Assignment-1 (Theory)

Question 1:

Design a database for an online bookstore using ER model. The bookstore includes information about the books, author, publisher, and customer. Each book is represented by its ISBN number, title, year and price. Each book has a unique ISBN number. Author of the book is characterized by author_id, name and address. Each author has a unique author_id. Address of the author includes city, state, country and pin_code. The association of author and book is represented by the relationship named as written_by. One book may have more than one author. Many books written by same author is available in the store. The publisher of the Book is represented by its name, address, phoneno. Publishers are uniquely identified by its name. One publisher may have multiple phoneno.s. Address of the publisher includes city, state, country and pin_code. The association of publisher and book is represented by the relationship named as published_by. One book is published by exactly one publisher. The customer of the store is represented by its email, name, address, phoneno. Each customer should have one unique email. Address of the publisher includes city, state, country and pin_code. One phoneno. is included for each customer in the database. Customer has a shopping_basket. The shopping_basket is represented by its basket_id. Association between customer and shopping_basket is represented by the relationship named as basket_of. One customer has exactly one shopping_basket. The shopping basket may contain many books. Same book can be included in multiple shopping_baskets. Association between book and shopping_basket is represented by the relationship named as contains. When book is added to shopping_basket a number field associated with relationship contains is updated.

Draw the ER diagram for the above online bookstore representing entity set, relationship set, mapping cardinality and participation constraint.

Solution:

The task is to design a database for an online bookstore using the Entity-Relationship (ER) model . This involves identifying entities, attributes, relationships, mapping cardinalities, and participation constraints.

Step 1: Understanding the Requirements

The problem describes an online bookstore system with the following key components:

1. Books : Represented by ISBN, title, year, and price.
2. Authors : Represented by author_id, name, and address (city, state, country, pin_code).
3. Publishers : Represented by name, address (city, state, country, pin_code), and multiple phone numbers.
4. Customers : Represented by email, name, address (city, state, country, pin_code), and phone number.

5. Shopping Baskets : Represented by basket_id, associated with customers, and containing books.

Additionally, the relationships are described:

- written_by : Between Book and Author (many-to-many).
- published_by : Between Book and Publisher (one-to-many).
- basket_of : Between Customer and Shopping_Basket (one-to-one).
- contains : Between Shopping_Basket and Book (many-to-many, with an additional "number" attribute).

Step 2: Identifying Entities and Attributes

We identify the entities and their attributes based on the description:

1. Book :
    - Attributes: **ISBN** (Primary Key), **title**, **year**, **price**.
2. Author :
    - Attributes: **author_id** (Primary Key), **name**, **address** (composite attribute with sub-attributes: **city**, **state**, **country**, **pin_code**).
3. Publisher :
    - Attributes: **name** (Primary Key), **address** (composite attribute with sub-attributes: **city**, **state**, **country**, **pin_code**), **phoneno** (multi-valued attribute).
4. Customer :
    - Attributes: **email** (Primary Key), **name**, **address** (composite attribute with sub-attributes: **city**, **state**, **country**, **pin_code**), **phoneno**.
5. Shopping_Basket :
    - Attributes: **basket_id** (Primary Key).

Step 3: Identifying Relationships

We identify the relationships between entities and their characteristics:

1. written_by :
    - Between **Book** and **Author**.
    - Cardinality: Many-to-many (a book can have multiple authors, and an author can write multiple books).
    - Participation: Total participation on both sides (every book must have at least one author, and every author must have written at least one book).

2. published_by :
    - Between **Book** and **Publisher**.
    - Cardinality: One-to-many (one publisher can publish many books, but each book is published by exactly one publisher).
    - Participation: Total participation on the **Book** side (every book must be published by a publisher).
3. basket_of :
    - Between **Customer** and **Shopping_Basket**.
    - Cardinality: One-to-one (each customer has exactly one shopping basket).
    - Participation: Total participation on both sides (every customer must have a shopping basket, and every shopping basket must belong to a customer).
4. contains :
    - Between **Shopping_Basket** and **Book**.
    - Cardinality: Many-to-many (a shopping basket can contain many books, and a book can appear in multiple shopping baskets).
    - Additional Attribute: **number** (tracks the quantity of a book in a shopping basket).
    - Participation: Partial participation on both sides (a shopping basket may or may not contain any books, and a book may or may not be in any shopping basket).

Step 4: Drawing the ER Diagram

The ER diagram will include:

1. Entities as rectangles.
2. Attributes as ovals connected to entities.
3. Composite Attributes as ovals with sub-attributes.
4. Multi-valued Attributes as double ovals.
5. Relationships as diamonds.
6. Mapping Cardinalities and Participation Constraints indicated on the relationships.

Here's how the ER diagram would look:

Entities and Attributes :

1. Book :
    - Rectangle labeled "Book".
    - Ovals for **ISBN** (PK), **title**, **year**, **price**.
2. Author :

- Rectangle labeled "Author".
- Ovals for **author_id** (PK), **name**, and composite attribute **address** with sub-attributes (**city**, **state**, **country**, **pin_code**).

3. Publisher :
    - Rectangle labeled "Publisher".
    - Ovals for **name** (PK), composite attribute **address** with sub-attributes (**city**, **state**, **country**, **pin_code**), and multi-valued attribute **phoneno**.
4. Customer :
    - Rectangle labeled "Customer".
    - Ovals for **email** (PK), **name**, composite attribute **address** with sub-attributes (**city**, **state**, **country**, **pin_code**), and **phoneno**.
5. Shopping_Basket :
    - Rectangle labeled "Shopping_Basket".
    - Oval for **basket_id** (PK).

Relationships :

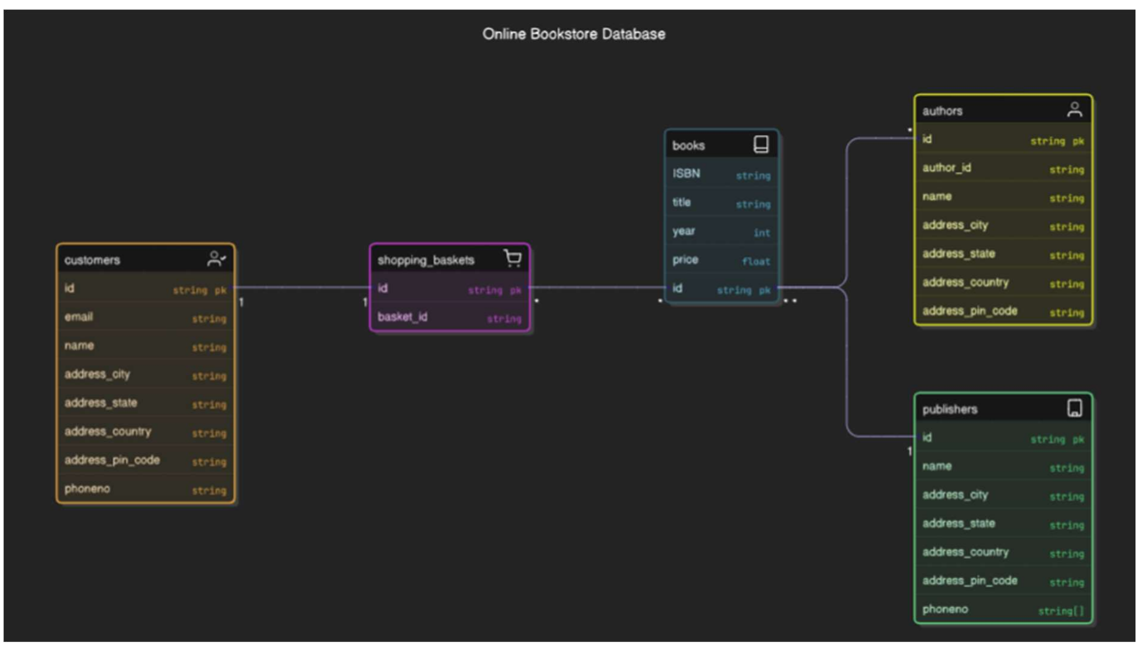
1. written_by :
    - Diamond labeled "written_by" connecting **Book** and **Author**.
    - Cardinality: Many-to-many.
    - Participation: Total on both sides.
2. published_by :
    - Diamond labeled "published_by" connecting **Book** and **Publisher**.
    - Cardinality: One-to-many.
    - Participation: Total on the **Book** side.
3. basket_of :
    - Diamond labeled "basket_of" connecting **Customer** and **Shopping_Basket**.
    - Cardinality: One-to-one.
    - Participation: Total on both sides.
4. contains :
    - Diamond labeled "contains" connecting **Shopping_Basket** and **Book**.
    - Cardinality: Many-to-many.
    - Additional Attribute: **number**.
    - Participation: Partial on both sides.

Step 5: Final Representation

The final ER diagram visually represents all the entities, attributes, relationships, cardinalities, and participation constraints as described above.

E-R Diagram:

Relational Schema diagram:



Question 2: Map the following ER diagram (Figure 1) to its corresponding relational schema. Also indicate the primary key and foreign key for the relational schema.

Step 1: Analyze the ER Diagram

The ER diagram represents a banking system with the following entities, relationships, and attributes:

1. **Entities** :
- Customer : Attributes: `cid` (Primary Key), `cname`, `phone`, `address`, `city`, `state`, `pincode`, `DOB`, `age()`.
- Account : Attributes: `accno` (Primary Key), `balance`.
- **Subtypes:**
- Saving_account : Attribute: `interest_rate`.
- Checking_account : Attribute: `overdraft_amount`.
- Loan : Attributes: `lno` (Primary Key), `lamt`.
- Branch : Attributes: `brid` (Primary Key), `brname`, `city`, `assets`.

2. **Relationships** :
- Borrower : Between `Customer` and `Loan` (many-to-many).
- Payment : Between `Loan` and `Payment` (one-to-many).
- Depositor : Between `Customer` and `Account` (many-to-many).
- Loan_branch : Between `Loan` and `Branch` (many-to-one).
- Account_branch : Between `Account` and `Branch` (many-to-one).

Step 2: Map Entities to Tables

Each entity becomes a table, with attributes becoming columns. Primary keys are explicitly marked.

1. Customer Table :
- Columns: `cid` (Primary Key), `cname`, `phone`, `address`, `city`, `state`, `pincode`, `DOB`, `age()`.
2. Account Table :
- Columns: `accno` (Primary Key), `balance`.
3. Saving_account Table :
- Columns: `accno` (Foreign Key referencing `Account(accno)`), `interest_rate`.
4. Checking_account Table :
- Columns: `accno` (Foreign Key referencing `Account(accno)`), `overdraft_amount`.
5. Loan Table :
- Columns: `lno` (Primary Key), `lamt`.
6. Branch Table :
- Columns: `brid` (Primary Key), `brname`, `city`, `assets`.
7. Payment Table :
- Columns: `pno` (Primary Key), `pdate`, `pamount`.

Step 3: Map Relationships to Tables

Relationships are handled based on their cardinality:

1. Borrower (many-to-many between Customer and Loan) :
- Create a junction table: `Borrower`.
- Columns: `cid` (Foreign Key referencing `Customer(cid)`), `lno` (Foreign Key referencing `Loan(lno)`).
- Composite Primary Key: `(cid, lno)`.
2. Payment (one-to-many between Loan and Payment) :
- Add a foreign key `lno` in the `Payment` table referencing `Loan(lno)`.
3. Depositor (many-to-many between Customer and Account) :
- Create a junction table: `Depositor`.
- Columns: `cid` (Foreign Key referencing `Customer(cid)`), `accno` (Foreign Key referencing `Account(accno)`).
- Composite Primary Key: `(cid, accno)`.
4. Loan_branch (many-to-one between Loan and Branch) :
- Add a foreign key `brid` in the `Loan` table referencing `Branch(brid)`.
5. Account_branch (many-to-one between Account and Branch) :
- Add a foreign key `brid` in the `Account` table referencing `Branch(brid)`.

Step 4: Final Relational Schema

The complete relational schema is as follows:

1. Customer :
- Columns: `cid` (Primary Key), `cname`, `phone`, `address`, `city`, `state`, `pincode`, `DOB`, `age()`.
2. Account :
- Columns: `accno` (Primary Key), `balance`, `brid` (Foreign Key referencing `Branch(brid)`).
3. Saving_account :
- Columns: `accno` (Primary Key, Foreign Key referencing `Account(accno)`), `interest_rate`.
4. Checking_account :
- Columns: `accno` (Primary Key, Foreign Key referencing `Account(accno)`), `overdraft_amount`.
5. Loan :
- Columns: `lno` (Primary Key), `lamt`, `brid` (Foreign Key referencing `Branch(brid)`).
6. Branch :
- Columns: `brid` (Primary Key), `brname`, `city`, `assets`.
7. Payment :
- Columns: `pno` (Primary Key), `pdate`, `pamount`, `lno` (Foreign Key referencing `Loan(lno)`).
8. Borrower :
- Columns: `cid` (Foreign Key referencing `Customer(cid)`), `lno` (Foreign Key referencing `Loan(lno)`).

- Composite Primary Key: `(cid, lno)`.

9. Depositor :
- Columns: `cid` (Foreign Key referencing `Customer(cid)`), `accno` (Foreign Key referencing `Account(accno)`).
- Composite Primary Key: `(cid, accno)`.

Question 3: Draw the schema diagram for the relational schema resulted in Question 2.

Step 1: Represent Tables as Rectangles

Each table is represented as a rectangle. Inside the rectangle, list the attributes of the table.

Step 2: Indicate Primary Keys

Underline the primary key(s) in each table. For example:

- In the `Customer` table, underline `cid`.
- In the `Account` table, underline `accno`.
- In the `Loan` table, underline `lno`.
- In the `Branch` table, underline `brid`.
- In the `Payment` table, underline `pno`.
- In the `Borrower` table, underline `(cid, lno)`.
- In the `Depositor` table, underline `(cid, accno)`.

Step 3: Indicate Foreign Keys

Use arrows to show foreign key relationships:

- Draw an arrow from `cid` in the `Borrower` table to `cid` in the `Customer` table.
- Draw an arrow from `lno` in the `Borrower` table to `lno` in the `Loan` table.
- Draw an arrow from `cid` in the `Depositor` table to `cid` in the `Customer` table.
- Draw an arrow from `accno` in the `Depositor` table to `accno` in the `Account` table.
- Draw an arrow from `brid` in the `Loan` table to `brid` in the `Branch` table.
- Draw an arrow from `brid` in the `Account` table to `brid` in the `Branch` table.
- Draw an arrow from `lno` in the `Payment` table to `lno` in the `Loan` table.

Step 4: Final Schema Diagram

Here's how the schema diagram would look:

1. Customer :

```
+------------------------------+
| Customer                     |
+------------------------------+
| cid (PK)                     |
| cname                        |
```

```
| phone                        |
| address                      |
| city                         |
| state                        |
| pincode                      |
| DOB                          |
| age()                        |
+------------------------------+
```

2. Account :

```
+------------------------------+
| Account                      |
+------------------------------+
| accno (PK)                   |
| balance                      |
| brid (FK → Branch)           |
+------------------------------+
```

3. Saving_account :

```
+------------------------------+
| Saving_account               |
+------------------------------+
| accno (PK, FK → Account)     |
| interest_rate                |
+------------------------------+
```

4. Checking_account :

```
+------------------------------+
| Checking_account             |
+------------------------------+
| accno (PK, FK → Account)     |
| overdraft_amount             |
+------------------------------+
```

5. Loan :

```
+------------------------------+
| Loan                         |
+------------------------------+
| lno (PK)                     |
| lamt                         |
| brid (FK → Branch)           |
+------------------------------+
```

6. Branch :

```
+------------------------------+
| Branch                       |
+------------------------------+
| brid (PK)                    |
| brname                       |
| city                         |
| assets                       |
```

```
+-----------------------------+
```

7. Payment :

```
+-----------------------------+
| Payment                     |
+-----------------------------+
| pno (PK)                    |
| pdate                       |
| pamount                     |
| lno (FK → Loan)             |
+-----------------------------+
```

8. Borrower :

```
+-----------------------------+
| Borrower                    |
+-----------------------------+
| cid (FK → Customer)         |
| lno (FK → Loan)             |
+-----------------------------+
```

9. Depositor :

```
+-----------------------------+
| Depositor                   |
+-----------------------------+
| cid (FK → Customer)         |
| accno (FK → Account)        |
+-----------------------------+
```

Step 5: Relationships

- Draw an arrow from `cid` in the `Borrower` table to `cid` in the `Customer` table.
- Draw an arrow from `lno` in the `Borrower` table to `lno` in the `Loan` table.
- Draw an arrow from `cid` in the `Depositor` table to `cid` in the `Customer` table.
- Draw an arrow from `accno` in the `Depositor` table to `accno` in the `Account` table.
- Draw an arrow from `brid` in the `Loan` table to `brid` in the `Branch` table.
- Draw an arrow from `brid` in the `Account` table to `brid` in the `Branch` table.
- Draw an arrow from `lno` in the `Payment` table to `lno` in the `Loan` table.

Question 4 in detail. The task is to compute the closure of the given set of functional dependencies (FDs) and list the candidate keys for the relation schema $r(A,B,C,D,E,F)$.

Step 1: Understanding the Problem

The given set of functional dependencies (FDs) is:

$F=\{A\to BC, CD\to E, B\to D, E\to A\}$

We need to:

1. Compute the closure of the set F.

2. List all the candidate keys for the relation schema $r(A,B,C,D,E,F)$.

Step 2: Definitions

1. Closure of a set of attributes ($X+$):
   - The closure $X+$ is the set of all attributes that can be functionally determined from X using the given FDs.
   - It is computed iteratively by applying the FDs until no new attributes can be added.
2. Candidate Key :
   - A candidate key is a minimal set of attributes that can uniquely identify all attributes in the relation.
   - To find candidate keys, we compute closures of subsets of attributes and check if they include all attributes in the relation.

Step 3: Compute the Closure of Each Attribute or Attribute Set

We compute the closure for each attribute or combination of attributes to determine the candidate keys.

(a) Compute $A+$:

- Start with $A+=\{A\}$.
- Apply $A \to BC$: Add $B,C \to A+=\{A,B,C\}$.
- Apply $B \to D$: Add $D \to A+=\{A,B,C,D\}$.
- Apply $CD \to E$: Add $E \to A+=\{A,B,C,D,E\}$.
- Apply $E \to A$: No new attributes are added.
- Final $A+=\{A,B,C,D,E\}$.

(b) Compute $B+$:

- Start with $B+=\{B\}$.
- Apply $B \to D$: Add $D \to B+=\{B,D\}$.
- No other FDs apply to $B+$.
- Final $B+=\{B,D\}$.

(c) Compute $C+$:

- Start with $C+=\{C\}$.
- No FDs directly involve C alone.
- Final $C+=\{C\}$.

(d) Compute *D*+:

- Start with $D+=\{D\}$.
- No FDs directly involve *D* alone.
- Final $D+=\{D\}$.

(e) Compute *E*+:

- Start with $E+=\{E\}$.
- Apply $E \rightarrow A$: Add $A \rightarrow E+=\{A,E\}$.
- Apply $A \rightarrow BC$: Add $B,C \rightarrow E+=\{A,B,C,E\}$.
- Apply $B \rightarrow D$: Add $D \rightarrow E+=\{A,B,C,D,E\}$.
- Final $E+=\{A,B,C,D,E\}$.

(f) Compute *F*+:

- Start with $F+=\{F\}$.
- No FDs involve *F*.
- Final $F+=\{F\}$.

Step 4: Identify Candidate Keys

A candidate key is a minimal set of attributes whose closure includes all attributes in the relation ($\{A,B,C,D,E,F\}$).

(a) Check *A*:

- From $A+=\{A,B,C,D,E\}$, *A* does not include *F*. Thus, *A* is not a candidate key.

(b) Check *E*:

- From $E+=\{A,B,C,D,E\}$, *E* does not include *F*. Thus, *E* is not a candidate key.

(c) Check *AF*:

- Start with $AF+=\{A,F\}$.
- Apply $A \rightarrow BC$: Add $B,C \rightarrow AF+=\{A,B,C,F\}$.
- Apply $B \rightarrow D$: Add $D \rightarrow AF+=\{A,B,C,D,F\}$.
- Apply $CD \rightarrow E$: Add $E \rightarrow AF+=\{A,B,C,D,E,F\}$.
- Since *AF*+ includes all attributes, *AF* is a candidate key.

(d) Check *EF*:

- Start with $EF+=\{E,F\}$.
- Apply $E \rightarrow A$: Add $A \rightarrow EF+=\{A,E,F\}$.

- Apply $A \rightarrow BC$: Add $B,C \rightarrow EF+=\{A,B,C,E,F\}$.
- Apply $B \rightarrow D$: Add $D \rightarrow EF+=\{A,B,C,D,E,F\}$.
- Since $EF+$ includes all attributes, EF is a candidate key.

(e) Check Other Combinations:

- Any other combination of attributes will either be non-minimal or fail to include all attributes. Thus, no additional candidate keys exist.

Step 5: Final Answer

1. Closure of Attributes :
   - $A+=\{A,B,C,D,E\}$
   - $B+=\{B,D\}$
   - $C+=\{C\}$
   - $D+=\{D\}$
   - $E+=\{A,B,C,D,E\}$
   - $F+=\{F\}$
2. Candidate Keys :
   - AF
   - EF

Conclusion

The closures of the attributes and the candidate keys have been computed step-by-step. The candidate keys for the relation schema $r(A,B,C,D,E,F)$ are:

Candidate Keys: $\{AF,EF\}$

Question 5 – Detailed Solution

Objective:
Analyze the relation schema `Student_Mark` to determine its normal form, evaluate if it satisfies **2NF**, and if not, **decompose** it into 2NF while ensuring **lossless join** and **dependency preservation**.

Step 1: Understanding the Schema

Relation Schema:
`Student_Mark(regd, name, course_id, title, grade)`

Functional Dependencies (FDs):
F = {
```
  regd → name,
  course_id → title,
  (regd, course_id) → grade
```
}

Step 2: Normal Form Definitions

- **1NF**: A relation is in 1NF if all attribute values are atomic.
  → Since no multi-valued attributes are mentioned, we assume 1NF is satisfied.
- **2NF**: A relation is in 2NF if:
  1. It is in 1NF, and
  2. It has **no partial dependency** — i.e., no non-prime attribute depends only on part of a candidate key.
- **Properties of Decomposition**:
  - **Lossless Join**: Original relation must be recoverable via joins.
  - **Dependency Preservation**: All original FDs must be represented in the decomposed relations.

Step 3: Finding Candidate Keys

Check the closure of attribute sets to find the candidate key(s):

- **Closure of (regd, course_id)**:
  Start: `{regd, course_id}`
  Apply FDs:
  `regd → name` → add `name`
  `course_id → title` → add `title`
  `(regd, course_id) → grade` → add `grade`
  Closure = `{regd, course_id, name, title, grade}` = all attributes
  ✅ So, **(regd, course_id)** is a **candidate key**

- No other combination produces all attributes, so it is the **only candidate key**.

Step 4: Checking for Partial Dependencies

- Candidate Key: **(regd, course_id)**
- Non-prime attributes: **name, title, grade**
- `regd → name`
  → Partial dependency (name depends only on part of the key)
- `course_id → title`
  → Partial dependency (title depends only on part of the key)
- `(regd, course_id) → grade`
  → Full dependency (grade depends on the entire key)

✅ **Conclusion**: The schema **is not in 2NF** due to partial dependencies.

Step 5: Decomposition into 2NF

To remove partial dependencies:

- **From `regd → name`**
  → Create: `R1(regd, name)`
  Key: `regd`
- **From `course_id → title`**
  → Create: `R2(course_id, title)`
  Key: `course_id`
- **Remaining relation**
  → `R3(regd, course_id, grade)`
  Key: `(regd, course_id)`

Step 6: Validating Decomposition

(a) Lossless Join

- `R1` and `R3` share `regd`, which is a key in `R1` → ✅
- `R2` and `R3` share `course_id`, which is a key in `R2` → ✅
  ✔️ Decomposition is **lossless**

(b) Dependency Preservation

- `regd → name` → in `R1`
- `course_id → title` → in `R2`
- `(regd, course_id) → grade` → in `R3`
  ✔️ All FDs are **preserved**

Step 7: Final Result

Normal Form:

- The original schema is in **1NF**, but **not in 2NF**.

2NF Decomposition:

- `R1(regd, name)` — Key: `regd`
- `R2(course_id, title)` — Key: `course_id`
- `R3(regd, course_id, grade)` — Key: `(regd, course_id)`

Decomposition Properties:

- **Lossless Join**: ✅
- **Dependency Preservation**: ✅

✅ **Final 2NF Decomposition:**

$$\boxed{R1(regd, name), \quad R2(course_id, title), \quad R3(regd, course_id, grade)}$$

Analyze the relation schema `Book(Title, Author, Catalog_no, Publisher, Year, Price)` to determine whether it satisfies **3NF**. If not, decompose it into 3NF while verifying the **lossless join** and **dependency preservation** properties.

Step 1: Understanding the Schema

Relation Schema:
`Book(Title, Author, Catalog_no, Publisher, Year, Price)`

Functional Dependencies (FDs):
F = {
```
 (Title, Author) → (Catalog_no, Price),
 Catalog_no → Title,
 Catalog_no → Publisher,
 Catalog_no → Year
```
}

Step 2: Definitions

- **Third Normal Form (3NF)**:
  A relation is in 3NF if:
  1. It is in 2NF, and
  2. For every FD `X → A`, one of the following holds:
     - `X` is a superkey, or
     - `A` is a prime attribute (part of a candidate key)
- **Candidate Key**: Minimal set of attributes that can uniquely identify all attributes in the relation
- **Prime Attribute**: Part of any candidate key
- **Non-Prime Attribute**: Not part of any candidate key
- **Decomposition Properties**:
  - **Lossless Join**: Reconstructs original relation without loss of data
  - **Dependency Preservation**: All original FDs are preserved in decomposed relations

---

Step 3: Identify Candidate Keys

Check closure of attribute sets using the given FDs:

- **Closure of (Title, Author)**:
  Start: `{Title, Author}`
  Apply `(Title, Author) → Catalog_no, Price` → add `Catalog_no, Price`
  Then:
  `Catalog_no → Title` (no change)
  `Catalog_no → Publisher` → add `Publisher`
  `Catalog_no → Year` → add `Year`

Closure = `{Title, Author, Catalog_no, Price, Publisher, Year}` → all attributes
✅ So, **(Title, Author)** is a **candidate key**

- No other minimal combinations cover all attributes → **Only candidate key**

Step 4: Check for 3NF Violations

- **(Title, Author) → Catalog_no, Price**
  → Left-hand side is a candidate key → ✅ satisfies 3NF
- **Catalog_no → Title**
  → Right-hand side `Title` is a **prime** attribute → ✅ satisfies 3NF
- **Catalog_no → Publisher**
  → `Publisher` is non-prime and `Catalog_no` is **not a superkey**
  → ❌ **Violates 3NF** (transitive dependency)
- **Catalog_no → Year**
  → Same issue as above: non-prime on RHS, LHS not a superkey → ❌ violates 3NF

✅ Conclusion: The schema is **not in 3NF** due to transitive dependencies via `Catalog_no`.

Step 5: Decompose into 3NF

To eliminate transitive dependencies:

- **Create R1** for `Catalog_no → Title, Publisher, Year`
  → `R1(Catalog_no, Title, Publisher, Year)`
  → Primary Key: `Catalog_no`
- **Remaining attributes** go into **R2**:
  → `R2(Title, Author, Catalog_no, Price)`
  → Primary Key: `(Title, Author)`

Step 6: Verify Decomposition

(a) Lossless Join

- `R1` and `R2` share `Catalog_no`, which is a **key in R1` → ✅
  ✔️ Decomposition is **lossless**

(b) Dependency Preservation

Check if all original FDs are represented:

- `(Title, Author) → Catalog_no, Price` → in `R2`

- `Catalog_no → Title, Publisher, Year` → in `R1`
  ✓ All FDs are **preserved**

Step 7: Final Result

Normal Form:

- The original schema is in **2NF** but **not in 3NF** due to transitive dependencies.

3NF Decomposition:

- `R1(Catalog_no, Title, Publisher, Year)` — Key: `Catalog_no`
- `R2(Title, Author, Catalog_no, Price)` — Key: `(Title, Author)`

Decomposition Properties:

- **Lossless Join**: ✅
- **Dependency Preservation**: ✅

✅ **Final 3NF Decomposition:**

$$\boxed{R1(Catalog\_no, Title, Publisher, Year), \quad R2(Title, Author, Catalog\_no, Price)}$$

Question 7 in detail. The task involves analyzing the given relation schema and functional dependencies to:

1. Prove that AG is a superkey using Armstrong's axioms .
2. Compute the canonical cover of the functional dependency set F.
3. Decompose the schema into 3NF based on the canonical cover.
4. Decompose the schema into BCNF .

The given relation schema is $r(A,B,C,D,E,G)$ with the following functional dependency set:

$F=\{A\rightarrow BCD, BC\rightarrow DE, B\rightarrow D, D\rightarrow A\}$.

Step 1: Prove That *AG* Is a Superkey

A superkey is a set of attributes that can uniquely identify all attributes in the relation. To prove that AG is a superkey, we compute its closure ($AG+$) and check if it includes all attributes in $r(A,B,C,D,E,G)$.

(a) Start with $AG+=\{A,G\}$.

(b) Apply $A\rightarrow BCD$:

- Add $B,C,D \rightarrow AG+=\{A,G,B,C,D\}$.

(c) Apply $BC\rightarrow DE$:

- Add $E \rightarrow AG+=\{A,G,B,C,D,E\}$.

(d) Check for additional FDs:

- No further attributes can be added using the remaining FDs ($B\rightarrow D$, $D\rightarrow A$).

(e) Final $AG+$:

- $AG+=\{A,B,C,D,E,G\}$, which includes all attributes in $r(A,B,C,D,E,G)$.

Thus, AG is a superkey .

Step 2: Compute the Canonical Cover

The canonical cover is a minimal set of functional dependencies that is equivalent to the original set F, with no redundant dependencies or extraneous attributes.

(a) Step 1: Simplify each FD by removing extraneous attributes.

1. $A \to BCD$:
   - Split into $A \to B$, $A \to C$, $A \to D$.
2. $BC \to DE$:
   - Split into $BC \to D$, $BC \to E$.
3. $B \to D$:
   - Already minimal.
4. $D \to A$:
   - Already minimal.

Updated F:

$F=\{A \to B, A \to C, A \to D, BC \to D, BC \to E, B \to D, D \to A\}$.

(b) Step 2: Remove redundant FDs.

1. Check $A \to B$:
   - Closure of A without $A \to B$: $A+=\{A,C,D\}$ (using $A \to C, A \to D$).
   - $B \in / A+$, so $A \to B$ is not redundant.
2. Check $A \to C$:
   - Closure of A without $A \to C$: $A+=\{A,B,D\}$ (using $A \to B, A \to D$).
   - $C \in / A+$, so $A \to C$ is not redundant.
3. Check $A \to D$:
   - Closure of A without $A \to D$: $A+=\{A,B,C\}$ (using $A \to B, A \to C$).
   - $D \in / A+$, so $A \to D$ is not redundant.
4. Check $BC \to D$:
   - Closure of BC without $BC \to D$: $BC+=\{B,C,E\}$ (using $BC \to E$).
   - $D \in / BC+$, so $BC \to D$ is not redundant.
5. Check $BC \to E$:
   - Closure of BC without $BC \to E$: $BC+=\{B,C,D\}$ (using $BC \to D$).
   - $E \in / BC+$, so $BC \to E$ is not redundant.
6. Check $B \to D$:
   - Closure of B without $B \to D$: $B+=\{B\}$.
   - $D \in / B+$, so $B \to D$ is not redundant.
7. Check $D \to A$:
   - Closure of D without $D \to A$: $D+=\{D\}$.
   - $A \in / D+$, so $D \to A$ is not redundant.

No redundant FDs are found.

(c) Final Canonical Cover:

$Fc=\{A \to B, A \to C, A \to D, BC \to D, BC \to E, B \to D, D \to A\}$.

Step 3: Decompose Into 3NF

To decompose into 3NF , we use the synthesis algorithm based on the canonical cover Fc.

(a) Create a relation for each FD in *Fc*:

1. $R1(A,B)$: From $A \rightarrow B$.
2. $R2(A,C)$: From $A \rightarrow C$.
3. $R3(A,D)$: From $A \rightarrow D$.
4. $R4(B,C,D)$: From $BC \rightarrow D$.
5. $R5(B,C,E)$: From $BC \rightarrow E$.
6. $R6(B,D)$: From $B \rightarrow D$.
7. $R7(D,A)$: From $D \rightarrow A$.

(b) Combine relations with overlapping keys:

- $R1(A,B)$, $R2(A,C)$, $R3(A,D)$ can be combined into $R1(A,B,C,D)$ since they share the same key A.
- $R4(B,C,D)$ and $R5(B,C,E)$ can be combined into $R2(B,C,D,E)$ since they share BC as a key.
- $R6(B,D)$ is redundant because $B \rightarrow D$ is already covered in $R2(B,C,D,E)$.
- $R7(D,A)$ is redundant because $D \rightarrow A$ is already covered in $R1(A,B,C,D)$.

(c) Final 3NF Decomposition:

$R1(A,B,C,D),R2(B,C,D,E)$.

Step 4: Decompose Into BCNF

To decompose into BCNF , we ensure that every determinant (left-hand side of an FD) is a superkey.

(a) Analyze $R1(A,B,C,D)$:

- FDs: $A \rightarrow B, A \rightarrow C, A \rightarrow D$.
- Candidate Key: A.
- All determinants (A) are superkeys. Thus, $R1$ is in BCNF .

(b) Analyze $R2(B,C,D,E)$:

- FDs: $BC \rightarrow D, BC \rightarrow E, B \rightarrow D$.
- Candidate Keys: BC.
- $B \rightarrow D$ violates BCNF because B is not a superkey.

(c) Decompose $R2$:

- From $B \rightarrow D$, create $R3(B,D)$.
- Remaining attributes: $R4(B,C,E)$.

(d) Final BCNF Decomposition:

$R1(A,B,C,D), R3(B,D), R4(B,C,E)$.

Final Answer

1. **Superkey Proof** :
   - AG is a superkey.
2. **Canonical Cover** :

$$F_c = \{A \to B,\ A \to C,\ A \to D,\ BC \to D,\ BC \to E,\ B \to D,\ D \to A\}.$$

3. **3NF Decomposition** :

$$R1(A, B, C, D),\ R2(B, C, D, E).$$

4. **BCNF Decomposition** :

$$\boxed{R1(A, B, C, D),\ R3(B, D),\ R4(B, C, E).}$$

Question 8 in detail. The task involves analyzing the given relation schema and functional dependencies to:

1. Determine if the schema satisfies 3NF .
2. If it does not satisfy 3NF , decompose it into 3NF while checking the properties of decomposition (lossless join and dependency preservation).
3. Determine if the schema satisfies BCNF .
4. If it does not satisfy BCNF , decompose it into BCNF .

The given relation schema is *r*(PAN, PI, DI, DRUG, QTY, COST) with the following functional dependency set:

F={PAN→PI,PI→DI,(PI, DRUG)→QTY,(DRUG, QTY)→COST}.

Step 1: Definitions

1. Third Normal Form (3NF) :
    - A relation is in 3NF if:
        - It is in 2NF (no partial dependencies).
        - There are no transitive dependencies (i.e., no non-prime attribute depends on another non-prime attribute).
2. Boyce-Codd Normal Form (BCNF) :
    - A relation is in BCNF if every determinant (left-hand side of an FD) is a superkey.
3. Candidate Key :
    - A candidate key is a minimal set of attributes that uniquely identifies all attributes in the relation.
4. Prime Attribute :
    - An attribute that is part of any candidate key.
5. Non-Prime Attribute :
    - An attribute that is not part of any candidate key.
6. Properties of Decomposition :
    - Lossless Join : The decomposition should allow us to reconstruct the original relation without losing data.
    - Dependency Preservation : All functional dependencies in the original schema must be preserved in the decomposed schema.

Step 2: Identify Candidate Keys

To determine the candidate keys, we compute the closure of subsets of attributes using the given FDs.

(a) Compute (PI, DRUG)+:

- Start with (PI, DRUG)+={PI, DRUG}.

- Apply PI→DI: Add DI → (PI, DRUG)+={PI, DRUG, DI}.
- Apply (PI, DRUG)→QTY: Add QTY → (PI, DRUG)+={PI, DRUG, DI, QTY}.
- Apply (DRUG, QTY)→COST: Add COST → (PI, DRUG)+={PI, DRUG, DI, QTY, COST}.
- Since (PI, DRUG)+ includes all attributes except PAN, add PAN using PAN→PI → (PI, DRUG)+={PAN, PI, DRUG, DI, QTY, COST}.

Thus, (PI, DRUG) is a candidate key .

(b) Check Other Combinations:

- Any other combination of attributes will either be non-minimal or fail to include all attributes. Thus, (PI, DRUG) is the only candidate key.

Step 3: Check for 3NF

To check if the schema satisfies 3NF , we examine the functional dependencies for transitive dependencies.

(a) Analyze PAN→PI:

- PI is a prime attribute (part of the candidate key (PI, DRUG)).
- This dependency does not violate 3NF because the right-hand side is a prime attribute.

(b) Analyze PI→DI:

- DI is a non-prime attribute.
- PI is not a superkey, but DI depends on PI, which creates a transitive dependency (PI→DI).

(c) Analyze (PI, DRUG)→QTY:

- QTY is a non-prime attribute.
- (PI, DRUG) is a candidate key, so this dependency does not violate 3NF .

(d) Analyze (DRUG, QTY)→COST:

- COST is a non-prime attribute.
- (DRUG, QTY) is not a superkey, but COST depends on (DRUG, QTY), which creates a transitive dependency ((DRUG, QTY)→COST).

Since there are transitive dependencies (PI→DI and (DRUG, QTY)→COST), the schema is not in 3NF .

Step 4: Decompose Into 3NF

To eliminate transitive dependencies, we decompose the schema into smaller relations such that each relation satisfies 3NF .

(a) Decompose PI→DI:

- Create a new relation *R*1(PI, DI).
- Primary Key: PI.

(b) Decompose (PI, DRUG)→QTY:

- Keep this dependency in the original relation *R*2(PAN, PI, DRUG, QTY).
- Primary Key: (PI, DRUG).

(c) Decompose (DRUG, QTY)→COST:

- Create a new relation *R*3(DRUG, QTY, COST).
- Primary Key: (DRUG, QTY).

(d) Remaining Attributes:

- Keep PAN→PI in *R*2(PAN, PI, DRUG, QTY).

Step 5: Verify Properties of Decomposition

(a) Lossless Join:

- To check for lossless join, use the rule that at least one common attribute between two relations must be a superkey in one of the relations.
- *R*1(PI, DI) and *R*2(PAN, PI, DRUG, QTY) share PI, which is a superkey in *R*1.
- *R*2(PAN, PI, DRUG, QTY) and *R*3(DRUG, QTY, COST) share (DRUG, QTY), which is a superkey in *R*3.
- Thus, the decomposition is lossless .

(b) Dependency Preservation:

- Original FDs:
    - PAN→PI: Preserved in *R*2.
    - PI→DI: Preserved in *R*1.
    - (PI, DRUG)→QTY: Preserved in *R*2.
    - (DRUG, QTY)→COST: Preserved in *R*3.
- All FDs are preserved in the decomposition.

Step 6: Check for BCNF

To check if the schema satisfies BCNF , we examine the determinants of each FD.

(a) Analyze PAN→PI:

- PAN is not a superkey, so this violates BCNF .

(b) Analyze PI→DI:

- PI is not a superkey, so this violates BCNF .

(c) Analyze (PI, DRUG)→QTY:

- (PI, DRUG) is a superkey, so this does not violate BCNF .

(d) Analyze (DRUG, QTY)→COST:

- (DRUG, QTY) is a superkey, so this does not violate BCNF .

Since PAN→PI and PI→DI violate BCNF , the schema is not in BCNF .

Step 7: Decompose Into BCNF

To decompose into BCNF , we ensure that every determinant is a superkey.

(a) Decompose PAN→PI:

- Create a new relation $R4$(PAN, PI).
- Primary Key: PAN.

(b) Remaining Attributes:

- Keep PI→DI in $R1$(PI, DI).
- Keep (PI, DRUG)→QTY in $R2$(PI, DRUG, QTY).
- Keep (DRUG, QTY)→COST in $R3$(DRUG, QTY, COST).

Final Answer

1. **3NF Decomposition** :

$$R1(\text{PI, DI}),\ R2(\text{PAN, PI, DRUG, QTY}),\ R3(\text{DRUG, QTY, COST}).$$

2. **BCNF Decomposition** :

$$\boxed{R1(\text{PI, DI}),\ R2(\text{PAN, PI}),\ R3(\text{PI, DRUG, QTY}),\ R4(\text{DRUG, QTY, COST}).}$$

Question 6 – Detailed Solution

We are given a relation schema:

Book(Title, Author, Catalog_no, Publisher, Year, Price)
with the following **functional dependencies (FDs)**:

F = {
  (Title, Author) → (Catalog_no, Price)
  Catalog_no → Title
  Catalog_no → Publisher
  Catalog_no → Year
}

Objective

1. Determine whether the schema is in **Third Normal Form (3NF)**.
2. If not, decompose it into 3NF.
3. Verify **lossless join** and **dependency preservation** properties of the decomposition.

Step 1: Key Concepts

- **3NF Definition**: A relation is in 3NF if, for every functional dependency `X → A`, one of the following holds:
  - `X` is a superkey, or
  - `A` is a **prime attribute** (part of some candidate key)
- **Candidate Key**: A minimal set of attributes that can uniquely determine all others in the relation.
- **Prime Attribute**: An attribute that is part of any candidate key.
- **Non-Prime Attribute**: Not part of any candidate key.
- **Lossless Join**: A decomposition is lossless if we can reconstruct the original relation without any loss of information.
- **Dependency Preservation**: All original functional dependencies must be enforceable in the decomposed relations.

Step 2: Identify Candidate Key(s)

Let's compute the closure of (Title, Author):

- Start with:
  (Title, Author)+ = {Title, Author}
- Apply FDs:
  - (Title, Author) → Catalog_no, Price → add: Catalog_no, Price

- Catalog_no → Title (already present)
- Catalog_no → Publisher → add: Publisher
- Catalog_no → Year → add: Year

Now:
(Title, Author)+ = {Title, Author, Catalog_no, Price, Publisher, Year} → all attributes covered
✅ **Candidate Key = (Title, Author)**

No other smaller combinations provide full closure, so this is the **only candidate key**.

Step 3: Check for 3NF

Examine each FD:

- **(Title, Author) → Catalog_no, Price**
  → LHS is a candidate key → ✅ satisfies 3NF
- **Catalog_no → Title**
  → RHS is a **prime** attribute → ✅ satisfies 3NF
- **Catalog_no → Publisher**
  → `Catalog_no` is not a superkey, and `Publisher` is non-prime → ❌ violates 3NF
- **Catalog_no → Year**
  → Same issue: non-superkey → non-prime → ❌ violates 3NF

✅ **Conclusion**: Schema **violates 3NF** due to transitive dependencies via `Catalog_no`.

Step 4: Decomposition into 3NF

To remove transitive dependencies:

(a) Create R1 with attributes from dependent FDs:

R1(Catalog_no, Title, Publisher, Year)
→ Primary Key: `Catalog_no`

(b) Place the remaining attributes in another relation:

R2(Title, Author, Catalog_no, Price)
→ Primary Key: (Title, Author)

Step 5: Verify Decomposition Properties

✅ Lossless Join

- Common attribute: `Catalog_no`
- `Catalog_no` is a key in **R1** → satisfies lossless join condition

✅ Dependency Preservation

- (Title, Author) → Catalog_no, Price → in R2
- Catalog_no → Title, Publisher, Year → in R1
  → All FDs preserved in the decomposition

Final Result

- **Is the original schema in 3NF?** ❌ No
- **3NF Decomposition:**
  - ✔ `R1(Catalog_no, Title, Publisher, Year)` — Key: `Catalog_no`
  - ✔ `R2(Title, Author, Catalog_no, Price)` — Key: `(Title, Author)`
- **Properties:**
  - ✔ **Lossless Join**
  - ✔ **Dependency Preservation**

$$\boxed{\text{3NF Decomposition: } R1(Catalog\_no, Title, Publisher, Year), \quad R2(Title, Author, Catalog\_no, Price)}$$

Question 7 in detail. The task involves analyzing the given relation schema $r(A,B,C,D,E,G)$ and its functional dependencies (FDs) to:

1. Prove that AG is a superkey using Armstrong's axioms .
2. Compute the canonical cover of the FD set F.
3. Decompose the schema into 3NF based on the canonical cover.
4. Decompose the schema into BCNF .

The given relation schema is $r(A,B,C,D,E,G)$ with the following FDs:

$F=\{A\rightarrow BCD, BC\rightarrow DE, B\rightarrow D, D\rightarrow A\}$.

Step 1: Prove That *AG* Is a Superkey

A superkey is a set of attributes that can uniquely identify all attributes in the relation. To prove that AG is a superkey, we compute its closure ($AG+$) and check if it includes all attributes in $r(A,B,C,D,E,G)$.

(a) Start with $AG+=\{A,G\}$.

(b) Apply $A\rightarrow BCD$:

- Add $B,C,D \rightarrow AG+=\{A,G,B,C,D\}$.

(c) Apply $BC\rightarrow DE$:

- Add $E \rightarrow AG+=\{A,G,B,C,D,E\}$.

(d) Check for additional FDs:

- No further attributes can be added using the remaining FDs ($B\rightarrow D$, $D\rightarrow A$).

(e) Final $AG+$:

- $AG+=\{A,B,C,D,E,G\}$, which includes all attributes in $r(A,B,C,D,E,G)$.

Thus, AG is a superkey .

Step 2: Compute the Canonical Cover

The canonical cover is a minimal set of functional dependencies that is equivalent to the original set F, with no redundant dependencies or extraneous attributes.

(a) Step 1: Simplify each FD by removing extraneous attributes.

1. $A \to BCD$:
    - Split into $A \to B$, $A \to C$, $A \to D$.
2. $BC \to DE$:
    - Split into $BC \to D$, $BC \to E$.
3. $B \to D$:
    - Already minimal.
4. $D \to A$:
    - Already minimal.

Updated F:

$F=\{A \to B, A \to C, A \to D, BC \to D, BC \to E, B \to D, D \to A\}$.

(b) Step 2: Remove redundant FDs.

1. Check $A \to B$:
    - Closure of A without $A \to B$: $A+=\{A,C,D\}$ (using $A \to C, A \to D$).
    - $B \in / A+$, so $A \to B$ is not redundant.
2. Check $A \to C$:
    - Closure of A without $A \to C$: $A+=\{A,B,D\}$ (using $A \to B, A \to D$).
    - $C \in / A+$, so $A \to C$ is not redundant.
3. Check $A \to D$:
    - Closure of A without $A \to D$: $A+=\{A,B,C\}$ (using $A \to B, A \to C$).
    - $D \in / A+$, so $A \to D$ is not redundant.
4. Check $BC \to D$:
    - Closure of BC without $BC \to D$: $BC+=\{B,C,E\}$ (using $BC \to E$).
    - $D \in / BC+$, so $BC \to D$ is not redundant.
5. Check $BC \to E$:
    - Closure of BC without $BC \to E$: $BC+=\{B,C,D\}$ (using $BC \to D$).
    - $E \in / BC+$, so $BC \to E$ is not redundant.
6. Check $B \to D$:
    - Closure of B without $B \to D$: $B+=\{B\}$.
    - $D \in / B+$, so $B \to D$ is not redundant.
7. Check $D \to A$:
    - Closure of D without $D \to A$: $D+=\{D\}$.
    - $A \in / D+$, so $D \to A$ is not redundant.

No redundant FDs are found.

(c) Final Canonical Cover:

$Fc=\{A \to B, A \to C, A \to D, BC \to D, BC \to E, B \to D, D \to A\}$.

Step 3: Decompose Into 3NF

To decompose into 3NF , we use the synthesis algorithm based on the canonical cover Fc.

(a) Create a relation for each FD in *Fc*:

1. $R1(A,B)$: From $A \rightarrow B$.
2. $R2(A,C)$: From $A \rightarrow C$.
3. $R3(A,D)$: From $A \rightarrow D$.
4. $R4(B,C,D)$: From $BC \rightarrow D$.
5. $R5(B,C,E)$: From $BC \rightarrow E$.
6. $R6(B,D)$: From $B \rightarrow D$.
7. $R7(D,A)$: From $D \rightarrow A$.

(b) Combine relations with overlapping keys:

- $R1(A,B)$, $R2(A,C)$, $R3(A,D)$ can be combined into $R1(A,B,C,D)$ since they share the same key A.
- $R4(B,C,D)$ and $R5(B,C,E)$ can be combined into $R2(B,C,D,E)$ since they share BC as a key.
- $R6(B,D)$ is redundant because $B \rightarrow D$ is already covered in $R2(B,C,D,E)$.
- $R7(D,A)$ is redundant because $D \rightarrow A$ is already covered in $R1(A,B,C,D)$.

(c) Final 3NF Decomposition:

$R1(A,B,C,D), R2(B,C,D,E)$.

Step 4: Decompose Into BCNF

To decompose into BCNF , we ensure that every determinant (left-hand side of an FD) is a superkey.

(a) Analyze $R1(A,B,C,D)$:

- FDs: $A \rightarrow B, A \rightarrow C, A \rightarrow D$.
- Candidate Key: A.
- All determinants (A) are superkeys. Thus, $R1$ is in BCNF .

(b) Analyze $R2(B,C,D,E)$:

- FDs: $BC \rightarrow D, BC \rightarrow E, B \rightarrow D$.
- Candidate Keys: BC.
- $B \rightarrow D$ violates BCNF because B is not a superkey.

(c) Decompose $R2$:

- From $B \rightarrow D$, create $R3(B,D)$.
- Remaining attributes: $R4(B,C,E)$.

(d) Final BCNF Decomposition:

$R1(A,B,C,D), R3(B,D), R4(B,C,E)$.

Final Answer

1. **Superkey Proof** :
   - AG is a superkey.
2. **Canonical Cover** :

$$F_c = \{A \to B,\, A \to C,\, A \to D,\, BC \to D,\, BC \to E,\, B \to D,\, D \to A\}.$$

3. **3NF Decomposition** :

$$R1(A, B, C, D),\, R2(B, C, D, E).$$

4. **BCNF Decomposition** :

$$\boxed{R1(A, B, C, D),\, R3(B, D),\, R4(B, C, E).}$$

Question 8 in detail. The task involves analyzing the given relation schema and functional dependencies to:

1. Determine if the schema satisfies 3NF .
2. If it does not satisfy 3NF , decompose it into 3NF while checking the properties of decomposition (lossless join and dependency preservation).
3. Determine if the schema satisfies BCNF .
4. If it does not satisfy BCNF , decompose it into BCNF .

The given relation schema is *r*(PAN, PI, DI, DRUG, QTY, COST) with the following functional dependency set:

F={PAN→PI,PI→DI,(PI, DRUG)→QTY,(DRUG, QTY)→COST}.

Step 1: Definitions

1. Third Normal Form (3NF) :
    - A relation is in 3NF if:
        - It is in 2NF (no partial dependencies).
        - There are no transitive dependencies (i.e., no non-prime attribute depends on another non-prime attribute).
2. Boyce-Codd Normal Form (BCNF) :
    - A relation is in BCNF if every determinant (left-hand side of an FD) is a superkey.
3. Candidate Key :
    - A candidate key is a minimal set of attributes that uniquely identifies all attributes in the relation.
4. Prime Attribute :
    - An attribute that is part of any candidate key.
5. Non-Prime Attribute :
    - An attribute that is not part of any candidate key.
6. Properties of Decomposition :
    - Lossless Join : The decomposition should allow us to reconstruct the original relation without losing data.
    - Dependency Preservation : All functional dependencies in the original schema must be preserved in the decomposed schema.

Step 2: Identify Candidate Keys

To determine the candidate keys, we compute the closure of subsets of attributes using the given FDs.

(a) Compute (PI, DRUG)+:

- Start with (PI, DRUG)+={PI, DRUG}.

- Apply PI→DI: Add DI → (PI, DRUG)+={PI, DRUG, DI}.
- Apply (PI, DRUG)→QTY: Add QTY → (PI, DRUG)+={PI, DRUG, DI, QTY}.
- Apply (DRUG, QTY)→COST: Add COST → (PI, DRUG)+={PI, DRUG, DI, QTY, COST}.
- Since (PI, DRUG)+ includes all attributes except PAN, add PAN using PAN→PI → (PI, DRUG)+={PAN, PI, DRUG, DI, QTY, COST}.

Thus, (PI, DRUG) is a candidate key .

(b) Check Other Combinations:

- Any other combination of attributes will either be non-minimal or fail to include all attributes. Thus, (PI, DRUG) is the only candidate key.

Step 3: Check for 3NF

To check if the schema satisfies 3NF , we examine the functional dependencies for transitive dependencies.

(a) Analyze PAN→PI:

- PI is a prime attribute (part of the candidate key (PI, DRUG)).
- This dependency does not violate 3NF because the right-hand side is a prime attribute.

(b) Analyze PI→DI:

- DI is a non-prime attribute.
- PI is not a superkey, but DI depends on PI, which creates a transitive dependency (PI→DI).

(c) Analyze (PI, DRUG)→QTY:

- QTY is a non-prime attribute.
- (PI, DRUG) is a candidate key, so this dependency does not violate 3NF .

(d) Analyze (DRUG, QTY)→COST:

- COST is a non-prime attribute.
- (DRUG, QTY) is not a superkey, but COST depends on (DRUG, QTY), which creates a transitive dependency ((DRUG, QTY)→COST).

Since there are transitive dependencies (PI→DI and (DRUG, QTY)→COST), the schema is not in 3NF .

Step 4: Decompose Into 3NF

To eliminate transitive dependencies, we decompose the schema into smaller relations such that each relation satisfies 3NF .

(a) Decompose PI→DI:

- Create a new relation *R*1(PI, DI).
- Primary Key: PI.

(b) Decompose (PI, DRUG)→QTY:

- Keep this dependency in the original relation *R*2(PAN, PI, DRUG, QTY).
- Primary Key: (PI, DRUG).

(c) Decompose (DRUG, QTY)→COST:

- Create a new relation *R*3(DRUG, QTY, COST).
- Primary Key: (DRUG, QTY).

(d) Remaining Attributes:

- Keep PAN→PI in *R*2(PAN, PI, DRUG, QTY).

Step 5: Verify Properties of Decomposition

(a) Lossless Join:

- To check for lossless join, use the rule that at least one common attribute between two relations must be a superkey in one of the relations.
- *R*1(PI, DI) and *R*2(PAN, PI, DRUG, QTY) share PI, which is a superkey in *R*1.
- *R*2(PAN, PI, DRUG, QTY) and *R*3(DRUG, QTY, COST) share (DRUG, QTY), which is a superkey in *R*3.
- Thus, the decomposition is lossless .

(b) Dependency Preservation:

- Original FDs:
    - PAN→PI: Preserved in *R*2.
    - PI→DI: Preserved in *R*1.
    - (PI, DRUG)→QTY: Preserved in *R*2.
    - (DRUG, QTY)→COST: Preserved in *R*3.
- All FDs are preserved in the decomposition.

Step 6: Check for BCNF

To check if the schema satisfies BCNF , we examine the determinants of each FD.

(a) Analyze PAN→PI:

- PAN is not a superkey, so this violates BCNF .

(b) Analyze PI→DI:

- PI is not a superkey, so this violates BCNF .

(c) Analyze (PI, DRUG)→QTY:

- (PI, DRUG) is a superkey, so this does not violate BCNF .

(d) Analyze (DRUG, QTY)→COST:

- (DRUG, QTY) is a superkey, so this does not violate BCNF .

Since PAN→PI and PI→DI violate BCNF , the schema is not in BCNF .

Step 7: Decompose Into BCNF

To decompose into BCNF , we ensure that every determinant is a superkey.

(a) Decompose PAN→PI:

- Create a new relation $R4$(PAN, PI).
- Primary Key: PAN.

(b) Remaining Attributes:

- Keep PI→DI in $R1$(PI, DI).
- Keep (PI, DRUG)→QTY in $R2$(PI, DRUG, QTY).
- Keep (DRUG, QTY)→COST in $R3$(DRUG, QTY, COST).

Final Answer

1. **3NF Decomposition** :

$$R1(\text{PI, DI}),\ R2(\text{PAN, PI, DRUG, QTY}),\ R3(\text{DRUG, QTY, COST}).$$

2. **BCNF Decomposition** :

$$\boxed{R1(\text{PI, DI}),\ R2(\text{PAN, PI}),\ R3(\text{PI, DRUG, QTY}),\ R4(\text{DRUG, QTY, COST}).}$$

Question 9 in detail. The task involves analyzing the given database schemas to determine the weakest normal form that the new database satisfies but the old one does not.

Step 1: Understanding the Problem

We are given two versions of a database schema for research articles in a journal:

Old Schema :

- Relation: *r*(VOLUME, NUMBER, STARTPAGE, ENDPAGE, TITLE, YEAR, PRICE)
- Primary Key: (VOLUME, NUMBER, STARTPAGE, ENDPAGE)
- Functional Dependencies (FDs):
    1. (VOLUME, NUMBER, STARTPAGE, ENDPAGE)→TITLE
    2. (VOLUME, NUMBER)→YEAR
    3. (VOLUME, NUMBER, STARTPAGE, ENDPAGE)→PRICE

New Schema :

- Relations:
    1. *R*1(VOLUME, NUMBER, STARTPAGE, ENDPAGE, TITLE, PRICE)
    2. *R*2(VOLUME, NUMBER, YEAR)
- Primary Keys:
    - For *R*1: (VOLUME, NUMBER, STARTPAGE, ENDPAGE)
    - For *R*2: (VOLUME, NUMBER)

The goal is to determine the weakest normal form satisfied by the new schema but not by the old schema.

Step 2: Definitions

To analyze the problem, we need to understand the following normal forms:

1. First Normal Form (1NF) :
    - A relation is in 1NF if all attributes contain atomic (indivisible) values.
    - Both the old and new schemas are assumed to be in 1NF because the problem does not mention any non-atomic attributes.
2. Second Normal Form (2NF) :
    - A relation is in 2NF if it is in 1NF and there are no partial dependencies (i.e., no non-prime attribute depends on part of a candidate key).
3. Third Normal Form (3NF) :
    - A relation is in 3NF if it is in 2NF and there are no transitive dependencies (i.e., no non-prime attribute depends on another non-prime attribute).
4. Boyce-Codd Normal Form (BCNF) :
    - A relation is in BCNF if every determinant (left-hand side of an FD) is a superkey.

Step 3: Analyze the Old Schema

(a) Candidate Key :

- The primary key is (VOLUME, NUMBER, STARTPAGE, ENDPAGE), which is the only candidate key.

(b) Check for Partial Dependencies :

- A partial dependency occurs when a non-prime attribute depends on part of the candidate key.
- The FD (VOLUME, NUMBER)→YEAR violates 2NF because:
    - YEAR is a non-prime attribute.
    - (VOLUME, NUMBER) is part of the candidate key (VOLUME, NUMBER, STARTPAGE, ENDPAGE).

Thus, the old schema is not in 2NF .

(c) Check for Transitive Dependencies :

- Since the schema is not in 2NF, it cannot be in 3NF or BCNF.

Step 4: Analyze the New Schema

(a) Relation *R*1(VOLUME, NUMBER, STARTPAGE, ENDPAGE, TITLE, PRICE):

- Candidate Key: (VOLUME, NUMBER, STARTPAGE, ENDPAGE).
- FDs:
    1. (VOLUME, NUMBER, STARTPAGE, ENDPAGE)→TITLE
    2. (VOLUME, NUMBER, STARTPAGE, ENDPAGE)→PRICE
- There are no partial or transitive dependencies in *R*1 because all non-prime attributes (TITLE, PRICE) depend directly on the entire candidate key.
- Thus, *R*1 is in 3NF and BCNF .

(b) Relation *R*2(VOLUME, NUMBER, YEAR):

- Candidate Key: (VOLUME, NUMBER).
- FDs:
    1. (VOLUME, NUMBER)→YEAR
- There are no partial or transitive dependencies in *R*2 because YEAR depends directly on the entire candidate key.
- Thus, *R*2 is in 3NF and BCNF .

Step 5: Compare the Old and New Schemas

- Old Schema :
  - Violates 2NF due to the partial dependency (VOLUME, NUMBER)→YEAR.
  - Therefore, the old schema is only in 1NF .
- New Schema :
  - Both $R1$ and $R2$ are in 3NF and BCNF because they eliminate the partial dependency present in the old schema.

Step 6: Determine the Weakest Normal Form

The weakest normal form satisfied by the new schema but not by the old schema is 2NF . This is because:

1. The old schema is not in 2NF due to the partial dependency (VOLUME, NUMBER)→YEAR.
2. The new schema eliminates this partial dependency and is therefore in 2NF.

Final Answer

$$\boxed{\text{The weakest normal form satisfied by the new schema but not by the old schema is 2NF.}}$$

Question 10 in detail. The task involves analyzing the given schemas and constraints to:

1. Determine the highest normal form satisfied by the schemas.
2. Normalize the schemas to 4NF .

The given schemas and constraints are:

Schemas :

1. **books(accessionno, isbn, title, author, publisher)**
2. **users(userid, name, deptid, deptname)**

Constraints :

- **accessionno → isbn**
- **isbn → title**
- **isbn → publisher**
- **isbn →→ author** (multi-valued dependency)
- **userid → name**
- **userid → deptid**
- **deptid → deptname**

Step 1: Definitions

To analyze the problem, we need to understand the following normal forms:

1. First Normal Form (1NF) :
   - A relation is in 1NF if all attributes contain atomic (indivisible) values.
2. Second Normal Form (2NF) :
   - A relation is in 2NF if it is in 1NF and there are no partial dependencies (i.e., no non-prime attribute depends on part of a candidate key).
3. Third Normal Form (3NF) :
   - A relation is in 3NF if it is in 2NF and there are no transitive dependencies (i.e., no non-prime attribute depends on another non-prime attribute).
4. Boyce-Codd Normal Form (BCNF) :
   - A relation is in BCNF if every determinant (left-hand side of an FD) is a superkey.
5. Fourth Normal Form (4NF) :
   - A relation is in 4NF if it is in BCNF and there are no multi-valued dependencies (MVDs) except those implied by the primary key.

Step 2: Analyze the books Schema

(a) Candidate Key :

- From the given functional dependencies:
  - **accessionno → isbn**
  - **isbn → title**, **isbn → publisher**
  - **isbn →→ author** (multi-valued dependency)
- The candidate key for **books** is accessionno because it uniquely identifies each tuple.

(b) Check for 1NF :

- All attributes (**accessionno**, **isbn**, **title**, **author**, **publisher**) are assumed to be atomic.
- Thus, the schema is in 1NF .

(c) Check for 2NF :

- A partial dependency occurs when a non-prime attribute depends on part of the candidate key.
- Here, all non-prime attributes (**isbn**, **title**, **publisher**, **author**) depend directly on the candidate key accessionno or indirectly through **isbn**.
- Thus, the schema is in 2NF .

(d) Check for 3NF :

- A transitive dependency occurs when a non-prime attribute depends on another non-prime attribute.
- Here:
  - **isbn → title**, **isbn → publisher**: These are transitive dependencies because **isbn** is not a candidate key.
- Thus, the schema is not in 3NF .

(e) Check for BCNF :

- A relation is in BCNF if every determinant is a superkey.
- Here:
  - **isbn → title**, **isbn → publisher**: **isbn** is not a superkey.
- Thus, the schema is not in BCNF .

(f) Check for 4NF :

- A multi-valued dependency (MVD) exists when two independent attributes depend on the same key.
- Here:
  - **isbn →→ author**: This is a multi-valued dependency because **author** has multiple values for a single **isbn**.
- Since MVDs exist, the schema is not in 4NF .

Step 3: Analyze the users Schema

(a) Candidate Key :

- From the given functional dependencies:
    - **userid → name**, **userid → deptid**
    - **deptid → deptname**
- The candidate key for **users** is userid because it uniquely identifies each tuple.

(b) Check for 1NF :

- All attributes (**userid**, **name**, **deptid**, **deptname**) are assumed to be atomic.
- Thus, the schema is in 1NF .

(c) Check for 2NF :

- A partial dependency occurs when a non-prime attribute depends on part of the candidate key.
- Here, all non-prime attributes (**name**, **deptid**, **deptname**) depend directly on the candidate key userid.
- Thus, the schema is in 2NF .

(d) Check for 3NF :

- A transitive dependency occurs when a non-prime attribute depends on another non-prime attribute.
- Here:
    - **deptid → deptname**: This is a transitive dependency because **deptid** is not a candidate key.
- Thus, the schema is not in 3NF .

(e) Check for BCNF :

- A relation is in BCNF if every determinant is a superkey.
- Here:
    - **deptid → deptname**: **deptid** is not a superkey.
- Thus, the schema is not in BCNF .

(f) Check for 4NF :

- There are no multi-valued dependencies (MVDs) in this schema.
- Thus, the schema satisfies 4NF .

Step 4: Highest Normal Form

- For the **books** schema:
    - The highest normal form satisfied is 2NF because it violates 3NF due to transitive dependencies and 4NF due to MVDs.

- For the **users** schema:
    - The highest normal form satisfied is 4NF because it satisfies all lower normal forms and has no MVDs.

Thus, the highest normal form satisfied by the schemas collectively is 2NF .

Step 5: Normalize to 4NF

To normalize the schemas to 4NF , we decompose them to eliminate multi-valued dependencies and ensure that every determinant is a superkey.

(a) Decompose books Schema :

- The MVD isbn →→ author must be resolved.
- Create a new relation for the multi-valued dependency:
    - $R1$(isbn, author)
    - Primary Key: (isbn, author)
- Keep the remaining attributes in the original relation:
    - $R2$(accessionno, isbn, title, publisher)
    - Primary Key: accessionno

(b) Decompose users Schema :

- The transitive dependency deptid → deptname must be resolved.
- Create a new relation for the transitive dependency:
    - $R3$(deptid, deptname)
    - Primary Key: deptid
- Keep the remaining attributes in the original relation:
    - $R4$(userid, name, deptid)
    - Primary Key: userid

Final Answer

1. **Highest Normal Form** :

    - The highest normal form satisfied by the schemas is:

$$\boxed{\text{2NF}}$$

2. **4NF Decomposition** :

    - For `books` :

$$R1(\text{isbn, author}),\ R2(\text{accessionno, isbn, title, publisher})$$

    - For `users` :

$$R3(\text{deptid, deptname}),\ R4(\text{userid, name, deptid})$$