Microsoft

Introduction to Database Systems

What Is a Database?

Database is an organized collection of data, stored electronically, designed for efficient storage, retrieval, and management of information

A **database** is an **organized collection of data**, stored electronically. It is designed to help users **store**, **search**, **update**, and **manage information** efficiently.

In Simple Terms:
Think of a database as a **digital version of a well-organized cupboard**. Each drawer (table) stores related items (data), and you can easily open, search, or rearrange them whenever needed.

Databases in a Connected World:

- A modern database is **not limited to one computer**.
- Data can come from **multiple computers**, devices, or servers connected through a **network or the internet**.
- This means people across the world can **access and update shared data** in real time.

Example:

- When you **order food online**, your request is stored in a database.
- The kitchen sees it immediately, even if you’re far away.
- The delivery app also pulls data from that same database to show your order status.

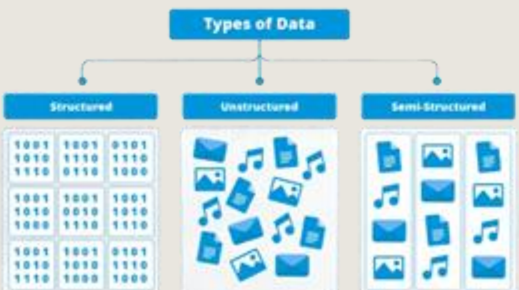
How Databases Help People Globally:

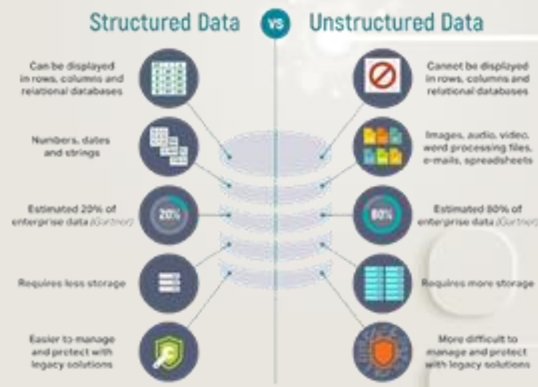
- **Centralized Access:** Data stored in one place can be accessed from anywhere
- **Real-time Sharing:** Multiple people or systems can update/view the data at the same time

- **Examples in Real Life:**
    - Google Drive files (stored in cloud databases)
    - Bank transactions across cities or countries
    - Social media platforms like Instagram or TikTok
    - Airline booking systems used by agents worldwide

What Is a Data?

Data is a collection of facts, figures, numbers, or observations that can be processed and analyzed to gain insights or make decisions.





What is Data?

- **Data** refers to raw facts and figures that by themselves may not have much meaning.
- It can be **numbers**, **text**, **images**, **videos**, **sensor readings**, or **events**.
- Once processed or organized, data can turn into **information** that helps us **understand**, **analyze**, or **make decisions**.

Examples:

- A list of students and their marks
- A log of temperature readings from a sensor
- A file containing customer reviews
- A spreadsheet of sales numbers

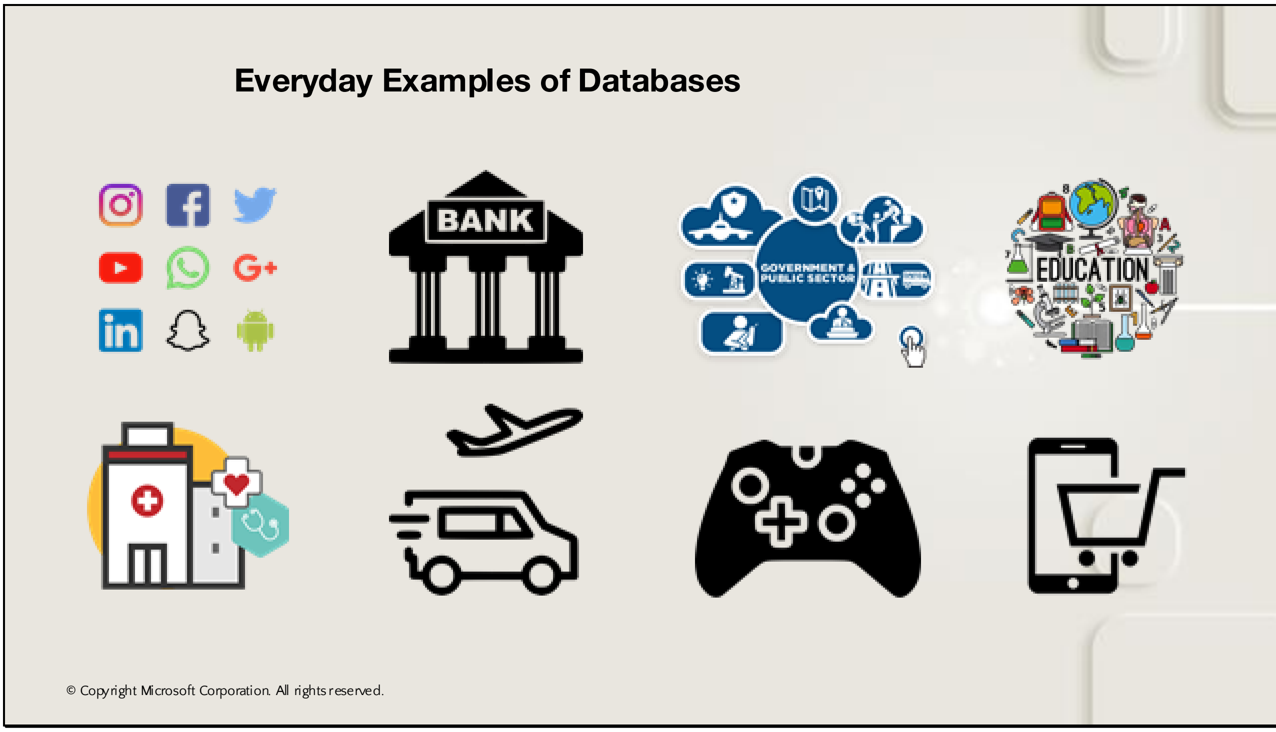
Types of Data:

1. **Structured Data**
   - Data that follows a fixed format or structure
   - Stored in tables (rows and columns), like in Excel or SQL databases
   - Example: Employee records (Name, ID, Department, Salary)

1. **Unstructured Data**
    - Data without a predefined format
    - Includes emails, images, videos, social media posts
    - Example: A folder full of resume PDFs or customer support chats
2. **Semi-Structured Data**
    - Data that is partially organized (has tags or markers)
    - Example: JSON, XML, or HTML documents used in web or API data

Why Data Matters:

- Organizations use data to **track performance**, **predict trends**, and **make smart decisions**.



We use databases **all the time** — often without realizing it!

Social Media Platforms

- Facebook, Instagram, TikTok: Databases store your posts, comments, likes, and friend lists.
- They also help recommend content by analyzing your behavior.

Banks & Finance

- Databases keep track of your account balance, transactions, loan history, and credit scores.
- ATMs and mobile apps use real-time database queries.

Government Systems

- CNIC/NADRA records, passport systems, and vehicle registration databases.
- Voting systems and taxation records are also stored and validated via databases.

Education

- University portals store student profiles, grades, attendance, and course materials.
- Online learning platforms like Coursera, Udemy, and LMS systems all rely on databases.

Healthcare

- Patient records, prescriptions, lab results, and doctor schedules are stored securely.
- Hospitals use databases for billing, insurance, and history tracking.

Travel & Transport

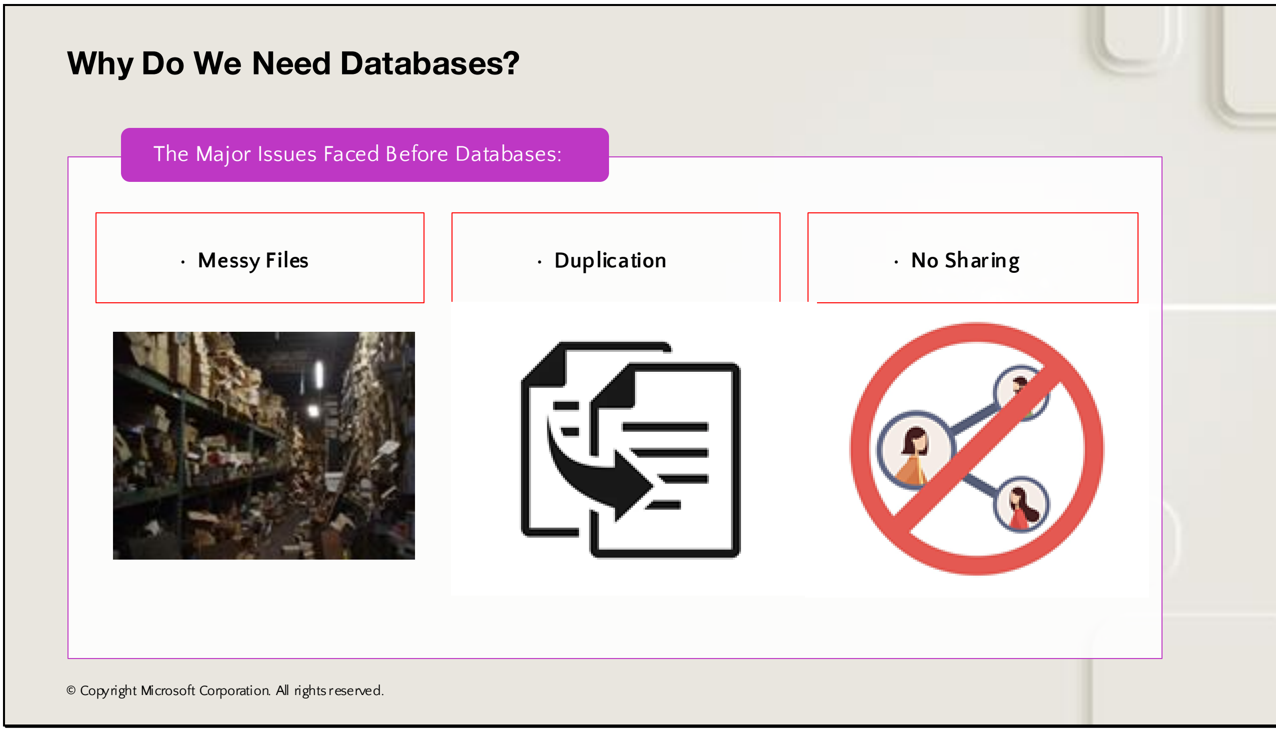
- Airlines use databases to manage flight bookings, ticketing, and seat reservations.
- Ride-hailing services like Uber or Careem track driver/passenger data and routes.

Games

- Online games use databases to store player progress, scores, in-game purchases, and leaderboards.

E-Commerce & Retail

- Websites like Daraz, Amazon, and Alibaba store product listings, customer data, orders, and reviews.
- Databases also power recommendations and inventory management.



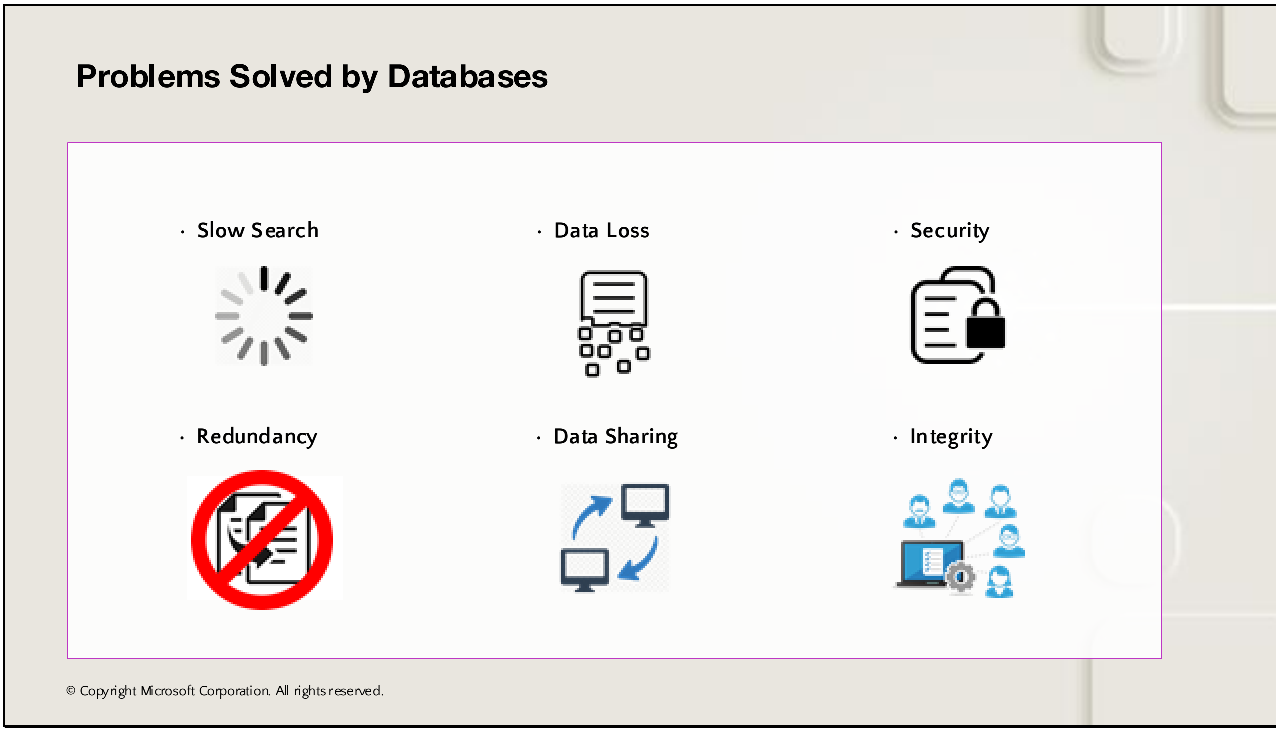
Example: Town-Wide School Exam – Paper-Based

Imagine it's 1960. You've conducted exams across 10 schools in your town. All answer sheets are paper-based and now stored in a big warehouse.

Major Problems Faced:

1. **Messy and Unorganized Files**
   - No standard format.
   - Some papers are missing names, others are mixed with wrong subjects.
   - Sorting them manually takes weeks.
2. **No Way to Detect Duplicates**
   - A student may submit two answer sheets with different names.
   - There's no automatic system to cross-check or validate data entries.
3. **Very Difficult to Share**
   - If someone in another town (or country) wants to help grade or audit:
     - You'll need to ship physical papers.
     - Risk of damage or loss in transit.
     - No real-time collaboration is possible.

1. **Slow Access and Retrieval**
    ○ Want to find a specific student's paper? It could take hours or days.
    ○ Manual search is inefficient and prone to human error.
2. **No Way to Analyze Results**
    ○ Calculating averages, pass/fail rates, or top performers is slow and error-prone.
    ○ You need to manually enter data into ledgers or typewriters.
3. **Low Data Security**
    ○ Anyone with physical access to the warehouse can steal, edit, or destroy records.
    ○ No encryption, no backup, no audit trail.



How Databases Solved the Exam Chaos

Imagine again you're running a town-wide exam, but **this time using a database system** instead of storing physical papers in boxes.

Problems Solved by Databases:

1. **Fast Search**
    - Want to find "Sara Ahmed's" Math exam? Just search by her name or roll number.
    - Results appear in **seconds**, not hours.
2. **No Redundancy or Duplicates**
    - The database prevents two students from using the same roll number.
    - It can **automatically detect** and block duplicate records.
3. **Improved Security**
    - Only authorized people can access or edit exam data.
    - Data is **password protected**, **encrypted**, and **backed up** regularly.
4. **Easy Sharing**
    - An examiner in another town or country can **instantly access** the same exam data online.

- No shipping, no delay, no paper getting lost.

1. **Data Integrity**
    - Everyone accessing the database sees **exactly the same records**.
    - If one person updates the score, the change is **logged and reflected everywhere**.
2. **Protection from Data Loss**
    - Even if the main computer crashes, data backups ensure nothing is lost.
    - No fear of fire, theft, or water damage like physical records.

Encryption

- **Symmetric Encryption**
  - One key is used for both encryption and decryption
  - Fast and efficient
  - Example algorithms: AES, DES
  - Challenge: Securely sharing the key
- **Asymmetric Encryption**
  - Uses a pair of keys: Public and Private
  - Data encrypted with one key can only be decrypted with the other
  - Example algorithms: RSA, ECC
  - Commonly used in secure communications like HTTPS

Asymmetric Encryption

- Asymmetric encryption is a type of encryption that uses **two different keys**:
  - **Public Key**: Shared with everyone.
  - **Private Key**: Kept secret by the owner.
  - These keys are mathematically linked, but you can't easily figure out the private key from the public one.
- **How It Works**
  - **Encrypting**: If someone wants to send you a secure message, they use your **public key** to encrypt it.
  - **Decrypting**: You use your **private key** to decrypt the message.
    - Only the person with the private key can decrypt and see the original message

Database Systems

- Database Systems are generally implemented as server applications, waiting for client connections on a specific port
- **Port 1433** is the **default communication port** used by **Microsoft SQL Server** when it talks to other computers over a network.
- SQL Server listens for incoming requests on port 1433.
  - When a client (like a computer running SQL Server Management Studio or an application) wants to connect to the database, it sends the request to the IP address of the server running SQL Server, along with this port.
- SQL Server can be configured with certificates to encrypt all communication between client and server

In network programming, a **port** is like a **channel or doorway** through which data travels between computers. Each application or service on a computer listens on a specific port number to send and receive data.

- **Microsoft SQL Server** is a database system that allows applications to store and retrieve data. When an application wants to connect to SQL Server over a network, it needs to know **where to send the request**. That's where **port 1433** comes in.
- **Port 1433** is the **default TCP port** that SQL Server uses to listen for incoming connections.
- It's part of the **TCP/IP protocol**, which is the foundation of most network communication.

How It Works in Network Programming

- **Client Application** (e.g., a web app or desktop app) wants to connect to SQL Server.
- It uses **network programming libraries** (like sockets in C#, Java, Python, etc.) to open a connection.
- The connection is directed to the **IP address of the server** and **port 1433**.
- SQL Server receives the request on port 1433, processes it, and sends back the response.

Microsoft

A Quick Timeline of Database History

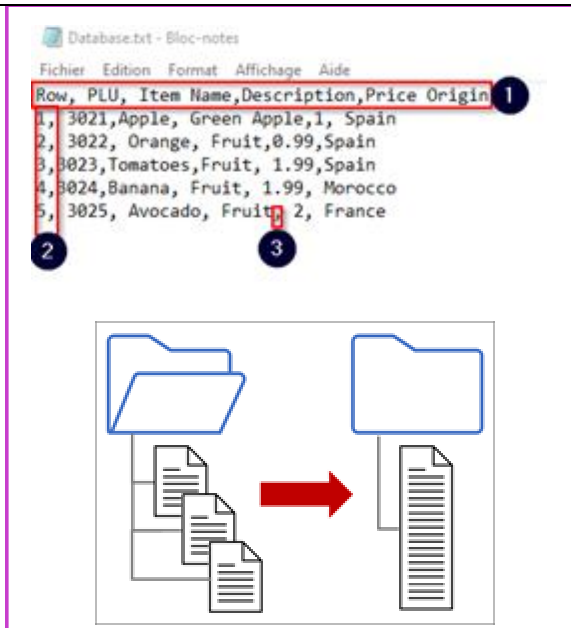
1960s: Flat Files

What were Flat Files?

- Data stored in plain text files (.txt, .csv).
- Each record was a line; each field separated by commas or tabs.

Limitations:

- No relationships between data.
- No efficient way to search or update data.
- A change in one file meant manually updating every related file (very error-prone!).



Example:

A company might keep separate files for *Employees*, *Departments*, and *Salaries* — with no connection between them.

1970s: The Relational Model Concept

An IBM researcher, **Dr. Edgar F. Codd,** proposed a new way to store and access data in 1970. Dr. Codd's model is the reason databases became structured and scalable. He called it "The Concept of Relational Models"

What is the Relational Model?

- Data is stored in tables (rows and columns)
- Tables can be linked with each other (will be discussed later how that is done)

Impact:

- Made data easier to manage and query.
- Ensured data remained consistent
- Led to the development of SQL (Structured Query Language), which is the language to work with relational data.

2000s: SQL Everywhere

Rise of Relational Database Management Systems (RDBMS):

- SQL became the standard way to query and manipulate data.
- Database Systems like **Oracle**, **SQL Server**, **MySQL** and **PostgreSQL** dominated the industry.

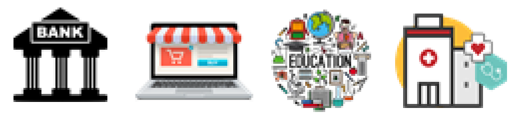
Used in:

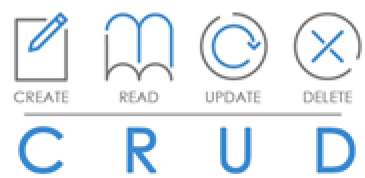
- Banking,
- E-commerce,
- Universities,
- Hospitals

— almost every serious system used SQL.

Why it mattered:

- SQL was powerful, easy to learn, and great for structured data.
- CRUD (Create, Read, Update, Delete) became standard database operations.





2010s: Big Data & NoSQL Era

What changed?

- Rapid growth of Social media, smartphones, and IoT devices created massive amounts of **unstructured data**.

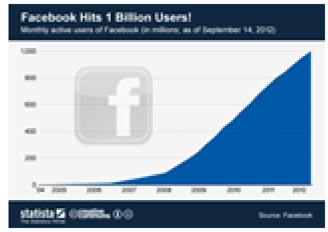
NoSQL vs SQL:

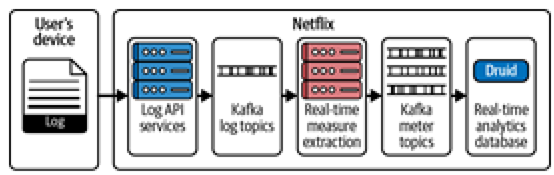
- **NoSQL** (Not Only SQL) databases like **MongoDB**, **Cassandra**, and **Redis** emerged.

- These are better for semi-structured data like JSON, documents, and graphs.

Use Cases:

- Real-time analytics
- Chat apps
- Content feeds
- Large-scale logging systems





Real-World Databases

MySQL

Firebase

Microsoft SQL Server

ORACLE

SQLite

MariaDB

mongoDB

Appendix

Supplemental Reading – Not covered in Quizzes/Exams

Data Related Industry Roles – Who does What with our Data:

In one simple line:

We have “Real-Life Applications” and “Database Management Tools” with many different “Types of Databases” containing “Professional Contexts” and “Personal Contexts” that we want to use with “Efficiency” and “Innovation” to help in good “Decision Making”.

What Are We Trying to Do with Data?

- Solve real-life problems
- Improve decision-making
- Innovate with personal & professional applications
- Use different **Database Tools** & **Database Types** efficiently
- Build systems that are **secure, fast, and reliable**

The team that makes it possible consists of the following roles:

Leadership Roles

They guide data strategy and drive innovation

- **Head of Data / CDO (Chief Data Officer)**

- Oversees all data activities, policies, and governance
  Focus: Alignment with business goals

- **IT Leadership / Decision Makers**
  Make investment decisions for database tools and technologies

Technical Skills Roles

They build, maintain, and test the databases

- **Database Administrator (DBA)**
  Manages database health, backups, and security

- **Database Developer**
  Designs tables, writes queries, builds stored procedures

- **Database Tester**
  Tests queries, database behavior, and performance

Analytical Roles

They turn raw data into insights

- **Data Analyst**
  Cleans, explores, and visualizes data
  Helps in reporting and informed decision-making

"Superhero" Roles

They rescue failing systems and consult across industries

- **Database Consultant**
  Works with companies to fix or redesign database systems
  Combines business knowledge with database expertise

Analytical + Creative + Communication Roles

They bridge the gap between business needs and database systems

- **Database Modeler**
  Designs the structure of data (ER diagrams, relationships)

- **Database Engineer**
  Builds systems to support large-scale data flows

- **Database Architect**
  Designs high-level systems: scalability, performance, integration

Database Architect

What they do:

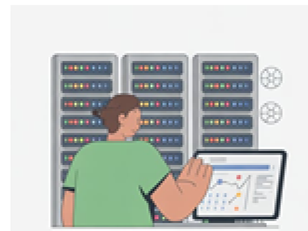
- Design how data is structured across the system.
- Decide on tables, relationships, keys, indexing strategies.
- Plan for performance, scalability, and security.

Real-World Example:
At a bank, a Database Architect would decide how to store customer accounts, transactions, and balances so the system is both fast and secure.

Database Administrator (DBA)

What they do:

- Manage and maintain databases after they've been built.
- Handle backups, security access, and performance tuning.
- Ensure the database is always running, even during failures.

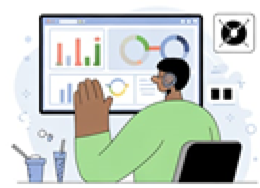


Real-World Example:
In a hospital, a DBA ensures that patient records are always accessible and backed up daily to avoid data loss.

Data Analyst

What they do:

- Use data to generate insights and reports.
- Run SQL queries to answer business questions.
- Build dashboards and data visualizations.



Real-World Example:
A retail store analyst checks sales trends to help managers decide which products to promote next month.

Data Engineer

What they do:

- Build pipelines that collect, clean, and organize data.
- Set up systems to move data from one system to another.
- Work closely with analysts and data scientists.

Real-World Example:
At Netflix, data engineers move viewing data from user devices into big data storage for analysis.

Backend Developer

What they do:

- Write the code that connects apps to the database.
- Create APIs and logic that apps use to store or retrieve data.
- Ensure apps run fast and securely.

Real-World Example:
A developer builds the login system of an app that checks a user's credentials against the database.