**Database:**

A database is a structured and organized collection of data where you don’t store data in files that is hectic and hard to manage.

**Anomalies**

Sure Talha, here are **3 real-world anomalies** in a **ride-sharing app** database (e.g., MySQL or PostgreSQL) if normalization isn’t handled properly:

**1. Insertion Anomaly**

**Problem:** Can’t add a new driver unless they’ve completed at least one ride.

**Example Table:**  
DriverRides(driver\_id, driver\_name, ride\_id, ride\_status)

**Issue:**  
If a new driver hasn't done any rides, you **can’t insert** their name without a dummy ride ID.

✅ **Fix:** Separate into Drivers and Rides tables.

**2. Update Anomaly**

**Problem:** Driver rating updated in one place, but not in another row.

**Example Table:**  
Rides(ride\_id, driver\_id, driver\_name, driver\_rating, ...)

**Issue:**  
If driver\_rating is updated for one ride, old rows still have outdated ratings.

✅ **Fix:** Move driver info to separate Drivers table, keep driver\_id as FK in Rides.

**3. Deletion Anomaly**

**Problem:** Deleting the last ride of a passenger also removes their contact info.

**Example Table:**  
PassengerRides(passenger\_id, name, contact, ride\_id)

**Issue:**  
Deleting all rides = you lose the **passenger record too**, even if they’re active.

✅ **Fix:** Normalize by having a Passengers table and Rides table.

**Normalization:**

Perfect, Talha. Let’s break down **normalization** in the **ride-sharing app** using 1NF, 2NF, 3NF, and 4NF—each with a **clear example**, **dependency explanation**, and **interview-ready points**.

**🔥 Goal of Normalization:**

Avoid **anomalies** (insert/update/delete), eliminate **redundancy**, and ensure **data integrity**.

**✅ 1NF (First Normal Form)**

**Rule:** Atomic values only (no repeating groups or arrays).

**❌ Bad (Unnormalized):**

DriverRides:

| ride\_id | driver\_id | driver\_name | stops |

|---------|-----------|-------------|-------------------|

| 101 | D01 | Ali | StopA, StopB, StopC |

* **Problem:** stops is a **multi-valued field**.
* **Dependency:** Partial dependence on ride\_id.

**✅ Fixed (1NF):**

RideStops:

| ride\_id | stop\_no | stop\_name |

|-------- |-------- |-----------|

| 101 | 1 | StopA |

| 101 | 2 | StopB |

| 101 | 3 | StopC |

📌 **Interview Tip:**  
“1NF ensures every column holds atomic data. In ride-sharing, stops per ride must be broken into rows, not kept as comma-separated values.”

**✅ 2NF (Second Normal Form)**

**Rule:** Must be in 1NF + No partial dependency (non-key column depends on part of a composite key).

**❌ Bad:**

RideDetails:

| ride\_id | driver\_id | driver\_name | fare |

|---------|-----------|-------------|------|

| 201 | D02 | Sara | 500 |

* Composite Key = (ride\_id, driver\_id)
* **Problem:** driver\_name depends only on driver\_id, not whole PK.

**✅ Fixed (2NF):**

* **Rides Table**: (ride\_id, driver\_id, fare)
* **Drivers Table**: (driver\_id, driver\_name)

📌 **Interview Tip:**  
“In 2NF, I split ride and driver data to remove partial dependency—especially in M:N relationships like rides and drivers.”

**✅ 3NF (Third Normal Form)**

**Rule:** Must be in 2NF + No transitive dependency (non-key depending on another non-key).

**❌ Bad:**

Driver:

| driver\_id | driver\_name | city\_id | city\_name |

|-----------|-------------|---------|-----------|

| D03 | Bilal | C01 | Lahore |

* **Transitive dependency**: city\_name → depends on city\_id, which depends on driver\_id.

**✅ Fixed (3NF):**

* **Drivers**: (driver\_id, driver\_name, city\_id)
* **Cities**: (city\_id, city\_name)

📌 **Interview Tip:**  
“3NF removes transitive dependencies—so city info should be in a separate table to avoid update anomalies.”

**✅ 4NF (Fourth Normal Form)**

**Rule:** No multi-valued dependencies (when one key maps to multiple independent attributes).

**❌ Bad:**

DriverSkillsVehicles:

| driver\_id | skill | vehicle\_type |

|-----------|---------------|--------------|

| D04 | First Aid | Car |

| D04 | Navigation | Bike |

* **Problem:** Skill and vehicle\_type are **independent**, but both repeat with driver\_id.

**✅ Fixed (4NF):**

* **DriverSkills**: (driver\_id, skill)
* **DriverVehicles**: (driver\_id, vehicle\_type)

📌 **Interview Tip:**  
“4NF separates independent multi-valued data—so skills and vehicles are stored without unnecessary duplication.”

**🧠 Summary Table**

| **Normal Form** | **Fixes** | **Key Interview Line** |
| --- | --- | --- |
| 1NF | Remove repeating groups | “Split comma-separated or nested fields into atomic rows.” |
| 2NF | Remove partial dependency | “Ensure non-key columns depend on the full primary key.” |
| 3NF | Remove transitive dependency | “Move dependent data (like city info) into separate tables.” |
| 4NF | Remove multi-valued dependency | “Separate independent lists like driver’s skills and vehicle types.” |