covered later):
    - **Data Marts**
    - **Data Lakes**
- **Relational vs. Non-Relational Warehousing**:
    - Traditionally used **relational** databases
    - Increasing use of **NoSQL** due to evolving data needs

🧠 **Big Data Stores**

- **Definition**:
  Data repositories designed to:
    - Store and process **very large-scale datasets**
    - Leverage **distributed computing and storage**
- **Purpose**:
    - Handle data at scale with flexibility
    - Critical in big data environments

🎯 **Importance of Data Repositories**

- Isolate and organize data
- Enable **credible and efficient** reporting and analytics
- Function as **data archives**

Summary
This video introduces the concept of **data repositories**, emphasizing their role in organizing and isolating data for business use, analytics, and storage. It outlines the main types: **databases** (both relational and non-relational), **data warehouses**, and **big data stores**. The video explains how databases are structured for storing and querying data, how relational databases use SQL and structured schemas, and how NoSQL databases address modern data demands. It also covers how **data warehouses** use the **ETL process** to consolidate data from various sources and how **big data stores** manage massive datasets through distributed infrastructures. The takeaway is a foundational understanding of where enterprise data resides and how it is structured for use.

Table: What We Learnt in the Video

Topic/Section	What We Learnt
Data Repository	A structured storage for data, used in operations, analytics, and archiving
Database	Stores, retrieves, and modifies data; managed by DBMS; used for querying
Relational Database	Uses structured tables and schemas; SQL-based; suited for complex queries
Non-Relational Database	Schema-less; handles unstructured data; scales well; ideal for big data
Database Management	DBMS allows data input, querying, and manipulation
ETL Process	Extracts, transforms, and loads data into a data warehouse
Data Warehouse	Central repository for consolidated, cleaned data for BI and analytics
Big Data Stores	Distributed systems handling large-scale, high-volume data
Importance of Repositories	Support clean data organization, reliable analysis, and long-term storage

Relational Database Management System

📘 Introduction to Relational Databases

- A **relational database** is a collection of data organized into **tables**.
- Each **table** has:
  - **Rows** (records)
  - **Columns** (attributes)
- Tables are linked (or related) using **common fields** (e.g., Customer ID).

🧾 Example: Customer and Transaction Tables

- **Customer Table**:
  - Attributes: Company ID, Company Name, Company Address, Company Primary Phone
  - Each row = a customer record
- **Transaction Table**:
  - Attributes: Transaction Date, Customer ID, Transaction Amount, Payment Method
- Relationship:
  - Linked via Customer ID

- Allows creation of **consolidated reports** (e.g., customer statements)

🔗 Relating Tables

- Combining tables through relationships allows:
    - Retrieval of **new tables/views**
    - Discovery of relationships and patterns
    - Better **decision-making insights**

💬 SQL – Structured Query Language

- Primary language used for:
    - Querying
    - Inserting
    - Updating
    - Deleting data
- Efficient for handling **large volumes** of data

🆚 Relational Databases vs. Spreadsheets

Aspect	Spreadsheets	Relational Databases
Structure	Flat files (rows & columns)	Tables with defined relationships
Scalability	Limited rows/columns	Optimized for large-scale data
Redundancy	Often duplicated data	Minimized via normalization
Data Integrity	Difficult to enforce	Strict data types & constraints
Performance	Slower with large data	High-speed queries
Security & Governance	Basic permissions	Controlled access & policy enforcement

---

🛡️ Advantages of Relational Databases

- 🔁 **Flexibility**:

- Easily add columns/tables, rename relationships during operations
- 📉 **Reduced Redundancy**:
  - Customer data stored once, referenced in transactions
- 💾 **Ease of Backup & Disaster Recovery**:
  - Export/import options
  - Cloud-based systems offer **continuous mirroring**
- ✅ **ACID Compliance**:
  - **A**tomicity: All parts of a transaction complete or none do
  - **C**onsistency: Data remains valid before & after transactions
  - **I**solation: Concurrent transactions don't interfere
  - **D**urability: Completed transactions survive system failure

---

🌐 **Popular Relational Database Systems**

- **Commercial**:
  - IBM DB2
  - Microsoft SQL Server
  - Oracle Database
- **Open-source**:
  - MySQL
  - PostgreSQL
- **Cloud-based (Database-as-a-Service)**:
  - Amazon RDS
  - Google Cloud SQL
  - IBM DB2 on Cloud
  - Oracle Cloud
  - SQL Azure

💼 Use Cases for Relational Databases

- **Online Transaction Processing (OLTP)**:
  - High-rate, transactional tasks
  - Fast response times
  - Multiple concurrent users
- **Data Warehousing / OLAP**:
  - Analyze historical data for **business intelligence**
- **IoT Solutions**:
  - Fast collection/processing from edge devices
  - Lightweight databases for high-speed operations

⚠️ Limitations of RDBMS

- 🧩 Not suitable for:
  - Semi-structured/unstructured data (e.g., multimedia, JSON)
  - Extensive analytics on non-tabular data
- 🔁 Migration Complexity:
  - Schema & data types must be identical between systems
- 📏 Data Length Limits:
  - Fields have a maximum size—oversized entries may be rejected

---

📝 Summary

Relational databases organize data into structured tables linked by common fields, enabling efficient querying and reporting. By using SQL, they support complex operations on large datasets while maintaining data integrity through ACID compliance. They are ideal for structured data, especially in high-volume applications like OLTP, data warehousing, and IoT. While they face limitations with unstructured data and migrations, their flexibility, reliability, and performance make them a cornerstone of modern data systems.

📊 Table: What We Learnt in the Video

Topic/Section	What We Learnt
Structure of Relational DBs	Data organized in tables with rows (records) and columns (attributes)

Table Relationships	Tables can be linked using common fields like Customer ID
SQL	Primary language used for querying and managing relational databases
Comparison to Spreadsheets	RDBMS offers scalability, structure, integrity, and minimized redundancy
Advantages of RDBMS	Flexibility, backup/recovery, ACID compliance, performance
Popular RDBMS Systems	Includes IBM DB2, MySQL, PostgreSQL, Oracle, SQL Server, and cloud options
Use Cases	OLTP, OLAP, and IoT applications
Limitations	Struggles with unstructured data, migration complexities, and field limits

NoSQL

🔍 Introduction to NoSQL

- **Definition**:
  NoSQL stands for **"Not Only SQL"** — a non-relational database design that supports flexible schemas for storing and retrieving data.
- **Purpose**:
  - Suited for modern needs like **cloud computing**, **big data**, and **high-volume** web/mobile apps.
  - Chosen for **scalability**, **performance**, and **ease of use**.
- **Clarification**:
  - "No" in NoSQL = "Not Only" SQL, *not* the literal "No".
- **Key Characteristics**:
  - Non-relational.
  - Schema-less (can store structured, semi-structured, unstructured data).
  - Typically doesn't use SQL; some support SQL-like querying.

🗂️ Types of NoSQL Databases

1. 🔑 Key-Value Stores

- **Structure**:
  - Data is stored in **key-value pairs**.
  - Keys = Unique identifiers.

- Values = Can be strings, numbers, or complex objects like JSON.

- **Use Cases**:
  - User session storage
  - User preferences
  - Real-time recommendations
  - In-memory caching
- **Limitations**:
  - Poor fit for complex querying or relationships between data.
  - Not ideal for multi-key querying.
- **Examples**:
  - **Redis**
  - **Memcached**
  - **DynamoDB**

---

2. 📄 Document-Based Databases

- **Structure**:
  - Stores each record and associated data in a **single document** (e.g., JSON, BSON).
  - Supports **flexible indexing** and **ad hoc queries**.
- **Use Cases**:
  - eCommerce platforms
  - Medical records
  - CRM systems
  - Analytics dashboards
- **Limitations**:
  - Not ideal for complex search queries or multi-operation transactions.
- **Examples**:

- **MongoDB**
- **DocumentDB**
- **CouchDB**
- **Cloudant**

---

3. 📊 Column-Based Databases

- **Structure**:
  - Data stored in **columns** (not rows).
  - **Column families** group related columns often accessed together.
- **Example Use**:
  - Grouping: Customer profile info (name, contact) vs. purchase history.
- **Advantages**:
  - High-speed access for heavy write loads.
  - Ideal for:
    - Time-series data
    - IoT
    - Weather data
- **Limitations**:
  - Not suitable for complex or frequently changing query patterns.
- **Examples**:
  - **Cassandra**
  - **HBase**

---

4. 🌐 Graph-Based Databases

- **Structure**:
  - Data stored as **nodes (entities)** and **edges (relationships)**.
  - Ideal for connected data.

- **Use Cases**:
  - Social networks
  - Fraud detection
  - Product recommendations
  - Access control
  - Network analysis
- **Limitations**:
  - Not optimized for high-volume transactions or large-scale analytics.
- **Examples**:
  - **Neo4j**
  - **CosmosDB**

✅ Advantages of NoSQL

- Handles **structured, semi-structured, and unstructured** data.
- Supports **distributed systems** across data centers — scalable in the cloud.
- **Cost-effective** scale-out on commodity hardware.
- **Simpler design** leads to:
  - Better availability
  - Higher agility
  - Faster iteration

🔄 Relational (RDBMS) vs. Non-Relational (NoSQL)

Feature	Relational Databases (RDBMS)	NoSQL Databases
Schema	Rigid, pre-defined	Flexible or schema-less
Cost	High (needs expensive infrastructure)	Lower (runs on commodity hardware)
Data Types	Structured only	Structured, semi-structured, unstructured
ACID Compliance	Fully supported	Often not supported or partially supported

Scalability	Vertical scaling (scale-up)	Horizontal scaling (scale-out)
Maturity	Very mature and well-documented	Newer, evolving rapidly
Best Use Cases	Traditional business apps, banking, inventory	Big data apps, IoT, mobile/web apps

---

📝 Summary

NoSQL databases are a modern alternative to traditional relational databases, designed to meet big data needs, cloud-native applications, and flexible data storage. They allow storage of diverse data types without a fixed schema and provide high scalability and performance. The main types include key-value, document, column, and graph-based databases — each suited for specific use cases. Although NoSQL lacks features like full ACID compliance, it offers cost-effective scaling and agility, making it an essential tool for today's data-heavy applications.

📋 Table: What We Learnt in the Video

Topic/Section	What We Learnt
NoSQL Introduction	A flexible, non-relational database suited for modern, large-scale applications
Key-Value Stores	Stores data as key-value pairs; good for caching, preferences, but limited querying
Document-Based Databases	Store data in documents; good for analytics and CRMs, not great for complex queries
Column-Based Databases	Store data in columns; best for time-series, fast writes; limited query flexibility
Graph-Based Databases	Uses nodes and relationships; ideal for social graphs, not for heavy analytics
NoSQL Advantages	Supports unstructured data, scalable, cost-effective, agile
RDBMS vs NoSQL Comparison	Relational is mature and structured; NoSQL is flexible, scalable, and modern

Data Marts, Data Lakes, ETL, and Data Pipelines

📦 Data Warehouse

- **Definition**: A centralized repository designed for reporting and analysis.
- **Characteristics**:
    - Multi-purpose storage.
    - Stores **structured data** that is **modeled** and **analysis-ready**.
    - Contains **cleansed, conformed, categorized** data.
    - Holds **both current and historical** data.
- **Use Case**:
    - Ideal for organizations with **massive operational data** needing **reporting** and **analytics**.
- **Benefits**:
    - Acts as a **single source of truth**.
    - Enables **operational and performance analytics**.

🧩 Data Mart

- **Definition**: A subsection of a data warehouse tailored for specific business units or purposes.
- **Characteristics**:
    - Business-specific repository.
    - Extracts **relevant data** for specific **stakeholders**.
    - Offers **isolated security** and **performance**.
- **Examples**:
    - Used by **sales** or **finance** teams for **quarterly reports** and **projections**.
- **Main Role**:
    - Facilitates **business-specific reporting and analytics**.

🌊 Data Lake

- **Definition**: A large repository for storing **raw data** in its native format.

- **Characteristics**:
  - Handles **structured**, **semi-structured**, and **unstructured** data.
  - Stores data with **metadata tagging**.
  - Retains **all source data** without exclusions.
  - Can be used as a **staging area** for data warehouses.
- **Use Case**:
  - Useful when dealing with **large volumes of data** with **undefined use cases**.
- **Main Role**:
  - Supports **predictive** and **advanced analytics**.

⚙️ ETL Process (Extract, Transform, Load)

🔍 Extract

- **Purpose**: Collect data from various sources.
- **Methods**:
  - **Batch Processing**: Moves large chunks at scheduled intervals.
    - *Tools*: Stitch, Blendo
  - **Stream Processing**: Real-time data movement and transformation.
    - *Tools*: Apache Samza, Apache Storm, Apache Kafka

🔧 Transform

- **Purpose**: Clean and convert raw data into usable formats.
- **Activities**:
  - Standardize **date formats**, **units**.
  - Remove **duplicates**, **irrelevant data**.
  - Enrich data (e.g., split names).
  - Create **relationships** across data.
  - Apply **business rules** and **data validation**.

📤 Load

- **Purpose**: Move transformed data to the target system.
- **Types**:
  - **Initial Loading**: Full data population.
  - **Incremental Loading**: Periodic updates and changes.
  - **Full Refresh**: Deletes existing data and reloads.
- **Important Checks**:
  - **Load verification**: Check for missing/null values, performance, and failures.
  - **Recovery mechanisms** must be in place.

🔁 Data Pipelines

- **Definition**: A broader term that includes all processes to move data from source to destination (ETL is a subset).
- **Capabilities**:
  - Can be configured for:
    - **Batch processing**
    - **Streaming data**
    - **Hybrid models**
- **Benefits**:
  - Handles both **long-running batch queries** and **smaller interactive queries**.
  - Suitable for **real-time updates**, e.g., **sensor data**.
- **Destinations**:
  - Typically a **data lake**, but can also be other applications or **visualization tools**.
- **Popular Tools**:
  - Apache Beam
  - Google DataFlow

📃 Summary

This content provides an in-depth exploration of data storage and processing systems in analytics, including data warehouses, data marts, and data lakes. It explains their purpose, structure, and use cases. It also details the ETL process, which prepares data for analysis through extraction, transformation, and loading. Additionally, it clarifies the role of data pipelines, which are broader systems facilitating data movement, including real-time streaming. Each system and process serves specific business needs, from historical reporting to predictive analytics, forming the backbone of modern data-driven decision-making.

📊 Table: What We Learnt in the Video

Topic/Section	What We Learnt
Data Warehouse	Centralized, structured, cleansed repository for reporting and analysis.
Data Mart	Subset of a data warehouse for a specific business area or function.
Data Lake	Stores raw data (all types) for undefined or future analytics.
ETL - Extract	Pulls data using batch or stream processing tools.
ETL - Transform	Cleans, standardizes, enriches, and validates data.
ETL - Load	Transfers data to the destination using different loading strategies and checks.
Data Pipelines	End-to-end data movement system that may include ETL; supports batch and real-time streaming.
Tools Mentioned	Stitch, Blendo, Apache Kafka, Samza, Storm, Beam, DataFlow.

Viewpoints: Considerations for Choice of Data Repository

🧠 Factors to Consider When Choosing a Data Repository

1. Use Case & Data Type

- **Purpose of the repository**: Understand the intended use
- **Type of data**:
  - Structured
  - Semi-structured
  - Unstructured
- **Schema awareness**: Do you know the schema beforehand?

2. Performance & Access

- **Data state**:
  - Data at rest

- Streaming data / Data in motion

- **Performance needs**:
  - High throughput?
  - Real-time access?
- **Access patterns**:
  - Frequent updates vs. long-term archival
  - Short intervals or long-running queries
- **Transaction vs. Analytics**:
  - OLTP (transaction processing)
  - OLAP (analytics, data warehousing)

3. Storage & Volume Requirements

- Volume of data:
  - Gigabytes, Terabytes, Petabytes?
- Frequency of access and update
- Backup and archival needs

4. Security & Compliance

- **Encryption needs**
- **Organization's security standards**
- **Data sovereignty or compliance mandates**

5. Organizational Standards & Preferences

- Some organizations have **approved lists** of repositories
- Often, choice is **not left to individuals**, but determined by **organizational IT standards**

🧰 **Types of Databases Used by Organizations**

1. Relational Databases (RDBMS)

- Enterprise DBs: IBM Db2, Oracle, Microsoft SQL Server
- Open-source: PostgreSQL, MySQL
- Use for: Structured data, transactional workloads

2. Document Stores

- MongoDB
- Use case: High data ingestion rates (GBs/TBs/day), semi-structured data

3. Wide Column Stores

- Apache Cassandra
- Use case: High-volume, high-write, horizontal scalability

4. Graph Databases

- Neo4j, Apache TinkerPop
- Use case: Social networks, recommendation engines, relationship mapping

5. Big Data & Analytical Systems

- Hadoop + MapReduce
- Use case: Mining petabytes of data for analytics

🛠️ Other Decision Criteria

1. Compatibility

- Compatibility with existing:
    - Programming languages
    - Tools
    - DevOps pipelines
    - Infrastructure & hosting platforms

2. Scalability

- Can the repository grow with organizational needs?
- Short-term vs. long-term performance considerations

3. Hosting Platform Considerations

- On-premises vs. cloud
- Popular choices:

- AWS RDS
- Amazon Aurora
- Google Cloud relational offerings

4. Cost & Skills

- Cost of licensing, maintenance, and cloud usage
- Internal expertise:
    - Organizations prefer tools their teams already know (e.g., Db2)
- Investment in training or upskilling

🧩 Choosing Multiple Repositories

- Most modern organizations use **multiple data repositories**
    - Enterprise RDBMS (e.g., Db2)
    - Open-source DB (e.g., PostgreSQL)
    - Unstructured data store (e.g., MongoDB)
- Different projects require different storage solutions

Summary

This video explores the key considerations organizations and data professionals weigh when choosing a data repository. The process depends on various factors like data structure, volume, performance needs, scalability, access patterns, and security. No single solution fits all use cases—many teams use multiple databases tailored to specific tasks. Examples include relational databases for structured data, document stores for high-ingestion semi-structured data, graph databases for network-based relationships, and Hadoop for large-scale analytics. Other considerations include compatibility with tools, internal expertise, hosting platforms, and cost.

Table: What We Learnt in the Video

Topic/Section	What We Learnt
Use Case & Data Structure	Determines whether to use relational, document, graph, or big data solutions
Data Volume & Performance	High volumes or streaming data may require document stores or wide column databases
Access Patterns	Whether data is accessed frequently, in real time, or archived affects the storage solution chosen
Security & Compliance	Data encryption and compliance needs influence repository selection
Organizational Standards	Companies may mandate use of certain databases depending on the task
Database Types	Various DB types exist: RDBMS, document stores, graph DBs, wide-column stores, Hadoop engines
Hosting Platforms	Cloud (AWS, GCP, Azure) vs. On-prem options also affect decision-making
Skills & Costs	In-house expertise and solution cost are major factors
Multiple Databases	Most organizations use a combination of databases based on project needs
Scalability	Solutions must be able to grow with organizational demands

Data Integration Platforms

📌 **Definition of Data Integration (According to Gartner)**

- **Discipline** that includes:
  - Practices
  - Architectural techniques
  - Tools
- **Purpose**:
  - Ingest, transform, combine, and provision data across various data types

---

📌 **Key Use Cases of Data Integration**

- Ensuring **data consistency** across multiple applications
- Supporting **Master Data Management (MDM)**
- Enabling **data sharing** between enterprises
- Facilitating **data migration** and **consolidation**

📌 **Role in Analytics and Data Science**

- **Processes Involved**:
  - Accessing, queueing, or extracting data from operational systems
  - Transforming and merging data (logically or physically)
  - Applying **data quality** and **governance** techniques
  - Delivering data for analytical purposes
- **Example**:
  To analyze customer behavior:
  - Extract customer data from sales, marketing, and finance systems
  - Provide a **unified view** of this combined data
  - Allow users to query and visualize data through a single interface

---

📌 **Relationship Between Data Integration, ETL, and Data Pipelines**

Concept	Role
Data Integration	Combines disparate data into a unified view
Data Pipeline	Covers the entire data journey (source to destination)
ETL (Extract, Transform, Load)	A process within data integration used for moving and transforming data

🔁 **Data pipelines enable** data integration
🔧 **ETL is a method** used in data integration

📌 **Capabilities of Modern Data Integration Platforms**

1. **Connectors & Adapters**:
   - Pre-built integrations with:
     - Databases
     - Flat files
     - APIs
     - Social media
     - CRM/ERP systems
2. **Open-source Architecture**:
   - Flexibility

    - Avoids vendor lock-in

3. **Processing Optimization**:
    - Supports both:
        - Batch processing
        - Continuous data streams
4. **Integration with Big Data Sources**:
    - Key factor in selecting integration platforms
5. **Additional Functionalities**:
    - Data quality
    - Governance
    - Security
    - Compliance
6. **Portability**:
    - Supports deployment across:
        - Single cloud
        - Multi-cloud
        - Hybrid environments

📌 **Vendors and Tools in the Market**

🔹 **IBM Data Integration Tools**

- IBM Information Server
- IBM Cloud Pak for Data
- IBM Cloud Pak for Integration
- IBM Data Replication
- IBM Data Virtualization Manager
- IBM InfoSphere Information Server on Cloud
- IBM InfoSphere DataStage

🔹 **Talend Tools**

- Talend Data Fabric
- Talend Cloud
- Talend Data Catalog
- Talend Data Management
- Talend Big Data
- Talend Data Services
- Talend Open Studio

🔹 **Other Commercial Vendors**

- SAP
- Oracle
- Denodo
- SAS
- Microsoft
- Qlik
- TIBCO

🔹 **Open-source/Cloud-based Tools**

- Dell Boomi
- Jitterbit
- SnapLogic

🔹 **Integration Platform as a Service (iPaaS)**

- Adeptia Integration Suite
- Google Cloud's Cooperation 534
- IBM Application Integration Suite on Cloud
- Informatica Integration Cloud

📌 **Trends in the Data Integration Space**

- Continuous evolution due to:
    - Emerging technologies
    - Growth in data volume, variety, and business use

- Demand for flexible, scalable platforms with:
    - Native cloud compatibility
    - Advanced governance and compliance features

Summary

This content explains **data integration** as a comprehensive discipline that enables businesses to combine and manage data from diverse sources for analytics, governance, and operational efficiency. Gartner defines it as involving the practices, architectures, and tools used to ingest, transform, and deliver data. The difference between data integration, ETL, and data pipelines is clarified, and the importance of data integration in analytics is emphasized. It also outlines essential capabilities of modern platforms, names leading vendors like IBM and Talend, and highlights the emergence of cloud-based and open-source solutions like iPaaS. The field is rapidly evolving to meet modern businesses' growing and complex data needs.

📊 Table: What We Learnt in the Video

Topic/Section	What We Learnt
Definition of Data Integration	A discipline involving tools and practices to combine and deliver data
Key Use Cases	Data consistency, MDM, sharing, migration, and consolidation
Role in Analytics	Supports data extraction, transformation, governance, and unified access
ETL vs Data Pipeline vs Integration	Data pipelines facilitate integration; ETL is a process within integration
Capabilities of Modern Tools	Connectors, open-source flexibility, big data support, governance, and portability
Key Vendors and Tools	IBM, Talend, SAP, Oracle, Microsoft, and open-source platforms
iPaaS Platforms	Cloud-based hosted integration services like Informatica and Adeptia
Industry Trends	Shift to cloud, focus on data quality, big data, and flexible deployment models

Lesson Summary: Welcome to Data Literacy

📚 Introduction to Data Literacy

- Data scientists require:
  - Awareness of **data storage** systems
  - Knowledge of **data organization** and **management**
  - Understanding of **data retrieval** options
- These systems enable:
  - Efficient **data discovery**
  - In-depth **data analysis**
  - Deriving insights from hidden patterns in data

🗃️ Data Repositories

- Purpose:
  - Enable retrieval of data in **usable formats**
- Choice depends on:
  - **Type of data**: structured, semi-structured, unstructured
  - **Organizational needs**

Types of Data:

Data Type	Description
Structured	Highly organized, fits in tables
Semi-structured	Partial structure (e.g., JSON, XML)
Unstructured	No predefined format (e.g., images, video)

🧮 Relational Databases (RDBMS)

- Designed for **structured data**
- Based on **tabular format** (rows and columns)
- Each table:
  - Focuses on a **specific topic**

- Columns contain **specific types of data**
- **Schema** defines relationships between tables
- **SQL** used for:
    - Data querying
    - Manipulation (Insert, Update, Delete)

✅ **Benefits:**

- Good for **data visualization** and **analysis**
- Enables **data integrity**:
    - Restrict fields to specific data types
    - Ensures consistency
- **Easy import/export** options
- **Efficient backup** and restoration

❌ **Limitations:**

- Poor handling of **semi-structured/unstructured data**
- **Slow performance** with large datasets
- Inflexible with evolving data structures
- Field length limits can **restrict storage**

🧩 **NoSQL Databases (Not Only SQL)**

- Designed for:
    - **Speed**
    - **Scalability**
    - **Flexibility**
- Handle **semi-structured** and **unstructured** data
- No need for **predefined schemas**

Types of NoSQL Databases:

Type	Description
Document-based	Stores documents (e.g., JSON); grouped in collections

Key-Value	Each data item stored as a **key-value pair**
Columnar	Stores data **by columns**, ideal for analytical workloads
Graph	Uses **nodes and relationships**; ideal for managing complex connections

---

🏢 **Big Data Storage Solutions**

📦 **Data Warehouse**

- Centralized, multipurpose storage
- Data is **pre-modeled and structured**
- Supports:
  - Reporting
  - Analytical querying
- Suitable for **massive amounts of operational data**

🧰 **Data Mart**

- Subsection of a data warehouse
- Tailored for:
  - Specific **business functions**
  - Certain **departments or user groups**
- Benefits:
  - Targeted analysis
  - Isolated performance and security

🌊 **Data Lake**

- Stores **all types of data** (structured to unstructured)
- Stores data in **native format**
- Uses **metadata** for classification and tagging

🔄 **Data Pipelines**

Definition:

- Systems for **collecting, transforming, and moving data**
- Ensures data flows efficiently from source to destination

ETL (Extract, Transform, Load):

- A **subset of data pipelines**

- Key stages:
    1. **Extract** – Pull raw data from sources
    2. **Transform** – Clean, enrich, reformat data
    3. **Load** – Store transformed data in target system for analysis
- Goal: Make raw data **analysis-ready**

📝 Summary

This video introduces essential concepts in data literacy for data scientists, focusing on technologies for storing, organizing, managing, and retrieving data. It covers different types of databases and repositories suitable for structured, semi-structured, and unstructured data. Relational databases (RDBMSs) are suited for structured data and offer consistency but struggle with scale and flexibility. In contrast, NoSQL databases provide scalable and schema-free solutions for more complex data types. For large-scale storage, options like data warehouses, data marts, and data lakes are explored. Finally, the video emphasizes the importance of data pipelines and ETL processes in preparing data for analysis, ensuring data scientists can derive meaningful insights efficiently.

📊 Table: What We Learnt in the Video

Topic/Section	What We Learnt
Data Repositories	Store different data types; must support retrieval in usable formats
Structured vs Unstructured	Different data types require different storage approaches
RDBMS	Structured data in tables; uses SQL; limited in flexibility and scalability
NoSQL Databases	Schema-less; handles semi/unstructured data; includes document, key-value etc
Data Warehouse	Centralized, structured data storage for reporting and analysis
Data Mart	Focused subset of a data warehouse for specific business functions
Data Lake	Stores raw data of all types with metadata classification
Data Pipeline	Manages flow of data through collection, transformation, and movement
ETL	Specific pipeline for extracting, transforming, and loading data

Summary: Data Literacy for Data Science

Congratulations! You have completed this lesson. At this point in the course, you know:

- The basics of data collection and organization methods.
- What RDBMS is and its significance.
- NoSQL databases and their flexible schema.
- Types of data storage and the ways to process data.
- The factors influencing data repository selection.
- The various data integration tools and the solutions they provide.