Below are 10 business scenarios. Your task is to select the most suitable database type for each scenario from the given options and provide a short justification for your choice.

Database Options:
Relational Database
NoSQL: Document Store
NoSQL: Key-Value Store
NoSQL: Column Store
NoSQL: Graph Database

Scenarios :

1. Student Enrollment System

A university needs to track student enrollments, courses, and grades while ensuring data integrity and relationships between students and courses.

2. Library Book Catalog

A library needs a system to store book details, authors, and availability, allowing structured queries for book searches.

3. E-commerce Website Product Catalog

An online store needs to store and manage products with varying attributes (e.g., different clothing sizes, colours, descriptions, and user reviews).

4. Real-Time Price Tracking System for Crypto Exchange

A cryptocurrency exchange needs to store and retrieve constantly fluctuating price data while ensuring high-speed access.

5. Friendship & Follower System in a Social Media App

A social media platform must manage friendships, followers, and mutual connections efficiently.

6. Customer Session Management for an Online Store

An e-commerce platform needs to store user session data, including cart contents, temporary preferences, and browsing history.

7. Fraud Detection in Online Banking Transactions

A banking system needs to detect suspicious transactions by analysing connections between accounts and transaction histories.

8. Recommendation Engine for a Movie Streaming Service

A streaming service like Netflix wants to recommend movies based on user preferences, viewing history, and similar user behaviour.

9. Log Analysis for Cybersecurity Monitoring

A company needs to analyse millions of system logs daily to detect unusual patterns or security threats in real time.

10. Supply Chain Management for a Global Retailer

A multinational retailer needs to track product shipments, supplier relationships, and warehouse inventories across different locations.

Answers:

Scenario	Database Type	Reasoning
1	Relational Database	Structured relationships between students, courses, and grades require data integrity and normalisation.
2	Relational Database	If storing structured data like books, authors or borrowers, an RDBMS works best.
3	NoSQL: Document Store or Relational Database	A Document Store is best for handling varying product attributes, while an RDBMS works if the product structure is fixed.
4	NoSQL: Key-Value Store or NoSQL: Column Store	A cryptocurrency exchange deals with real-time price updates and historical price analysis. It needs a high-performance database that can handle millions of price updates per second while allowing users to access historical trends for market analysis. Key-value Stores offer fast lookups of real-time price updates. Traders and analysts need to analyse price trends over days, weeks, and months; Column Stores efficiently handle historical price analytics.
5	NoSQL: Graph Database	Friendship and follower networks form a graph-like structure where users are connected via relationships. Unlike relational databases that rely on foreign keys and joins, a Graph Database directly stores relationships, making queries like "Who are my mutual friends?" much faster. Queries like "Find all mutual friends of Alice and Bob" are instantaneous, whereas an RDBMS would require multiple expensive JOIN operations.

6	NoSQL: Key-Value Store	An online store needs to manage customer session data, including shopping carts, browsing preferences, and login sessions. Since session data is temporary, frequently accessed, and requires fast retrieval, a NoSQL Key-Value Store is the best choice.
7	NoSQL: Graph Database or Relational Database	Graph databases excel at handling data where relationships between entities are key. In the case of fraud detection, the connections between accounts, transactions, and users can be modelled as a graph. For example, an account can be linked to other accounts it interacted with through transactions. RDBMS ensures data integrity and consistency throughout. This is important when tracking transactions because you need to ensure the accuracy of financial data. If the transactions are straightforward, such as debit and credit logs with clearly defined relationships between accounts and amounts, an RDBMS can work well. Fraud detection might involve comparing transaction data across accounts for anomalies, which is possible in a relational model.
8	NoSQL: Graph Database or NoSQL: Document Store	Graph Database is excellent when the recommendation system relies on complex relationships and connections between users, movies, and other entities for example: genres, actors, or directors. Graph databases allow you to analyse how users are connected to other users or movies. A Document Store (e.g., MongoDB) is great for handling flexible, semi-structured data. In this case, a document can represent a movie, a user, or a collection of user preferences. The data can be stored in a JSON-like format, making it easy to store various attributes such as movie titles, genres, ratings, or a list of movies a user has watched.
9	NoSQL: Column Store or NoSQL:	A Column Store is optimised for handling large volumes of data with a schema that can evolve

	Document Store	over time. It is particularly useful for storing and analysing time-series data like logs, where data is organised by columns rather than rows. Each log entry can be stored with a key and a set of associated columns that hold the log details like timestamp, event type, IP address, user, action. A Document Store is optimised for handling semi-structured data, where each log entry is stored as a document. Document Stores provide flexibility in terms of schema design, allowing logs to be stored in a more flexible, key-value format with varying attributes like timestamp, source, message, severity.
10	NoSQL: Document Store or Relational Database	Relational Databases (e.g., MySQL, PostgreSQL) are well-suited for managing structured data with defined relationships. In a supply chain system, there are usually clear and well-defined entities, such as suppliers, products, shipments, warehouses, and inventories. These entities are interconnected, and the data is often normalised into different tables (e.g., Suppliers, Products, Shipments, Inventory, Orders), with relationships between them represented by foreign keys. A Document Store (e.g., MongoDB) is suitable for handling semi-structured or flexible data. In a global supply chain system, different product categories, warehouses, or suppliers may have different attributes, making a schema-less, flexible storage solution like a Document Store more adaptable. With a Document Store, each entity (e.g., a product, order, or shipment) can be stored as a document (typically in JSON or BSON format), allowing for flexibility in handling product information, shipment details, or customer data without rigid schema constraints.

1. **Student Enrollment System**

NoSQL: Document Store – Student records have a fixed structure, making a schema-based relational database better.
NoSQL: Key-Value Store – The data is too complex for simple key-value pairs.
NoSQL: Column Store – Not optimised for transactional operations.
NoSQL: Graph Database – Not needed unless tracking complex relationships like alumni networks.

2. **Library Book Catalog**

NoSQL: Key-Value Store – Searching for books by different attributes is inefficient.
NoSQL: Column Store – Designed for analytics, not transactional operations.
NoSQL: Graph Database – Unnecessary unless modelling complex book recommendations.

3. **E-commerce Website Product Catalog**

NoSQL: Key-Value Store – Products need filtering and sorting, which key-value stores don't handle well.
NoSQL: Column Store – Not suitable for hierarchical product data.
NoSQL: Graph Database – Not needed unless modelling complex product recommendations.

4. **Real-Time Price Tracking for Crypto Exchange**

Relational Database – Not optimised for high-speed reads and writes.
NoSQL: Document Store – JSON storage is inefficient for real-time updates.
NoSQL: Graph Database – Price data doesn't involve complex relationships.

5. **Friendship & Follower System in a Social Media App**

Relational Database – Joins on large user datasets slow down queries.
NoSQL: Document Store – Poor at handling highly interconnected data.
NoSQL: Key-Value Store – Relationships between users are too complex for key-value pairs.
NoSQL: Column Store – Not designed for network-based relationships.

6. **Customer Session Management for an Online Store**

Relational Database – Overhead is too high for temporary data.
NoSQL: Document Store – Overkill for simple session storage.
NoSQL: Column Store – Not designed for fast session retrieval.
NoSQL: Graph Database – No need for complex relationships.

7. **Fraud Detection in Online Banking Transactions**

NoSQL: Document Store – JSON is not efficient for querying connections.
NoSQL: Key-Value Store – No ability to analyse patterns in transaction data.
NoSQL: Column Store – Not designed for fraud detection queries.

8. **Recommendation Engine for a Movie Streaming Service**

Relational Database – Joins slow down recommendations on large datasets.
NoSQL: Key-Value Store – Cannot analyse user behaviour patterns.
NoSQL: Column Store – Not optimised for real-time recommendations.

9. **Log Analysis for Cybersecurity Monitoring**

Relational Database – Querying large log data is slow.
NoSQL: Key-Value Store – Logs need filtering and analysis, not just lookups.
NoSQL: Graph Database – Not needed unless analysing user attack paths.

10. Supply Chain Management for a Global Retailer

NoSQL: Key-Value Store – Poor for managing complex inventory relationships.
NoSQL: Column Store – Not ideal for transactional updates.
NoSQL: Graph Database – Not needed unless mapping complex supplier relationships.