
PLURALSIGHT

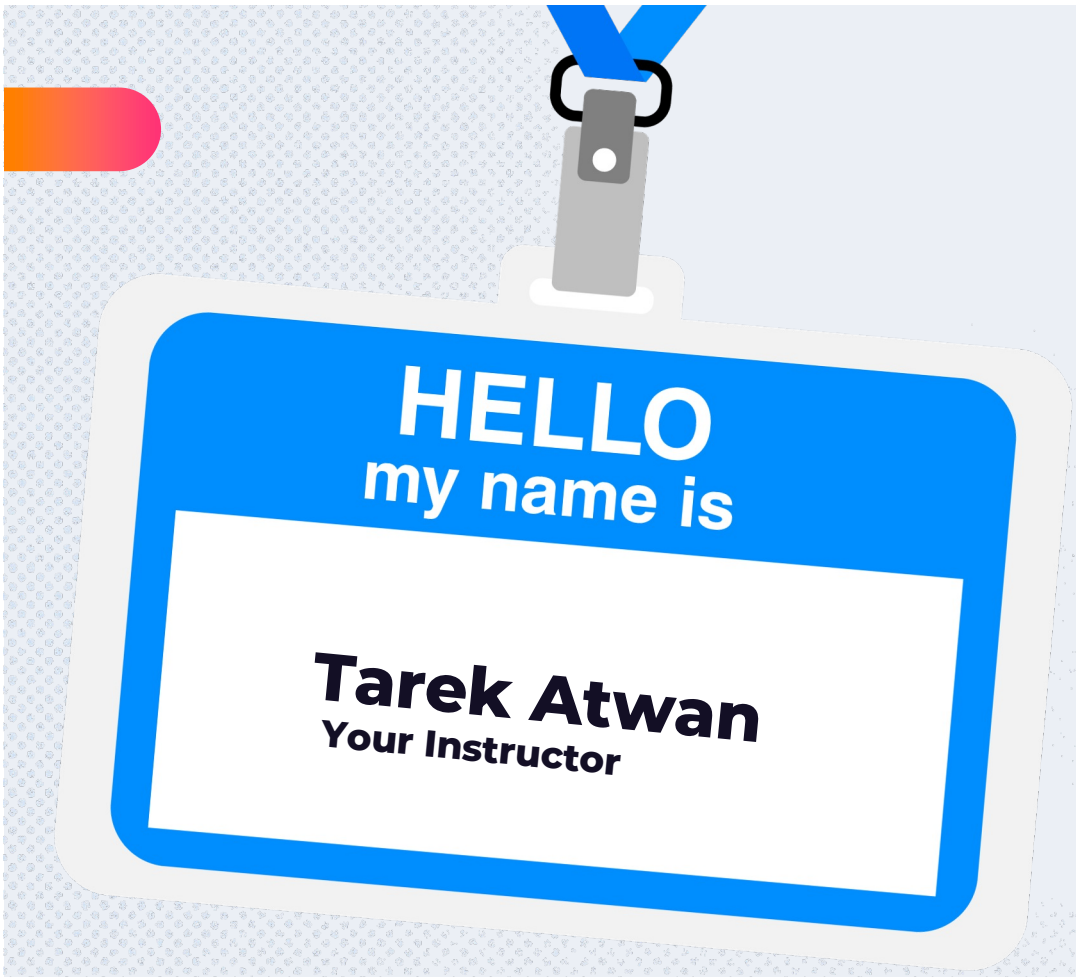
Data Engineering

THE HARTFORD

Tech Catalyst Bootcamp

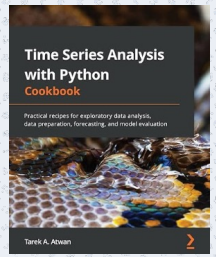
Tarek Atwan
Instructor, Pluralsight

PLURALSIGHT



About Me:

- Book Author
- 18+ Years Consulting Services
- 5+ Years Instructor
- 2 Startups
- Book Author "Time Series Analysis with Python Cookbook"
- World Traveler



About Me:

- 4+ Years of Training Experience
- Generative AI Practitioner
- Expert in Data Engineering and Data Science Realms
- Technical Speaker

My pledge to you:

I will..

- Make this interactive
- Ask you questions
- Ensure everyone can speak
- Create an inclusive learning environment
- Use an on-screen timer for breaks

...also, if you have an accessibility need, please let me know!

Data Engineering Bootcamp Goal

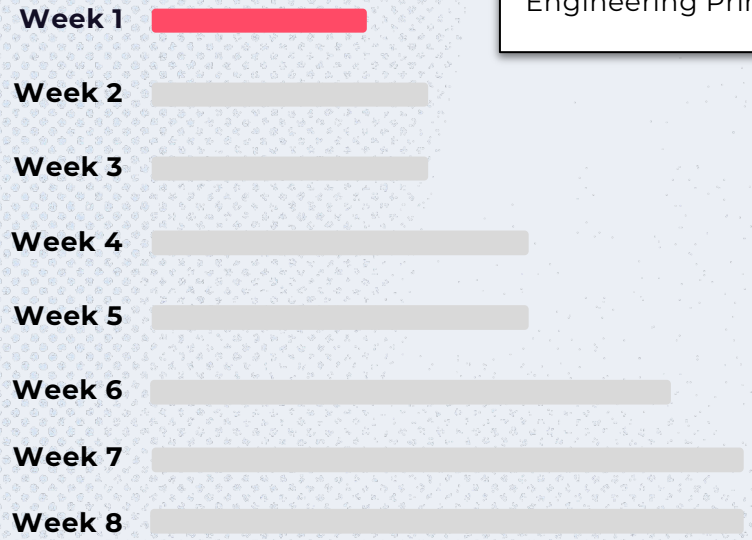
At the end of this bootcamp, you will:

Become a **data Engineering Ninja**

Weekly Breakdown

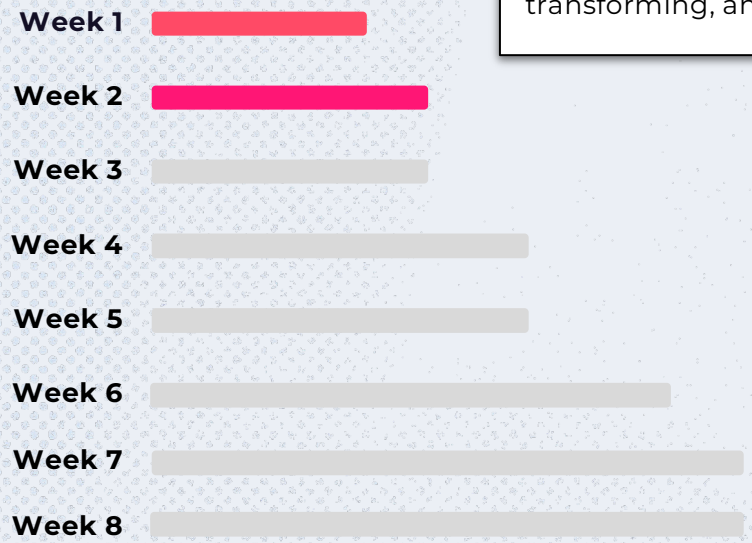
Week 1

Data Engineering on the Cloud: Introduction to Data Engineering Fundamentals including Data Architecture Fundamentals, Data Engineering Primer, a Data Engineering on AWS.



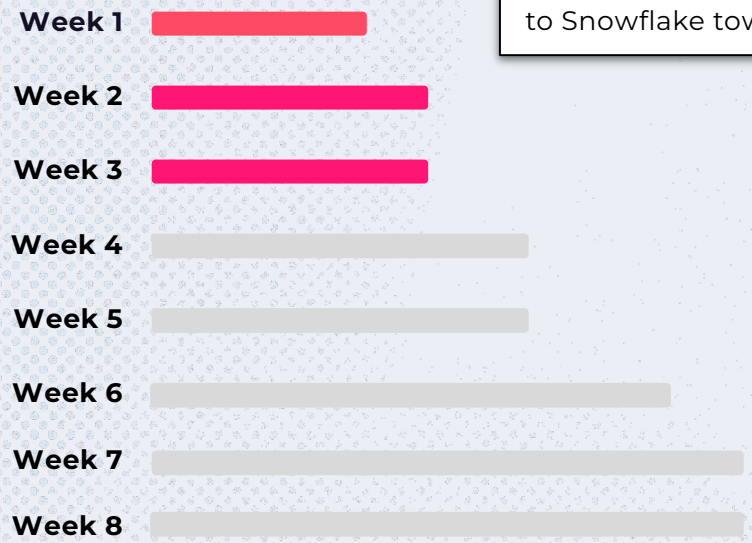
Week 2

Python for Data Engineering & Analytics: You will be introduced to the pandas library and learn about several I/O capabilities for ingesting, transforming, and loading data using pandas..



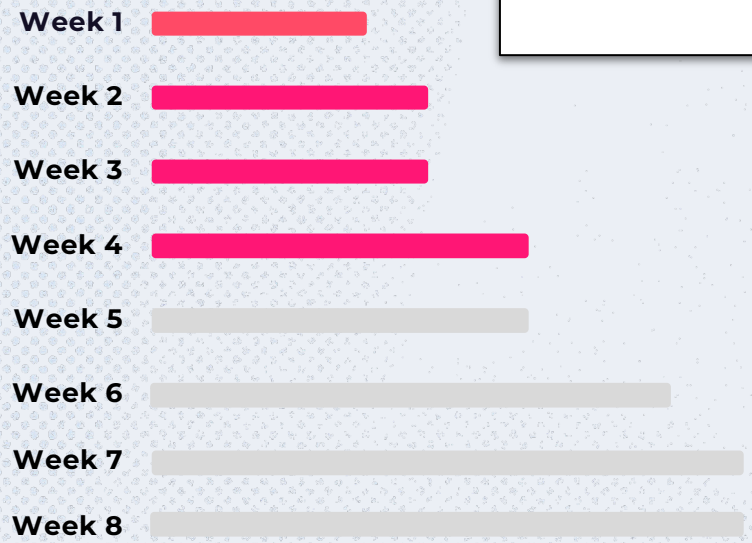
Week 3

SQL for Data Engineering and Analytics: We will start learning about Relational Database Systems, SQL, and go from an intro to SQL to more advanced SQL techniques using PostgreSQL. You will be introduced to Snowflake toward the end of this week.



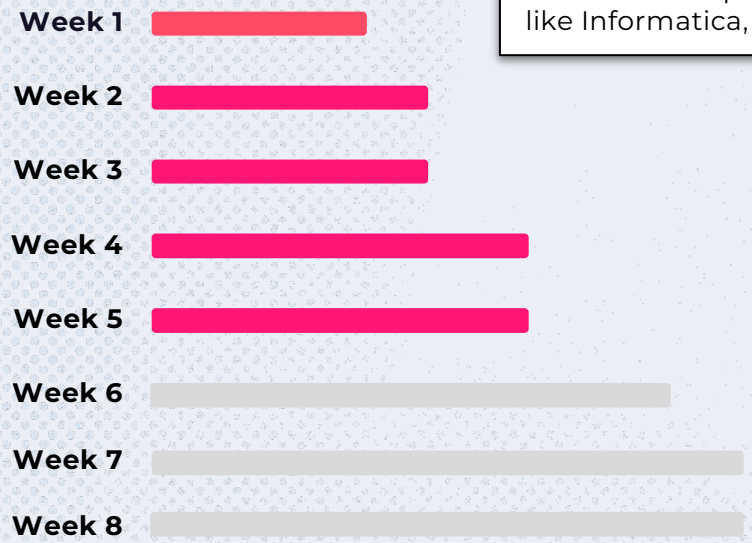
Week 4

Advanced Snowflake: We will continue using SQL and advanced SQL with a focus on Snowflake and its capabilities.



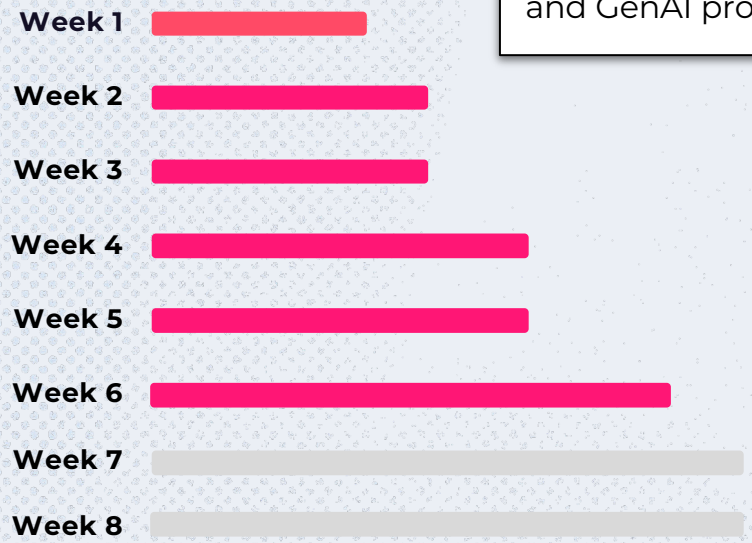
Week 5

Building Data Engineering Pipelines: In this section you will have hands-on experience building data pipelines on AWS. Learn about Linux and review important terminal commands. You will learn about other tools, like Informatica, for creating an ETL pipeline.



Week 6

DevOps, DataOps, MLOps, and LLMOps: You will learn about the role of a Data Engineer in Machine Learning, Artificial Intelligence, and GenAI process and workflow.

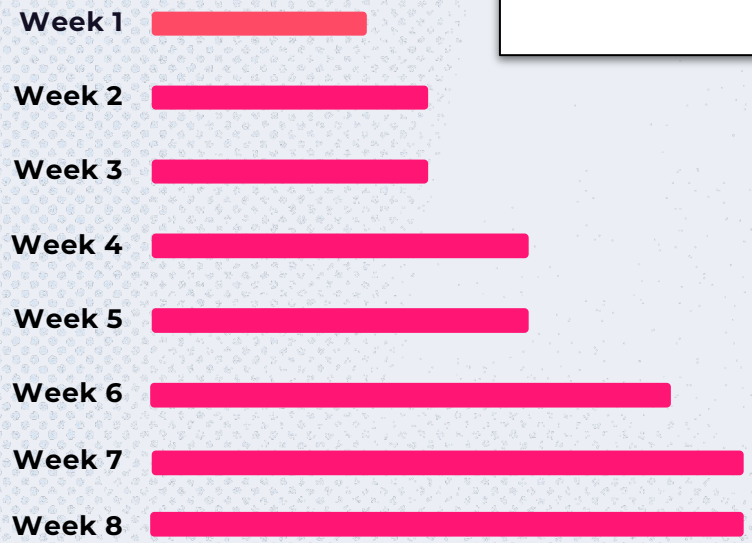


Week 7

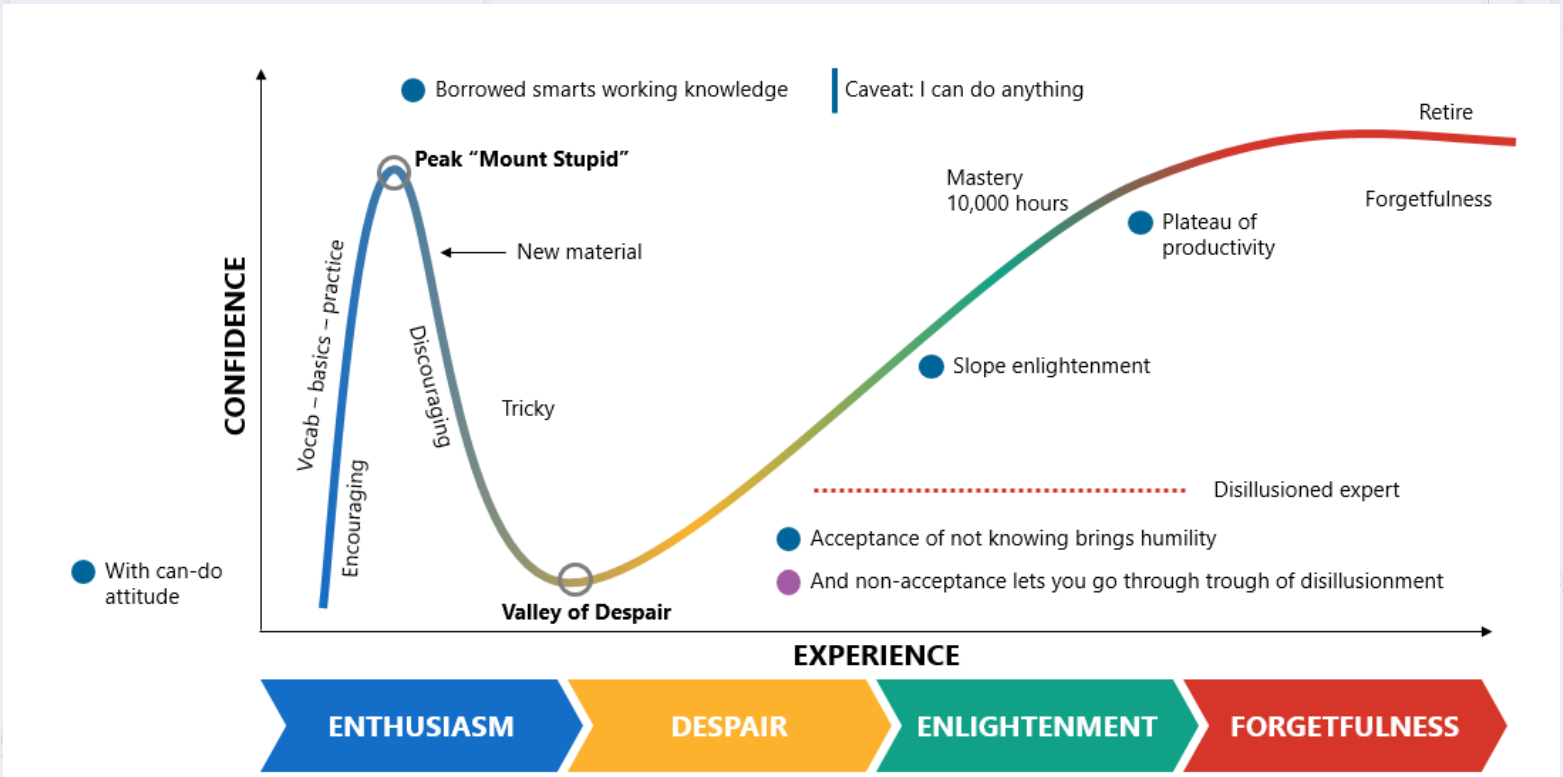
Data Visualization Fundamentals: You will learn about Data Visualization best practices and get hands-on experience using Tableau and Thoughtspot

Week 8

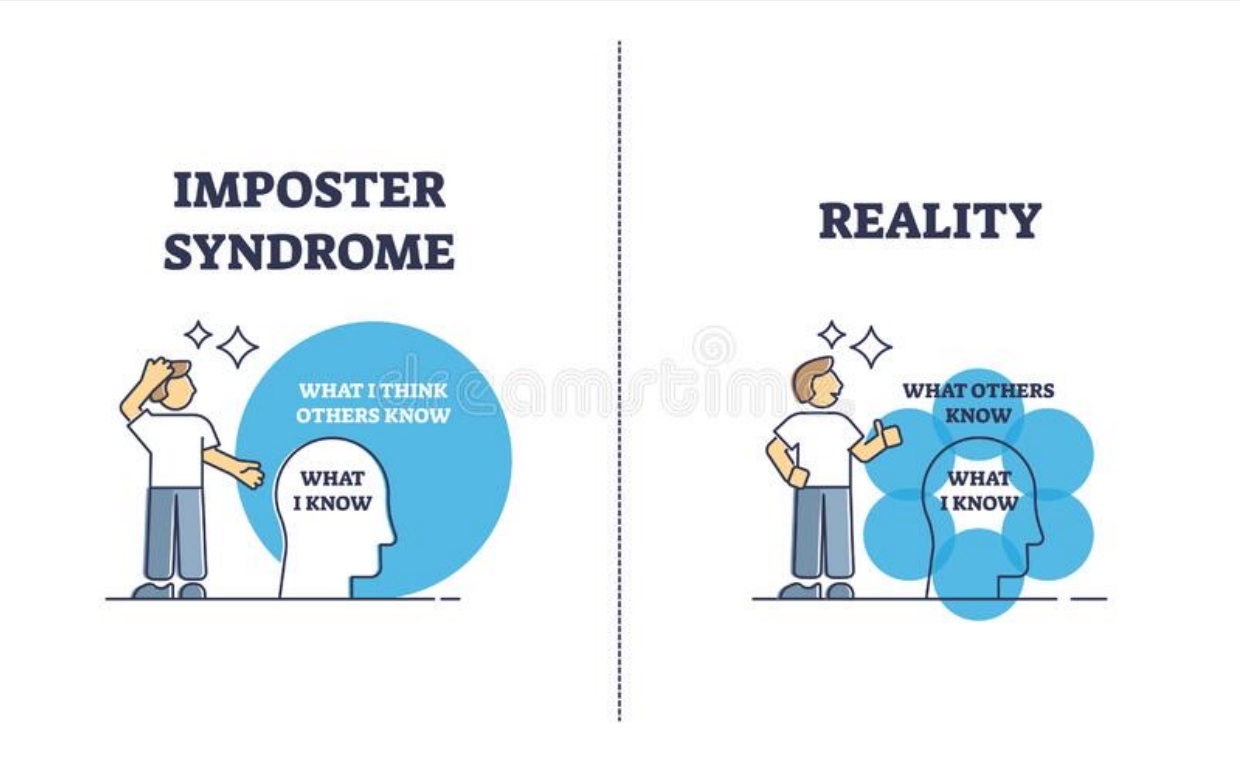
Capstone Project : End-to-End project to demonstrate what you have learned.



Dunning Kruger Effect



Imposter Syndrome



Data Engineering on the Cloud

Week 1

What is Data Engineering

Data Engineering

Data engineering is the process of **designing** and **building** systems to **collect**, **process**, and **manage** large amounts of data from various sources and formats. It involves creating **data pipelines**, **data warehouses**, and **data lakes** to enable the analysis and use of data for business insights and **decision-making**.

Data engineers are responsible for ensuring that data is **reliable**, **consistent**, and easily **accessible** for data scientists and other stakeholders.

Data Engineering Tasks

Some of the common tasks a data engineer might perform when working with data include:

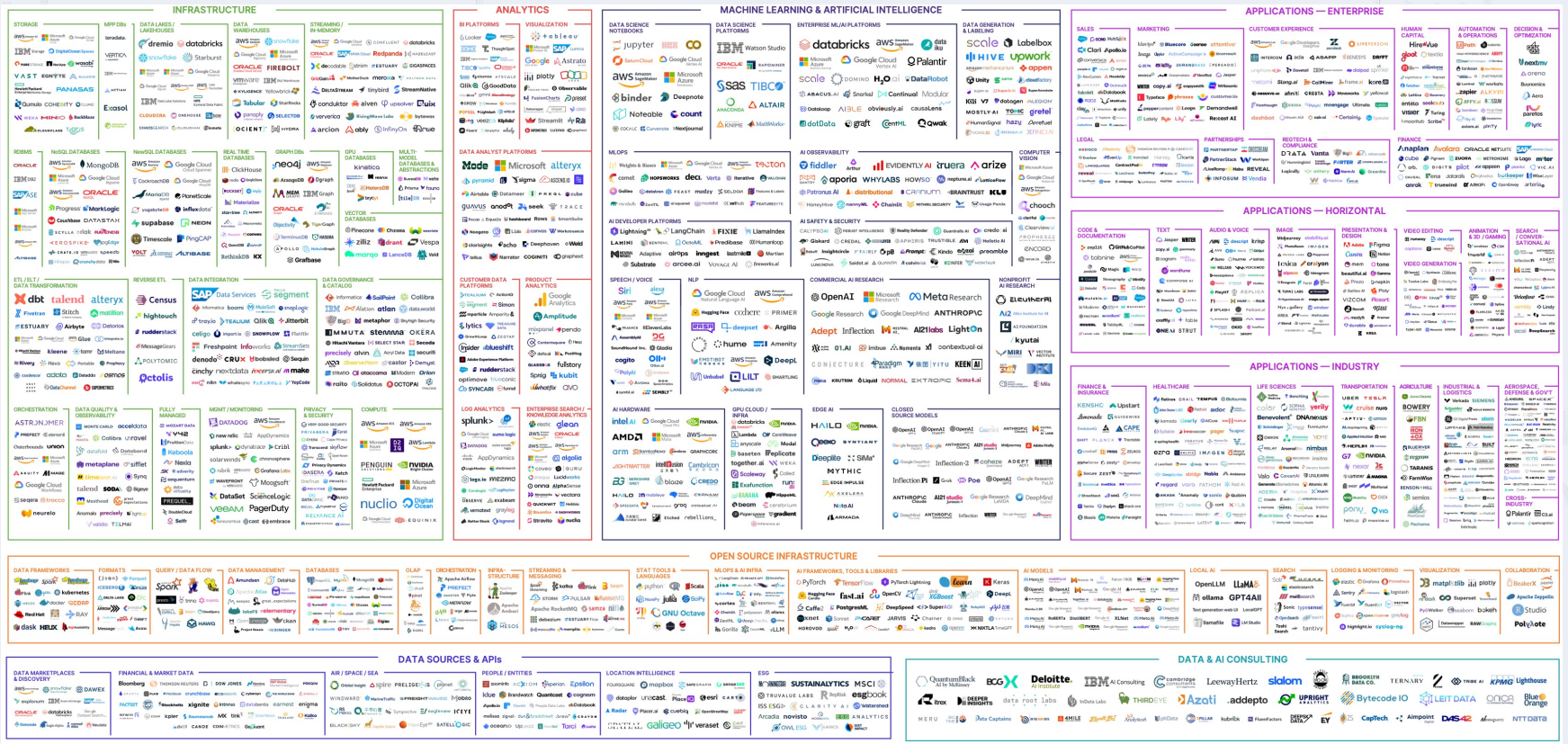
- Acquire datasets that align with business needs
- Develop algorithms to transform data into useful, actionable information
- Build, test, and maintain database pipeline architectures
- Collaborate with management to understand company objectives
- Create new data validation methods and data analysis tools
- Ensure compliance with data governance and security policies

Data Engineering Skills

Data Analyst	Data Engineer	Data Scientist
Data Warehousing	Data Warehousing & ETL	Statistical & Analytical skills
Adobe & Google Analytics	Advanced programming knowledge	Data Mining
Programming knowledge	Hadoop-based Analytics	Machine Learning & Deep learning principles
Scripting & Statistical skills	In-depth knowledge of SQL/ database	In-depth programming knowledge (SAS/R/ Python coding)
Reporting & data visualization	Data architecture & pipelining	Hadoop-based analytics
SQL/ database knowledge	Machine learning concept knowledge	Data optimization
Spread-Sheet knowledge	Scripting, reporting & data visualization	Decision making and soft skills

Data Analytics Landscape

The Landscape of Data Analytics



Current and Future Analytics Technology Stack

Considerations

- **License**: Open Source vs. Commercial
- **Level of Coding**: Coding vs. Code-Free
- **Scaling**: Enterprise vs. Departmental
- **Deployment**: On-Premise vs. Cloud
- **User Accessibility**: Technical users vs. Non-Technical users

What is the end goal

Data-Driven Decision Making

What is it?

- Using Facts, Metrics, and Data to Guide Strategic Business Decisions

Why is it important?

- Enhance Accuracy
- Reduce Risks
- Fosters Proactive Strategies

Data-driven decision making refers to the process of using data, facts, metrics, and insights to guide strategic business decisions that align with goals, objectives, and initiatives

Being a Data-Driven Organization

What is it?

- Leverage data at every level for better decision making

Why is it important?

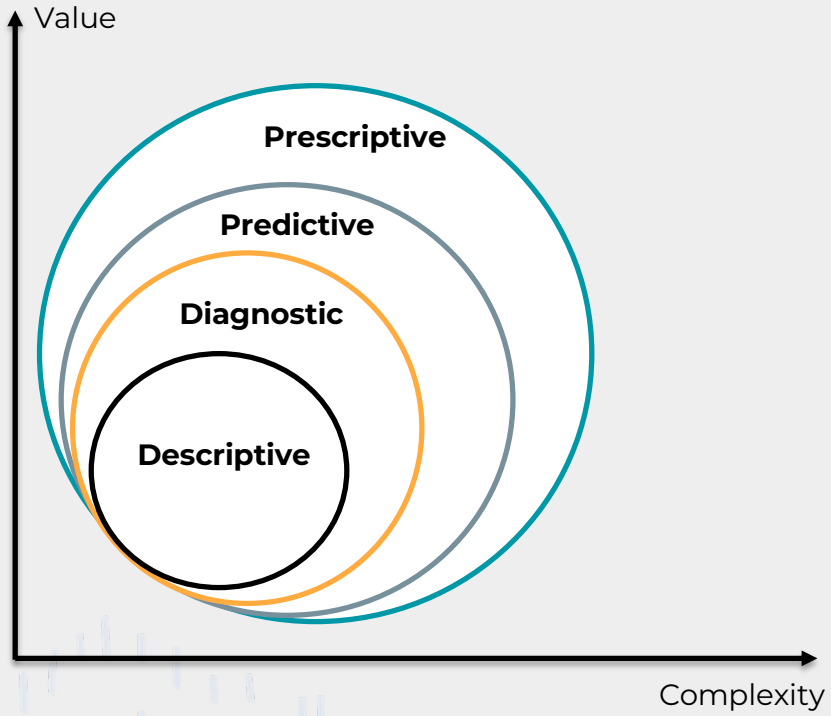
- Ensures higher quality
- Efficient operations
- Increased innovation
- Data democratization

Data democratization

Empowering individuals to use data for informed decisions

A data-driven organization is one that uses data insights to guide its decision-making processes. It is a long-term continuous process that requires addressing business needs, people mindset, and culture change

Types of Data Analytics



Descriptive
What is happening in my business?

Diagnostic
Why is it happening?

Predictive
What is likely to happen?

Prescriptive
What do I need to do?

What is Big Data

What is Big Data

What is it?

- Massive volume of data that can't be processed using traditional databases or techniques
- Volume, Variety, Velocity (Three V's)
- 4 Vs, 5 Vs, 6Vs ..etc

Why is it important?

- Uncover hidden patterns and correlations
- Uncover deeper insights
- **The importance of big data lies in how it is used.**

What are some challenges?

- Storage
- Processing
- Analysis
- Visualization

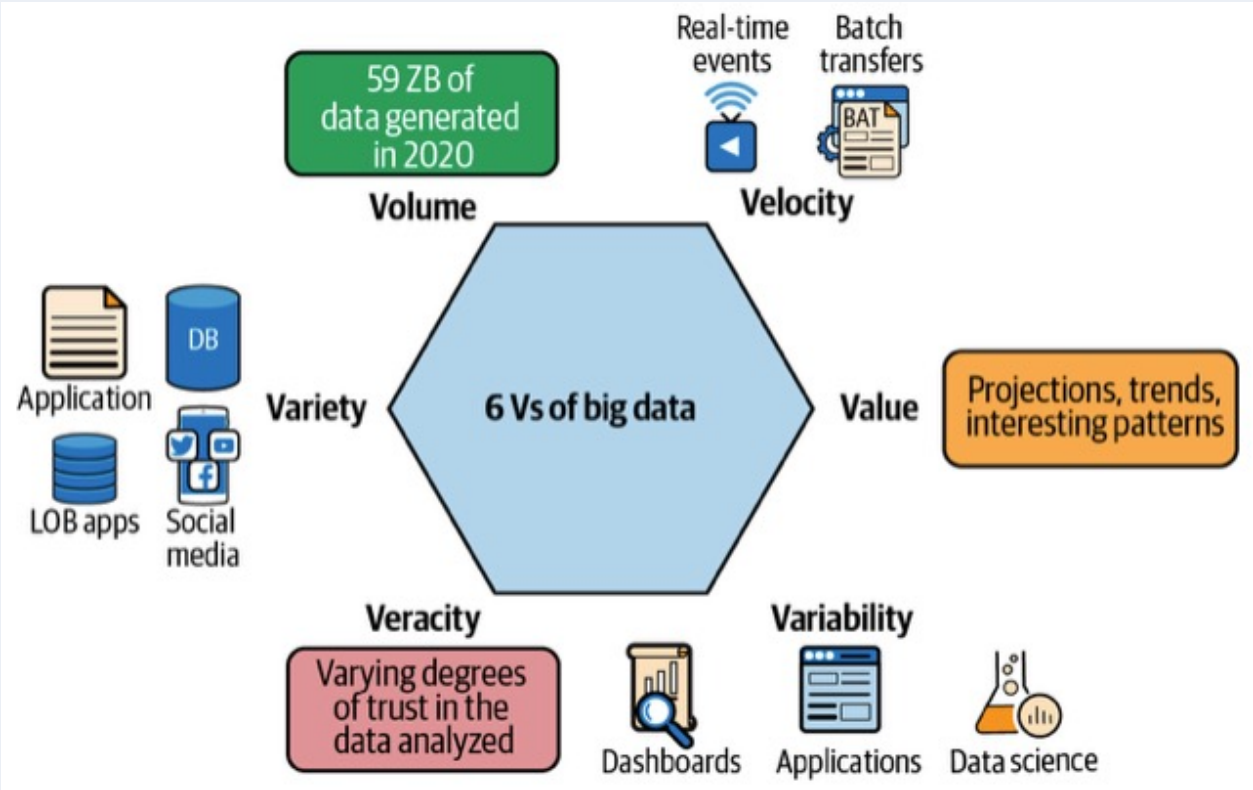
What's Big Data?

Big data is high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization

—Doug Laney, Gartner

Volume	Velocity	Variety
• Result of data being continuously gathered • Terabytes rather than Gigabytes	• Require continuously updated analysis	• Not suitable for traditional databases • Unstructured text, images, video, citation graphs...

What is Big Data



Diversity in Data Sources

Storage Location

- On-Premises
- Cloud

Storage Type (Examples)

- **Databases** (Relational, Non-Relational)
  - MySQL, SQL Server, MongoDB
- **File-based**
  - Excel, CSV, JSON, XML

Data Forms and Speeds

Forms:

- **Structured** – Highly organized like (example: Excel Spreadsheet)
- **Unstructured** – No pre-defined structure (example: Emails or a Social Media Post)
- **Semi-Structured** – A hybrid (example: XML or JSON)

Speeds:

- **Interval-Based**
  - Daily, Weekly, Monthly
- **Real-Time**

Different forms and speeds serve varied analytics needs

Structured Data

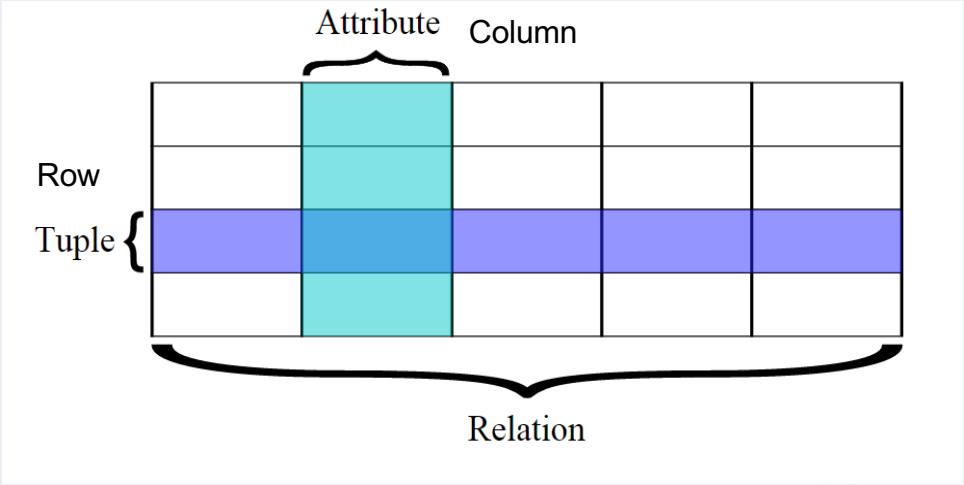
Data that is organized in a defined manner or schema, typically found in relational databases.

Characteristics:

- Easily queryable
- Organized in rows and columns
- Has a consistent structure

Examples:

- Database tables
- CSV files with consistent columns
- Excel spreadsheets



Structured Data

DEFINITION:

- Data that is stored and is usable in a way in which its **type is well-defined**.
- Structured data definitions (schema) include **format** (i.e. number, text, bit) **and length/precision**.

COMMON SOURCES:

- Relational Database
- Excel Spreadsheet
- OLAP Cube
- XML (well-formed)
- CSV files with consistent columns

Structured Data Key Terms

- Schema
- Table
- Index
- View
- Partition
- Constraint

- Database
- Data Mart
- Data Warehouse
- Row/Record/Case
- Attribute/Field/Variable/Value
- Key

Unstructured Data Defined

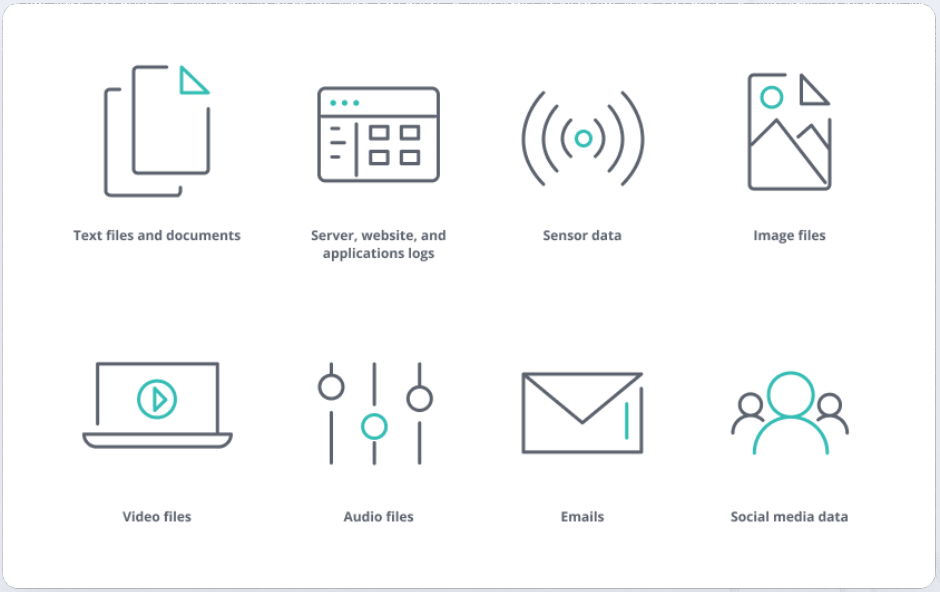
Data that doesn't have a predefined structure or schema.

Characteristics:

- Not easily queryable without preprocessing
- May come in various formats

Examples:

- Text files without a fixed format
- Video and Audio files
- Images
- Emails and word processing documents



Unstructured Data Defined

DEFINITION:

- Data that does not have a specific schema applied.
- Schemas are typically set at query runtime.
- Unstructured data tends to be text-heavy.

COMMON SOURCES:

- Flat Files
- Multi-media Files
- Sensor Data
- Feeds

Semi-structured

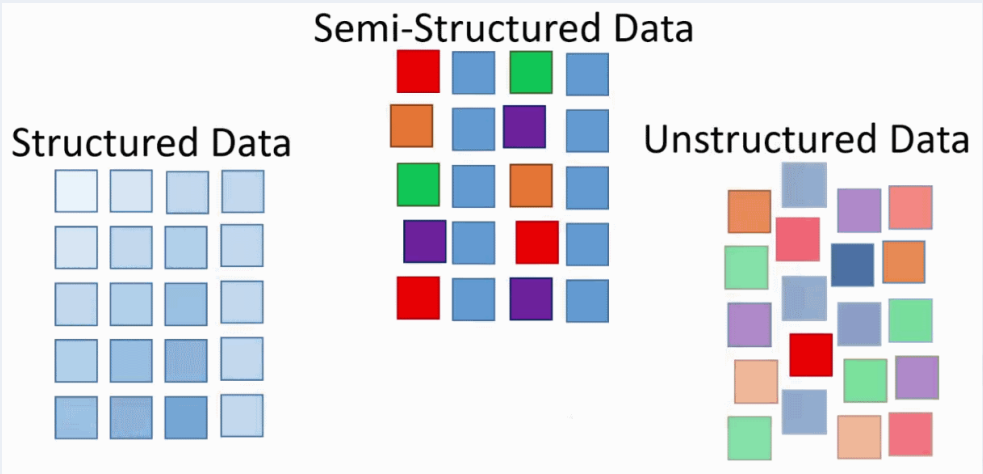
Data that is not as organized as structured data but has some level of structure in the forms of tags, hierarchies, or other patterns.

Characteristics:

- Elements might be tagged or categorized in some way
- More flexible than structured data but not as chaotic as unstructured data

Examples:

- XML and JSON files
- Email headers (which have a mix of structured fields like data, subject, etc., and unstructured data in the body).
- Log files with varied formats



Semi-structured

DEFINITION:

Data that is neither raw or fitted to a conventional database system

COMMON SOURCES:

- CSV
- XML
- JSON
- NoSQL

Structured vs Semi-Structured vs Unstructured

Unstructured data

The university has 5600 students.
John's ID is number 1, he is 18 years old and already holds a B.Sc. degree.
David's ID is number 2, he is 31 years old and holds a Ph.D. degree. Robert's ID is number 3, he is 51 years old and also holds the same degree as David, a Ph.D. degree.

Semi-structured data

```
<University>
  <Student ID="1">
    <Name>John</Name>
    <Age>18</Age>
    <Degree>B.Sc.</Degree>
  </Student>
  <Student ID="2">
    <Name>David</Name>
    <Age>31</Age>
    <Degree>Ph.D. </Degree>
  </Student>
  ....
</University>
```

Structured data

ID	Name	Age	Degree
1	John	18	B.Sc.
2	David	31	Ph.D.
3	Robert	51	Ph.D.
4	Rick	26	M.Sc.
5	Michael	19	B.Sc.

Unstructured	Semi-structured	Structured
PDFs, JPEGs, MP3, Movies, ...	CSV, JSON, XML, MongoDB, ...	Oracle, MSSQL, MySQL, DB2, ...

Sources of Data

Internal

- Internal data most relevant to tactical
- Granularity is very low
- Tends to have more quality issues
- Serves as corporate asset
- Can be augmented with external data

Sources of Data

External

- Great external sources exist, but often at a cost
- Granularity is often highly summarized
- Used often to provide macro lens context
- Data Validation source
- Many large corporations purchase external data on subscription

Working with Big Data

Working with Big Data effectively requires

- Collecting and storing data efficiently and cost-effectively
- Processing data in a timely manner
- Analyzing the results

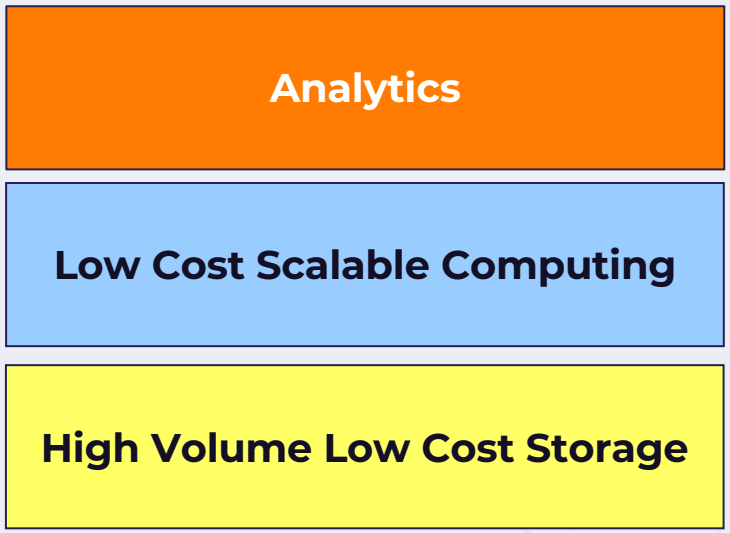
Developments in technology for Big Data

- Reliable, low-cost, scalable storage
- Cheap processing power on demand
- Better tools and algorithms for processing and analyzing data

Big Data Stack

A Big Data stack typically has three layers

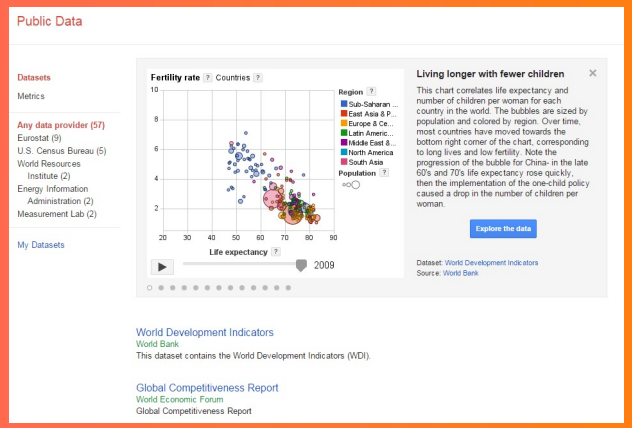
The technology and tools at each layer vary widely dependent on business needs



Exercise: External Data Sources

- Data.gov

Google public data explorer



Modern Data Architecture

What is Everyone Talking about in 2024

ChatGPT

Artificial Intelligence

Chat Bots

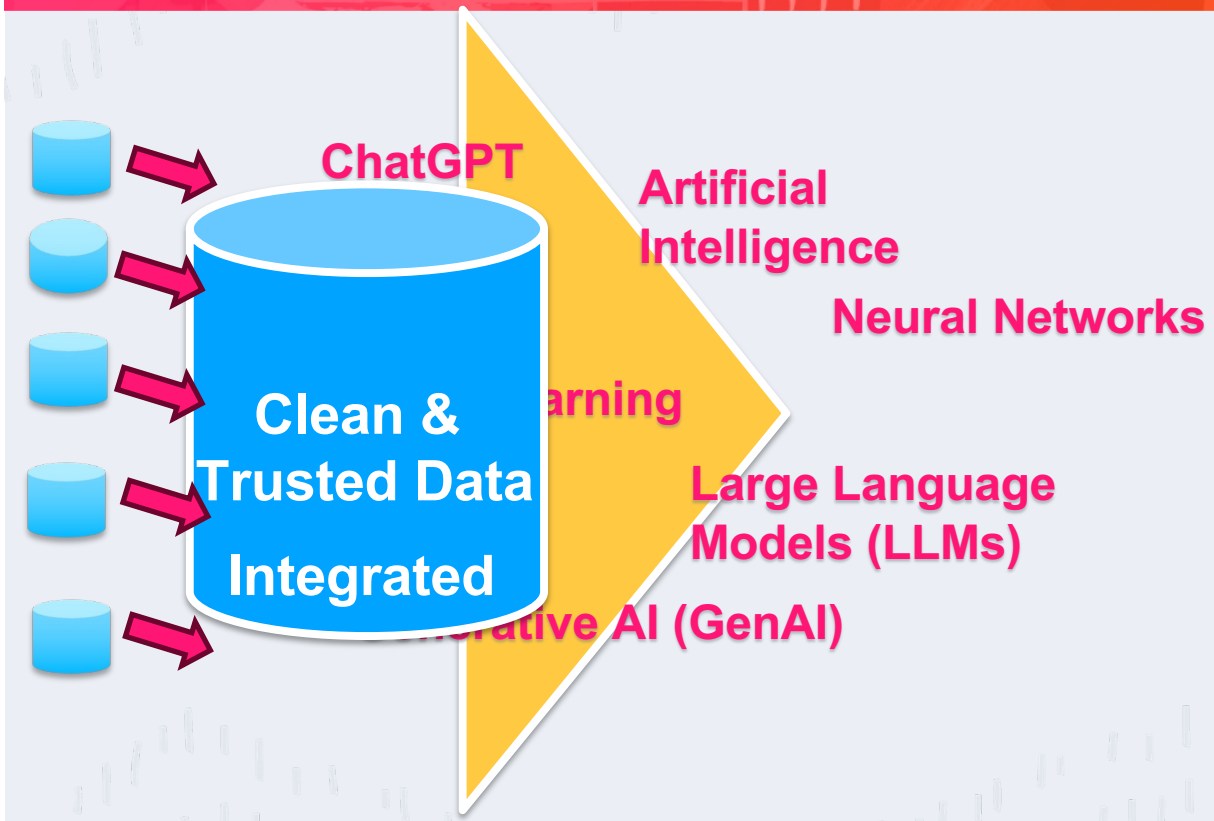
Neural Networks

Deep Learning

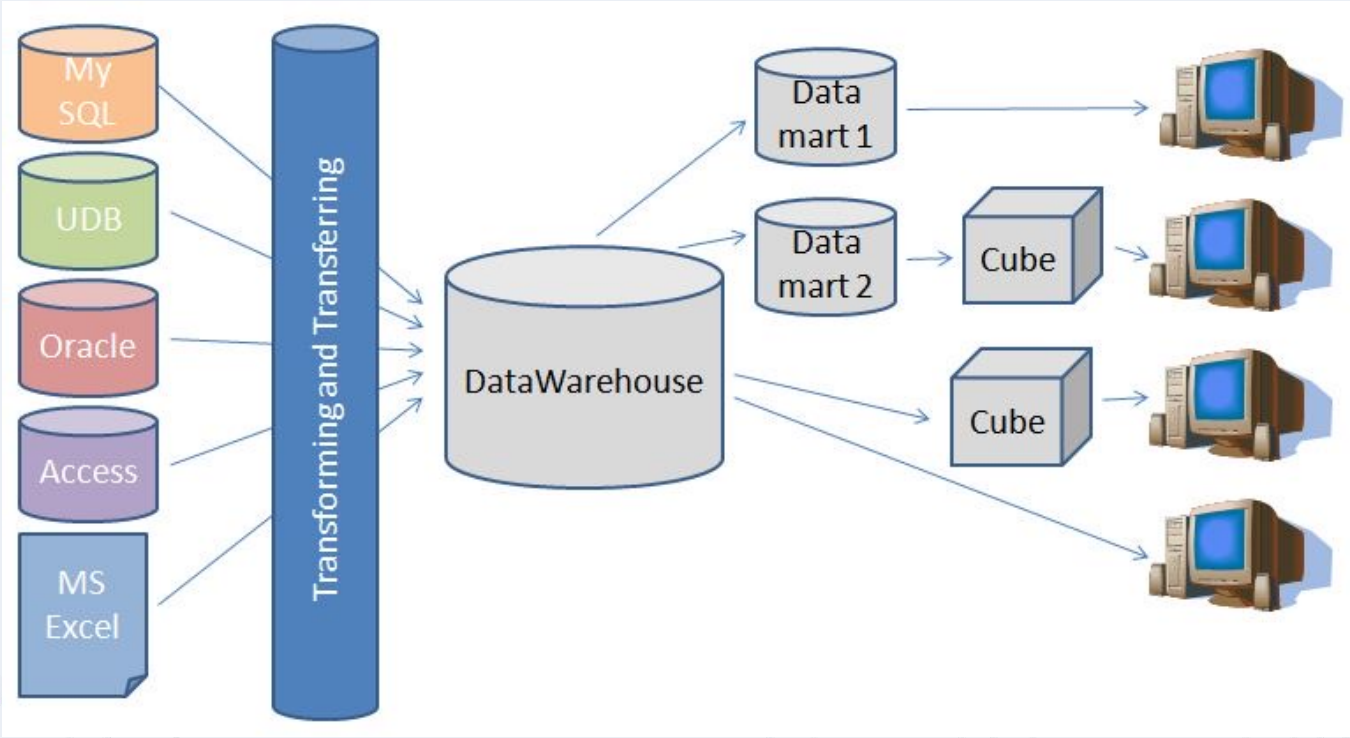
Large Language Models (LLMs)

Generative AI (GenAI)

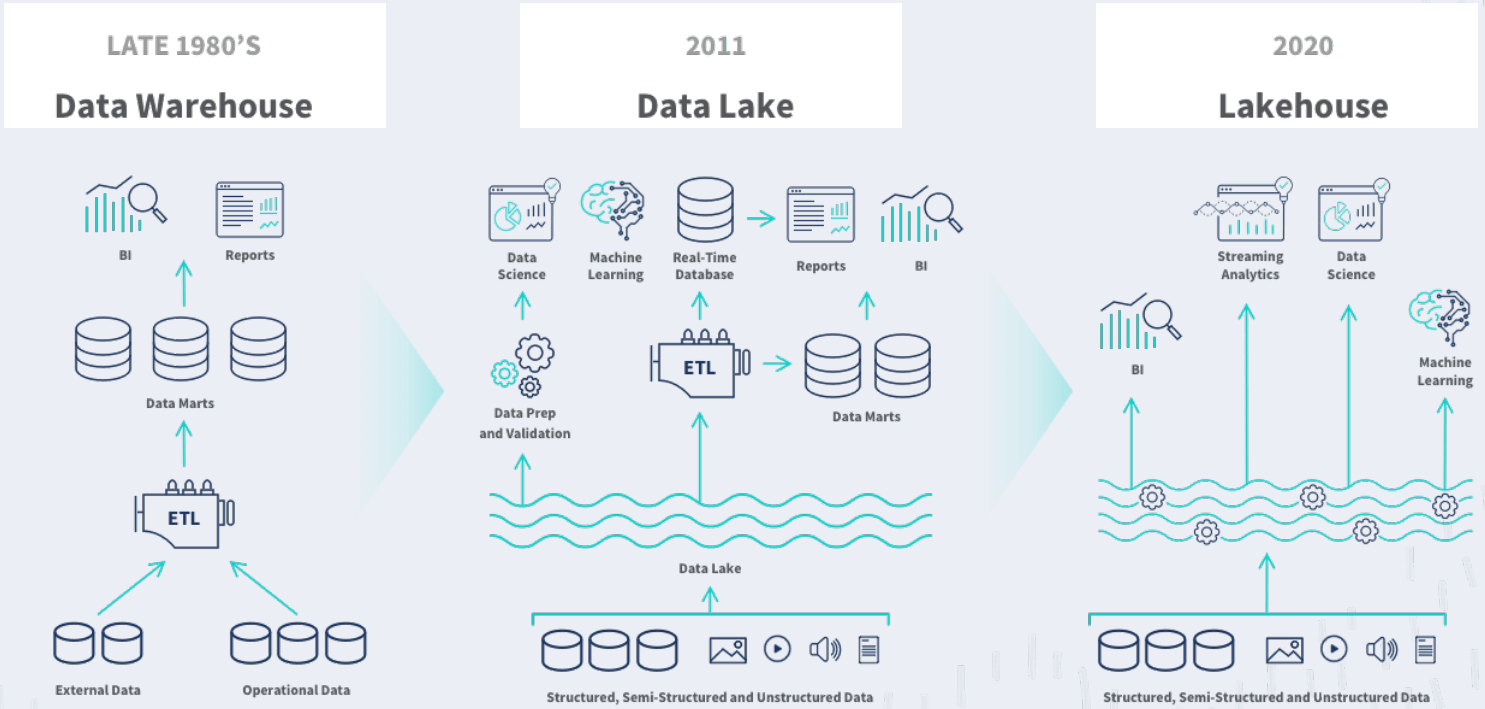
What No One Likes to Talk About?



Making Data Usable



Modern Data Architecture



Data Warehouse

Data warehouses ingest and hold highly structured and unified data to support specific business intelligence and analytics needs. The data has been **transformed** and fit into a **defined schema**.

Pros:

- Little or no data prep needed. This makes it faster and easier for analysts and business users to access and analyze the data.
- Unified, harmonized data offers a single source of truth, building trust in data insights and decision-making across business lines.
- Designed for complex queries and analysis. Optimized for read-heavy operations.

Cons:

- Not appropriate for storing unstructured or semi-structured data.
- Traditional data warehouses can cost more than data lakes, due to additional operational costs.

Data Warehouse

amazon REDSHIFT

Google BigQuery

Azure Synapse Analytics

snowflake

Data Lake

Data lakes ingest and hold **raw data** in a wide variety of formats to directly support data science and machine learning. Massive volumes of **structured** and **unstructured** data. Data teams can build data pipelines and schema-on-read transformations to make data stored in a data lake available for BI and analytics tools.

Pros:

- Storage costs and operational costs can be less compared to a data warehouse.
- Data scientists or end-to-end self-service BI tools can gain access to a broader range of data far faster than in a data warehouse because it is kept in a raw state.
- Supports batch, real-time, and stream processing

Cons:

- Streaming and appending data can be difficult leading to inconsistency and isolation.
- Data lakes don't support transactional data and don't enforce data quality.

Data Lake

AWS S3 Storage

Azure Data Lake Storage Gen2

Data Lake vs Data Warehouse (DW)

Schema:

- Data Warehouse: **Schema-on-write** (predefined schema before writing data)
  Extract – Transform – Load (ETL)
- Data Lake: **Schema-on-read (**schema is defined at the time of reading data)
  Extract – Load – Transform (ELT)

Data Types:

- Data Warehouse: Primarily structured data
- Data Lake: Both structured and unstructured data

Agility:

- Data Warehouse: Less agile due to predefined schema
- Data Lake: More agile as it accepts raw data without a predefined structure

Processing:

- Data Warehouse: ETL (Extract, Transform, Load)
- Data Lake: ELT (Extract, Load, Transform) or just Load for storage purposes

Cost:

- Data Warehouse: Typically more expensive because of optimizations for complex queries
- Data Lake: Cost-effective storage solutions, but costs can rise when processing large amounts of data

Data Lakehouse

The data lakehouse can be more flexible than the traditional data warehouse or data lake architecture in that it can eliminate data redundancy and improve data quality while offering lower cost storage. **ETL pipelines** provide the critical link between the unsorted lake layer and the integrated warehouse layer.

Pros:

- A single data repository requires less time and budget to administer than a multiple-solution system. Simplified data governance due to having a single control point.
- Lower data storage cost. Less data movement and redundancy.
- Simplified schema management.
- ACID-compliant transaction support

Cons:

- Can be expensive and time-consuming to migrate from traditional data warehouses.

Data Lake and Lakehouse

databricks

CLOUDERA

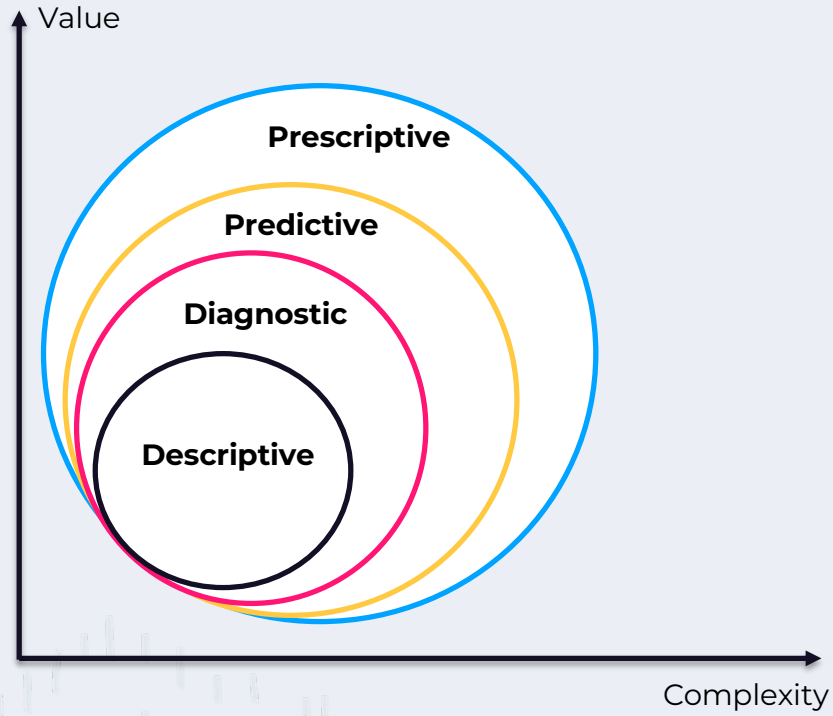
AWS Lake Formation

DELTA LAKE

DATA LAKES / LAKEHOUSES

dremio
databricks
snowflake
Starburst
Microsoft Azure HDInsight
Microsoft Azure Data Lake Storage
Google Cloud Dataproc
Google Cloud BigLake
aws Amazon EMR
aws Lake Formation
IBM Data Lake Solutions
HPE Ezmeral Data Fabric
CLOUDERA
ONEHOUSE
Qubole
CHAOSSEARCH
CLOUDIAN
iomete

Types of Data Analytics



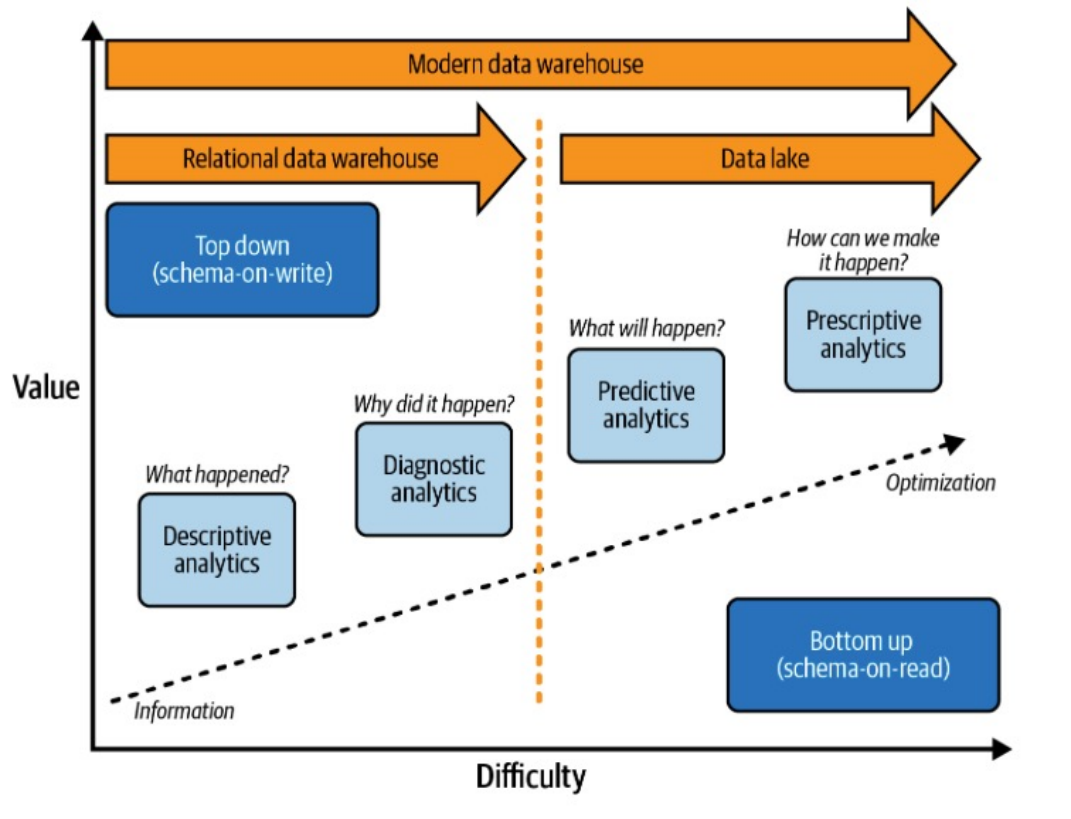
Descriptive
What is happening in my business?

Diagnostic
Why is it happening?

Predictive
What is likely to happen?

Prescriptive
What do I need to do?

Big Data Analytics



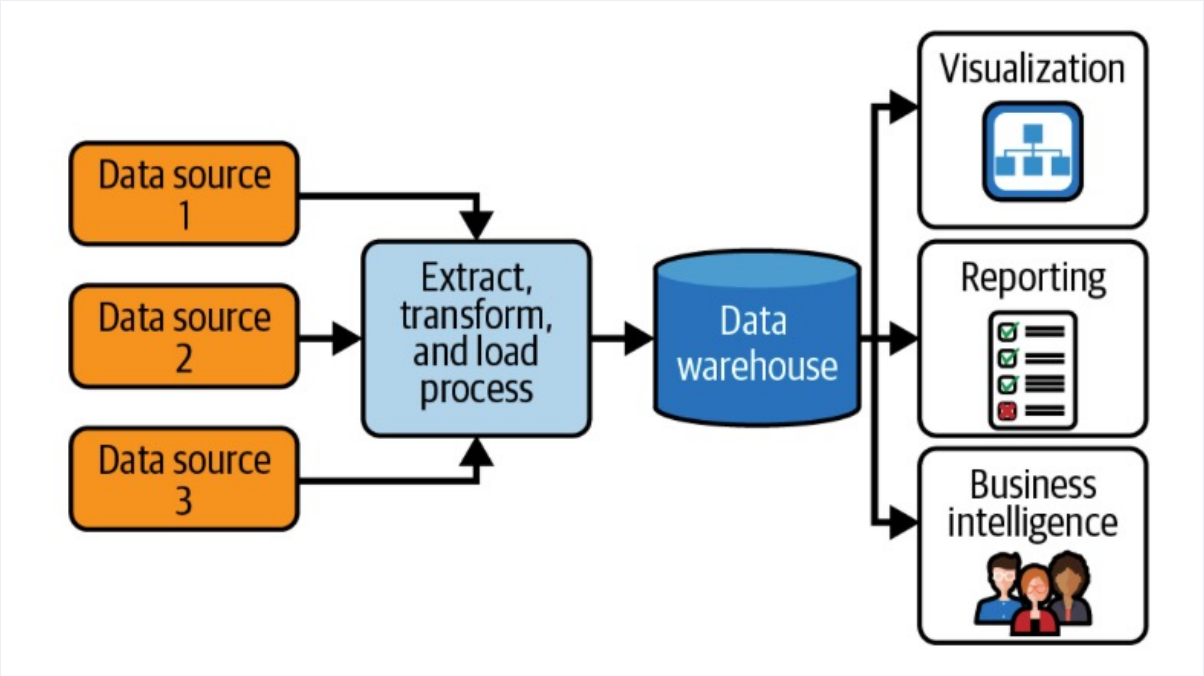
Data Warehouse Architecture

Historical Analysis

Structured Data

Descriptive Analytics

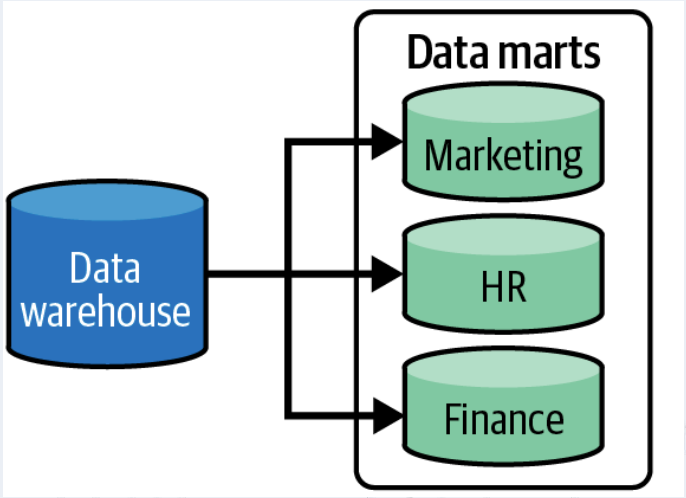
Diagnostic Analytics



Data Mart Architecture

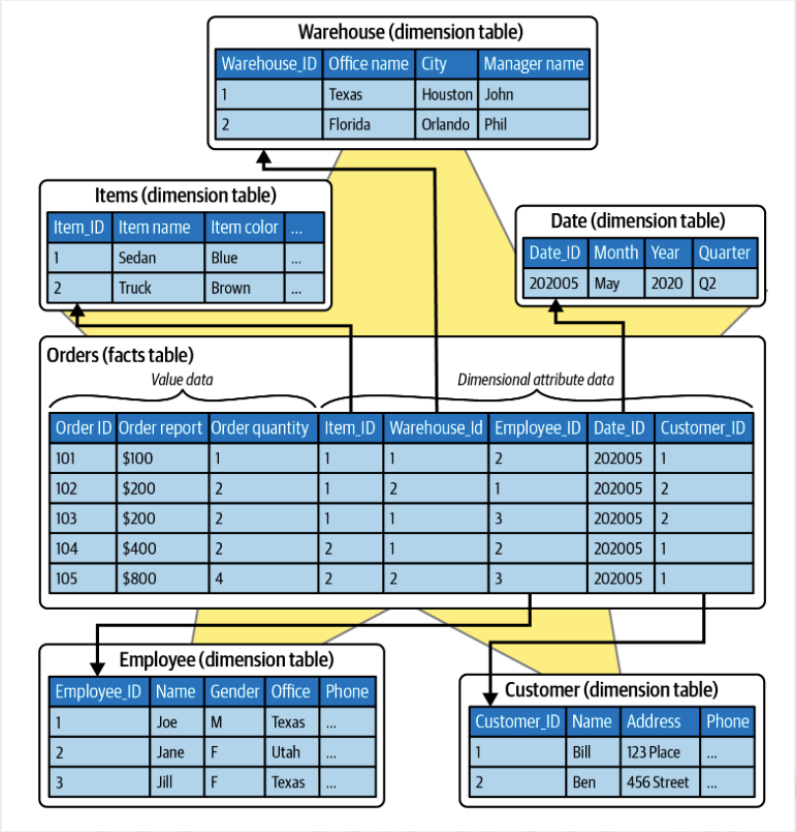
Historical Analysis | Structured Data | Descriptive Analytics | Diagnostic Analytics

It is usually designed as a **focused repository** of data that is optimized to meet the specific needs of a department or a business line within an organization (like finance, human resources, or marketing). Because of its narrower scope, a data mart can provide users with a more streamlined and accessible view of information than a DW can.



Data Warehouse Modeling

Star Schema



Data Lake

Raw Data

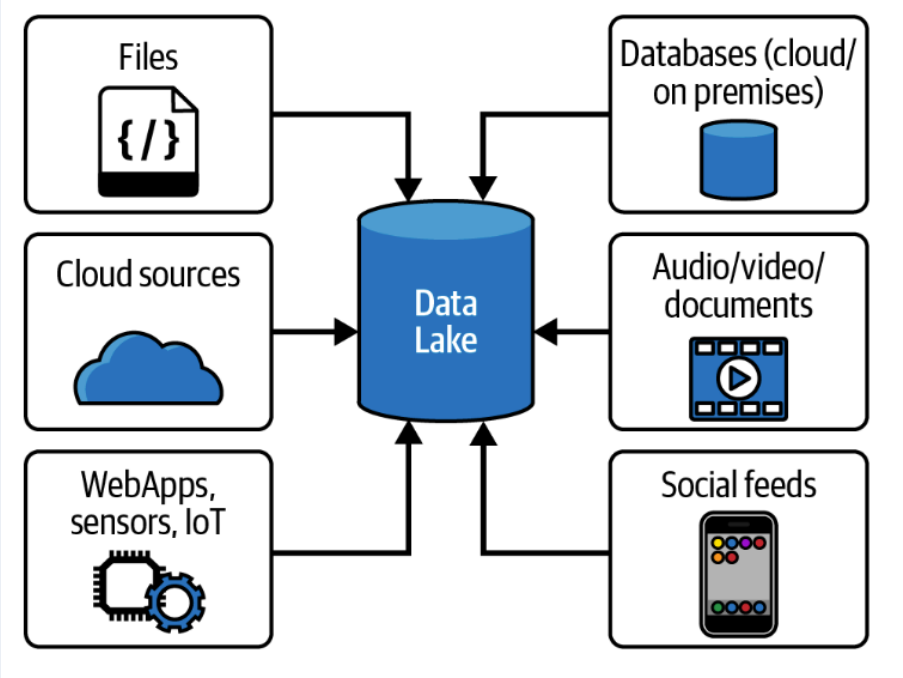
Structured Data

Unstructured Data

Predictive Analytics

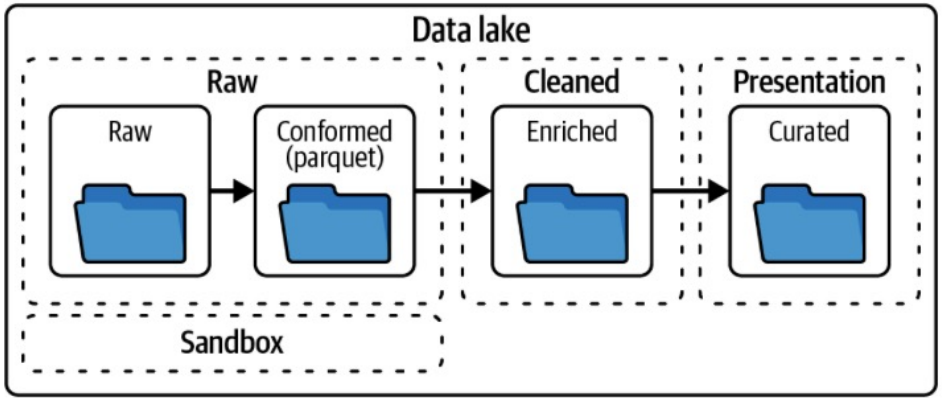
Prescriptive Analytics

Schema On-Read



Data Lake – Best Practices

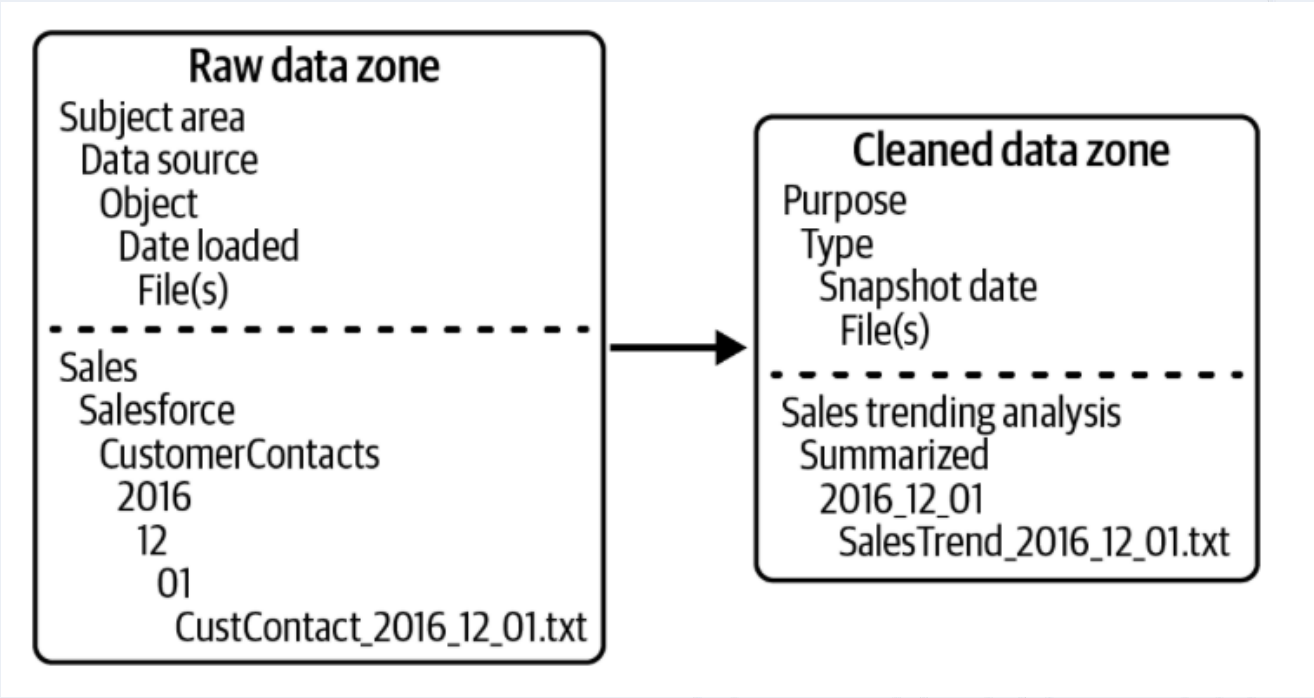
Schema On-Read



- **Raw Layer**
  - Immutable data store for historical reference – Raw data “as is”
- **Conformed Layer**
  - Example, file types converted into Parquet file format
- **Cleansed Layer**
  - Data is cleaned, integrated, and consolidated into consumable datasets
- **Presentation Layer**
  - Business logic applied to clean data which can involve aggregations or summarization

Data Lake – Example Folder Structure

Schema On-Read



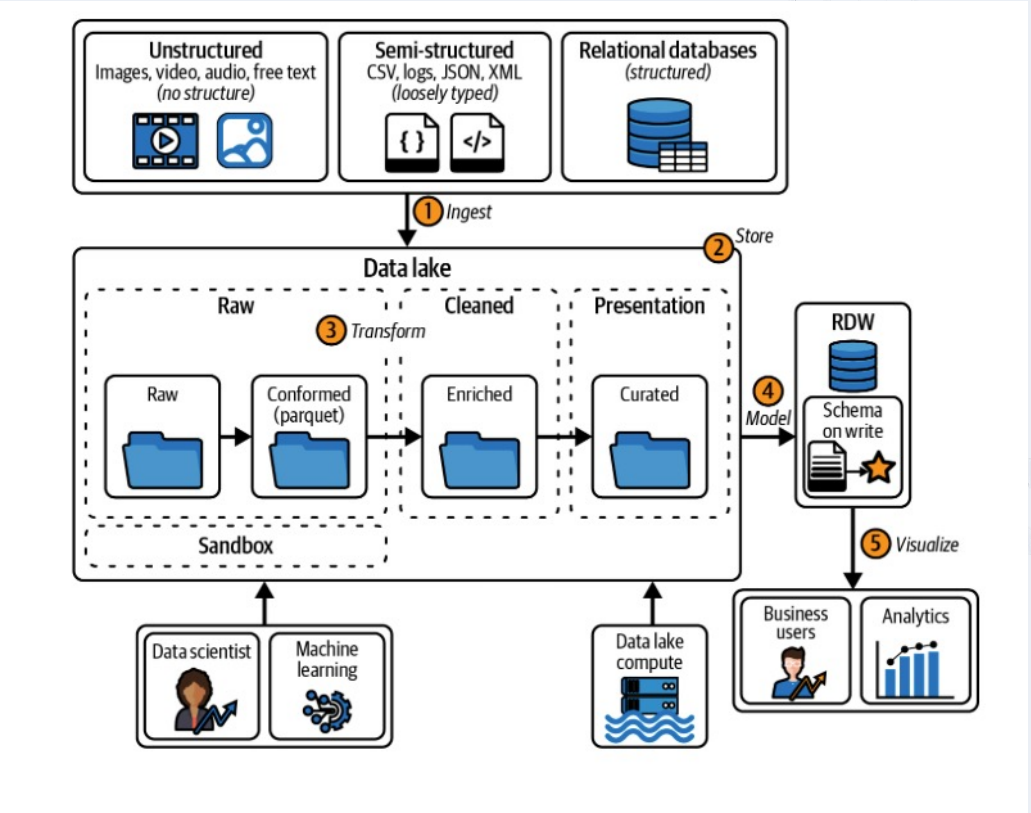
Data Lake – Journey into Modern Data Architecture

Benefits

- Integration of multiple data sources
- Scalability
- Real-time analytics
- Improved performance
- Flexibility

Drawbacks

- Complexity
- Initial high cost
- Technical skills required
- Vendor dependency (if using cloud)



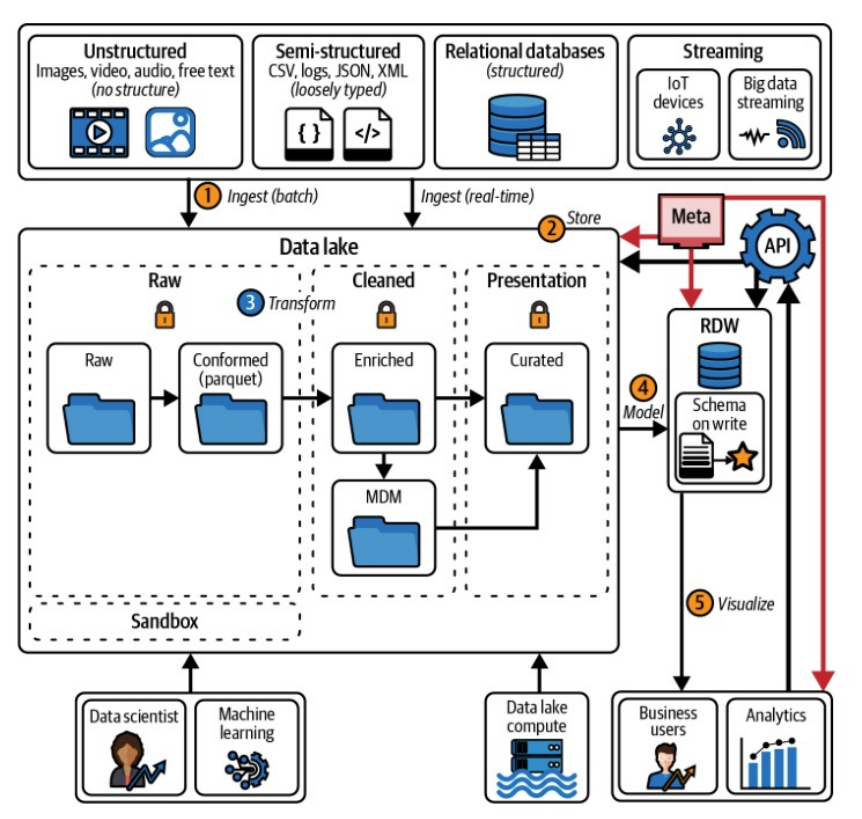
Data Fabric Architecture

It adds

- Data Access Policies
- Metadata Catalog
- Master Data Management
- APIs

Drawback

- Can be resource intensive to build this
  - Cost
  - Training
  - Integration



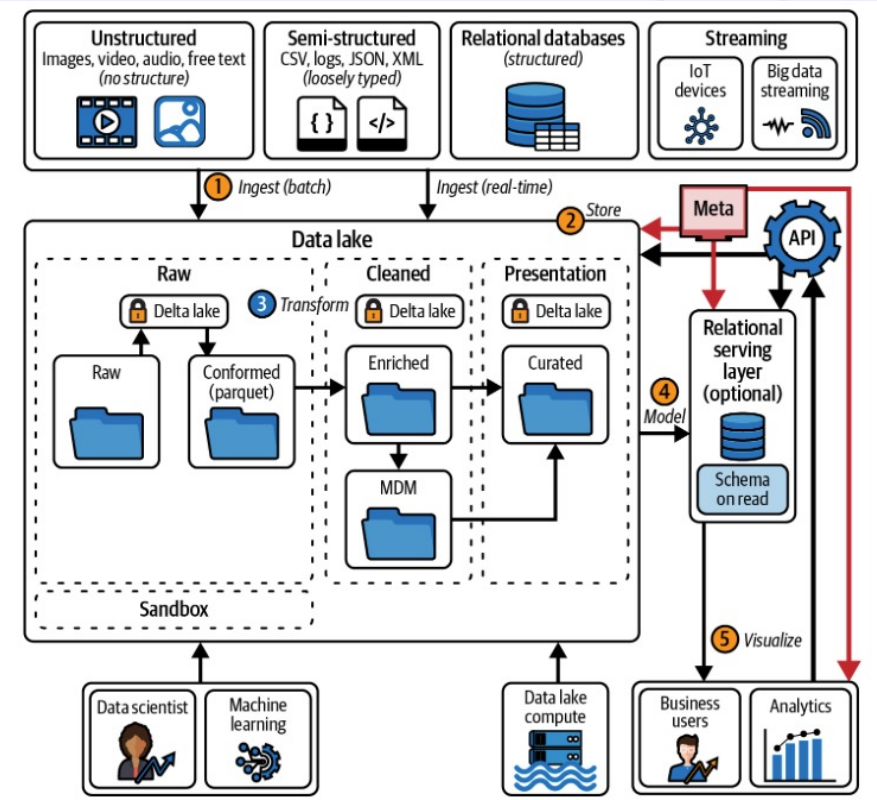
Data Lakehouse Architecture

It adds

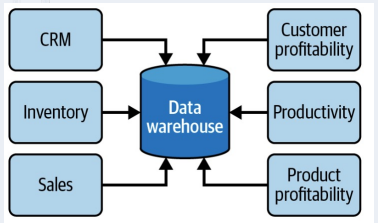
- Transaction Support (ACID)
- Time Travel
- Caching
- Query optimization

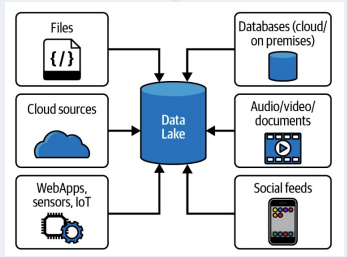
Drawback

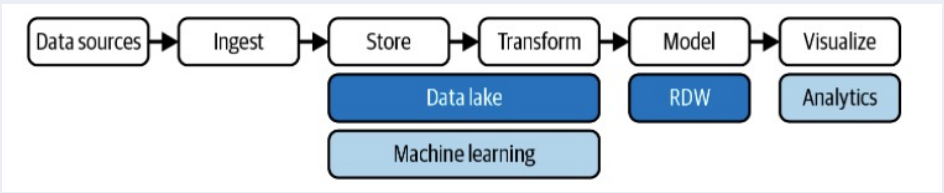
- Can be resource intensive to build this

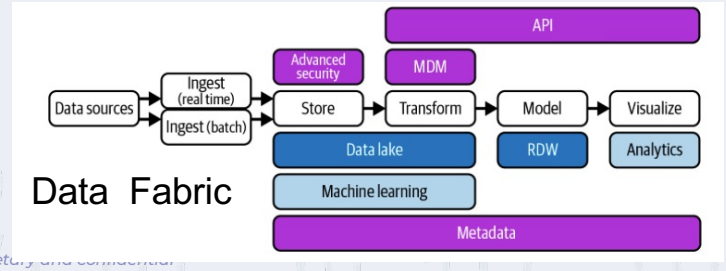


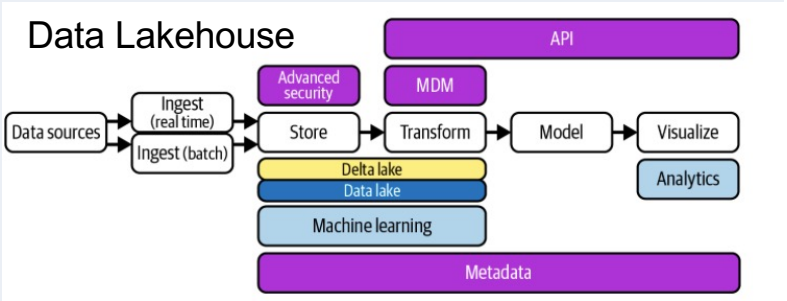
Evolution Summary











Common File Formats

CSV (Comma-Separated Value)

Description:

Text-based format that represents data in a tabular form where each line corresponds to a row and values within a row are separated by commas.

When to Use:

- For small to medium datasets.
- For data interchange between systems with different technologies.
- For human-readable and editable data storage.
- Importing/Exporting data from databases or spreadsheets.

Systems:

Databases (SQL-based), Excel, Pandas in Python, R, and many ETL tools

CSV (Comma-Separated Value)

csv

Copy code

```csv
id,name,age,email
1,John Doe,29,john.doe@example.com
2,Jane Smith,34,jane.smith@example.com
```

XML (Extensible Markup Language)

Description:

Markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It uses tags to define data structures and is highly flexible.

When to Use:

- For data interchange in complex systems where data validation and structure are crucial.
- For configuration files and metadata storage.
- In systems where data needs to be both human-readable and easily processed by machines.
- For documents that require a strict hierarchical structure.

Systems:

Web services (SOAP), document storage systems, configuration management systems, data interchange between enterprise applications.

XML (Extensible Markup Language)

xml Copy code

```xml
<records>
  <record>
    <id>1</id>
    <name>John Doe</name>
    <age>29</age>
    <email>john.doe@example.com</email>
  </record>
  <record>
    <id>2</id>
    <name>Jane Smith</name>
    <age>34</age>
    <email>jane.smith@example.com</email>
  </record>
</records>
```

JSON (JavaScript Object Notation)

Description:

Lightweight, text-based, and human-readable data interchange format that represents structured or semi- structured data based on key-value pairs.

When to Use:

- Data interchange between a web server and a web client.
- Configurations and settings for software applications.
- Use cases that need a flexible schema or nested data structures.

Systems:

Web browsers, many programming languages (like JavaScript, Python, Java, etc.), RESTful API, NoSQL Databases (like MongoDB)

JSON (JavaScript Object Notation)

json Copy code

```
[
  {
    "id": 1,
    "name": "John Doe",
    "age": 29,
    "email": "john.doe@example.com"
  },
  {
    "id": 2,
    "name": "Jane Smith",
    "age": 34,
    "email": "jane.smith@example.com"
  }
]
```

AVRO

Description:

Binary format that stores both the data and its schema, allowing it to be processed later with different systems without needing the original system's context.

When to Use:

- With big data and real-time processing systems.
- When schema evolution (changes in data structure) is needed.
- Efficient serialization for data transport between systems.

Systems:

Apache Kafka, Apache Spark, Apache Flink, Hadoop ecosystem.

AVRO

Schema

json Copy code

```
{
  "type": "record",
  "name": "User",
  "fields": [
    {"name": "id", "type": "int"},
    {"name": "name", "type": "string"},
    {"name": "age", "type": "int"},
    {"name": "email", "type": "string"}
  ]
}
```

Data

json Copy code

```
[
  {"id": 1, "name": "John Doe", "age": 29, "email": "john.doe@example.com"},
  {"id": 2, "name": "Jane Smith", "age": 34, "email": "jane.smith@example.com"}
]
```

Parquet

Description:

Columnar storage format optimized for analytics. Allows for efficient compression and encoding schemes.

When to Use:

- Analyzing large datasets with analytics engines.
- Use cases where reading specific columns instead of entire records is beneficial.
- Storing data on distributed systems where I/O operations and storage need optimization.

Systems:

Hadoop ecosystem, Apache Spark, Apache Hive, Apache Impala, Amazon Redshift Spectrum.

ORC (Optimized Row Columnar)

Description:

Columnar storage format designed to improve the storage and processing efficiency of data. It is optimized for read-heavy operations and supports complex data types.

When to Use:

- For storing and analyzing large datasets in distributed systems.
- When you need efficient compression to save storage space and improve I/O performance.
- In use cases where you need fast read and scan performance, especially for columnar data processing.
- For data warehousing and big data analytics scenarios.

Systems:

Hadoop ecosystem, Apache Hive, Apache Spark, Apache HBase, Amazon Athena.

Avro, ORC, and Parquet

Feature	Avro	Parquet	ORC
Data Format	Row-oriented	Columnar	Columnar
Schema Evolution	Excellent	Good	Good
Compression	Snappy, Deflate, Bzip2	Snappy, Gzip, LZO	Zlib, Snappy
Read Performance	Good for all rows	Excellent for reading specific columns	Excellent for reading specific columns
Write Performance	Excellent	Good	Good
Support for Complex Data Types	Yes	Yes	Yes
Use Cases	Data serialization, streaming data	Analytical queries, BI tools	Analytical queries, BI tools
Schema Storage	Included in file	Separate schema file	Separate schema file
Interoperability	Good (used in Hadoop ecosystem, Kafka)	Good (used in Hadoop ecosystem, Presto, Spark)	Good (used in Hadoop ecosystem, Hive, Spark)
Tooling and Libraries	Avro libraries available for most languages	Supported by major big data tools	Supported by major big data tools
File Size	Larger due to row-oriented storage	Smaller due to columnar compression	Smaller due to columnar compression
Suitability for Small Files	Good	Not optimal	Not optimal
Complex Queries	Moderate	Excellent	Excellent
Splitable	Yes	Yes	Yes

SQL Databases

Database Ranking

https://db-engines.com/en/ranking

- Complete ranking
- Relational DBMS
- Key-value stores
- Document stores
- Time Series DBMS
- Graph DBMS
- Search engines
- RDF stores
- Object oriented DBMS
- Vector DBMS
- Wide column stores
- Multivalue DBMS
- Spatial DBMS
- Native XML DBMS
- Event Stores
- Content stores
- Navigational DBMS

421 systems in ranking, June 2024

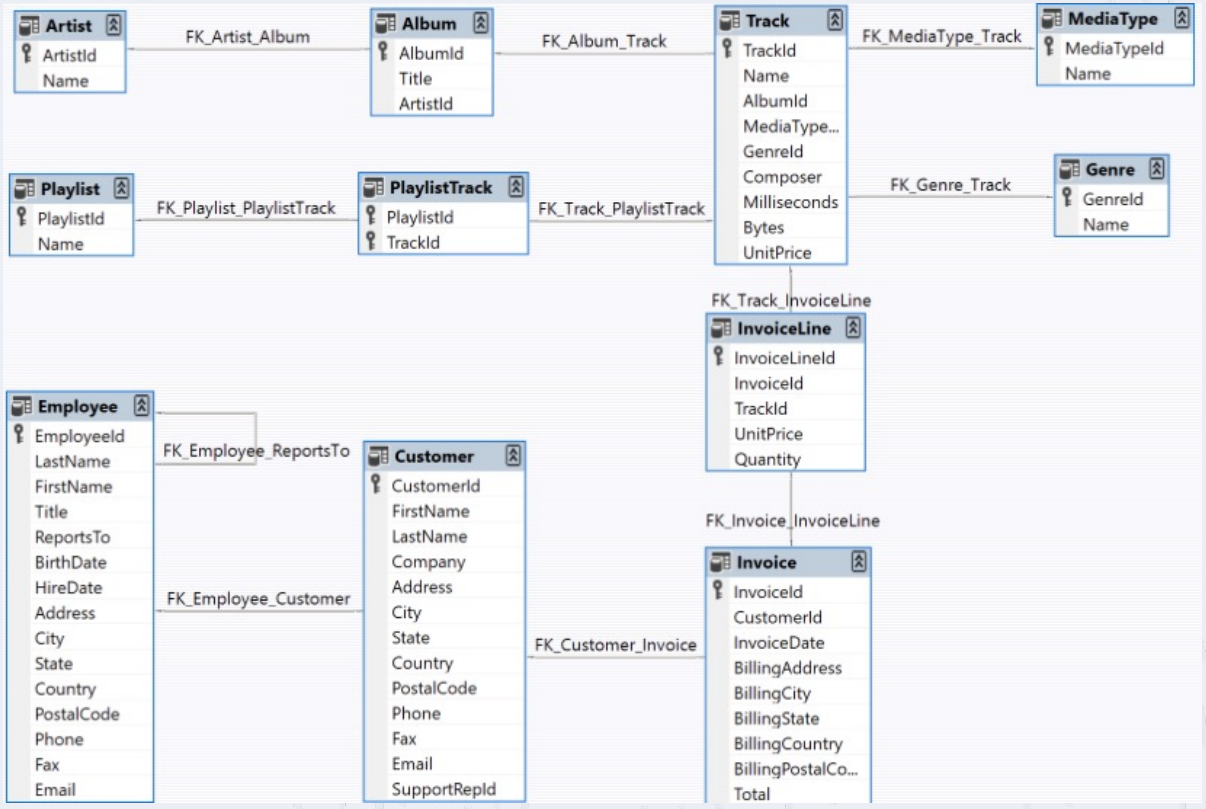
Rank			DBMS	Database Model	Score		
Jun 2024	May 2024	Jun 2023			Jun 2024	May 2024	Jun 2023
1.	1.	1.	Oracle	Relational, Multi-model	1244.08	+7.79	+12.61
2.	2.	2.	MySQL	Relational, Multi-model	1061.34	-22.39	-102.59
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	821.56	-2.73	-108.50
4.	4.	4.	PostgreSQL	Relational, Multi-model	636.25	-9.30	+23.43
5.	5.	5.	MongoDB	Document, Multi-model	421.08	-0.58	-4.29
6.	6.	6.	Redis	Key-value, Multi-model	155.94	-1.86	-11.41
7.	7.	↑ 8.	Elasticsearch	Search engine, Multi-model	132.83	-2.52	-10.92
8.	↑ 9.	↑ 11.	Snowflake	Relational	130.36	+9.03	+16.23
9.	↓ 8.	↓ 7.	IBM Db2	Relational, Multi-model	125.90	-2.56	-18.99
10.	10.	10.	SQLite	Relational	111.41	-2.91	-19.81

Relational Databases

- A relational database management system (RDBMS) is a software layer of tools and services that manages relational tables. In practice, the terms RDBMS and relational database are considered to be synonyms. A relational database provides a consistent interface between applications, users, and relational database. Organizations use RDBMS for managing large amounts of business critical information from various departments.
- Multiple users can work with the same database in different ways. For example, they can perform database operations and aggregate key data points without creating data redundancy. Relational database management systems also give your database administrator greater access control over your data.

Relational Databases - ERDs

Entities, their data types, and relationships are all illustrated in the diagram.



How Relational Databases Work

Data Model

- **Tables:** Represent real-world objects or concepts (entities).
- **Attributes:** Columns in a table holding specific kinds of data.
- **Fields:** Store the actual values of attributes.
- **Primary Key:** Unique identifier for each row in a table.
- **Foreign Key:** References the primary key of another table, creating logical connections between tables.
- **Example:** An orders table can have a foreign key (customer ID) linking to the customer table.

How Relational Databases Work

SQL (Structured Query Language)

- **Primary Interface:** Used to communicate with relational databases.
- **ANSI Standard:** Became a standard in 1986, supported by all popular relational database engines.
- **Functions:** Used to update, delete, store, retrieve data, and manage the database.
- **Ease of Use:** Uses common English keywords and integrates well with programming languages like Java.

How Relational Databases Work

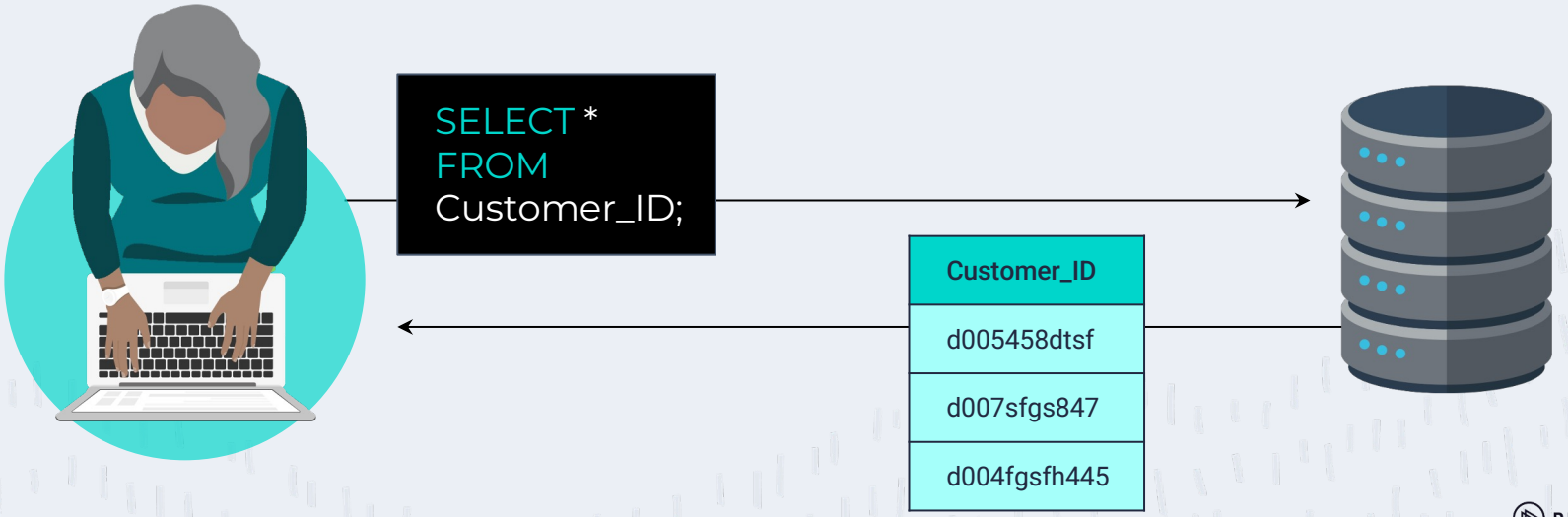
Transactions

- **Definition:** One or more SQL statements forming a single logical unit of work.
- **All-or-Nothing:** Entire transaction must complete, or none of the components go through.
- **Outcome:** Results in a COMMIT (successful) or a ROLLBACK (unsuccessful).
- **Properties:** Treated coherently, reliably, independently, and isolated from other transactions.

SQL

SQL (often pronounced "sequel") stands for Structured Query Language.

It is a powerful tool that enables programmers to create, populate, manipulate, and access databases. It also provides an easy method for dealing with server-side storage.



SQL

Data using SQL is stored in tables on the server, much like spreadsheets you would create in Microsoft Excel.

This makes the data easy to visualize and search.

Customer_ID	Date_ID
d005458dtsf	6/26/2019
d007sfgs847	8/3/2018
d004fgsfh445	12/3/2018

Order_ID	Customer_ID	Date_ID
10001	d005458dtsf	6/26/2019
10002	d007sfgs847	8/3/2018
10003	d004fgsfh445	12/3/2018

CRUD Operations

Create **R**ead **U**pdate **D**elete is a set of operations used with persistent storage.

Create	INSERT INTO table (column1, column2, column3)
Read	SELECT * FROM table
Update	UPDATE table SET column1 = VALUE WHERE id = 1
Delete	DELETE FROM table WHERE id = 5

These tools are fundamental to all programming languages, not just SQL.

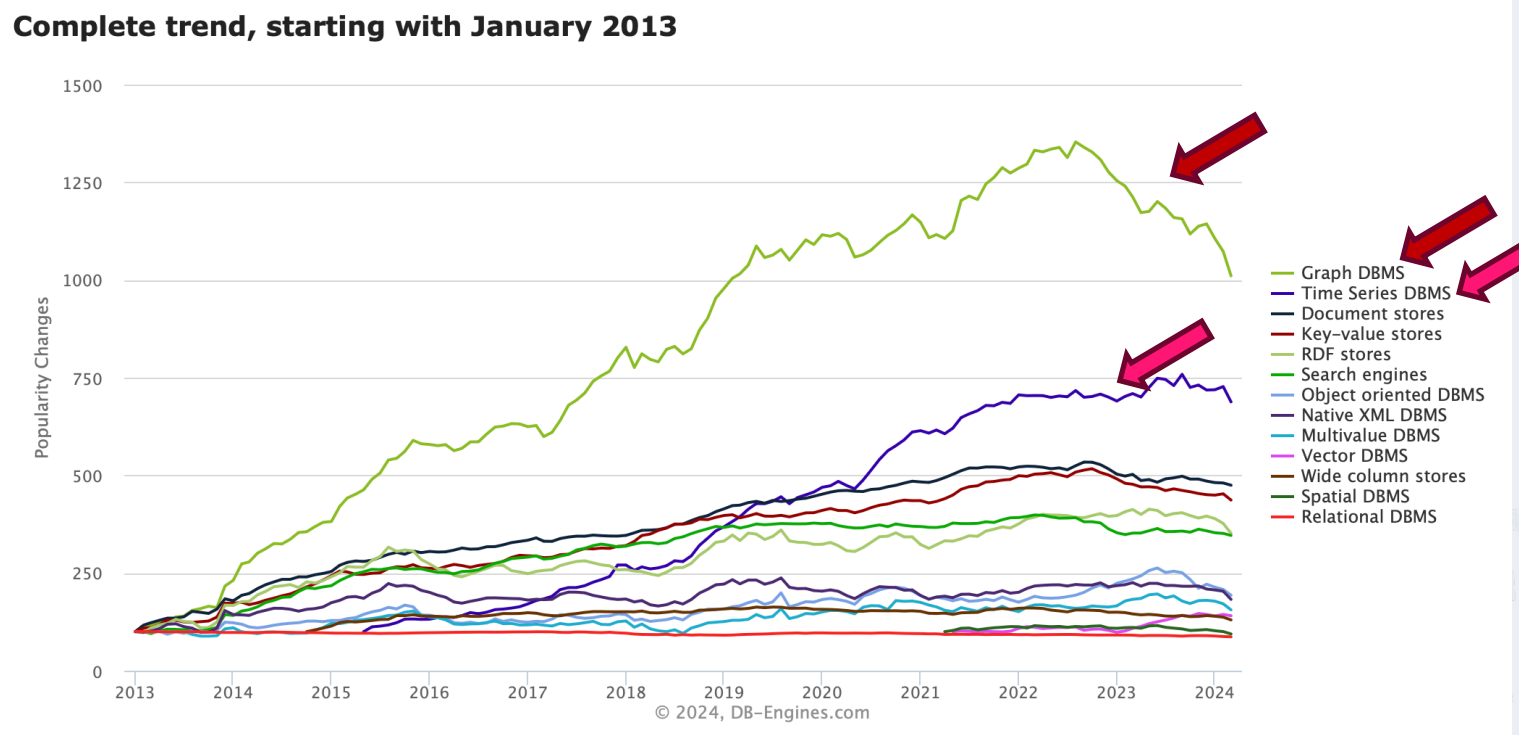
SQL JOINS

`INNER JOIN`	Returns records that have matching values in both tables.
`LEFT JOIN`	Returns all records from the left table and the matched records from the right table.
`RIGHT JOIN`	Returns all records from the right table and the matched records from the left table.
`CROSS JOIN`	Returns records that match every row of the left table with every row of the right table. This type of join has the potential to make very large tables.
`FULL OUTER JOIN`	Places null values within the columns that do not match between the two tables, after an inner join is performed.

NoSQL Databases

Database Trends

https://db-engines.com/en/ranking_categories

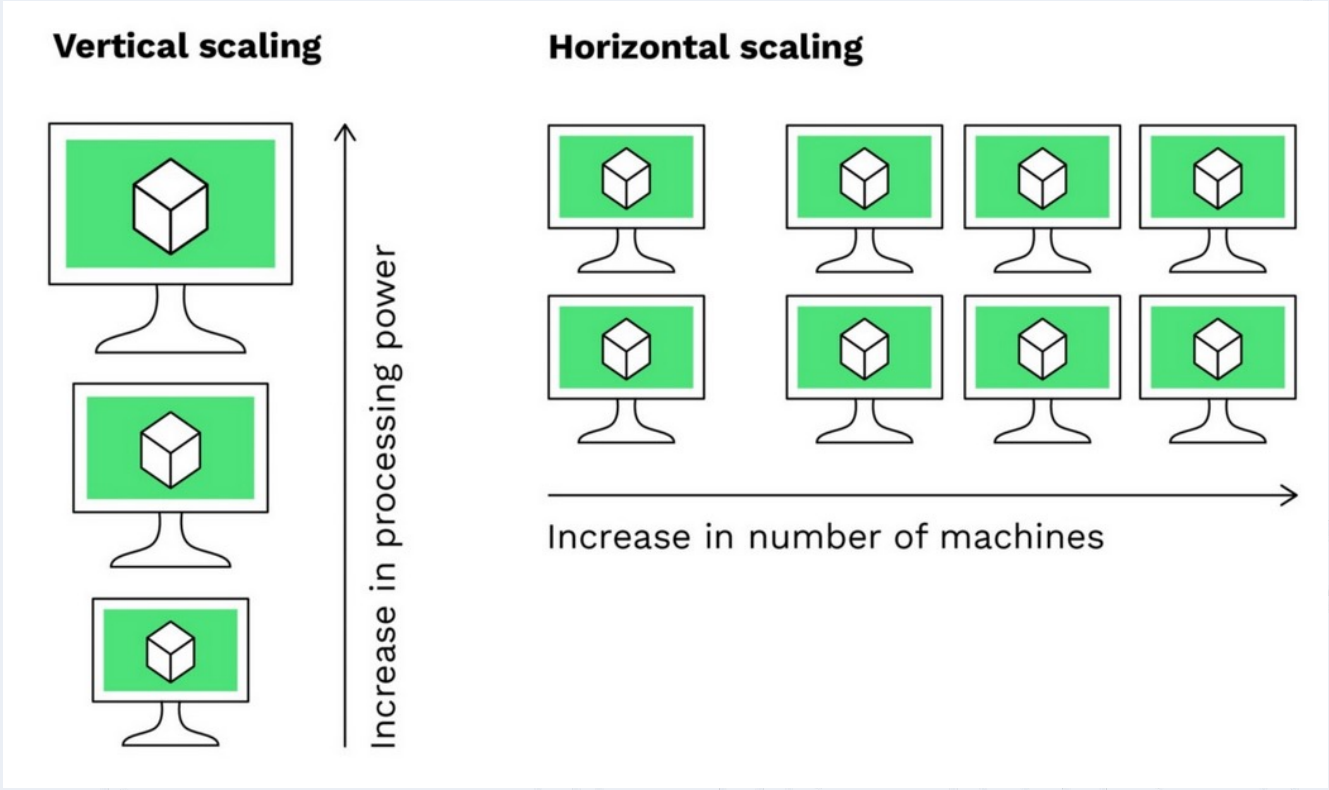


NoSQL Databases

Key Features of NoSQL Databases

1. **Flexible Data Models**: NoSQL databases store data in various formats such as JSON, XML, and key-value pairs, which allows for more flexible schema design and easier adaptation to changing data structures.
2. **Scalability**: NoSQL databases are designed for **horizontal scaling**, which enables them to handle large volumes of data and high traffic without significant performance degradation.
3. **Simple Design**: NoSQL databases often have simpler designs compared to relational databases, making them easier to implement and manage.
4. **High Availability**: NoSQL databases are designed to provide high availability and reliability by replicating data across multiple nodes and ensuring that data is accessible even if some nodes go offline.
5. **Support for SQL-like Query Languages**: Many NoSQL databases support SQL-like query languages, making it easier for developers familiar with SQL to work with these databases

Scalability



SQL vs NoSQL Terminology

SQL	MongoDB	DynamoDB	Cassandra	Couchbase
Table	Collection	Table	Table	Data bucket
Row	Document	Item	Row	Document
Column	Field	Attribute	Column	Field
Primary key	ObjectId	Primary key	Primary key	Document ID
Index	Index	Secondary index	Index	Index
View	View	Global secondary index	Materialized view	View
Nested table or object	Embedded document	Map	Map	Map
Array	Array	List	List	List

Types of NoSQL Databases

Document Databases

- Store data in flexible, semi-structured documents, often in JSON or XML format.
- Allow for nested data structures and flexible schemas, making them well-suited for handling unstructured and semi-structured data.
- Examples include MongoDB and Couchbase.

Key-Value Stores

- Store data as simple key-value pairs, where the value can be anything from a simple string to a complex object.
- Provide fast read/write operations and are optimized for simple data access.
- Examples include Redis and Memcached.

Types of NoSQL Databases

Column-Oriented Stores

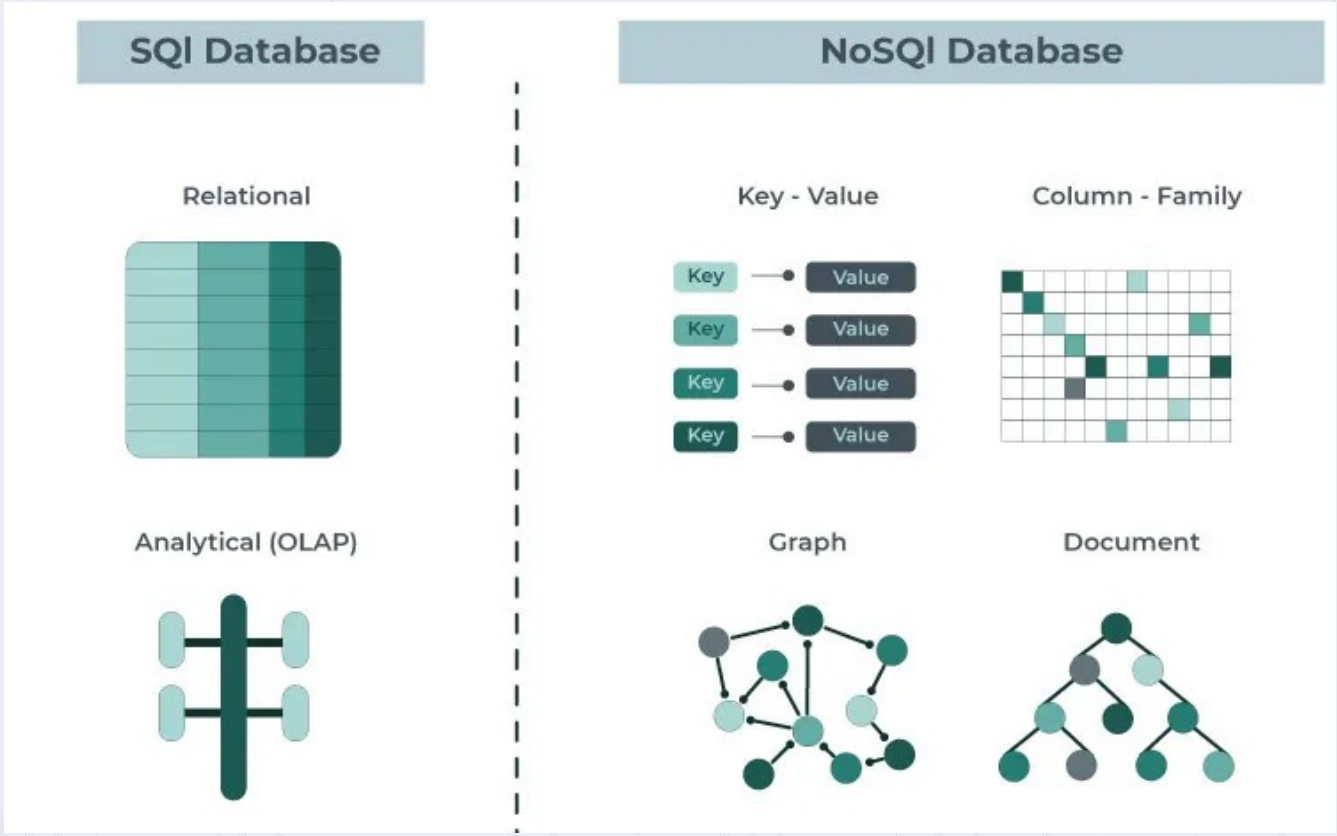
- Store data in columns instead of rows, which allows for efficient querying and analysis of large datasets.
- Organize data into column families, which are sets of columns treated as a single entity.
- Examples include Cassandra and HBase.

Graph Databases

- Focus on the relationships between data, storing it in the form of nodes and edges.
- Designed to handle complex, interconnected data and are well-suited for applications that require traversing and analyzing relationships.
- Examples include Neo4j and Amazon Neptune.

abases:

SQL vs NoSQL



Thank you!

If you have any additional questions, please reach out to me at: (email address).