Normalization and Functional Dependencies

(a) Update Anomalies in the Given Table

The given table is susceptible to the following update anomalies:

1. Insertion Anomaly:
   - If a new contract is introduced, but no staff is assigned yet, we cannot store contract details.

2. Deletion Anomaly:
   - If we remove a staff member, we may lose information about a contract (e.g., contract C1025 might be lost if all staff leave).

3. Update Anomaly:
   - If a hotel changes its location, multiple rows must be updated. If any row is missed, inconsistency arises.

(b) Identifying Functional Dependencies

Assumptions:
- NIN uniquely identifies each staff member.
- Each contract is assigned to one or more staff.
- Each hotel has a unique hotel number (hNo) and location (hLoc).

Functional Dependencies:
1. NIN -> eName (Each NIN has a unique staff name)
2. contractNo -> hours (Each contract defines specific working hours)
3. hNo -> hLoc (Each hotel number is uniquely associated with one location)

(c) Normalization to 3rd Normal Form (3NF)

Step 1: 1NF (Eliminate Repeating Groups)
- The table is already in 1NF since all attributes have atomic values.

Step 2: 2NF (Remove Partial Dependencies)
- Remove dependencies where non-key attributes depend only on part of the primary key.
- Create new tables:

  1. Staff(NIN, eName)
  2. Contract(contractNo, hours)
  3. Hotel(hNo, hLoc)

4. Assignment(NIN, contractNo, hNo)

Step 3: 3NF (Remove Transitive Dependencies)

- hLoc is only dependent on hNo, so we ensure it is in a separate Hotel table.
- Final Relations:
  1. Staff(NIN, eName) - Primary Key: NIN
  2. Contract(contractNo, hours) - Primary Key: contractNo
  3. Hotel(hNo, hLoc) - Primary Key: hNo
  4. Assignment(NIN, contractNo, hNo) - Primary Key: (NIN, contractNo, hNo)