DNA Homework -4

Team ID: 5

Krishak Aneja	Nidhish Jain	Saiyam Jain	Varun Gupta
2023101106	2023101071	2023101135	2023101108

PART ONE

Part 1: Analyze Attribute Subsets

(a) ABC

1. **Dependencies**:
   - From F, we have AB → C, AC → B, and BC → A that are relevant for this subset.
   - **Minimal cover**: { AB → C, AC → B, BC → A }
2. **Normal Form**:
   - To determine the strongest normal form, we examine each FD in the minimal cover:
     - AB → C: AB is a candidate key for ABC, so this dependency is in BCNF.
     - AC → B: Similarly, AC is a candidate key, so this dependency is also in BCNF.
     - BC → A: BC is a candidate key, so this dependency is also in BCNF.
   - Since all dependencies in the minimal cover satisfy BCNF, the relation ABC is **already in BCNF**.
3. **Decomposition**:
   - No decomposition is needed, as ABC is in BCNF.

(b) ABCDE

1. **Dependencies**:
   - From F, the relevant dependencies are AB → C, AC → B, AD → E, B → D, and BC → A.
   - **Minimal cover**: { AB → C, AC → B, AD → E, B → D, BC → A }
2. **Normal Form**:
   - Candidate keys: {AB, AC, BC}
   - Since B → D violates BCNF, ABCDE is **not in BCNF**.
   - AD→E and B→D violate 3NF as AD and B are not Candidate keys
   - B→D also violates 2NF because there is a partial dependency as key is AB but D is dependent on only B.

    - Max Normal form is 1NF
3. **Decomposition**:
    - To bring ABCDE to BCNF, decompose it into relations that do not violate BCNF:
        - Split ABCDE into BD and ABCE.
        - ABCE does not violate any FDs.
    - Final BCNF decomposition: { BD, ABCE }

(c) ABCG

1. **Dependencies**:
    - From F, the relevant dependencies are AB → C, AC → B, BC → A, AD → E, B → D and E → G. We get the dependency AB → G syllogistically.
    - **Minimal cover**: { AB → C, AC → B, BC → A, AB → G} (E, D are not relevant here). Candidate keys: {AB, BC, AC}
2. **Normal Form**:
    - Using similar reasoning as in (a), ABCG is in **BCNF** because all FDs involve candidate keys.
3. **Decomposition**:
    - No decomposition is needed, as ABCG is in BCNF.

(d) DEGH

1. **Dependencies**:
    - The only relevant dependency from F is E → G.
    - **Minimal cover**: { E → G }
    - Candidate keys: {DEH}
2. **Normal Form**:
    - E is not a candidate key for DEGH, so E → G violates BCNF.
    - DEGH is **not in BCNF**.
    - We can see that there is partial dependency as E→G so it violates 2NF and hence 3NF and BCNF.
    - Max Normal Form IS 1NF
3. **Decomposition**:
    - Decompose DEGH by splitting out EG.
    - Final BCNF decomposition: { DEH, EG } with all dependencies on Candidate keys.

(e) ABCEH

1. **Dependencies**:
   - Relevant dependencies from F are AB→C , AC→B , BC→A , AB→E (as B → D, AD → E)
   - **Minimal cover**: { AB → C, AC → B, AB→E, BC → A }
   - Candidate keys: {ABH, BCH, ACH}
2. **Normal Form**:
   - The dependency AB → E causes a violation of 2NF (and thus BCNF).
   - ABCEH is **not in BCNF**.
   - It is also not in 2NF.
   - Max Normal Form is 1NF.
3. **Decomposition**:
   - R1(ABE)
   - R2(ABCH)
   - Now we have all dependencies on Candidate Keys. Therefore, It is BCNF and thus no further decomposition is needed

---

Part 2: Analyzing Decompositions

Now, we evaluate the given decompositions for **dependency-preservation** and **lossless-join**. Note, the candidate keys of R are {AB, BC, AC}

(a) { AB, BC, ABDE, EG }

1. **Dependency-Preservation**:
   - We check if all dependencies in F can be represented in this decomposition.
   - This decomposition is **dependency-preserving** because each dependency in F can be derived from one or more of the decomposed relations.
2. **Lossless-Join**:
   - Since ABDE contains the key attributes for R, this decomposition is **lossless-join**.

(b) { ABC, ACDE, ADG }

1. **Dependency-Preservation**:
   - Check if each dependency in F can be represented in this decomposition.
   - This decomposition is **not dependency-preserving**, as some dependencies, such as B → D, cannot be derived from the decomposed relations.
2. **Lossless-Join**:
   - Since ACDE contains a candidate key, this decomposition is **lossless-join**.

PART TWO

Original Table: PURCHASES

(Example of a valid state of the Table:)

<u>Customer ID</u>	**<u>Order ID</u>**	**<u>Product ID</u>**	**Cust Name**	**Product Name**	**Phn Nos**	**Day**	**Discount**
C001	O1001	P001	Alice	ProductA	1234567890, 0987654321	Monday	10%
C002	O1002	P002	Bob	ProductB	9876543210	Tuesday	15%

Given Functional Dependencies:

1. Composite Key: **Customer ID + Order ID + Product ID**
2. **Phn Nos** is a multi-valued attribute (i.e., a customer may have multiple phone numbers).
3. **Day → Discount**: The day determines the discount.
4. **Customer ID → Phn Nos**: A customer has a set of phone numbers.
5. **Customer ID → Cust Name**: A customer has a unique name.
6. **Product ID → Product Name**: A product has a unique name.
7. **Order ID → Day**: An order is placed on a specific day.

---

a) Convert to 1NF (First Normal Form)

1. **Eliminate repeating groups**: The multivalued attribute **Phn Nos** needs to be addressed by creating separate rows for each phone number.
   After conversion, each attribute will have atomic values, and there will be no repeating groups. **Phn Nos** becomes a part of the key of the table. (Another way to resolve this without creating partial dependencies would be to create a separate table with just **Phn Nos** and **Customer ID**, but this is anyways going to be done during conversion to 2NF)

Table in 1NF:

Customer ID	**Order ID**	**Product ID**	**Cust Name**	**Product Name**	**Phn Nos**	**Day**	**Discount**
C001	O1001	P001	Alice	ProductA	1234567890	Monday	10%
C001	O1001	P001	Alice	ProductA	0987654321	Monday	10%
C002	O1002	P002	Bob	ProductB	9876543210	Tuesday	15%

In 1NF, we have eliminated the multi-valued attribute **Phn Nos** by introducing separate rows for each phone number.

---

b) Convert to 2NF (Second Normal Form)

For 2NF:

1. The table should already be in **1NF**.
2. We must eliminate partial dependencies. A partial dependency is when a non-prime attribute depends on only part of a composite key.

Functional Dependencies:

- **Customer ID → Cust Name**
- **Customer ID → Phn Nos**
- **Product ID → Product Name**
- **Order ID → Day**
- **Day → Discount**

Steps to remove partial dependencies:

- **Customer ID → Cust Name** and **Customer ID → Phn Nos** are dependent only on **Customer ID**, part of the composite key. Thus, we can create separate tables for **Customer** and **Phone Numbers**.
- **Product ID → Product Name** is dependent only on **Product ID**, so create a separate **Product** table.
- **Order ID → Day** is dependent only on **Order ID**, so create a separate **Order** table.
- **Day → Discount** shows that **Discount** depends on the **Day**. Syllogistically, **Order ID → Discount** and thus **Discount** can be placed in the table with Order ID as key.

Tables after conversion to 2NF:

Customer Table :
(Cust Name is not a prime attribute)

Customer ID	**Cust Name**
C001	Alice
C002	Bob

Phone Numbers :
(In this table **Phn Nos** is a prime attribute)

Customer ID	**Phn Nos**
C001	1234567890
C001	0987654321
C002	9876543210

Product Table :

Product ID	**Product Name**
P001	ProductA
P002	ProductB

Order Table :

Order ID	**Day**	**Discount**
O1001	Monday	10%
O1002	Tuesday	15%

Purchases Table (linking the normalized tables)

Customer ID	**Order ID**	**Product ID**
C001	O1001	P001
C001	O1001	P001
C002	O1002	P002

In 2NF, we've removed partial dependencies by creating separate tables for customer details, phone numbers, products, and orders, while the **Purchases** table only holds the foreign keys.

---

c) Convert to 3NF (Third Normal Form)

For 3NF:

1. The table should already be in **2NF**.
2. We need to eliminate transitive dependencies. A transitive dependency occurs when one non-prime attribute depends on another non-prime attribute through a third.

Functional Dependencies:

- **Day** → **Discount** is a transitive dependency because **Discount** depends on **Day**, and **Day** depends on **Order ID**.

Steps to remove transitive dependencies:

- We need to separate **Discount** based on **Day** into its own table.

Tables after conversion to 3NF:

Customer Table :

Customer ID	**Cust Name**
C001	Alice
C002	Bob

Phone Numbers :

Customer ID	**Phn Nos**
C001	1234567890
C001	0987654321
C002	9876543210

Product Table :

Product ID	**Product Name**
P001	ProductA
P002	ProductB

Order Table :

Order ID	**Day**
O1001	Monday
O1002	Tuesday

Day-Discount Table :

Day	**Discount**
Monday	10%
Tuesday	15%

Purchases Table (linking the normalized tables)

Customer ID	**Order ID**	**Product ID**
C001	O1001	P001
C001	O1001	P001
C002	O1002	P002

In 3NF, we've removed the transitive dependency (**Day → Discount**) by putting the discount in its own table. Now, each non-prime attribute depends only on the key (no transitive dependencies).
