REVIEW 4: NORMALISATION AND DEPENDENCIES

TABLE CUSTOMER:

create table customer(account_no int primary key, Name char(20), phone_no int, Email_id varchar(10),DOB date, address varchar(10), user_id varchar(25) unique);

Functional Dependencies:

account_no → Name, phone_no, Email_id, DOB, address, user_id

Normalization Form:

1NF: Atomic values in columns.

2NF: No partial dependencies.

3NF: No transitive dependencies.

Decomposed Table (SQL*Plus):

CREATE TABLE Customer_Details (

account_no INT PRIMARY KEY,

Name CHAR(20),

phone_no INT,

Email_id VARCHAR(10),

DOB DATE,

address VARCHAR(10)

);

CREATE TABLE Customer_Account (

account_no INT PRIMARY KEY,

user_id VARCHAR(25) UNIQUE,

FOREIGN KEY (account_no) REFERENCES Customer_Details(account_no)

);

Pitfalls:

Partial Dependencies: Columns such as phone_no, Email_id, address might depend on the primary key account_no rather than being fully dependent on it. For instance, if phone_no changes, it might require updating multiple rows in the table.

TABLE PAYMENT:

```
CREATE TABLE Payment (
Reference_no int PRIMARY KEY,
Payment_amount DECIMAL(10, 2) NOT NULL,
Mode_of_payment VARCHAR(50) NOT NULL,
Transaction_id VARCHAR(50) NOT NULL UNIQUE,
Account_no int,
Date_of_purchase DATE NOT NULL,
FOREIGN KEY (Account_no) REFERENCES Account(Account_no)
);
```

Functional Dependencies:

Reference_no → Payment_amount, Mode_of_payment, Transaction_id, Account_no, Date_of_purchase

Normalization Form:

1NF: Atomic values in columns.

2NF: No partial dependencies.

3NF: No transitive dependencies.

Decomposed Table (SQL*Plus):

```
-- Assume Account is decomposed to Account_Details and Account_Transactions
CREATE TABLE Payment (
  Reference_no INT PRIMARY KEY,
  Payment_amount DECIMAL(10, 2) NOT NULL,
  Mode_of_payment VARCHAR(50) NOT NULL,
  Transaction_id VARCHAR(50) NOT NULL UNIQUE,
  Account_no INT,
```

Date_of_purchase DATE,

FOREIGN KEY (Account_no) REFERENCES Account_Details(Account_no)

);

Pitfalls:

Redundancy: Depending on the broader schema, there might be redundancy if payment-related information is also stored elsewhere (e.g., customer's account details).

TABLE CREDIT CARD:

CREATE TABLE CreditCard (

card_id INT PRIMARY KEY,

customer_id INT NOT NULL,

card_number VARCHAR(16) NOT NULL,

expiration_date DATE NOT NULL

card_holder_name VARCHAR(255) NOT NULL,

billing_address varchar(20)

FOREIGN KEY (customer_id) REFERENCES customer(account_no)

);

Functional Dependencies:

card_id → customer_id, card_number, expiration_date, card_holder_name

Normalization Form:

1NF: Atomic values in columns.

2NF: No partial dependencies.

3NF: No transitive dependencies.

Decomposed Table (SQL*Plus):

-- Assume Customer is decomposed to Customer_Details and Customer_Account

CREATE TABLE CreditCard (

card_id INT PRIMARY KEY,

customer_id INT,

card_number VARCHAR(16) NOT NULL,

expiration_date DATE NOT NULL,

card_holder_name VARCHAR(255) NOT NULL,

FOREIGN KEY (customer_id) REFERENCES Customer_Account(account_no)

);

Pitfalls:

Transitive Dependencies: If card_number determines expiration_date or card_holder_name, this can lead to transitive dependencies.

TABLE DEBIT CARD:

CREATE TABLE DebitCard (

customer_id INT primary key,

card_number VARCHAR(16) NOT NULL,

expiration_date DATE NOT NULL,

card_holder_name VARCHAR(255) NOT NULL,

amount decimal(18,2),

FOREIGN KEY (customer_id) REFERENCES customer(account_no)

);

Functional Dependencies:

customer_id → card_number, expiration_date, card_holder_name, amount

Normalization Form:

1NF: Atomic values in columns.

2NF: No partial dependencies.

3NF: No transitive dependencies.

Decomposed Table (SQL*Plus):

-- Assume Customer is decomposed to Customer_Details and Customer_Account

CREATE TABLE DebitCard (

customer_id INT PRIMARY KEY,

card_number VARCHAR(16) NOT NULL,

expiration_date DATE NOT NULL,

card_holder_name VARCHAR(255) NOT NULL,

amount DECIMAL(18, 2),

FOREIGN KEY (customer_id) REFERENCES Customer_Account(account_no)

);

Pitfalls:

Transitive Dependencies: Similar to the CreditCard table, there may be transitive dependencies if expiration_date or card_holder_name rely on card_number rather than the primary key.

TABLE LOAN1:

CREATE TABLE loan(

account_no int,

loan_type int,

loan_no varchar(20),

amount int,

interest int,

PRIMARY KEY (account_no),

FOREIGN KEY (account_no) REFERENCES customer(account_no));

Functional Dependencies:

account_no → loan_type, loan_no, amount, interest

Normalization Form:

1NF: Atomic values in columns.

2NF: No partial dependencies.

3NF: No transitive dependencies.

Decomposed Table (SQL*Plus):

-- Assume Customer is decomposed to Customer_Details and Customer_Account

CREATE TABLE Loan_Details (

account_no INT PRIMARY KEY,

loan_type INT,

loan_no VARCHAR(20),

amount INT,

interest INT,

FOREIGN KEY (account_no) REFERENCES Customer_Account(account_no)

);

Pitfalls:

Functional Dependencies: If attributes like loan_type, loan_no, amount, or interest depend on each other rather than solely on the primary key (account_no), it could lead to functional dependency issues.