Last Name: Rachmat First Name: Anthony Student ID: 26339003

1. [20 pts]

Start by looking at the existing UserPhone table from the HW2 solution. Suppose that, in addition to the fields listed there, a new **OS** field is added whose value is defined for mobile phones only, and whose type is an ENUM of 'Windows', 'Android', 'iOS', and 'Other'. Similar to the existing UserPhone table, this modified UserPhone table's DDL includes all of its fields in its primary key. The new CREATE TABLE statement is given below:

```sql
CREATE TABLE UserPhone (
    user_id VARCHAR(20),
    type ENUM('HOME', 'OFFICE', 'MOBILE'),
    number VARCHAR(20),
    os ENUM('Windows', 'Android', 'iOS', 'Other'),
    PRIMARY KEY (user_id, type, number, os),
    FOREIGN KEY (user_id) REFERENCES User (user_id)
);
```

(a) [5 pts] What non-trivial functional dependencies does this current table have, if any? List them all:

There are no non-trivial functional dependencies because each of the dependencies (the RHS) would be a subset of the LHS because all attributes are a part of the key.

(b) [5 pts] Looking at this new CREATE TABLE statement, your boss says that this design doesn't really make sense. When you ask her why, she says "Pull out your phone. Now show me your other phone with the same phone number!" "Ah, got it!" you say. Think about how the world of devices works and then correct the above design by offering a sensible PRIMARY KEY clause below:

PRIMARY KEY (user_id, number)

(c) [5 pts] Given your new PRIMARY KEY clause, what non-trivial functional dependencies does the revised UserPhone table have? List them all here:

(user_id, number) -> type
(user_id, number) -> os

(d) [5 pts] Given the new PRIMARY KEY clause, what normal form is the UserPhone table currently in after following your boss' corrective advice? Briefly show your reasoning.

UserPhone is in BCNF for three reasons; there is no partial dependency, no transitive dependency, and (user_id, number) is a super key. There is no partial dependency because neither the type or os are determined by only user_id or number. There is no transitive dependency because user_id and number directly infer type and os. Type and os are attributes that are functionally dependent on (user_id, number). In other words, the super key (user_id, number) is determining all relations.

2. [20 pts]

Suppose that a coworker, someone who hasn't taken CS-122A (and is not well-versed in ER modeling), merges the OrderItems and Products table together as a replacement for both of them. This person cites "performance" as their reason for doing so, stating that OrderItems and Products are commonly queried as a pair (through SQL joins). Consequently, the resulting CREATE TABLE statement now becomes:

```
CREATE TABLE OrderItemsProducts (
    item_id VARCHAR(15),
    qty INTEGER NOT NULL,
    selling_price DECIMAL(5, 2) NOT NULL,
    order_id VARCHAR(20),
    product_id VARCHAR(20) NOT NULL,
    product_category ENUM(...) NOT NULL,
    product_name VARCHAR(30) NOT NULL,
    product_description VARCHAR(50) NOT NULL,
    product_list_price DECIMAL(5, 2) NOT NULL,
    PRIMARY KEY (item_id, order_id),
    FOREIGN KEY (order_id) REFERENCES Orders (order_id) ON DELETE CASCADE
);
```

(a) [5 pts] What non-trivial functional dependencies does this current table have, if any? List them here:

(item_id, order_id) -> qty

(item_id, order_id) -> selling price

(item_id, order_id) -> product_id

(item_id, order_id) -> product_category

(item_id, order_id) -> product_name

(item_id, order_id) -> product_description

(item_id, order_id) -> product_list_price

(product_id) -> product_category

(product_id) -> product_name

(product_id) -> producct_description

(proudcut_id) -> product_list_price

(b) [5 pts] What is the highest normal form that this current table is in? Answer and explain why.

OrderItemsProducts is in 2NF because it has no partial dependencies but contains transitive dependencies with item_id,order_id -> product_id and product_id -> product_(...). It has no partial dependencies because none of the RHS are dependent on solely one attribute of the primary key on the LHS.

(c) [5 pts] For this “performance-oriented” design, what are the implications on the database in terms of the amount of space that will be used by the new design (versus the old design)?

In the new design with a merged OrderItemsProducts, this table takes up more space because it redundantly stores the same information. For example, if a customer wanted to buy three of the same product, the new design would store three rows of the same product thus taking up more space. The old design with a separate OrderItems and Products table stores only one row of the product information by using the product_id.

(d) [5 pts] For this “performance-oriented” design, what are the implications on the database in terms of stocking products that haven’t been ordered yet?

The implications of this design makes it so the stores are not able to have products in stock until someone places an order. The new design does not take into consideration how stores affect the product.

3. [20 pts]

Consider the following relation:

H	**J**	**G**
h_1	j_5	g_1
h_8	j_4	g_3
h_1	j_5	g_2
h_3	j_8	g_4
h_4	j_4	g_3
h_9	j_5	g_2

(a) [15 pts] Given the current state of the database, for each one of the following functional dependencies answer a) Does this functional dependency hold in the above relation instance [Yes/No]? b) If your answer to the previous question was no, explain why by listing a tuple that causes a violation.

i) G ⟶ H

No because g_2 maps to h_1 and h_9.

ii) H ⟶ J

Yes

iii) J ⟶ H

No because j_4 maps to h_8 and h_4.

iv) H ⟶ G

No because h_1 maps to g_1 and g_2.

v) G ⟶ J

Yes

(b) [5 pts] List all ***potential*** candidate keys (if there are any) for the above relation.

(G,H)

Normalizing a schema with a set of FDs can be done automatically by computers. Complete questions 3 and 4 with the help of the normalization tool provided by Griffith University - *but try each part by hand first*! Use the problems to cement your understanding and use the tool to check your answers.

4. [20 pts]

R(A, B, C, D, E)

(All attributes contain only atomic values.)

FD1: B ⟶ D

FD2: A ⟶ BC

FD3: A ⟶ E

FD4: CD ⟶ E

(a) [5 pts] Compute A+, the attribute closure of attribute A. Show your work as well as the final result.

A->BC

A-> E

A -> BCE

B->D

A-> BCDE

Final answer: A+ = {A,B,C,D,E}

(b) [5 pts] List the candidate keys of R. Is this related to your answer to (a)? Why?

The candidate key of R is A. This is related to our answer in (a) because both answers are consistent with the idea that A is able to infer BCDE.

(c) [5 pts] What's the highest normal form that R satisfies and **why**?

2NF because there are no partial dependencies and the relationship also contains transitivity which is why it is not 3NF.

(d) [5 pts] If R is not already at least in 3NF, then normalize R into 3NF and show the resulting relation(s) and specify their candidate keys. Make sure that your 3NF decomposition is both lossless-join and dependency-preserving. Note: If R was already in 3NF, then just list the candidate keys of R. What is the highest normal form that your answer now satisfies?

3NF Normalization:

- R(A,B,C): candidate key is A
- R(B,D): candidate key is B
- R(C,D,E): candidate key is C

This new relation is lossless-join because when R(A,B,C) U R(B,D) U R(C,D,E) there is no lost data. This new relationship is dependency-preserving because the functional dependencies of R(A,B,C) U R(B,D) U R(C,D,E) are equal to the original functional dependencies before decomposition. The relationship is in BCNF because the super key A determines all attributes of the relation.

5. [20 pts]

R(A, B, C, D, E, F, G)

(All attributes contain only atomic values.)

FD1: B ⟶ D

FD2: AB ⟶ C

FD3: AB ⟶ E

FD4: BC ⟶ E

FD5: F ⟶ G

(a) [5 pts] Compute A+, the attribute closure of attribute A. Show your work and final result.

There is nothing in the attribute closure of A because by only knowing A there are no functional dependencies that allow you to infer the value of any other attribute.

(b) [5 pts] What is the minimal cover for the given set of FDs?

{B->D, AB->C, BC->E, F->G}

(c) [5 pts] Normalize R into BCNF and show the resulting relation(s) and their candidate keys.

BCNF Normalization

R(A,B,F): candidate key is (A,B,F)

R(B,D): candidate key is (B)

R(A,B,C): candidate key is (A,B)

R(B,C,E): candidate Key is (B,C)

R(F,G): candidate key is (F,G)

(d) [5 pts] Is the decomposition in part (c) dependency-preserving? Why or why not?

Yes, the decomposition is dependency-preserving because the outputted relationships reflect the original functional dependency constraints.