UCS310: Database Management System Project



Pharmacy Management System

EVENSEM 2018-19

SUBMITTED BY SUBMITTED TO

Abhishek Bansal: 101703022(COE1) Ms. Harkiran Kaur

Abhishek Maira: 101703026(COE2)

Akshita Goyal:101703052(COE1) SUBMITTED ON

07 May 2019

DEVELOPED IN

Oracle Sql Plus for SQL and PL/SQL

INDEX

1. Introduction 3
2. ER diagram of the database 4
3. Conversion of ER diagram to tables 5
4. Normalized 6
5. PL/SQL queries and snap shots 8
6. Snap Shots of the tables 12
7. References 23

INTRODUCTION

REQUIREMENT ANALYSIS

**Problem Statement: Develop a general purpose Pharmacy Management System to be used by a parent pharmacy.**

**Objective**

It is a user friendly application for pharmacists which reduces the burden of their work and helps them to manage pharmacy sections like medicine management, billing etc. which improves processing efficiency. It deals with the automating task of maintaining bills. In pharmacy, billing management is the key process.

**Need**

The process of designing and implementing record keeping systems has significant accountability and resource implications for an organization. Feasibility analysis will help you make informed and transparent decisions at crucial points during the developmental process to determine whether it is operationally, economically and technically realistic to proceed with a particular course of action.

**Functional requirement**

1. The admin’s privileges are role based, thereby allowing role based access control of security.

2. Each user’s IP address is stored in the database for security.

3. Any user of the system is banned after 4 failed authentication processes.

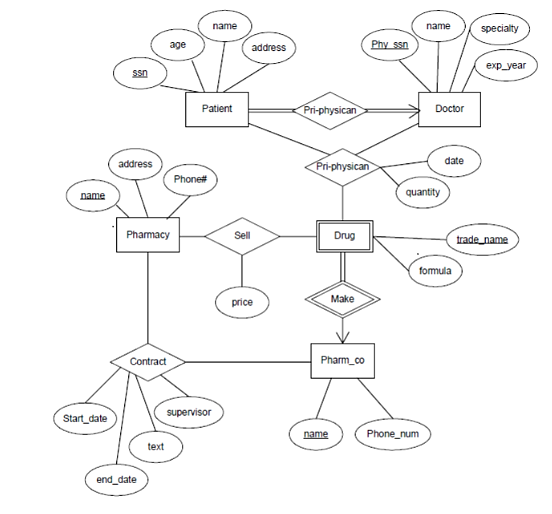
4. Users can view products out of stock but cannot card them to cart.

5. Each user has a unique identifier.

**Brief introduction about technique used:**

With the processing as described above, it was possible to start coding for the project. This tool uses PLSQL procedures to insert values into some tables, the ones with auto incrementing primary keys. This is so that the programmer doesn’t need to insert the values of those fields explicitly. These values are inbuilt in the code of the procedures. Another quite possible challenge is iterative searches in the densely populated tables. To solve these issues, the tool is equipped with PL/SQL functions that search as per the need and return a VARCHAR2 literal with the result. This result can then be displayed on the screen, or used for further processing at the backend, as per the requirement. On successful insertions of records in any table, Triggers are called which inform the insertion status to the user on the DBMS console. This output can certainly be used to process and check status according to requirement of future scaling of the project. All the fields capable of Auto incrementing values are supported by sequences which can be used to auto-generate the next value of the tuple.

ER DIAGRAM OF THE DATABASE



CONVERSION OF ER DIAGRAM TO TABLES

Patient

ssn age name address

Doctor

phy\_ssn name speciality exp\_years

Drug

trade\_name Pharm\_co.name formula

Pharm\_co

name phone\_num

Pharmacy

name address phone#

Pri\_physician

ssn phy\_ssn

Prescription

ssn phy\_ssn Pharm\_co.name trade\_name date quantity

Make

Pharm\_co.name trade\_name

Sell

Pharmacy.name Pharm\_co.name trade\_name

Contract

Pharmacy.name Pharm\_co.name start\_date end\_date text supervisor

**NORMALIZED TABLES**

**patient**

|  |  |  |  |
| --- | --- | --- | --- |
| **ssn** | **age** | **name** | **address** |

**doctor**

|  |  |  |  |
| --- | --- | --- | --- |
| **Phy\_ssn** | **name** | **speciality** | **Exp\_years** |

**drug**

|  |  |
| --- | --- |
| **Trade.name** | **formula** |

**Pharm\_co**

|  |  |
| --- | --- |
| **name** | **Phone\_num** |

**Pri\_physician**

|  |  |
| --- | --- |
| **ssn** | **Phy\_ssn** |

**prescription**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ssn** | **Phy\_ssn** | **Pharmco\_name** | **Trade\_name** | **quantity** |

**Make**

|  |  |
| --- | --- |
| **Pharm co.name** | **Trade\_name** |

**Sell**

|  |  |  |
| --- | --- | --- |
| **Pharmacy.name** | **Pharmco.name** | **Trade\_name** |

**Contract**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Pharmacy.name** | **Pharmco.name** | **Start\_date** | **End\_date** | **text** |

|  |  |  |
| --- | --- | --- |
| **Pharmacy.name** | **Pharmco.name** | **superviser** |

PL/SQL QUERIES

Following are the PL/Sql queries used along with comments about their usage.

**Procedure-1**

**//displaying medicines with largest quantites**

SET SERVEROUTPUT ON;

create or replace procedure printquantity\_largest is

MEDNAME prescription.trade\_name%TYPE;

prescriptionquantity prescription.quantity%TYPE;

CURSOR TEMP1 IS SELECT trade\_name,quantity FROM prescription

ORDER BY quantity DESC;

BEGIN

OPEN TEMP1;

LOOP

FETCH TEMP1 INTO MEDNAME,prescriptionquantity;

EXIT WHEN TEMP1%ROWCOUNT>5 OR TEMP1%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(MEDNAME||prescriptionquantity);

END LOOP;

CLOSE TEMP1;

END;

/

**Procedure-2**

**//displaying medicines with smallest quantites**

Create or replace procedure printquantity\_smallest is

MEDNAME prescription.trade\_name%TYPE;

prescriptionquantity prescription.quantity%TYPE;

CURSOR TEMP1 IS SELECT trade\_name,quantity FROM prescription

ORDER BY quantity;

BEGIN

OPEN TEMP1;

LOOP

FETCH TEMP1 INTO MEDNAME,prescriptionquantity;

EXIT WHEN TEMP1%ROWCOUNT>5 OR TEMP1%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(MEDNAME||prescriptionquantity);

END LOOP;

CLOSE TEMP1;

END;

/

**Procedure-3**

**//displaying patient details**

create or replace procedure patient\_details(patient\_name varchar) is

v\_age number(10);

v\_name varchar(20);

v\_address varchar(100);

BEGIN

select age,name,address into v\_age,v\_name,v\_address from Patient where name=patient\_name;

DBMS\_OUTPUT.PUT\_LINE('NAME: ' || v\_name);

DBMS\_OUTPUT.PUT\_LINE('ADDRESS: ' || v\_address);

DBMS\_OUTPUT.PUT\_LINE('AGE: ' || v\_age);

END;

/

**Procedure-4**

**//displaying doctor details**

create or replace procedure doctor\_details(doctor\_name varchar) is

v\_expyears number(10);

v\_name varchar(20);

v\_speciality varchar(20);

BEGIN

select name,speciality,exp\_years into v\_name,v\_speciality,v\_expyears from Doctor where name=doctor\_name;

DBMS\_OUTPUT.PUT\_LINE('NAME: ' || v\_name);

DBMS\_OUTPUT.PUT\_LINE('SPECIALITY: ' || v\_speciality);

DBMS\_OUTPUT.PUT\_LINE('EXP\_YEARS: ' || v\_expyears);

END;

/

**Procedure-5**

**//** **pharmacy company that makes a particular drug**

create or replace procedure get\_pharmco(tradename varchar) is

v\_name varchar(20);

BEGIN

select pharmco\_name into v\_name from Drug where trade\_name=tradename;

DBMS\_OUTPUT.PUT\_LINE('PHARMCO\_NAME: ' || v\_name);

END;

/

**Procedure-6**

**// insert/delete/update combined procedure**

create or replace procedure MasterInsertUpdateDelete

(id INTEGER, v\_name VARCHAR,v\_age NUMBER,v\_address VARCHAR,StatementType nvarchar:= '' )

AS

BEGIN

IF StatementType='Insert' THEN

insert into Patient (ssn,name,age,address) values( id,v\_name,v\_age,v\_address);

END IF;

IF StatementType='Select' THEN

select \* from Patient;

END IF;

IF StatementType='Update' THEN

UPDATE Patient SET

name:=v\_name,age:=v\_age,

address:=v\_address

WHERE ssn:= id;

END IF;

ELSIF StatementType='Delete' THEN

DELETE FROM Paient WHERE ssn:=id;

END IF;

end;

/

**Procedure-7**

**//trigger check before updating number of medicines to be prescribed**

CREATE OR REPLACE TRIGGER UPDATE\_CHECK

BEFORE UPDATE ON prescription

FOR EACH ROW

BEGIN

IF :NEW.quantity<:OLD.quantity THEN

RAISE\_APPLICATION\_ERROR(-20000,'NEW quantity CANNOT

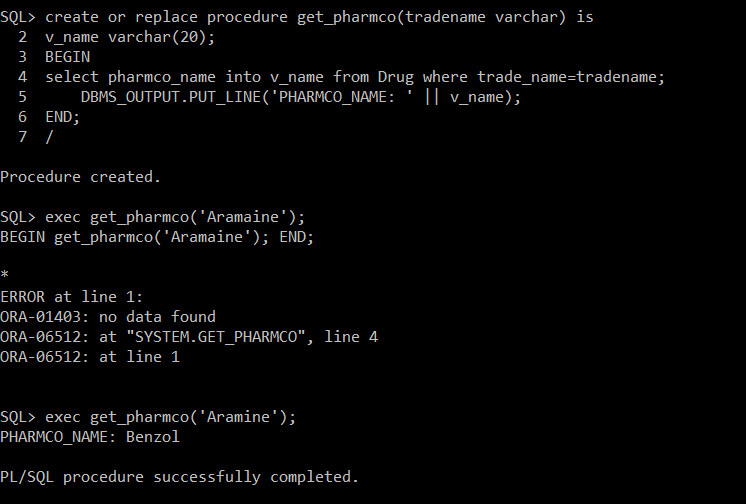
BE LESS THAN OLD quantity');

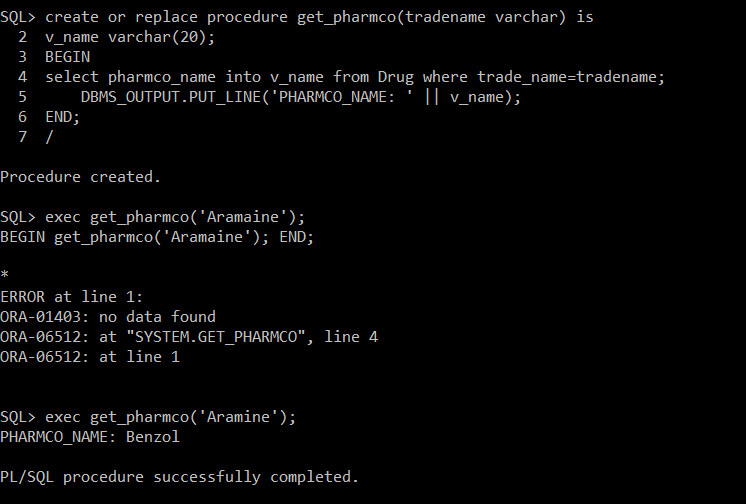
END IF;

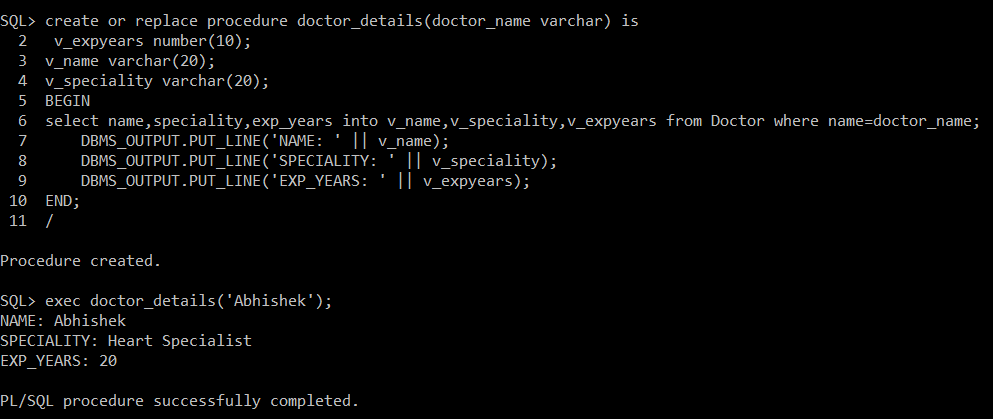
END;

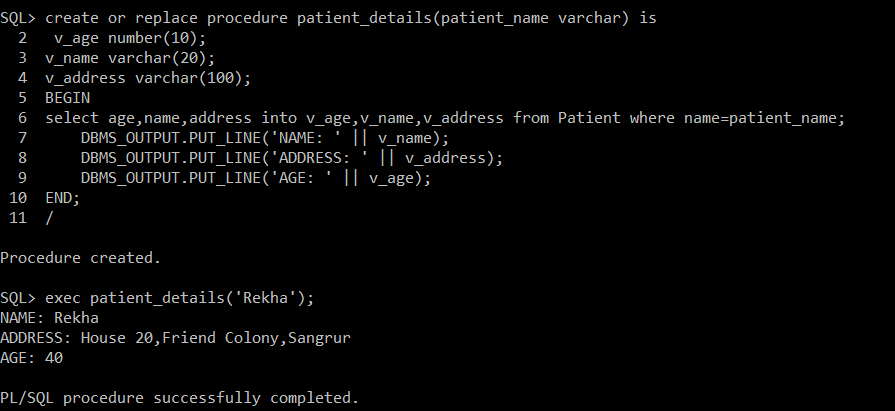
/

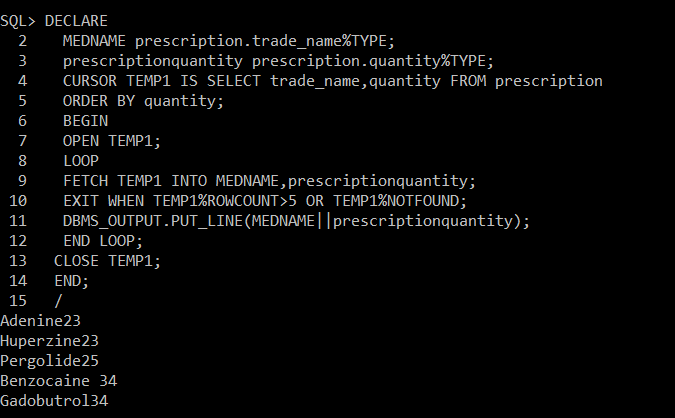
SNAP SHOTS OF THE PL/SQL QUERIES

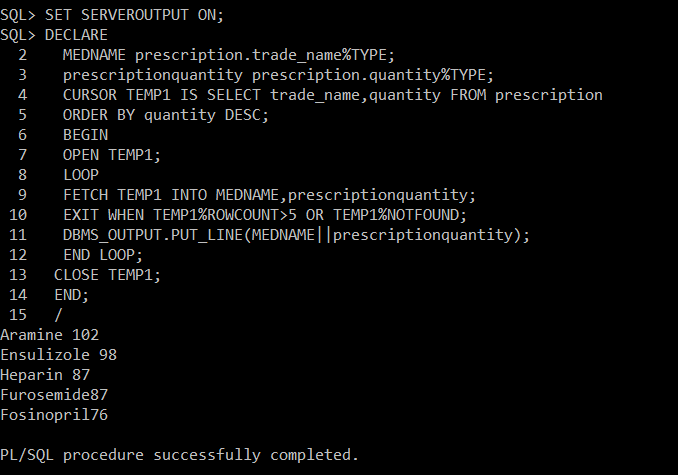




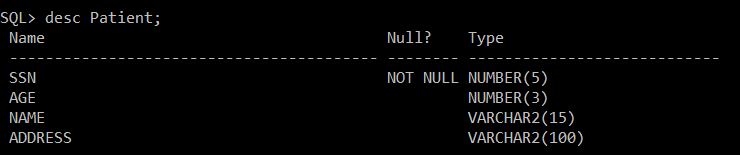


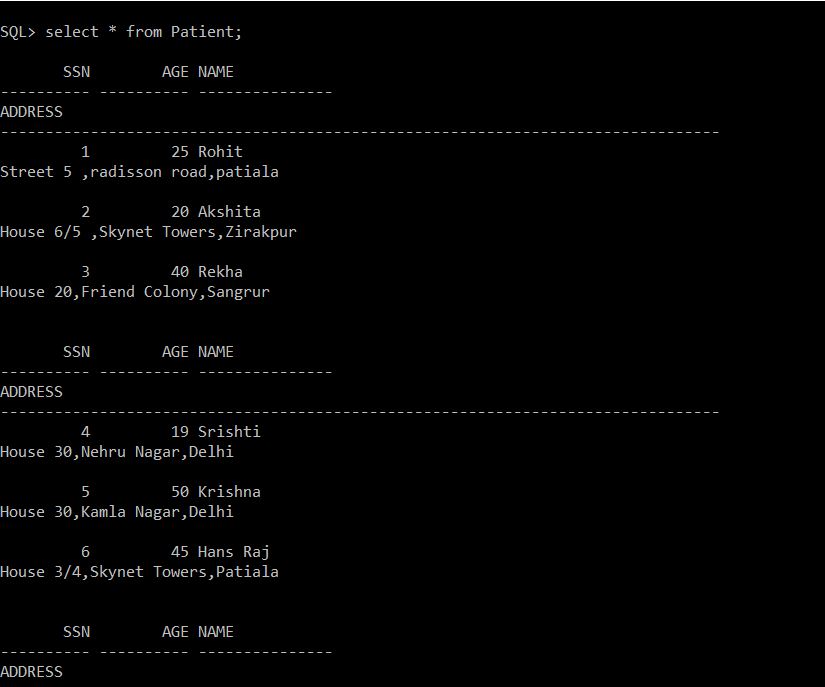


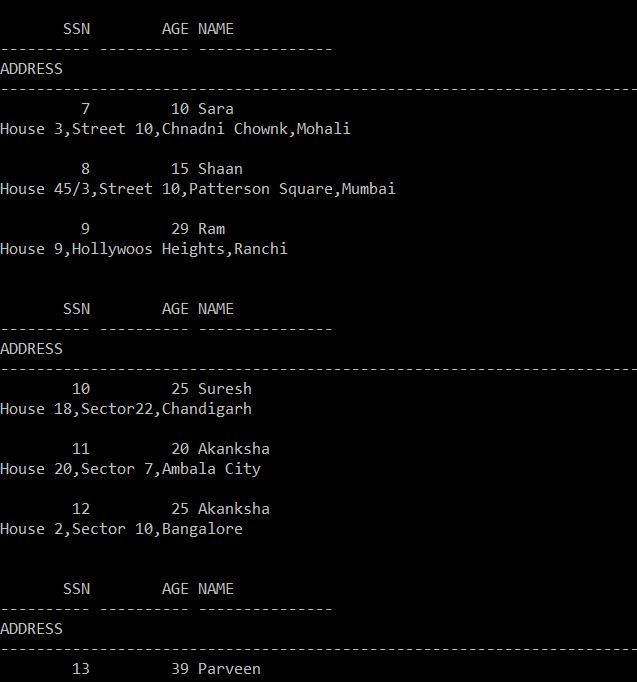




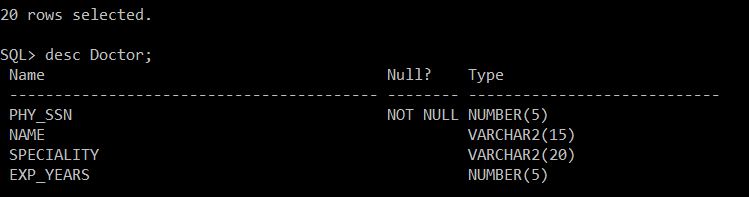
SNAP SHOTS OF SOME OF THE TABLES CREATED

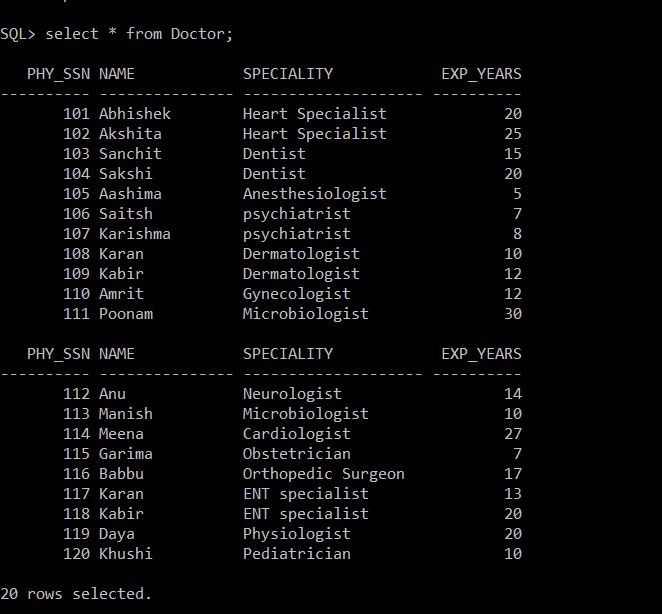


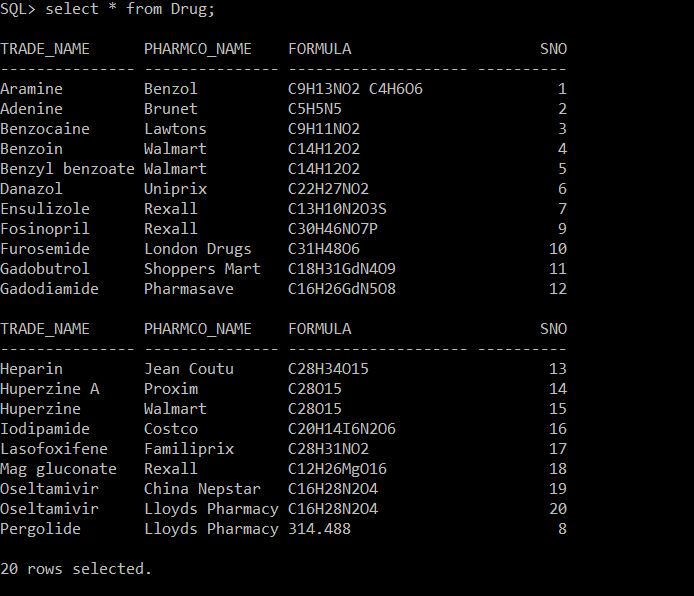


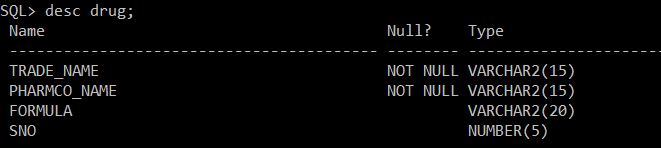


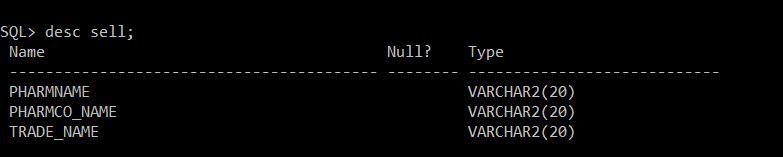


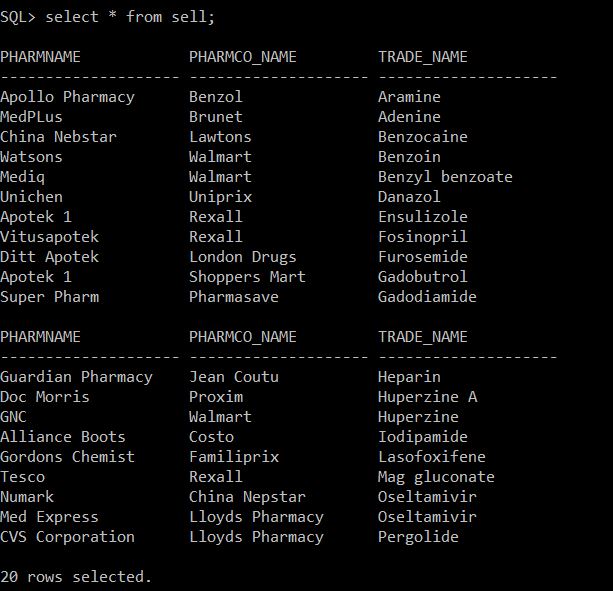


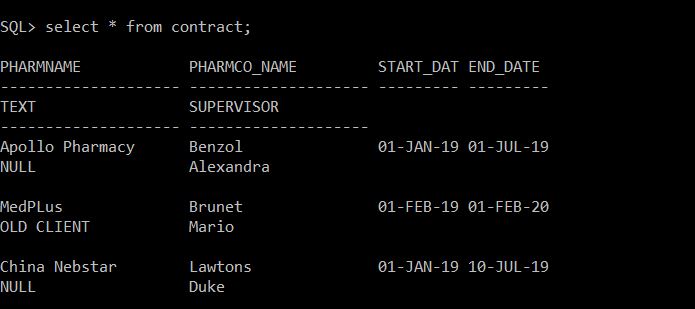


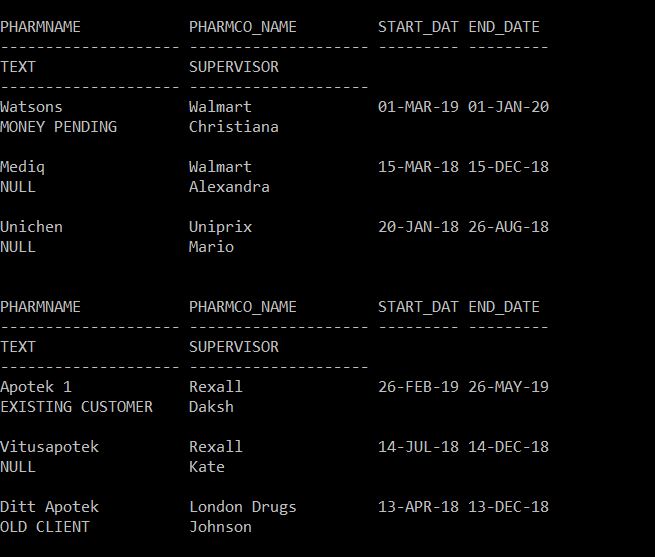


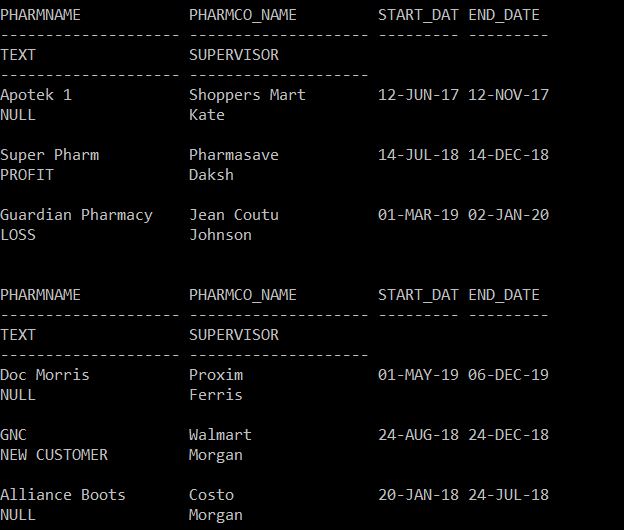


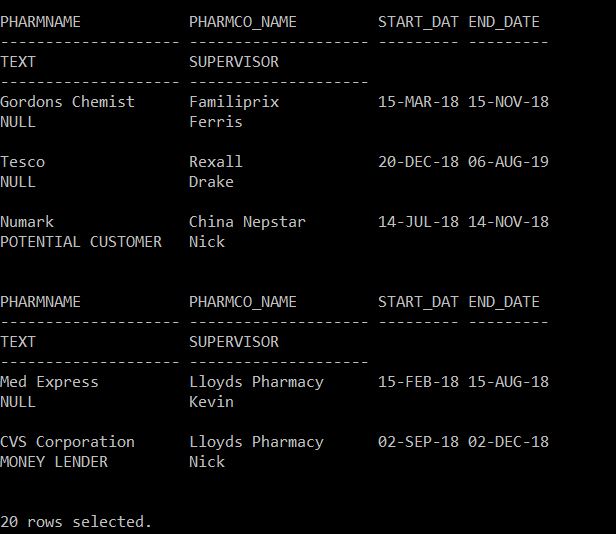


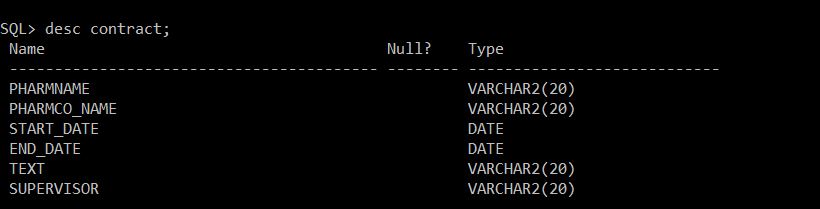












REFERENCES

<https://en.wikipedia.org/wiki/PL/SQL>

<https://github.com/sanchit-nayyar/SQL_Project>

<https://www.youtube.com/watch?v=-OR7zLzCh_I&list=PLL_LQvNX4xKyiExzq9GKwORoH6nvaRnOQ&index=15>

<https://sites.google.com/site/dbms310cse/lecture-notes>