DBMS PROJECT MILESTONE-1 ER DIAGRAM

PROJECT TITLE:-

COLLEGE DISPENSARY DATABASE

BY-

KUSHAGRA KANCHAN - U22CS113

JIGAR JAT - U22CS114

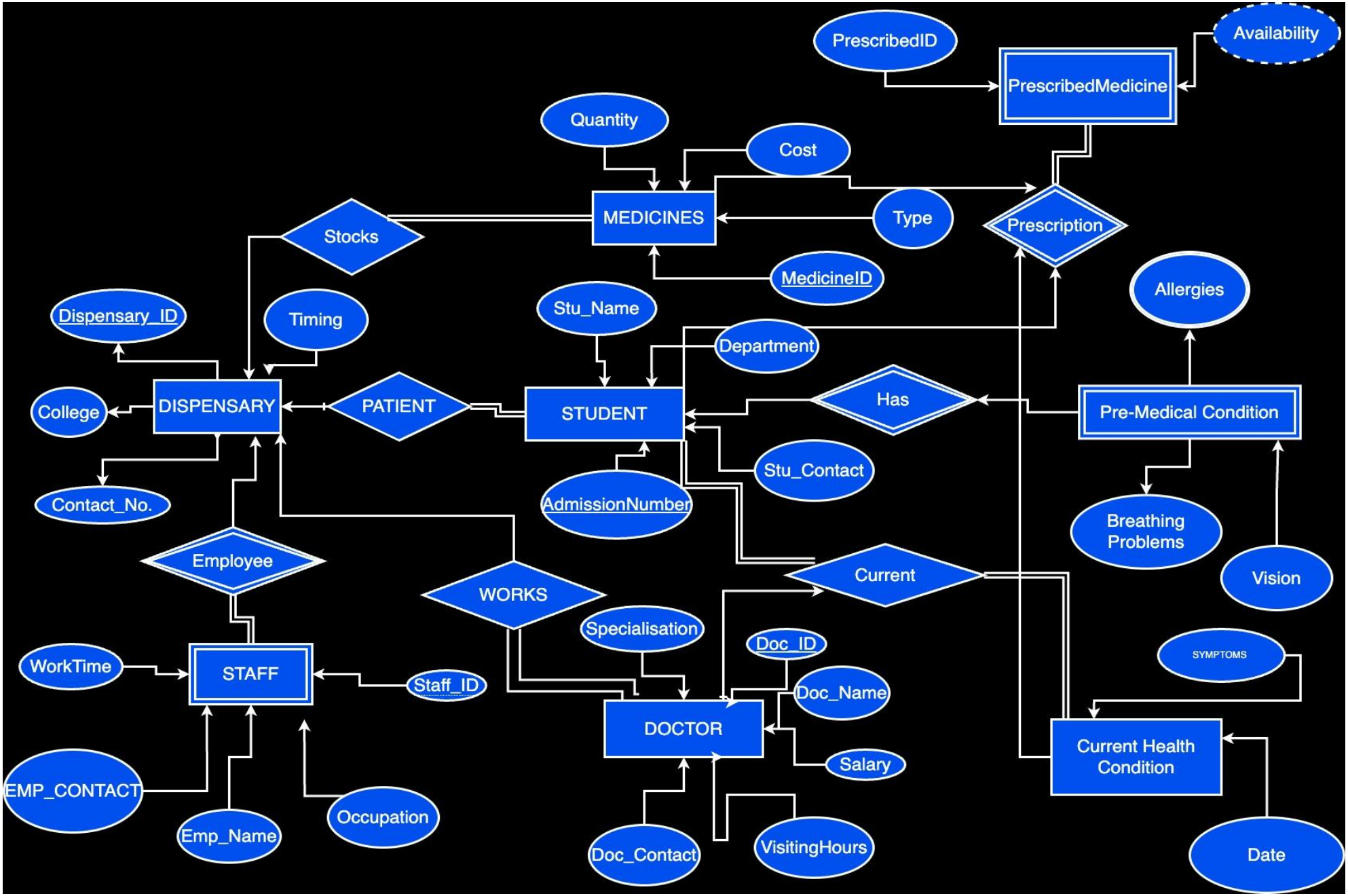
LAKSHITA JAWANDHIYA - U22CS126

AJAY SOLANKI - U22CS138

ENTITY	TYPE	ATTRIBUTES	KEYS	NORMALIZATION
DISPENSARY	STRONG	Dispensary_ID,College,Contact, Timing	PRIMARY KEY = Dispensary_ID	Fourth normal form (4NF)
STUDENT	STRONG	Stu_Name,AdmissionNumber, Department,Stu_Contact	PRIMARY KEY = AdmissionNumber FOREIGN KEY = Dispensary_ID	Fourth normal form (4NF)
MEDICINES	STRONG	Quantity, Cost, Type, MedicineID	PRIMARY KEY = MedicineID FOREIGN KEY = Dispensary_ID	Fourth normal form (4NF)
DOCTOR	STRONG	Specialization, Doc_ID, Doc_Name, Salary, VisitingHours, Doc_Contact	PRIMARY KEY = Doc_ID FOREIGN KEY = Dispensary_ID	Fourth normal form (4NF)
STAFF	WEAK	Staff_ID, WorkTime, Emp_Contact, Emp_Name, Occupation	PRIMARY KEY = Staff_ID FOREIGN KEY = Dispensary_ID	Fourth normal form (4NF)
CURRENT HEALTH CONDITIONS	STRONG	Symptoms, Date	FOREIGN KEY = AdmissionNumber	Fourth normal form (4NF)
PRE-MEDICAL CONDITIONS	WEAK	Allergies, Breathing Problems, Vision	FOREIGN KEY = AdmissionNumber	Fourth normal form (4NF)

PRESCRIBED MEDICINE	WEAK	PrescribedID, Availability	PRIMARY KEY = PrescribedID FOREIGN KEY = MedicineID	Fourth Normal form(4NF)

ER DIAGRAM —————————>


PrescribedID
Availability
PrescribedMedicine
Quantity
Cost
MEDICINES
Type
Stocks
Prescription
MedicineID
Allergies
Dispensary_ID
Timing
Stu_Name
Department
College
DISPENSARY
PATIENT
STUDENT
Has
Pre-Medical Condition
Stu_Contact
Contact_No.
AdmissionNumber
Breathing Problems
Vision
Employee
WORKS
Current
Specialisation
Doc_ID
SYMPTOMS
WorkTime
STAFF
Staff_ID
Doc_Name
DOCTOR
Salary
Current Health Condition
EMP_CONTACT
Emp_Name
Occupation
Doc_Contact
VisitingHours
Date

DBMS PROJECT MILESTONE-2

PL/SQL SCRIPTING DOCUMENT

PROJECT TITLE: *COLLEGE DISPENSARY DATABASE*

BY-

KUSHAGRA KANCHAN U22CS113

JIGAR JAT U22CS114

LAKSHITA JAWANDHIYA U22CS126

AJAY SOLANKI U22CS138

SQL SCRIPTS

```
SET SERVEROUTPUT ON;
CREATE TABLE DISPENSARY (
    Dispensary_ID INT PRIMARY KEY,
    College VARCHAR(255),
    Contact VARCHAR(255),
    Times VARCHAR(255)
);


CREATE TABLE STUDENT (
    Stu_Name VARCHAR(255),
    AdmissionNumber INT PRIMARY KEY,
    Department VARCHAR(255),
    Stu_Contact VARCHAR(255),
    Dispensary_ID INT,
    FOREIGN KEY (Dispensary_ID) REFERENCES
DISPENSARY(Dispensary_ID)
);

-- Creating table MEDICINES
CREATE TABLE MEDICINES (
    MedicineID INT PRIMARY KEY,
    Quantity INT,
    MedCost DECIMAL(10, 2),
    MedType VARCHAR(255),
    Dispensary_ID INT,
    MinAmount INT,
    ReOrder INT default 0,
    FOREIGN KEY (Dispensary_ID) REFERENCES
DISPENSARY(Dispensary_ID)
);
```

```
-- Creating table DOCTOR
CREATE TABLE DOCTOR (
    Specialization VARCHAR(255),
    Doc_ID INT PRIMARY KEY,
    Doc_Name VARCHAR(255),
    Salary DECIMAL(10, 2),
    VisitingHours VARCHAR(255),
    Doc_Contact VARCHAR(255),
    Dispensary_ID INT,
    FOREIGN KEY (Dispensary_ID) REFERENCES
DISPENSARY(Dispensary_ID)
);

-- Creating table STAFF
CREATE TABLE STAFF (
    Staff_ID INT PRIMARY KEY,
    WorkTime VARCHAR(255),
    Emp_Contact VARCHAR(255),
    Emp_Name VARCHAR(255),
    Occupation VARCHAR(255),
    Dispensary_ID INT,
    FOREIGN KEY (Dispensary_ID) REFERENCES
DISPENSARY(Dispensary_ID)
);

-- Creating table CURRENT_HEALTH_CONDITIONS
CREATE TABLE CURRENT_HEALTH_CONDITIONS (
    Symptoms VARCHAR(255),
    VisitDate DATE,
    AdmissionNumber INT,
    FOREIGN KEY (AdmissionNumber) REFERENCES
STUDENT(AdmissionNumber)
);
```

```
-- Creating table PRE_MEDICAL_CONDITIONS
CREATE TABLE PRE_MEDICAL_CONDITIONS (
    Allergies VARCHAR(255),
    BreathingProblems VARCHAR(255),
    Vision VARCHAR(255),
    AdmissionNumber INT,
    FOREIGN KEY (AdmissionNumber) REFERENCES
STUDENT(AdmissionNumber)
);

-- Creating table PRESCRIBED_MEDICINE
CREATE TABLE PRESCRIBED_MEDICINE (
    PrescribedID INT PRIMARY KEY,
    MedAvailability VARCHAR(255),
    MedicineID INT,
    FOREIGN KEY (MedicineID) REFERENCES
MEDICINES(MedicineID)
);

-- Create a trigger to handle quantity decrement and reorder flag
update
CREATE OR REPLACE TRIGGER UpdateMedicineQuantity
AFTER INSERT ON PRESCRIBED_MEDICINE
FOR EACH ROW
DECLARE
    v_MinAmount MEDICINES.MinAmount%TYPE;
    v_Quantity MEDICINES.Quantity%TYPE;
BEGIN
    -- Fetch the minimum amount and current quantity of the
prescribed medicine
    SELECT MinAmount, Quantity INTO v_MinAmount, v_Quantity
    FROM MEDICINES
    WHERE MedicineID = :NEW.MedicineID;

    -- Decrement the quantity by 1
```

```
    UPDATE MEDICINES
    SET Quantity = Quantity - 1
    WHERE MedicineID = :NEW.MedicineID;

    -- Check if the quantity is lower than the minimum amount and
update the reorder flag
    IF (v_Quantity - 1) < v_MinAmount THEN
        UPDATE MEDICINES
        SET ReOrder = 1
        WHERE MedicineID = :NEW.MedicineID;
   END IF;
END;
/
CREATE OR REPLACE TRIGGER UpdateReOrderFlag
BEFORE UPDATE OF Quantity, MinAmount ON MEDICINES
FOR EACH ROW
BEGIN
    IF :NEW.Quantity >= :NEW.MinAmount THEN
        :NEW.ReOrder := 0;
    ELSIF :NEW.Quantity < :NEW.MinAmount THEN
        :NEW.ReOrder := 1;
    END IF;
END UpdateReOrderFlag;
/


CREATE OR REPLACE PROCEDURE
insert_prescribed_medicine(
    p_prescribed_id IN INT,
    p_med_availability IN VARCHAR2,
    p_medicine_id IN INT
) AS
BEGIN
    INSERT INTO PRESCRIBED_MEDICINE (PrescribedID,
MedAvailability, MedicineID)
```

```
    VALUES (p_prescribed_id, p_med_availability, p_medicine_id);
    COMMIT;
END insert_prescribed_medicine;
/
CREATE OR REPLACE PROCEDURE
delete_prescribed_medicine(
    p_prescribed_id IN INT
) AS
BEGIN
    DELETE FROM PRESCRIBED_MEDICINE
    WHERE PrescribedID = p_prescribed_id;
    COMMIT;
END delete_prescribed_medicine;
/
CREATE OR REPLACE PROCEDURE
update_prescribed_medicine(
    p_prescribed_id IN INT,
    p_med_availability IN VARCHAR2,
    p_medicine_id IN INT
) AS
BEGIN
    UPDATE PRESCRIBED_MEDICINE
    SET MedAvailability = p_med_availability, MedicineID =
p_medicine_id
    WHERE PrescribedID = p_prescribed_id;
    COMMIT;
END update_prescribed_medicine;
/
CREATE OR REPLACE PROCEDURE update_medicine_quantity(
    p_medicine_id IN INT,
    p_new_quantity IN INT,
    p_new_minamount IN INT
) AS
BEGIN
    UPDATE MEDICINES
```

```
    SET Quantity = p_new_quantity,
    MinAmount = p_new_minamount
    WHERE MedicineID = p_medicine_id;
    COMMIT;
END update_medicine_quantity;
/
CREATE OR REPLACE PROCEDURE insert_medicine(
    p_medicine_id IN INT,
    p_quantity IN INT,
    p_med_cost IN DECIMAL,
    p_med_type IN VARCHAR2,
    p_dispensary_id IN INT,
    p_min_amount IN INT
) AS
BEGIN
    INSERT INTO MEDICINES (MedicineID, Quantity, MedCost,
MedType, Dispensary_ID, MinAmount)
    VALUES (p_medicine_id, p_quantity, p_med_cost, p_med_type,
p_dispensary_id, p_min_amount);
    COMMIT;
END insert_medicine;
/
CREATE OR REPLACE PROCEDURE update_medicine(
    p_medicine_id IN INT,
    p_quantity IN INT,
    p_med_cost IN DECIMAL,
    p_med_type IN VARCHAR2,
    p_dispensary_id IN INT,
    p_min_amount IN INT,
    p_reorder IN INT
) AS
BEGIN
    UPDATE MEDICINES
    SET Quantity = p_quantity,
        MedCost = p_med_cost,
```

```
        MedType = p_med_type,
        Dispensary_ID = p_dispensary_id,
        MinAmount = p_min_amount,
        ReOrder = p_reorder
    WHERE MedicineID = p_medicine_id;
    COMMIT;
END update_medicine;
/
CREATE OR REPLACE PROCEDURE delete_medicine(
    p_medicine_id IN INT
) AS
BEGIN
    DELETE FROM MEDICINES
    WHERE MedicineID = p_medicine_id;
    COMMIT;
END delete_medicine;
/

CREATE OR REPLACE PROCEDURE ReorderMedicine IS
MedID Medicines.MedicineID%TYPE;
MCost Medicines.MedCost%TYPE;
MType Medicines.MedType%TYPE;
CURSOR ReorderMed IS
SELECT MedicineID,MedCost,MedType
    FROM Medicines
    WHERE Reorder = 1;
BEGIN
OPEN ReorderMed;
LOOP
    FETCH ReorderMed INTO MedID, Mcost, Mtype;
    EXIT WHEN ReorderMed%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(MedID ||' -> '|| MCost || ' -> ' ||
Mtype);
END LOOP;
END;
```

```
/
-- Create the stored function to return the count of rows with
ReOrder = 1
CREATE OR REPLACE FUNCTION GetReOrderCount RETURN
INT IS
  v_count INT;
BEGIN
  SELECT COUNT(*)
  INTO v_count
  FROM MEDICINES
  WHERE ReOrder = 1;

  RETURN v_count;
END GetReOrderCount;
/

BEGIN
    -- Inserting data into DISPENSARY table
    INSERT INTO DISPENSARY (Dispensary_ID, College, Contact,
Times)
    VALUES (1, 'ABC College', '1234567890', '9:00 AM - 5:00 PM');

    -- Inserting data into STUDENT table
    INSERT INTO STUDENT (Stu_Name, AdmissionNumber,
Department, Stu_Contact, Dispensary_ID)
    VALUES ('John Doe', 1001, 'Computer Science', '9876543210',
1);

    -- Inserting data into MEDICINES table
    INSERT INTO MEDICINES (MedicineID, Quantity, MedCost,
MedType, Dispensary_ID)
    VALUES (1, 50, 10.99, 'Painkiller', 1);

    -- Inserting data into DOCTOR table
```

```
    INSERT INTO DOCTOR (Specialization, Doc_ID, Doc_Name,
Salary, VisitingHours, Doc_Contact, Dispensary_ID)
    VALUES ('Cardiology', 2001, 'Dr. Smith', 100000.00, '10:00 AM -
2:00 PM', '9876543210', 1);

    -- Inserting data into STAFF table
    INSERT INTO STAFF (Staff_ID, WorkTime, Emp_Contact,
Emp_Name, Occupation, Dispensary_ID)
    VALUES (10001, 'Full Time', '9999999999', 'Jane Doe', 'Nurse',
1);

    -- Inserting data into CURRENT_HEALTH_CONDITIONS table
    INSERT INTO CURRENT_HEALTH_CONDITIONS (Symptoms,
VisitDate, AdmissionNumber)
    VALUES ('Fever', SYSDATE, 1001);

    -- Inserting data into PRE_MEDICAL_CONDITIONS table
    INSERT INTO PRE_MEDICAL_CONDITIONS (Allergies,
BreathingProblems, Vision, AdmissionNumber)
    VALUES ('Pollen', 'Asthma', 'Normal', 1001);

    -- Inserting data into PRESCRIBED_MEDICINE table
    INSERT INTO PRESCRIBED_MEDICINE (PrescribedID,
MedAvailability, MedicineID)
    VALUES (1001, 'In Stock', 1);

    -- Commit the transaction here to make the changes permanent
    COMMIT;
END;
/
-- Insert the first row
INSERT INTO DISPENSARY (Dispensary_ID, College, Contact,
Times)
VALUES (2, 'College A', 'Contact A', 'Times A');
```

```
-- Insert the second row
INSERT INTO DISPENSARY (Dispensary_ID, College, Contact,
Times)
VALUES (3, 'College B', 'Contact B', 'Times B');

-- Insert the third row
INSERT INTO DISPENSARY (Dispensary_ID, College, Contact,
Times)
VALUES (4, 'College C', 'Contact C', 'Times C');

-- Continue with additional rows as needed
INSERT INTO DISPENSARY (Dispensary_ID, College, Contact,
Times)
VALUES (5, 'College D', 'Contact D', 'Times D');

INSERT INTO DISPENSARY (Dispensary_ID, College, Contact,
Times)
 VALUES (6, 'College E', 'Contact E', 'Times E');



-- Insert the second row
INSERT INTO STUDENT (Stu_Name, AdmissionNumber,
Department, Stu_Contact, Dispensary_ID)
VALUES ('Jane Doe', 1002, 'Electrical Engineering', 'Contact B', 2);

-- Insert the third row
INSERT INTO STUDENT (Stu_Name, AdmissionNumber,
Department, Stu_Contact, Dispensary_ID)
VALUES ('Alice Johnson', 1003, 'Mechanical Engineering', 'Contact
C', 3);

-- Continue with additional rows as needed
INSERT INTO STUDENT (Stu_Name, AdmissionNumber,
Department, Stu_Contact, Dispensary_ID)
```

```
VALUES ('Bob Williams', 1004, 'Civil Engineering', 'Contact D', 4);

INSERT INTO STUDENT (Stu_Name, AdmissionNumber,
Department, Stu_Contact, Dispensary_ID)
VALUES ('Eve Wilson', 1005, 'Chemistry', 'Contact E', 5);

-- Insert the first row

-- Insert the second row
INSERT INTO MEDICINES (MedicineID, Quantity, MedCost,
MedType, Dispensary_ID, MinAmount)
VALUES (2, 50, 5.99, 'Antibiotic', 2, 10);

-- Insert the third row
INSERT INTO MEDICINES (MedicineID, Quantity, MedCost,
MedType, Dispensary_ID, MinAmount)
VALUES (3, 200, 25.99, 'Allergy Medication', 3, 30);

-- Continue with additional rows as needed
INSERT INTO MEDICINES (MedicineID, Quantity, MedCost,
MedType, Dispensary_ID, MinAmount)
VALUES (4, 75, 7.99, 'Cough Syrup', 1, 15);

INSERT INTO MEDICINES (MedicineID, Quantity, MedCost,
MedType, Dispensary_ID, MinAmount)
VALUES (5, 120, 12.49, 'Painkiller', 2, 25);

-- Insert the first doctor
INSERT INTO DOCTOR (Specialization, Doc_ID, Doc_Name,
Salary, VisitingHours, Doc_Contact, Dispensary_ID)
VALUES ('Cardiologist', 1, 'Dr. Smith', 120000.00, 'Mon-Fri 9 AM - 5
PM', 'Contact Doc 1', 1);

-- Insert the second doctor
```

```
INSERT INTO DOCTOR (Specialization, Doc_ID, Doc_Name,
Salary, VisitingHours, Doc_Contact, Dispensary_ID)
VALUES ('Pediatrician', 2, 'Dr. Johnson', 95000.00, 'Tue-Thu 10 AM
- 3 PM', 'Contact Doc 2', 2);

-- Insert the first staff member
INSERT INTO STAFF (Staff_ID, WorkTime, Emp_Contact,
Emp_Name, Occupation, Dispensary_ID)
VALUES (1, 'Full-Time', 'Contact Staff 1', 'John Smith', 'Nurse', 1);

-- Insert the second staff member
INSERT INTO STAFF (Staff_ID, WorkTime, Emp_Contact,
Emp_Name, Occupation, Dispensary_ID)
VALUES (2, 'Part-Time', 'Contact Staff 2', 'Alice Johnson',
'Receptionist', 2);

-- Insert the first health condition
INSERT INTO CURRENT_HEALTH_CONDITIONS (Symptoms,
VisitDate, AdmissionNumber)
VALUES ('Fever, Cough', TO_DATE('2023-11-01', 'YYYY-MM-DD'),
1001);

-- Insert the second health condition
INSERT INTO CURRENT_HEALTH_CONDITIONS (Symptoms,
VisitDate, AdmissionNumber)
VALUES ('Sore Throat, Headache', TO_DATE('2023-11-03',
'YYYY-MM-DD'), 1002);

-- Insert the first pre-medical condition
INSERT INTO PRE_MEDICAL_CONDITIONS (Allergies,
BreathingProblems, Vision, AdmissionNumber)
VALUES ('Pollen Allergy', 'Asthma', '20/20', 1001);

-- Insert the second pre-medical condition
```

```
INSERT INTO PRE_MEDICAL_CONDITIONS (Allergies,
BreathingProblems, Vision, AdmissionNumber)
VALUES ('None', 'None', '20/20', 1002);

UPDATE medicines
SET minamount = 10
WHERE medicineid = 1;

DECLARE
  reorder_count INT;
BEGIN
  reorder_count := GetReOrderCount;
  DBMS_OUTPUT.PUT_LINE('Count of rows with ReOrder = 1: ' ||
reorder_count);
END;
/
```

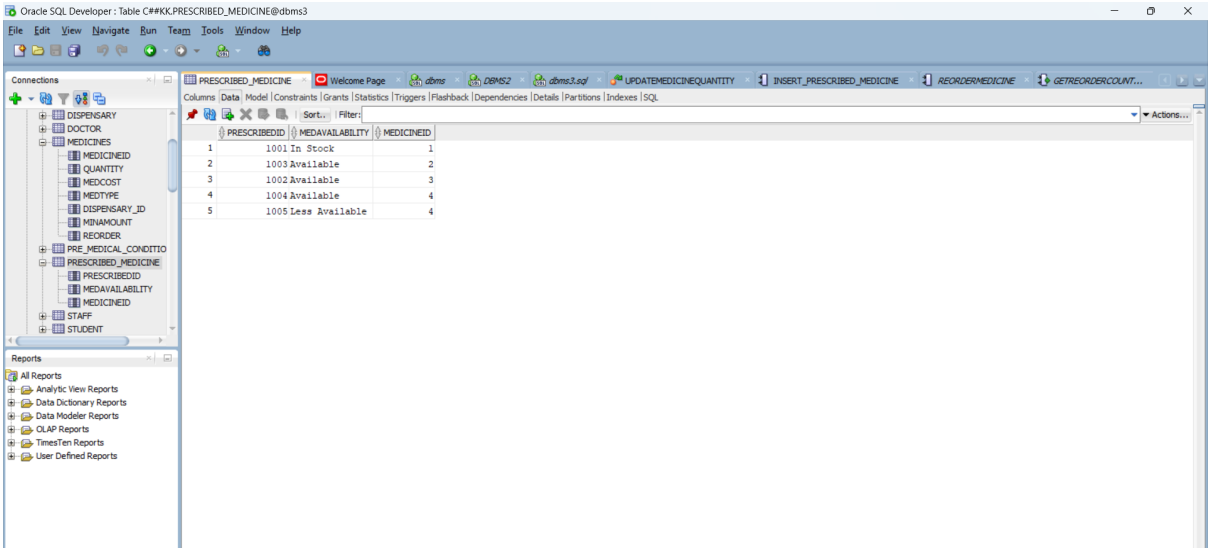
QUERY OUTPUTS

1.

```
EXEC
insert_prescribed_medicine(1005,'Less
Available',4);
EXEC
update_prescribed_medicine(1002,'Not
available',1);
```

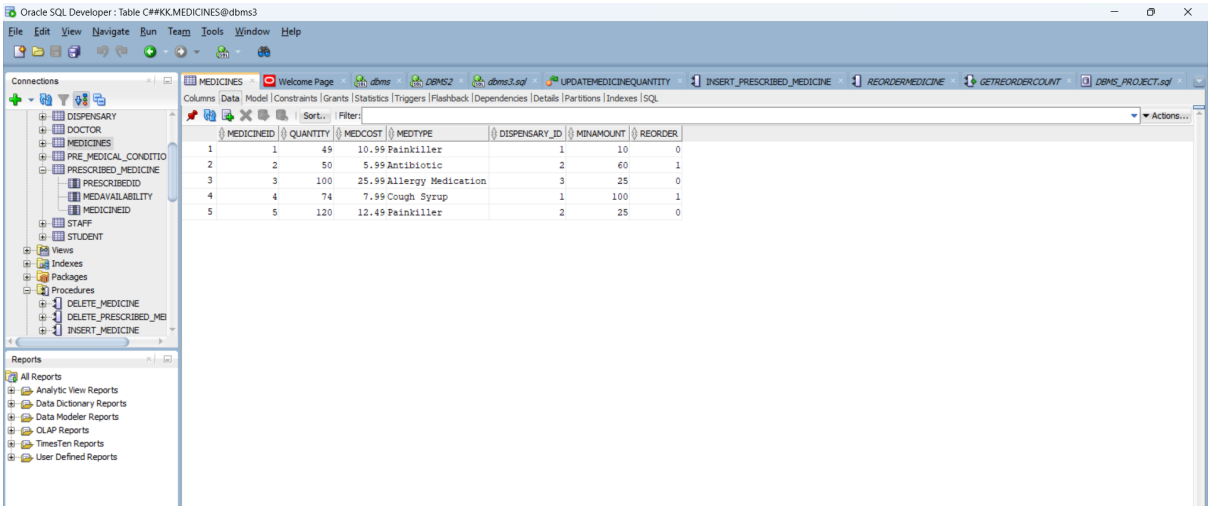
EXEC
delete_prescribed_medicine(1002);

DESCRIPTION: THESE QUERIES UPDATE, DELETE, INSERT ROWS IN THE TABLE.



2.
EXEC
update_medicine_quantity(3,100,25);

DESCRIPTION: UPDATES MEDICINE QUANTITY OF ID 3 AND CHANGES REORDER FLAG AS WELL.

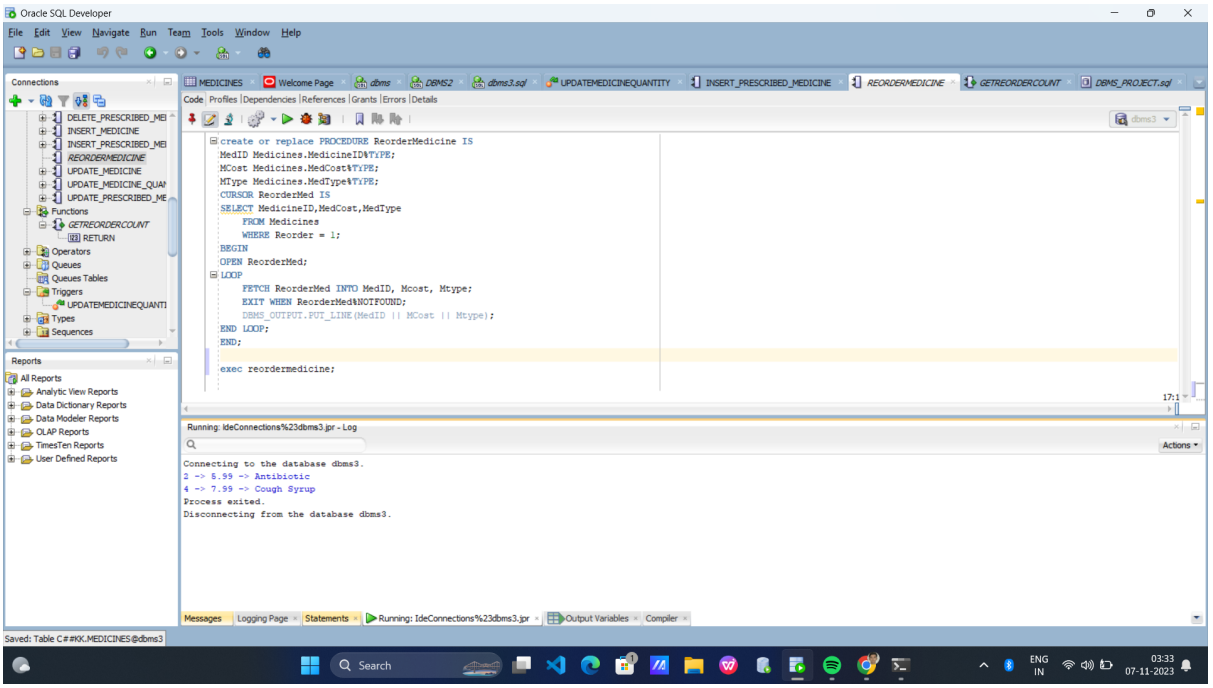


3.

```
create or replace PROCEDURE
ReorderMedicine IS
MedID Medicines.MedicineID%TYPE;
MCost Medicines.MedCost%TYPE;
MType Medicines.MedType%TYPE;
CURSOR ReorderMed IS
SELECT MedicineID,MedCost,MedType
    FROM Medicines
    WHERE Reorder = 1;
BEGIN
OPEN ReorderMed;
LOOP
```

```
    FETCH ReorderMed INTO MedID,
Mcost, Mtype;
    EXIT WHEN ReorderMed%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(MedID ||
MCost || Mtype);
END LOOP;
END;

exec reordermedicine;
```

DESCRIPTION: Stores the medicine in a **cursor** whose quantity has dropped below the minamount and need to be reordered.



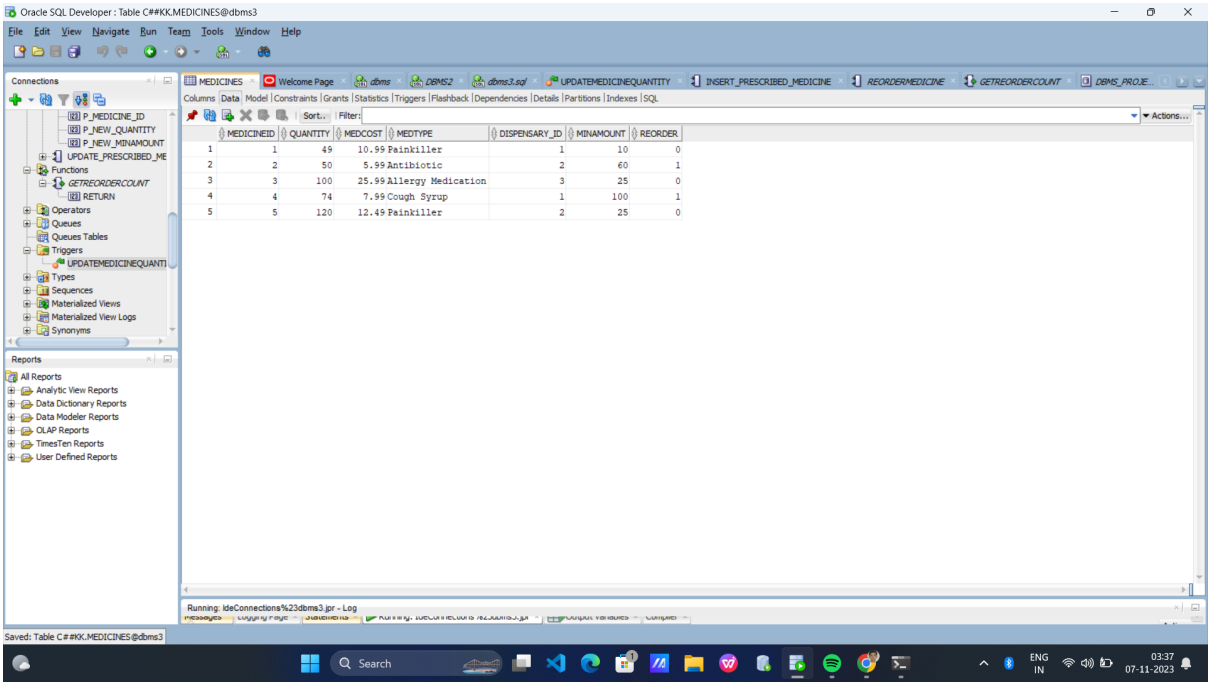
4.

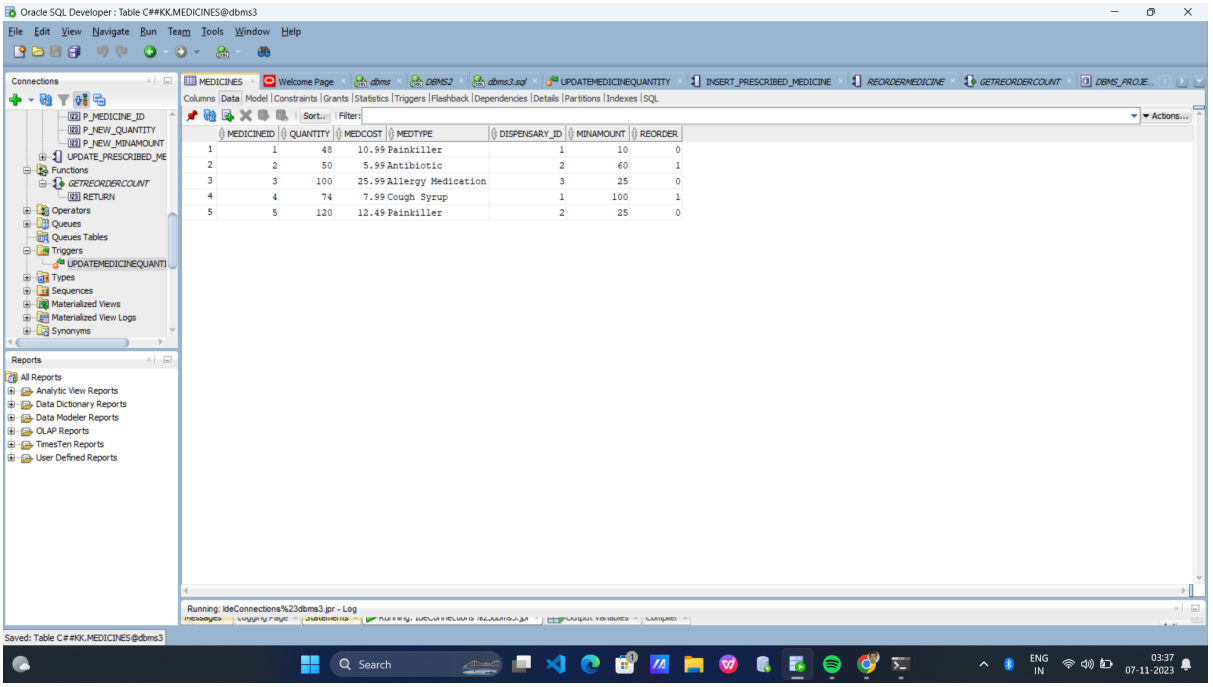
```
create or replace TRIGGER
UpdateMedicineQuantity
AFTER INSERT ON
PRESCRIBED_MEDICINE
FOR EACH ROW
DECLARE
    v_MinAmount
MEDICINES.MinAmount%TYPE;
    v_Quantity MEDICINES.Quantity%TYPE;
BEGIN
    -- Fetch the minimum amount and current
quantity of the prescribed medicine
    SELECT MinAmount, Quantity INTO
v_MinAmount, v_Quantity
    FROM MEDICINES
    WHERE MedicineID = :NEW.MedicineID;

    -- Decrement the quantity by 1
    UPDATE MEDICINES
    SET Quantity = Quantity - 1
    WHERE MedicineID = :NEW.MedicineID;
```

```
    -- Check if the quantity is lower than the
minimum amount and update the reorder
flag
    IF (v_Quantity - 1) < v_MinAmount THEN
        UPDATE MEDICINES
        SET ReOrder = 1
        WHERE MedicineID =
:NEW.MedicineID;
   END IF;
END;
```

DESCRIPTION: This trigger performs the function that whenever data is inserted in prescribed medicine the quantity of the particular medicine decreases.

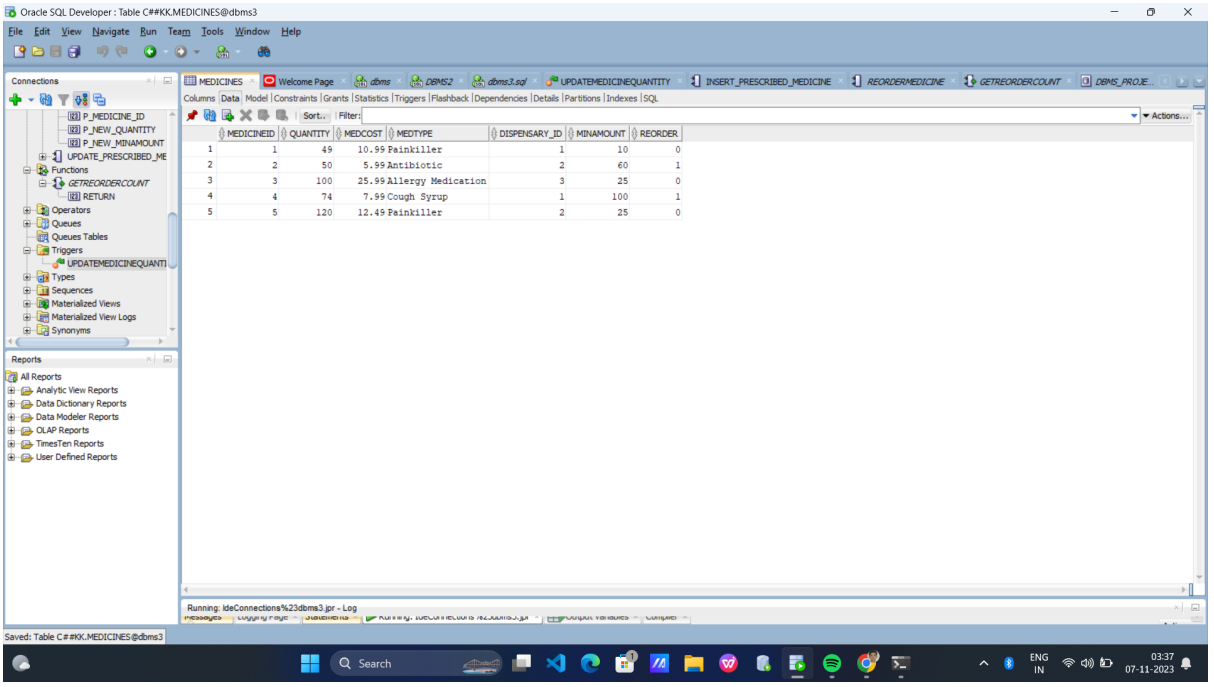
BEFORE AND AFTER INSERTION -


Oracle SQL Developer : Table C##KK.MEDICINES@dbms3
File Edit View Navigate Run Team Tools Window Help
Connections
Reports
MEDICINES
Welcome Page
Columns Data Model Constraints Grants Statistics Triggers Flashback Dependencies Details Partitions Indexes SQL
Sort... Filter: Actions...
MEDICINEID QUANTITY MEDCOST MEDTYPE DISPENSARY_ID MINAMOUNT REORDER
1 1 49 10.99 Painkiller 1 10 0
2 2 50 5.99 Antibiotic 2 60 1
3 3 100 25.99 Allergy Medication 3 25 0
4 4 74 7.99 Cough Syrup 1 100 1
5 5 120 12.49 Painkiller 2 25 0
Running: IdeConnections%23dbms3.jpr - Log
Saved: Table C##KK.MEDICINES@dbms3
Search
ENG IN
03:37 07-11-2023

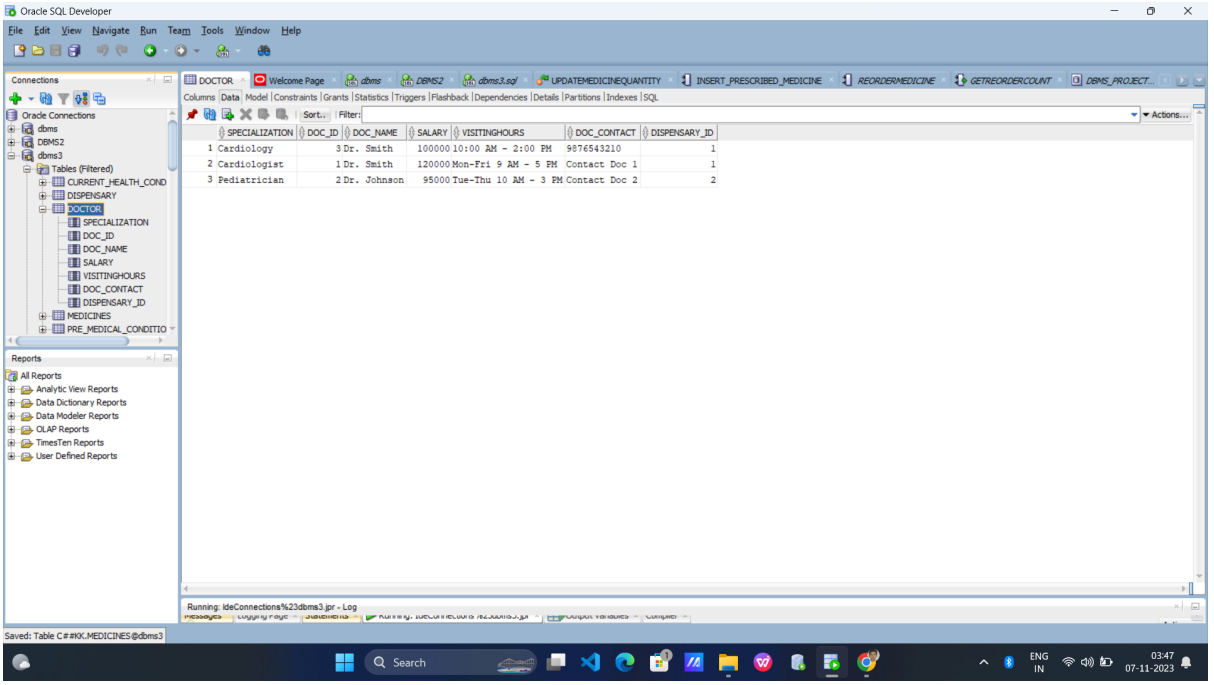

Oracle SQL Developer : Table C##KK.MEDICINES@dbms3
File Edit View Navigate Run Team Tools Window Help
Connections
Reports
MEDICINES
Welcome Page
Columns Data Model Constraints Grants Statistics Triggers Flashback Dependencies Details Partitions Indexes SQL
Sort... Filter: Actions...
MEDICINEID QUANTITY MEDCOST MEDTYPE DISPENSARY_ID MINAMOUNT REORDER
1 1 48 10.99 Painkiller 1 10 0
2 2 50 5.99 Antibiotic 2 60 1
3 3 100 25.99 Allergy Medication 3 25 0
4 4 74 7.99 Cough Syrup 1 100 1
5 5 120 12.49 Painkiller 2 25 0
Running: IdeConnections%23dbms3.jpr - Log
Saved: Table C##KK.MEDICINES@dbms3
Search
ENG IN
03:37 07-11-2023

TABLES-

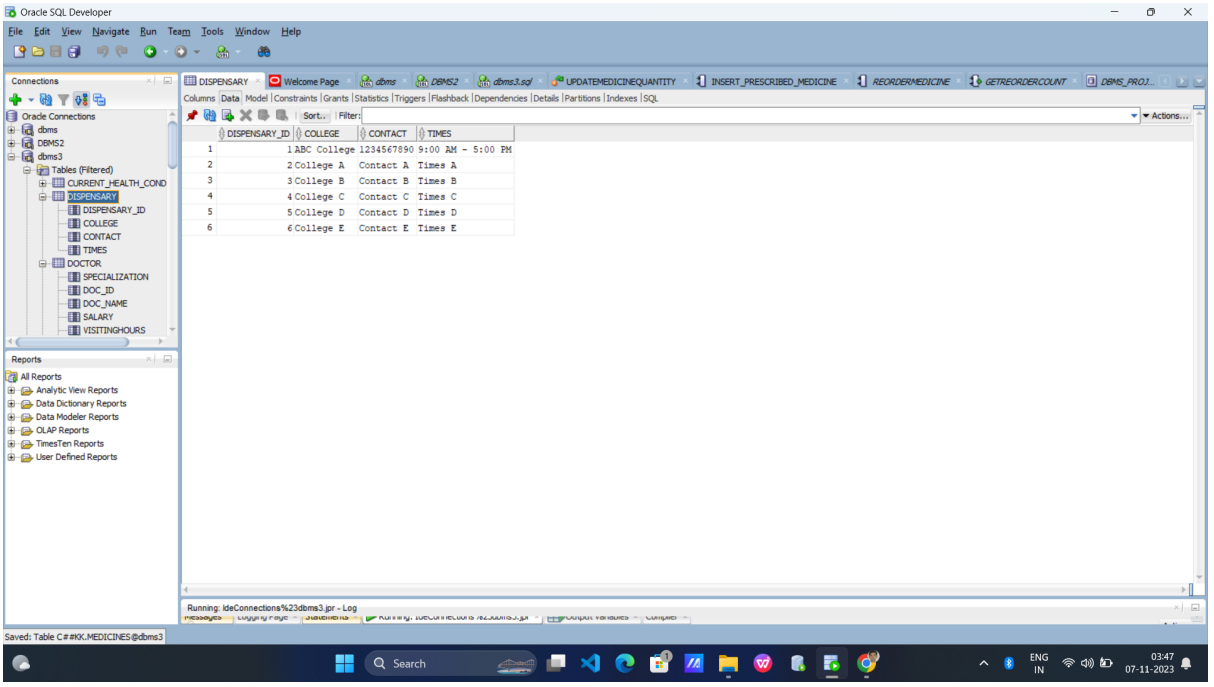
1. MEDICINES



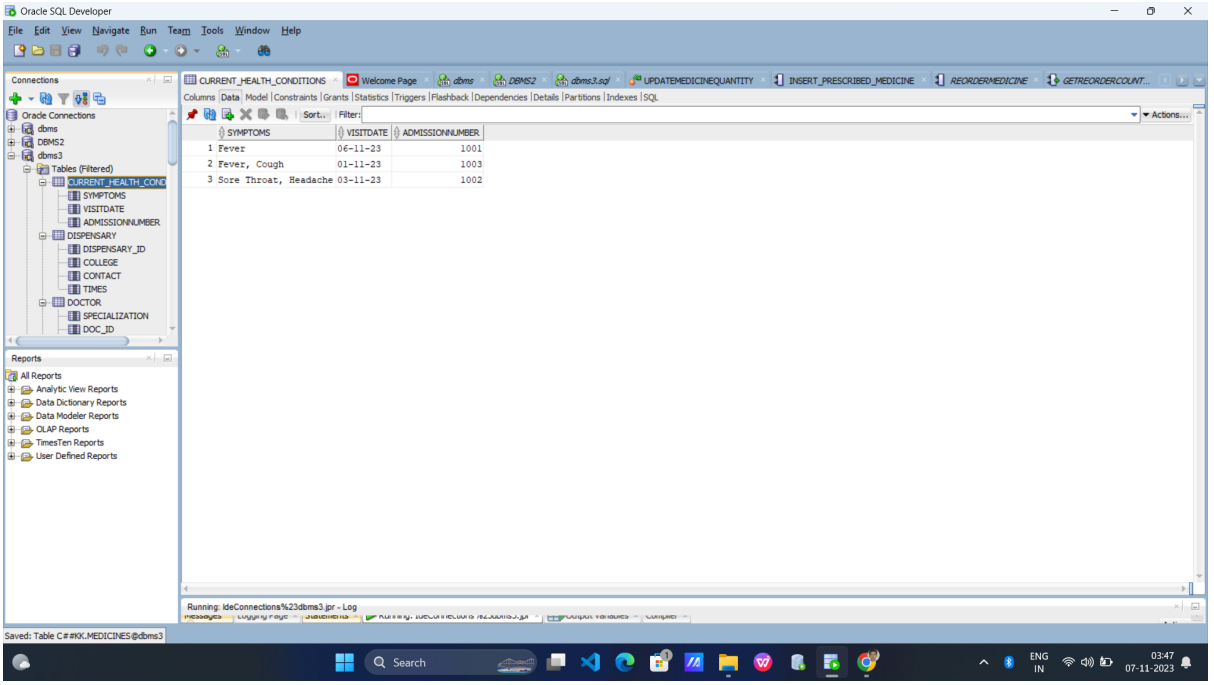
2. DOCTORS



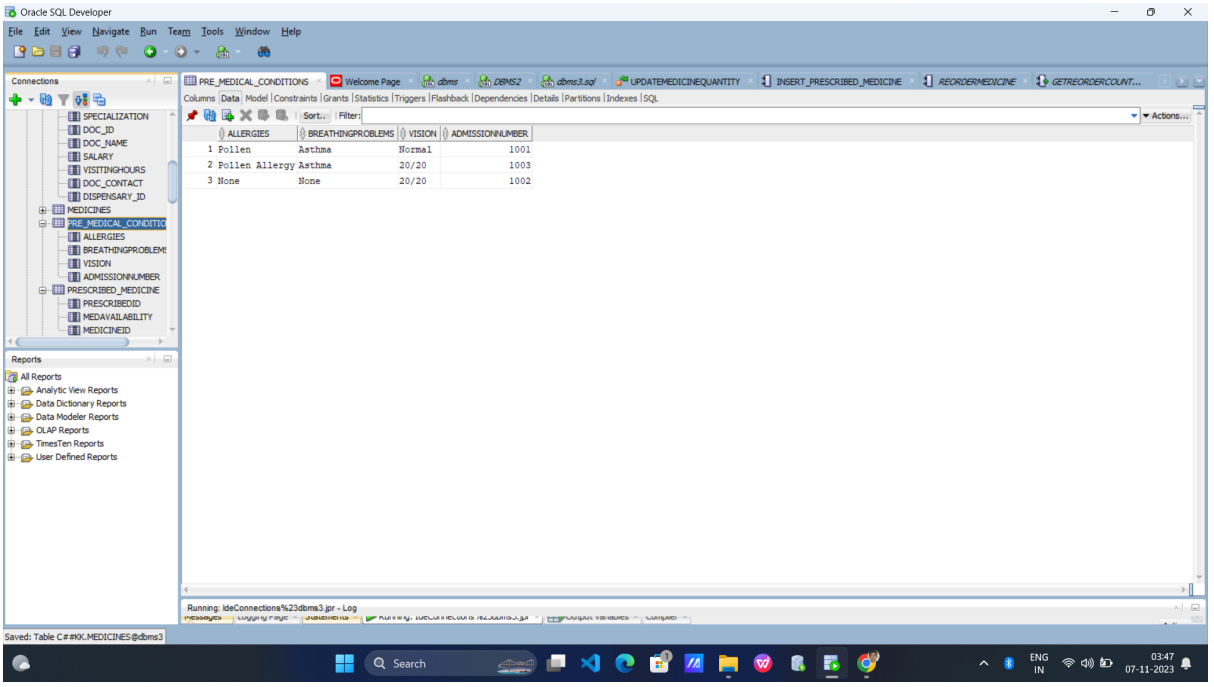
3. DISPENSARY



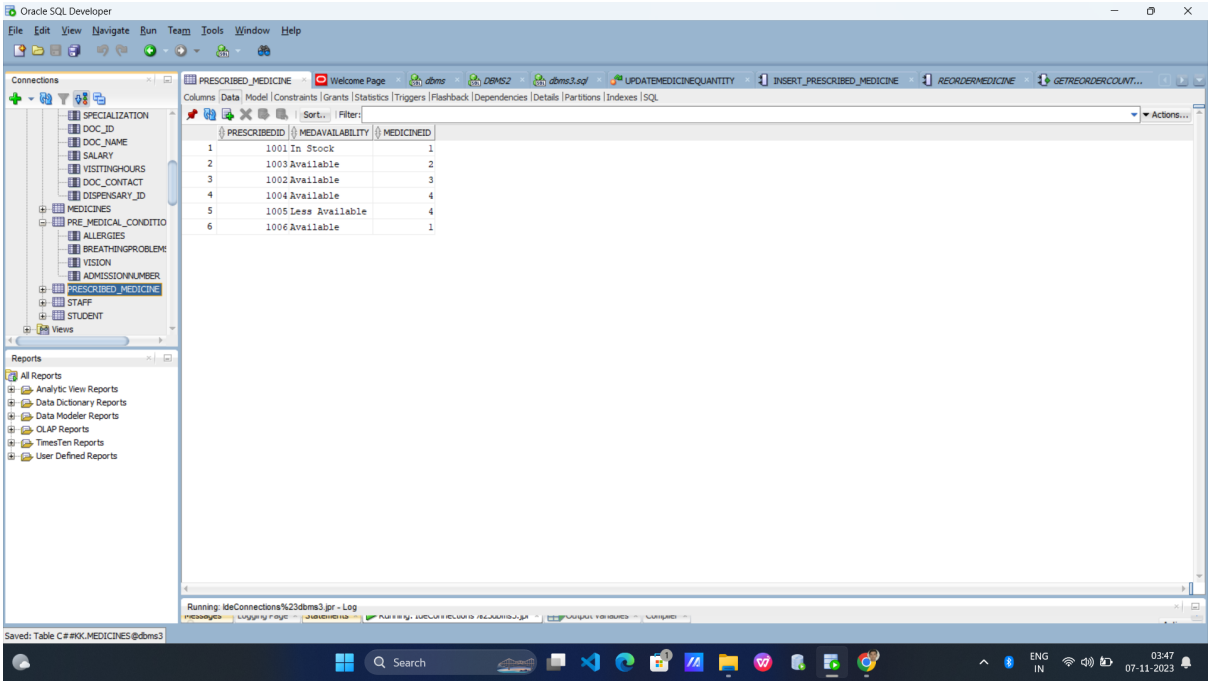
4. CURRENT HEALTH CONDITIONS



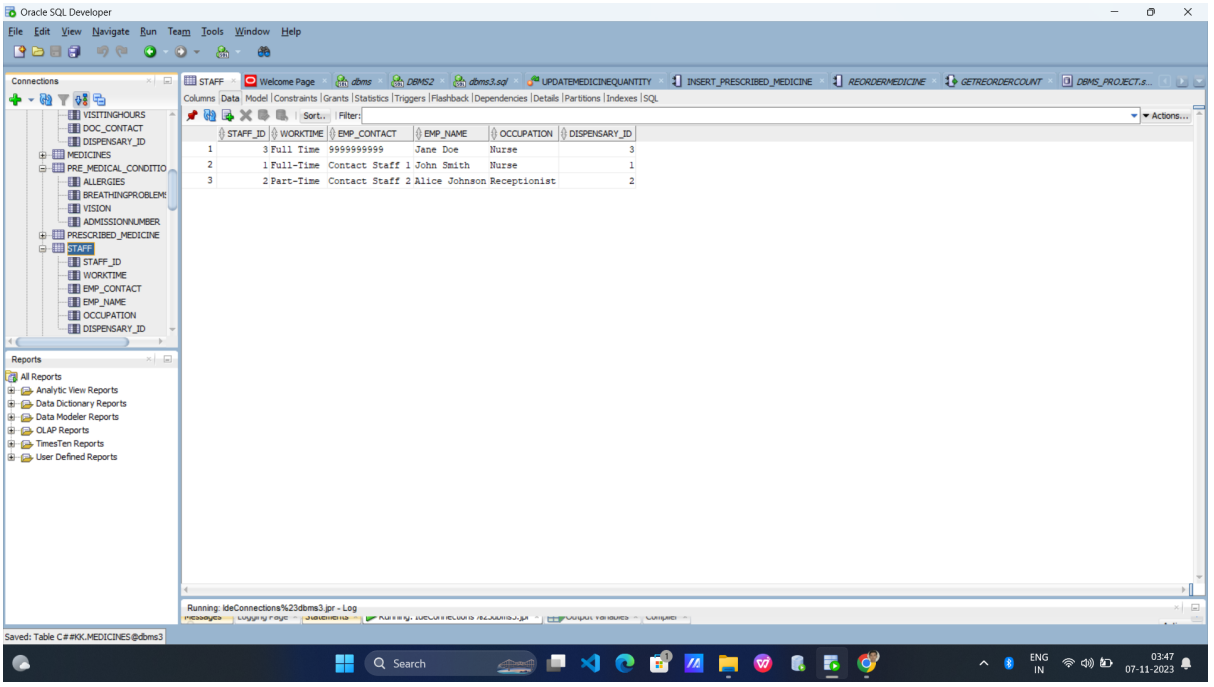
5. PRE-MEDICAL_CONDITIONS



6. PRESCRIBED MEDICINE



7. STAFF



8. STUDENT