Solutions

Each answer shows the steps, then the **result**.

2.1

- any two of: data redundancy (same data in several files); data inconsistency (copies disagree); data dependence (programs tied to the file format); weak security/sharing/querying

2.2

- primary key = field(s) that **uniquely identify** each record
- foreign key = a field that refers to the **primary key of another table**

2.3

- a **thing** the database stores data about (e.g. a customer, a book) — one table per entity

2.4

- all values are **atomic** (single values) — no repeating groups in a field

3.1

(a) any two: redundancy; inconsistency; data dependence; poor sharing/security

(b) e.g. each fact is stored **once** in a table, so there is no redundancy/inconsistency

3.2

- record = one **row** (one instance of the entity)
- field = one **column** (one piece of data)
- primary key = field(s) that **uniquely identify** a record

3.3

- every non-key field depends on **the whole primary key** and on **nothing but the key**
- values are atomic (1NF) and there is no partial dependency (2NF)
- no non-key field depends on another non-key field (no transitive dependency)

3.4

- create a **link (junction) table** (e.g. `STUDENT_CLUB`)
- it holds a foreign key to `STUDENT` and a foreign key to `CLUB`
- each row records one student-in-one-club, so many-to-many becomes two one-to-many relationships

3.5

(a) every foreign key value must match an existing primary key in the related table (or be null)

- so the database cannot hold a reference to a record that does not exist

(b) a **one-to-many** relationship

- the foreign key (`CustomerID`) is held in the `ORDER` table — the "many" side