1. What is database normalization?

Normalization is a process of organizing the columns and tables of a relational database to minimize data redundancy and improve data integrity.

1. Why is normalization important?

Normalization is crucial for several reasons that contribute to a healthy and efficient database;

1. Reduce data redundancy: By eliminating duplicate data, its saves storage space and prevents inconsistencies that can arise when the same data is stored in multiple places.
2. Improve data integrity: Ensures that data is consistent and accurate across the database. When data changes, it only needs to be updated in one place.
3. Enhances Data Consistency: Prevents update, insertion, and deletion anomalies, which are problems that occur when data is not properly organized.
4. Simplifies Queries: While it might seem counterintuitive due to more joins, a normalized schema often makes it easier to write clear and specific queries because each piece of information has a designated, logical home.
5. Better Database Design: Promotes a cleaner, more logical, and more scalable database structure.

Imagine a school database. If you store student details (name, address, phone) directly in a Courses table for every course a student takes, you'll repeat the student's name and address multiple times. If the student moves, you'd have to update their address in many places, risking errors. Normalization would put student details in a separate Students table, and the Courses table would just have a StudentID to link to it.

Types of normal forms (1NF, 2NF, 3NF, BCNF)?

Normalization is defined through a series of "normal forms." Each normal form builds upon the previous one, adding stricter rules to reduce redundancy and improve data integrity. The most commonly discussed normal forms are:

**What is Denormalization?**

**🔹 English:**

**Denormalization** is the process of combining normalized tables into a single table to improve read performance, even if it introduces some redundancy

**Example:**

**Normalized:**

**Students Table:**

| **StudentID** | **Name** | **DeptID** |
| --- | --- | --- |
| 1 | Rakib | CSE |

**Departments Table:**

| **DeptID** | **DeptName** |
| --- | --- |
| CSE | Computer Science |

👉 Student এর ডিপার্টমেন্ট নাম জানতে JOIN করতে হবে।

**Denormalized:**

| **StudentID** | **Name** | **DeptID** | **DeptName** |
| --- | --- | --- | --- |
| 1 | Rakib | CSE | Computer Science |

👉 এখন JOIN ছাড়াই সব তথ্য একসাথে — Faster Read, কিন্তু DeptName বারবার repeat হচ্ছে।

**✅ Difference between Normalization and Denormalization**

| **Feature** | **Normalization** | **Denormalization** |
| --- | --- | --- |
| 🔍 Purpose | Reduce redundancy, ensure consistency | Improve read/query performance |
| 📦 Data Storage | Less storage, compact | More storage, duplicate data |
| 🔁 Joins | More joins needed | Fewer joins |
| ⚠️ Redundancy | Redundancy removed | Redundancy introduced |
| 🧩 Data Integrity | High data integrity | Lower integrity (more chances of error) |

## Advantages of Normalization

1. 🔁 **Removes redundancy** — একি ডেটা বারবার রাখতে হয় না
2. ✅ **Improves data integrity** — ডেটা ভুল হবার সম্ভাবনা কমে
3. 📚 **Easy to update & maintain** — একটা জায়গা থেকে সব আপডেট করা যায়
4. 💾 **Saves storage** — অপ্রয়োজনীয় ডেটা রিমুভ হয়
5. 🔍 **Better organization** — ডেটার গঠন থাকে পরিপাটি

## ✅ Disadvantages of Normalization

1. 🔀 **More complex queries** — অনেক JOIN করতে হয়
2. 🐌 **Slower reads** — ডেটা পড়তে সময় লাগে
3. 👨‍💻 **Hard for beginners** — নতুনদের বুঝতে অসুবিধা হতে পারে
4. 💢 **Difficult for reporting** — রিপোর্ট তৈরি করতে বেশি table access করতে হয়