DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By

SRN :	Name :	Evaluation date and time:
PES1201801994	H GURURAGHAVENDA	

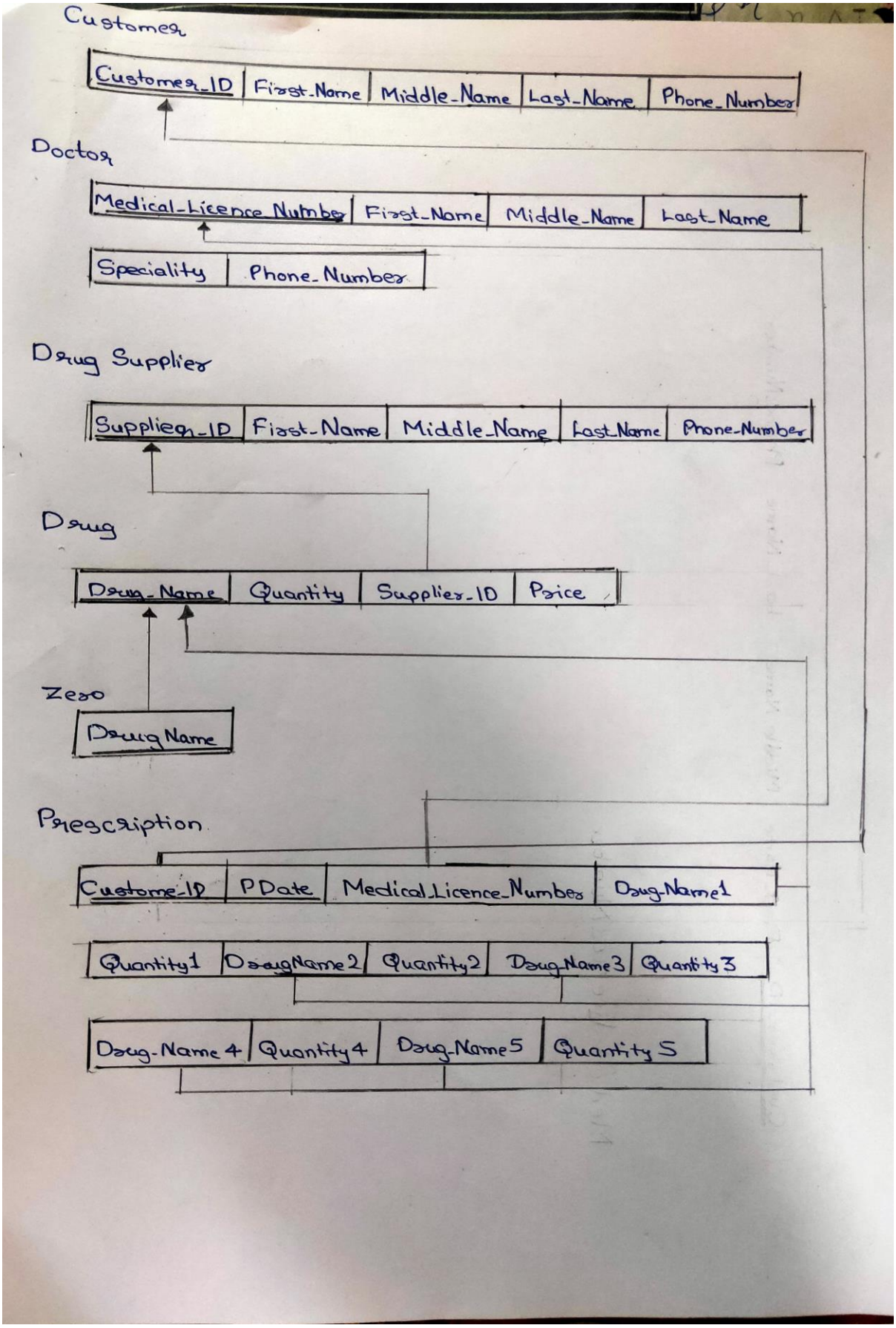
Functional Dependencies	**1**
Identifying Keys based on FDs	**1**
Normalization & testing for lossless join property	**2+1**
DDL: Table creation with all constraints	**2+**Error! Bookmark not defined.
Triggers	**5**
SQL Queries	**2**
Viva / modifications (Unit III/ IV concepts)	**2+2**

<< Pharmacy Database>>

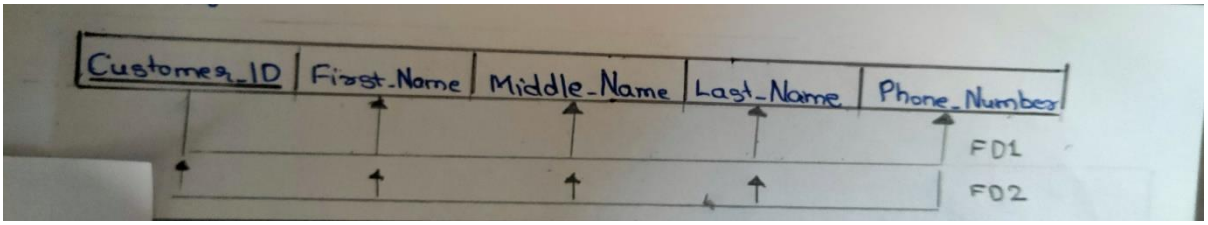
Problem statement:

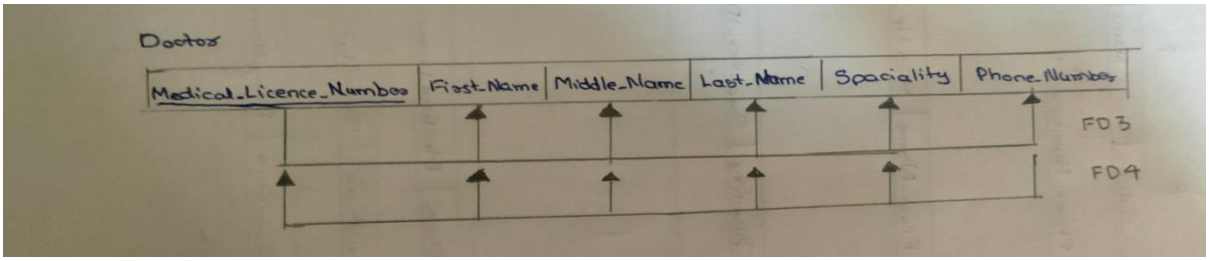
A Pharmacy database needs to store information about customers(identified by Customer_Id, with First_Name, Middle_Name, Last_Name and Phone_Number attributes),
Doctor(identified by Medical_Licence_Number, with First_Name, Middle_Name, Last_Name, Speciality and Phone_Number attributes), Drug_Supplier(identified with Supplier_ID, First_Name, Middle_Name, Last_Name and Phone_Number attributes), Drug(identified with Drug Name, with Supplier_ID, Quantity and price attributes), Zero(identified with Drug Name) for storing the drug names whose quantity is zero, And Prescription(identified with customer_ID and PDate and with Medical_Licence_Number of the doctor who issued the prescription and the Drug Names from 1 to 5 and their respective quantities.

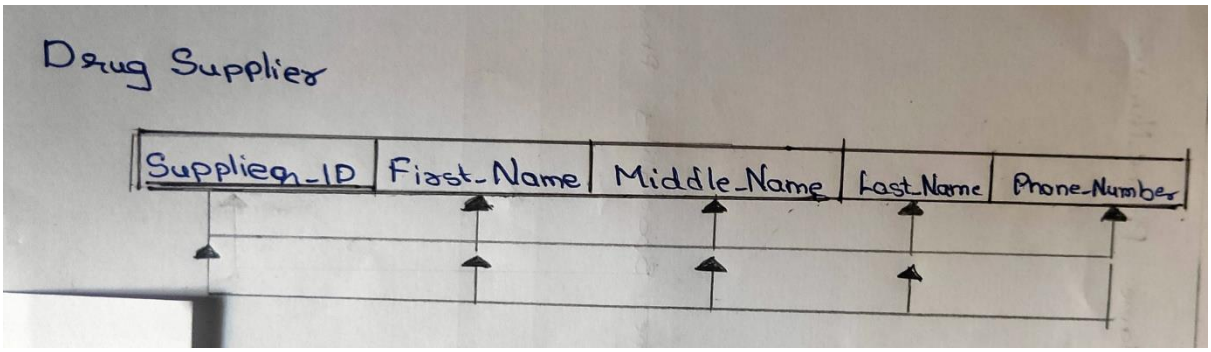
Database Schema: (show all the tables and the constraints)

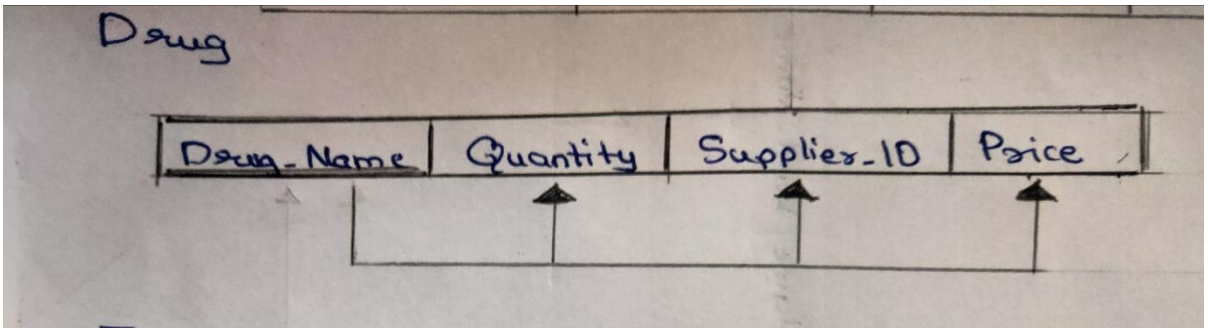


Functional Dependencies: (List based on your application constraints)









Candidate keys: (Justify how did you get these as keys)

In Customer Table: Phone_Number is Candidate key because it can uniquely define a person (customer).
In Doctor Table : Phone_Number is Candidate key because it can uniquely define a person (Doctor).
In Drug_supplier Table: Phone_Number is Candidate key because it can uniquely define a Drug_supplier.

Primary keys:

In Customer Table: Customer_ID - because it can uniquely define a customer.
In Doctor Table: Medical_Licence_Number - because it uniquely defines a doctor
In Drug supplier Table: Supplier_ID – because it uniquely defines a supplier.
In Drug Table: Drug_Name - each drug is uniquely defined by the drug name.
In prescription Table: Customer_ID and PDate together define a prescription.

Normalization and testing for lossless join property:

A relation is in **first normal form** if and only if the domain of each attribute contains only atomic values and the value of each attribute contains only a single value from that domain.

This Database is in first normal form because all the attributes are atomic.

A relation is in **second normal form** if every non-prime attribute A in R is fully functionally dependent on the primary key of R.

i.e there is no partial dependencies.

In table pharmacy

Customer_ID
PDate
Medical_License_Number
Drug_Name1
Quantity1
Drug_Name2
Quantity2`
Drug_Name3
Quantity3
Drug_Name4
Quantity4
Drug_Name5
Quantity5
Total

In this table if we add customer name to this table, this becomes partially dependent since the primary key for this table is combination of customer_ID and PDate but by only Customer_ID we can get the customer name.

A relation R is in **third normal form** if, whenever a nontrival function dependency X->A holds on R, then either X is super key of R or A is a prime attribute of R.
i.e there is no transitive dependency.

This Database **is in third normal form** since it doesn't violate the 3NF as it doesn't have a transitive dependency.

A relation R is in **BNCF** if, whenever a nontrivial function dependency X->A holds on R, then X is only super key of R.

This Database is **not in BNCF** since phone number is unique for every row.

A relation R is in **fourth normal form** with respect to a set dependencies F(that includes fictional dependencies and multivalued dependencies) if, for every nontrivial multivalued dependency X->>Y in F, X is a superkey for R.

i.e the **4NF** is a level of database normalization where there is no non-trivial multivalued dependencies other than a candidate key. It builds on the first three normal forms .

By the above definition the database **follows the 4NF** as it has no multi-valued dependency

A relation R is in **fifth normal form** if and only if every non-trivial join dependency in that table is implied by the candidate keys.
The database **is in 5NF**

DDL:

Create table scripts here. Ensure integrity constraints are defined.

Add sample insert statements as well, that you would be using for demo.

table creation commads

```
CREATE TABLE Customer(
    Customer_ID CHAR(5) NOT NULL PRIMARY KEY,
    First_Name VARCHAR(15) NOT NULL,
    Middle_Name VARCHAR(15) NOT NULL,
    Last_Name VARCHAR(15) NOT NULL,
    Phone_Number CHAR(10) NOT NULL  CHECK(length(Phone_Number) = 10) );

CREATE TABLE Doctor(
    Medical_License_Number CHAR(5) NOT NULL PRIMARY KEY,
    First_Name VARCHAR(15) NOT NULL,
    Middle_Name VARCHAR(15) NOT NULL,
    Last_Name VARCHAR(15) NOT NULL,
    Speciality VARCHAR(15),
    Phone_Number CHAR(10) NOT NULL  CHECK(length(Phone_Number) = 10) );

CREATE TABLE Drug_Supplier(
```

```
    Supplier_ID CHAR(5) NOT NULL PRIMARY KEY,
    First_Name VARCHAR(15) NOT NULL,
    Middle_Name VARCHAR(15) NOT NULL,
    Last_Name VARCHAR(15) NOT NULL,
    Phone_Number CHAR(10) NOT NULL  CHECK(length(Phone_Number) = 10));


CREATE TABLE Drug(
    Drug_Name VARCHAR(30) NOT NULL PRIMARY KEY,
    Supplier_ID CHAR(5) NOT NULL,
    Quantity INTEGER NOT NULL,
    Price INTEGER NOT NULL,
    FOREIGN KEY (Supplier_ID)
      REFERENCES Drug_Supplier (Supplier_ID));

CREATE TABLE Zero(
    Drug_Name VARCHAR(30) NOT NULL PRIMARY KEY,
    FOREIGN KEY (Drug_Name)
      REFERENCES Drug (Drug_Name));


CREATE TABLE Prescription(
    Customer_ID CHAR(5) NOT NULL ,
    PDate Date NOT NULL,
    Medical_License_Number CHAR(5) NOT NULL ,
    Drug_Name1 VARCHAR(30) NOT NULL,
    Quantity1 INTEGER DEFAULT 0,
    Drug_Name2 VARCHAR(30) DEFAULT NULL,
    Quantity2 INTEGER DEFAULT 0,
    Drug_Name3 VARCHAR(30) DEFAULT NULL,
    Quantity3 INTEGER DEFAULT 0,
    Drug_Name4 VARCHAR(30) DEFAULT NULL,
    Quantity4 INTEGER DEFAULT 0,
    Drug_Name5 VARCHAR(30) DEFAULT NULL,
    Quantity5 INTEGER DEFAULT 0,
    FOREIGN KEY (Customer_ID)
      REFERENCES Customer (Customer_ID),
    FOREIGN KEY (Medical_License_Number)
      REFERENCES Doctor (Medical_License_Number),
    FOREIGN KEY (Drug_Name1)
      REFERENCES Drug (Drug_Name),
    FOREIGN KEY (Drug_Name2)
      REFERENCES Drug (Drug_Name),
    FOREIGN KEY (Drug_Name3)
      REFERENCES Drug (Drug_Name),
    FOREIGN KEY (Drug_Name4)
      REFERENCES Drug (Drug_Name),
    FOREIGN KEY (Drug_Name5)
      REFERENCES Drug (Drug_Name),
    PRIMARY KEY(customer_ID,PDate)
    );

###Insert into commands
```

```
INSERT INTO Customer Values('00001','A','B','C',1111111111);
INSERT INTO Customer Values('00002','D','E','F',1111111112);
INSERT INTO Customer Values('00003','G','H','I',1111111113);
INSERT INTO Customer Values('00004','J','K','L',1111111114);
INSERT INTO Customer Values('00005','M','N','O',1111111115);

INSERT INTO Doctor Values('10001','A','B','C','ABC',3111111110);
INSERT INTO Doctor Values('10002','D','E','F','XYZ',3111111111);

INSERT INTO Drug_Supplier Values('20001','A','B','C',2111111111);
INSERT INTO Drug_Supplier Values('20002','D','E','F',2111111112);
INSERT INTO Drug_Supplier Values('20003','G','H','I',2111111113);



INSERT INTO Drug Values('AA','20000',12,14);
INSERT INTO Drug Values('BB','20001',10,12);
INSERT INTO Drug Values('CC','20002',100,13);
INSERT INTO Drug Values('DD','20002',130,14);
INSERT INTO Drug Values('EE','20003',140,15);
```

Triggers:

1. Identify a constraint to implement as a trigger and write the English statement for that.
2. Write the trigger creation statement along with any stored procedures/functions involved.

Trigger zero is for when the drug quantity is 0:

```
CREATE TRIGGER zero AFTER UPDATE ON Drug
    WHEN NEW.Quantity =0
            BEGIN
                    INSERT INTO Zero Values(NEW.Drug_Name);
            END;
```

Trigger NOTzero is for when the drug quantity is changed from zero to other number :

```
 CREATE TRIGGER NOTzero AFTER UPDATE ON Drug
    WHEN NEW.Quantity <>0 AND OLD.Quantity=0
            BEGIN
                    DELETE FROM Zero WHERE Drug_Name=NEW.Drug_Name;
            END;
```

SQL Queries:

<Write a few english sentences and SQL queries for them. Ensure at least 2 correlated-nested Advanced and 2 aggregate queries. >

1) List of names of suppliers form whom no drugs are purchased

```
CREATE VIEW V1 AS SELECT Supplier_ID,First_Name,Middle_Name,Last_Name
FROM Drug_Supplier AS ds WHERE ds.Supplier_ID  IN
( SELECT Supplier_ID FROM Drug_Supplier EXCEPT SELECT Supplier_ID FROM Drug);

SELECT * FROM V1
```

2) Speciality of the doctors of the customers who have visited till now.

```
SELECT DISTINCT Speciality
 FROM
( Doctor NATURAL JOIN Prescription );
```

3) Total day's revenue of a particular day

```
SELECT SUM(Total)
 FROM Prescription
 WHERE PDate="2020-05-28";
```

4) Revenue from a particular doctor.

```
Medical_License_Number , SUM(Total)
 FROM
 (  Doctor NATURAL JOIN Prescription )
  GROUP BY Medical_License_Number;
```