# Yaniv Erlich 205958333, Yaacov biblow (ben ari) 205460942, nevo sharabi 316360304

**System Requirement –**

**The new Data Base will provide an efficient way to manage the clinic’s data, manage the doctors schedule and keep track of their session and prescriptions.  
The system also manages the patient medical history by keeping a record of the patient’s previous sessions including his analysis and diagnosis.**

* **General Description**
* A **Clinic** has a name and address. The clinic has **Doctors** and a **managing doctor**. **Patients** makes appointments to get treated by a doctor in the clinic.
* In a **Clinic** there are several doctors that have a doctor id , first name, last name , dob,age, gender and years of experience.
* Each doctor has a work **schedule** containing the day and starting hour and ending hour that he’s available for sessions to treat patients.
* The **patients** coming to the clinic have a patient id,first name, last name , age ,dob and gender.
* A **session** between a **patient** and a **doctor** occours in a specific **room** and might result in an **analysis** or a **diagnosis** and a **prescription** given by the doctor.
* Prescription contains issue date and **lines**, each line contains **medicine** and a dose .
* Each **room** has a number and a specific type.

**There are three types of users that use the system:**

**Patient:**

The patient makes an appointment for a session, may view their future appointments and see information about their past appointments including the prescription, analysis and diagnosis that was given to them.

**Doctor:**

A doctor can view his work schedule, view the medical history (past analysis and diagnosis) of his patients and all the prescriptions he has given.

**Managing Doctor:**

The managing doctor is in charge of the schedule of the doctors and view all the information of the system including the sessions between the doctors and patients.

* **Entities Description**
* **PK, FK**
* **Clinic** – Name, Address.
* **Schedule**– sc\_id, sc\_day, sc\_start\_hour, sc\_end\_hour,d\_id ,c\_name
* **Doctor** – d\_id, d\_f\_name, d\_l\_name,d\_dob,d\_age,d\_gender,d\_years\_of\_exp,manager\_id
* **Patient**–p\_id, p\_f\_name, p\_last\_name,p\_gender,p\_dob,p\_age
* **Session** – s\_id, s\_date, s\_hour, d\_id, p\_id,r\_num
* **Room** – r\_num, r\_type
* **Analysis** – a\_id, a\_RBC,a\_WBC,a\_HGB,s\_id
* **Diagnosis**– diag\_id,diag\_name
* **Diagnosis of Session**– diag\_id , s\_id
* **Prescription**– prsc\_id,prsc\_date,s\_id
* **Medicine –** m\_id,m\_name
* **Lines in prescription**– line\_num,lip\_amount,m\_id,prsc\_id

The SQL Code

**Table Creation:**

**Clinic**:

CREATE TABLE Clinic(c\_name CHAR(25) NOT NULL,

c\_addressCHAR(50) NOT NULL,

PRIMARY KEY(c\_name))

ENGINE = InnoDB;

**Doctor:**

CREATE TABLE Doctor ( d\_id INT NOT NULL,

d\_f\_name CHAR(25) NOT NULL,

d\_l\_name CHAR(25) NOT NULL,

d\_gender BOOL NOT NULL,

d\_years\_of\_exp INT,

d\_dob DATE NOT NULL,

d\_age FLOAT, manager\_id INT,

PRIMARY KEY (d\_id)) ENGINE=INNODB;

**Patient:**

CREATE TABLE Patient

( p\_id INT NOT NULL,

p\_f\_name CHAR(25) NOT NULL,

p\_l\_name CHAR(25) NOT NULL,

p\_gender BOOL NOT NULL,

p\_dob DATE NOT NULL,

p\_age FLOAT,

PRIMARY KEY (p\_id)) ENGINE=INNODB;

**Schedule:**

CREATE TABLE Schedules

( sc\_id INT NOT NULL,

d\_id INT NOT NULL,

c\_name CHAR(25) NOT NULL,

sc\_day CHAR(10),

sc\_start\_hour TIME,

sc\_end\_hour TIME,

PRIMARY KEY (sc\_id),

CONSTRAINT fk\_doctor\_id\_1 FOREIGN KEY (d\_id) REFERENCES Doctor (d\_id),

CONSTRAINT fk\_clinic FOREIGN KEY (c\_name) REFERENCES Clinic (c\_name))

ENGINE=INNODB;

**Room:**

CREATE TABLE Room

( r\_num INT NOT NULL,

r\_type CHAR(25),

PRIMARY KEY (r\_num))

ENGINE=INNODB;

**Session:**

CREATE TABLE Session

( s\_id INT NOT NULL,

d\_id INT NOT NULL,

p\_id INT NOT NULL,

r\_num INT NOT NULL,

s\_date DATE,

s\_hour TIME,

PRIMARY KEY (s\_id),

CONSTRAINT fk\_doctor\_id\_2 FOREIGN KEY (d\_id) REFERENCES Doctor (d\_id),

CONSTRAINT fk\_patient\_id\_1 FOREIGN KEY (p\_id) REFERENCES Patient (p\_id),

CONSTRAINT fk\_room\_num\_1 FOREIGN KEY (r\_num) REFERENCES Room (r\_num) )

ENGINE=INNODB;

**Prescription:**

CREATE TABLE Prescription

( prsc\_id INT NOT NULL,

s\_id INT NOT NULL,

prsc\_date CHAR(25),

PRIMARY KEY (prsc\_id),

CONSTRAINT fk\_session\_id\_1 FOREIGN KEY (s\_id) REFERENCES Session (s\_id))

ENGINE=INNODB;

**Analysis:**

CREATE TABLE Analysis

( a\_id INT NOT NULL,

s\_id INT NOT NULL,

a\_RBC FLOAT,

a\_WBC FLOAT,

a\_HGB FLOAT,

PRIMARY KEY (a\_id),

CONSTRAINT fk\_session\_id\_2 FOREIGN KEY (s\_id) REFERENCES Session (s\_id))

ENGINE=INNODB;

**Diagnosis:**

CREATE TABLE Diagnosis

( diag\_id INT NOT NULL,

diag\_name CHAR(30),

PRIMARY KEY (diag\_id))

ENGINE=INNODB;

**DOS(Diagnosis of Session) :**

CREATE TABLE DOS

( diag\_id INT NOT NULL,

s\_id INT NOT NULL,

CONSTRAINT fk\_diagnosis\_id\_1 FOREIGN KEY (diag\_id) REFERENCES Diagnosis (diag\_id),

CONSTRAINT fk\_session\_id\_3 FOREIGN KEY (s\_id) REFERENCES Session (s\_id))

ENGINE=INNODB;

CREATE TABLE Medicine

( m\_id INT NOT NULL,

m\_name CHAR(30),

PRIMARY KEY (m\_id))

ENGINE=INNODB;

**Lines In Prescription:**

CREATE TABLE Lines\_In\_Prescription

( line\_num INT NOT NULL,

prsc\_id INT NOT NULL,

m\_id INT NOT NULL,

lip\_amount INT,

PRIMARY KEY (prsc\_id, line\_num), CONSTRAINT fk\_prescription\_id\_1 FOREIGN KEY (prsc\_id) REFERENCES Prescription (prsc\_id),

CONSTRAINT fk\_medicine\_id\_1 FOREIGN KEY (m\_id) REFERENCES Medicine (m\_id)) ENGINE=INNODB;;

Insertion:

insert into **Doctor** values

(341270111, "Ross", "Geller", 0, 20, '1994-9-8', null, null),

(341270222, "Rachel", "Green", 1, 10, '1969-2-11', null, 341270111),

(341270333, "Monika", "Geller", 1, 12, '1964-6-15', null, 341270111),

(341270444, "Phoebe", "Buffay", 1, 5, '1963-7-30', null, 341270111),

(341270555, "Joey", "Tribbiani", 0, 8, '1967-7-25', null, 341270111),

(341270666, "Chandler", "Bing", 0, 14, '1969-8-19', null, 341270111);

INSERT INTO **Room** VALUES

(1,"Lab"),(2,"Lab"),

(3,"Examination"),

(4,"Examination"),

(5,"Examination"),

(6,"Examination");

INSERT INTO **Diagnosis** VALUES

(100,"Covid-19"),

(101,"Flue"),

(102,"Heart Stroke"),

(103,"Cancer"),

(104,"Diabetes"),

(105,"Headaches"),

(106,"Stomach Aches");

insert into **Clinic** values

("Ocean Medical Clinic",

"Trinity House 1-3 Ocean Village");

insert into **Medicine** values

(1,'levothyroxine’),

(2,'rosuvastatin’),

(3,'albuterol’),

(4,'fluticasone’),

(5,'esomeprazole’),

(6,'insulin’),

(7,'glargine’),

(8,'sitagliptin’),

(9,'tiotropium’),

(10,'pregabalin’),

(11,'Advil’);

insert into **Patient** values

('321561231','Frank','Kohen','0','1960-11-10’,null),

('213562145','Estelle','Costanza','1','1970-11-10', null),

('342341231','Susan','Ross','1','1980-11-10', null),

('634612239','Morty','Seinfeld','0','1990-11-10', null),

('352345232','jerry','Seinfeld','0','1965-11-10', null),

('321561234','George','Costanza','0','1982-11-10 null),

('321565435','Elaine','Benes','1','1995-11-10', null),

('323454237','Cosmo','Kramer','0','2000-11-10', null),

('312853231',' Joe','Davola','0','1957-11-10', null),

('223324236','Kenny','Bania','0','1955-11-10', null);

insert into **Schedules** values

(221, 341270111, "Ocean Medical Clinic", "Sunday", '10:00:00', '16:00:00’),

(222, 341270111, "Ocean Medical Clinic", "Thursday", '12:00:00', '17:00:00’),

(223, 341270222, "Ocean Medical Clinic", "Monday", '08:00:00', '16:00:00’),

(224, 341270222, "Ocean Medical Clinic", "Friday", '07:30:00', '13:00:00’),

(225, 341270333, "Ocean Medical Clinic", "Tuesday", '08:00:00', '14:00:00’),

(226, 341270333, "Ocean Medical Clinic", "Thursday", '08:00:00', '12:00:00’),

(227, 341270444, "Ocean Medical Clinic", "Monday", '10:00:00', '16:00:00’),

(228, 341270444, "Ocean Medical Clinic", "Wednesday", '12:00:00', '18:00:00’),

(229, 341270555, "Ocean Medical Clinic", "Wednesday", '08:00:00', '14:00:00’),

(230, 341270555, "Ocean Medical Clinic", "Moday", '14:00:00', '20:00:00’),

(231, 341270666, "Ocean Medical Clinic", "Sunday", '08:00:00', '16:00:00’),

(232, 341270666, "Ocean Medical Clinic", "Monday", '08:00:00', '16:00:00’);

insert into **Analysis** values

(3000, 1014, 7.2, 3.4, 12.1),

(3001, 1015, 5.5, 8.9, 16.0),

(3002, 1016, 4.21, 3.98, 19.1),

(3003, 1017, 5.64, 7.02, 20),

(3004, 1018, 5.0, 4.0, 12.12);

insert into **Session** values

(1000, 341270111, 321561231, 3, '2022-1-2', '10:00:00’),

(1001, 341270111, 213562145, 3, '2022-1-2', '10:30:00’),

(1002, 341270222, 213562145, 4, '2022-1-3', '08:00:00’),

(1003, 341270222, 312853231, 4, '2022-1-3', '08:30:00’),

(1004, 341270333, 342341231, 5, '2022-1-4', '13:00:00’),

(1005, 341270333, 634612239, 5, '2022-1-4', '13:30:00’),

(1006, 341270444, 352345232, 6, '2022-1-5', '12:00:00’),

(1007, 341270444, 634612239, 6, '2022-1-5', '13:00:00’),

(1008, 341270333, 223324236, 3, '2022-1-6', '08:00:00’),

(1009, 341270111, 323454237, 4, '2022-1-6', '12:00:00’),

(1010, 341270222, 321565435, 3, '2022-1-7', '07:00:00’),

(1011, 341270222, 321561234, 3, '2022-1-7', '08:00:00’),

(1012, 341270666, 321565435, 4, '2022-1-9', '08:00:00’),

(1013, 341270666, 321561234, 5, '2022-1-9', '15:00:00’),

(1014, 341270222, 321561234, 1, '2022-1-10', '08:00:00’),

(1015, 341270333, 352345232, 2, '2022-1-11', '09:00:00’),

(1016, 341270222, 634612239, 1, '2022-1-10', '09:00:00’),

(1017, 341270333, 342341231, 2, '2022-1-11', '10:00:00’),

(1018, 341270222, 213562145, 1, '2022-1-10', '10:00:00');

insert into **DOS** values

(100,1000),

(105,1000),

(103,1001),

(105,1001),

(102,1002),

(104,1003),

(106,1004),

(101,1005),

(101,1006),

(104,1007),

(104,1008),

(106,1009),

(105,1010),

(105,1011),

(105,1012),

(103,1013);

insert into **Prescription** values

(4000, 1003, '2022-1-3'),

(4001, 1002, '2022-1-3'),

(4002, 1004, '2022-1-4'),

(4003, 1006, '2022-1-5'),

(4004, 1007, '2022-1-5'),

(4005, 1008, '2022-1-6');

insert into **Lines\_In\_Prescription** values

(1, 4000, 6, 10),

(1, 4001, 2, 10),

(2, 4001, 3, 10),

(1, 4002, 4, 5),

(2, 4002, 5, 6),

(1, 4003, 11, 10),

(2, 4003, 8, 11),

(3, 4003, 9, 12),

(1, 4004, 6, 10),

(1, 4005, 6, 10);

**calculate Patient table age:**

DELIMITER $$

CREATE PROCEDURE UpdatePatientAge( in pid INT)

BEGIN

# get patient DOB

SELECT Patient.p\_dob INTO @DOB FROM Patient

WHERE Patient.p\_id = pid; # get patient age based on DOB

SELECT timestampdiff(YEAR, @DOB, CURDATE()) INTO @AGE;

# update patient age

UPDATE Patient SET Patient.p\_age = @age

WHERE Patient.p\_id = pid;END$$DELIMITER ;

DROP PROCEDURE IF EXISTS UpdatePatientTableAge;DELIMITER $$CREATE PROCEDURE UpdatePatientTableAge()

BEGIN

DECLARE n INT DEFAULT 0;

DECLARE i INT DEFAULT 0;

SELECT count(\*) INTO n from Patient;

SET i = 0;

WHILE i < n DO

SELECT Patient.p\_id INTO @pid FROM Patient LIMIT i, 1;

call UpdatePatientAge(@pid);

SET i = i + 1;

END WHILE;

END$$

DELIMITER ;

call UpdatePatientTableAge();

**Triggers:**

CREATE TABLE Patient\_Log (

p\_id INT NOT NULL,

p\_f\_name\_new CHAR(25),

p\_f\_name\_old CHAR(25),

p\_l\_name\_new CHAR(25),

p\_l\_name\_old CHAR(25),

p\_gender\_new BOOL,

p\_gender\_old BOOL,

p\_dob\_new DATE,

p\_dob\_old DATE,

p\_age\_new FLOAT,

p\_age\_old FLOAT,

command\_ts timestamp,

command char(10)

) ENGINE=INNODB;

delimiter $

CREATE TRIGGER Patient\_inst\_trigg AFTER INSERT ON Patient

FOR EACH ROW

BEGIN

INSERT INTO Patient\_Log VALUES(new.p\_id, new.p\_f\_name, null, new.p\_l\_name, null, new.p\_gender, null, new.p\_dob, null, new.p\_age, null, now(), 'INSERT');

END$

delimiter ;

delimiter $

CREATE TRIGGER Patient\_updt\_trigg AFTER UPDATE ON Patient

FOR EACH ROW

BEGIN

INSERT INTO Patient\_Log VALUES(new.p\_id, new.p\_f\_name, old.p\_f\_name, new.p\_l\_name, old.p\_l\_name, new.p\_gender, old.p\_gender, new.p\_dob, old.p\_dob, new.p\_age, old.p\_age, now(), 'UPDATE');

END$

delimiter ;

delimiter $

CREATE TRIGGER Patient\_del\_trigg AFTER DELETE ON Patient

FOR EACH ROW

BEGIN

INSERT INTO Patient\_Log VALUES(old.p\_id, null, old.p\_f\_name, null, old.p\_l\_name, null, old.p\_gender, null, old.p\_dob, null, old.p\_age, now(), 'DELETE');

END$

delimiter ;

CREATE TABLE Session\_Log (

s\_id INT NOT NULL,

d\_id INT NOT NULL,

p\_id INT NOT NULL,

r\_num INT NOT NULL,

s\_date\_new DATE,

s\_date\_old DATE,

s\_hour\_new TIME,

s\_hour\_old TIME,

command\_ts timestamp,

command char(10)

) ENGINE=INNODB;

delimiter $

CREATE TRIGGER Session\_inst\_trigg AFTER INSERT ON Session

FOR EACH ROW

BEGIN

INSERT INTO Session\_Log VALUES(new.s\_id, new.d\_id,new.p\_id,new.r\_num,new.s\_date,null,new.s\_hour,null, now(), 'INSERT');

END$

delimiter;

delimiter $

CREATE TRIGGER Session\_updt\_trigg AFTER UPDATE ON Session

FOR EACH ROW

BEGIN

INSERT INTO Session\_Log VALUES(new.s\_id, new.d\_id,new.p\_id,new.r\_num,new.s\_date,old.s\_date,new.s\_hour,old.s\_hour, now(), 'UPDATE');

END$

delimiter ;

delimiter $

CREATE TRIGGER Session\_del\_trigg AFTER DELETE ON Session

FOR EACH ROW

BEGIN

INSERT INTO Session\_Log VALUES(old.s\_id, old.d\_id,old.p\_id,old.r\_num,null,old.s\_date,null,old.s\_hour, now(), 'DELETE');

END$

delimiter ;

CREATE TABLE Doctor\_log (

d\_id INT NOT NULL,

d\_f\_name\_new CHAR(25) ,

d\_f\_name\_old CHAR(25) ,

d\_l\_name\_new CHAR(25) ,

d\_l\_name\_old CHAR(25) ,

d\_gender\_new BOOL ,

d\_gender\_old BOOL ,

d\_years\_of\_exp\_new INT,

d\_years\_of\_exp\_old INT,

d\_dob\_new DATE ,

d\_dob\_old DATE ,

d\_age\_new FLOAT,

d\_age\_old FLOAT,

manager\_id\_new INT,

manager\_id\_old INT,

command\_ts timestamp,

command char(10)

) ENGINE=INNODB;

delimiter $

CREATE TRIGGER Doctor\_inst\_trigg AFTER INSERT ON Doctor

FOR EACH ROW

BEGIN

INSERT INTO Doctor\_Log VALUES(new.d\_id, new.d\_f\_name,null,new.d\_l\_name,null,new.d\_gender,null,new.d\_years\_of\_exp,null,new.d\_dob,null,new.d\_age,null,new.manager\_id,null, now(), 'INSERT');

END$

delimiter;

delimiter $

CREATE TRIGGER Doctor\_updt\_trigg AFTER UPDATE ON Doctor

FOR EACH ROW

BEGIN

INSERT INTO Doctor\_Log VALUES(new.d\_id, new.d\_f\_name,old.d\_f\_name,new.d\_l\_name,old.d\_l\_name,new.d\_gender,old.d\_gender,new.d\_years\_of\_exp,old.d\_years\_of\_exp,new.d\_dob,old.d\_dob,new.d\_age,old.d\_age,new.manager\_id,old.manager\_id, now(), 'UPDATE');

END$

delimiter ;

delimiter $

CREATE TRIGGER Doctor\_del\_trigg AFTER DELETE ON Doctor

FOR EACH ROW

BEGIN

INSERT INTO Doctor\_Log VALUES(old.d\_id, null,old.d\_f\_name,null,old.d\_l\_name,null,old.d\_gender,null,old.d\_years\_of\_exp,null,old.d\_dob,null,old.d\_age,null,old.manager\_id, now(), 'DELETE');

END$

delimiter ;

CREATE TABLE Schedules\_Log (

sc\_id INT,

d\_id INT,

c\_name CHAR(25),

sc\_day\_new CHAR(10),

sc\_day\_old CHAR(10),

sc\_start\_hour\_new TIME,

sc\_start\_hour\_old TIME,

sc\_end\_hour\_new TIME,

sc\_end\_hour\_old TIME,

command\_ts timestamp,

command char(10)

) ENGINE=INNODB;

delimiter $

CREATE TRIGGER Schedules\_inst\_trigg AFTER INSERT ON Schedules

FOR EACH ROW

BEGIN

INSERT INTO Schedules\_Log VALUES(new.sc\_id, new.d\_id, new.c\_name, new.sc\_day, null, new.sc\_start\_hour, null, new.sc\_end\_hour, null, now(), 'INSERT');

END$

delimiter ;

delimiter $

CREATE TRIGGER Schedules\_updt\_trigg AFTER UPDATE ON Schedules

FOR EACH ROW

BEGIN

INSERT INTO Schedules\_Log VALUES(new.sc\_id, new.d\_id, new.c\_name, new.sc\_day, old.sc\_day, new.sc\_start\_hour, old.sc\_start\_hour, new.sc\_end\_hour, old.sc\_end\_hour, now(), 'UPDATE');

END$

delimiter ;

delimiter $

CREATE TRIGGER Schedules\_del\_trigg AFTER DELETE ON Schedules

FOR EACH ROW

BEGIN

INSERT INTO Schedules\_Log VALUES(old.sc\_id, old.d\_id, old.c\_name, null, old.sc\_day, null, old.sc\_start\_hour, null, old.sc\_end\_hour, now(), 'DELETE');

END$

delimiter ;

CREATE TABLE Medicine\_Log (

m\_id INT,

m\_name\_new CHAR(30),

m\_name\_old CHAR(30),

command\_ts timestamp,

command char(10)

) ENGINE=INNODB;

delimiter $

CREATE TRIGGER Medicine\_inst\_trigg AFTER INSERT ON Medicine

FOR EACH ROW

BEGIN

INSERT INTO Medicine\_Log VALUES(new.m\_id,new.m\_name,null, now(), 'INSERT');

END$

delimiter;

delimiter $

CREATE TRIGGER Medicine\_updt\_trigg AFTER UPDATE ON Medicine

FOR EACH ROW

BEGIN

INSERT INTO Medicine\_Log VALUES(new.m\_id,new.m\_name,old.m\_name, now(), 'UPDATE');

END$

delimiter ;

delimiter $

CREATE TRIGGER Medicine\_del\_trigg AFTER DELETE ON Medicine

FOR EACH ROW

BEGIN

INSERT INTO Medicine\_Log VALUES(old.m\_id,null,old.m\_name, now(), 'DELETE');

END$

delimiter ;

CREATE TABLE Diagnosis\_Log (

diag\_id INT,

diag\_name\_new CHAR(30),

diag\_name\_old CHAR(30),

command\_ts timestamp,

command char(10)

) ENGINE=INNODB;

delimiter $

CREATE TRIGGER Diagnosis\_inst\_trigg AFTER INSERT ON Diagnosis

FOR EACH ROW

BEGIN

INSERT INTO Diagnosis\_Log VALUES(new.diag\_id, new.diag\_name, null, now(), 'INSERT');

END$

delimiter ;

delimiter $

CREATE TRIGGER Diagnosis\_updt\_trigg AFTER UPDATE ON Diagnosis

FOR EACH ROW

BEGIN

INSERT INTO Diagnosis\_Log VALUES(new.diag\_id, new.diag\_name, old.diag\_name, now(), 'UPDATE');

END$

delimiter ;

delimiter $

CREATE TRIGGER Diagnosis\_del\_trigg AFTER DELETE ON Diagnosis

FOR EACH ROW

BEGIN

INSERT INTO Diagnosis\_Log VALUES(old.diag\_id, null, old.diag\_name, now(), 'DELETE');

END$

delimiter ;

**Queries:**

**Patient Queries:**

**# 1 - Display all Prescription(including lines) written for a patient (using patient\_id)**

CREATE PROCEDURE GetPrcPerPait1( in pai\_Id varchar(255))

BEGIN

select line.prsc\_id,

Medicine.m\_name,

line.lip\_amount as amount from Lines\_In\_Prescription as line,

(select pre.prsc\_id from Prescription as pre,Session as si,

Patient as pai where pai.p\_id = si.p\_id and pre.s\_id = si.s\_id and pai.p\_id = pai\_Id) as tmp,Medicine

where tmp.prsc\_id = line.prsc\_id and line.m\_id=Medicine.m\_id;

END$$

DELIMITER ;

call GetPrcPerPait1('213562145');

**# 2 - for a given patient id and doctor name display all sessions that the patient had with the doctor**

DELIMITER $$

CREATE PROCEDURE DisplaySessionOfPatientWithDoctor(in pid INT, dname CHAR(25))

BEGIN

select \* from patient, session, doctor

where patient.p\_id = session.p\_id and session.d\_id = doctor.d\_id and patient.p\_id = pid and doctor.d\_f\_name like dname;

END$$DELIMITER ;

call DisplaySessionOfPatientWithDoctor(321565435, "rachel");

**# 3 - for a given doctor id and day, display the schedule of that doctor**

DELIMITER $$

CREATE PROCEDURE DisplayDoctorScheduleOfDay( in did INT, scday CHAR(25))

BEGIN

select \* from doctor,

Schedules where doctor.d\_id = schedules.d\_id and doctor.d\_id = did and schedules.sc\_day like scday;

END$$

DELIMITER ;

call DisplayDoctorScheduleOfDay(341270666, "sunday");

**# 4 - for a given patient id display all of his diagnostics**

DELIMITER $$

CREATE PROCEDURE DisplayDIagnosticsOfPatient( in pid INT

)BEGIN

select \* from patient, session, DOS, Diagnosis

where patient.p\_id = session.p\_id and session.s\_id = DOS.s\_id and DOS.diag\_id = diagnosis.diag\_id and patient.p\_id = pid;

END$$

DELIMITER ;

call DisplayDIagnosticsOfPatient(321561231);

**# 5 - for a given patient id show all sessions that had more than 1 line in the prescription**

DELIMITER $$

CREATE PROCEDURE DisplayPrescriptionWIthMoreThanOneLine( in pid INT)

BEGIN

select \*from patient, session,

(select prescription.s\_id,

prescription.prsc\_id,

count(lip.prsc\_id) as lip\_count from prescription,

Lines\_In\_Prescription as lip

where prescription.prsc\_id = lip.prsc\_id group by lip.prsc\_id) as TMP

where patient.p\_id = session.p\_id and session.s\_id = TMP.s\_id and TMP.lip\_count > 1 and patient.p\_id = pid;

END$$

DELIMITER ;

call DisplayPrescriptionWIthMoreThanOneLine(213562145);

**# 6 - for a given patient id show all sessions that had more than 1 diagnosis**

DELIMITER $$

CREATE PROCEDURE DisplaySessionsOfPatientWithMoreThanOneDiagnostic( in pid INT)

BEGIN

select \* from patient,session,

(select DOS.s\_id, count(DOS.s\_id) as count from patient, session, DOS

where patient.p\_id = session.p\_id and session.s\_id = DOS.s\_id group by DOS.s\_id) as TMP

where patient.p\_id = session.p\_id and session.s\_id = TMP.s\_id and TMP.count > 1 and patient.p\_id = pid;

END

$$DELIMITER ;

call DisplaySessionsOfPatientWithMoreThanOneDiagnostic(321561231);

**# 7 - for a given patient id display all doctors info that had a session with that patient**

DELIMITER $$

CREATE PROCEDURE DisplayDoctorInfoThatHadSessionWithPatient( in pid INT)

BEGIN

select distinct \* from doctor,

(select session.d\_id from patient, session

where patient.p\_id = session.p\_id and patient.p\_id = pid) as TMP

Where doctor.d\_id = TMP.d\_id;

END$$

DELIMITER ;

call DisplayDoctorInfoThatHadSessionWithPatient(213562145);

**# 8 - for a given patient id show all sessions that had an analysis**

DELIMITER $$

CREATE PROCEDURE DisplayAnlysisOfPatient( in pid INT)

BEGIN

select \* from Analysis, Session, Patient

where Analysis.s\_id = Session.s\_id and Patient.p\_id = Session.p\_id and Patient.p\_id = pid;

END$$

DELIMITER ;

call DisplayAnlysisOfPatient(321561234);

**# 9 - for a given patient id count the number of sessions**

DELIMITER $$

CREATE PROCEDURE ShowNumberOfSessionForAPatient( in pid INT)

BEGIN

select \*, count(session.p\_id) as number\_of\_sessions from session where session.p\_id = pid group by session.p\_id;

END$$

DELIMITER ;

call ShowNumberOfSessionForAPatient(213562145);

**# 10 - for a given patient id display the rooms he had sessions in**

DELIMITER $$

CREATE PROCEDURE ShowRoomsThatPatientVisited( in pid INT)

BEGIN

select room.\* from session, room where session.p\_id = pid and session.r\_num = room.r\_num;

END$$

DELIMITER ;

call ShowRoomsThatPatientVisited(213562145);

**Doctor Queries:**

**# 1 - for a given doctor id show all the prescriptions he wrote**

DELIMITER $$

CREATE PROCEDURE DisplayPrescriptionsWrittenByDoctor( in did INT)

BEGIN

select \* from session, prescription where session.s\_id = prescription.s\_id and session.d\_id = did;

END$$

DELIMITER ;

call DisplayPrescriptionsWrittenByDoctor(341270222);

**# 2 - for a given doctor id show all his patients**

DELIMITER $$

CREATE PROCEDURE ShowPatientsOfDoctor( in did INT)

BEGIN

select patient.\* from patient, session where patient.p\_id = session.p\_id and session.d\_id = did;

END$$

DELIMITER ;

call ShowPatientsOfDoctor(341270222);

**# 3 - for a given doctor id display his manager info**

DELIMITER $$

CREATE PROCEDURE DisplayManagerOfDoctor( in did INT)

BEGIN

select \* from doctor,

(select doctor.manager\_id from doctor where doctor.d\_id = did) as TMP

where doctor.d\_id = TMP.manager\_id;

END$$

DELIMITER ;

call DisplayManagerOfDoctor(341270222);

**# 4 - for a given doctor id show all sessions that had more than 1 line in prescription**

DELIMITER $$

CREATE PROCEDURE ShowPrescriptionsWithMoreThanOneLineWrittenByDoctor( in did INT)

BEGIN

select \*from session,

(select prescription.prsc\_id, prescription.s\_id, count(prescription.prsc\_id) as lip\_count

from prescription, Lines\_In\_Prescription as lip

Where prescription.prsc\_id = lip.prsc\_id

group by prescription.prsc\_id) as TMPwhere session.d\_id = 341270222 and session.s\_id = TMP.s\_id and TMP.lip\_count > 1;

END$$

DELIMITER ;

call ShowPrescriptionsWithMoreThanOneLineWrittenByDoctor(341270222);

**# 5 - for a given doctor id count the number of session he had.**

DELIMITER $$

CREATE PROCEDURE ShowNumberOfsessionOfDoctor( in did INT)

BEGIN

select TMP.d\_id, TMP.session\_count from

(select \*, count(session.d\_id) as session\_count from session group by session.d\_id) as TMP

where TMP.d\_id = did;

END$$

DELIMITER ;

call ShowNumberOfsessionOfDoctor(341270222);

**# 6 - for a given doctor id display the medicines he prescribed**

DELIMITER $$

CREATE PROCEDURE ShowMedicinesDoctorPrescribed( in did INT)

BEGIN

select medicine.\* from medicine

inner join

(select prescription.s\_id, prescription.prsc\_id, lip.m\_id

from session, prescription, Lines\_In\_Prescription as lip

where session.s\_id = prescription.s\_id and prescription.prsc\_id = lip.prsc\_id and session.d\_id = did) as TMP

ON TMP.m\_id = medicine.m\_id;

END$$

DELIMITER ;

call ShowMedicinesDoctorPrescribed(341270222);

**# 7 - for a given doctor id display the number of prescriptions he wrote**

DELIMITER $$

CREATE PROCEDURE SHowNumberOfPrescriptionsWrittenByDoctor( in did INT)

BEGIN

select count(session.d\_id) as num\_of\_prsc\_written from session, prescription

where session.s\_id = prescription.s\_id and session.d\_id = did group by session.d\_id;

END$$

DELIMITER ;

call SHowNumberOfPrescriptionsWrittenByDoctor(341270222);

**# 8 - for a given doctor id show the number of diagnostics he gave**

DELIMITER $$

CREATE PROCEDURE ShowNumOfDiagnosisByDoctor( in did INT)

BEGIN

select count(session.d\_id) as NUM\_OF\_DIAG\_BY\_DOCTOR from Session, DOS where Session.s\_id = DOS.s\_id and session.d\_id = did group by session.d\_id;

END$$

DELIMITER ;

call ShowNumOfDiagnosisByDoctor(341270111);

**# 9 - for a given doctor id show the session that had a diagnosis of diabetes**

DELIMITER $$

CREATE PROCEDURE ShowSessionofDoctorWithDiabetes( in did INT)

BEGIN

select session.\*, diagnosis.diag\_name as Diagnosis from diagnosis, DOS, Session

where diagnosis.diag\_id = DOS.diag\_id and DOS.s\_id = session.s\_id and diagnosis.diag\_name like "diabetes" and session.d\_id = did;

END$$

DELIMITER ;

call ShowSessionofDoctorWithDiabetes(341270222);

**# 10 - for a given doctor id display his schedule**

DELIMITER $$

CREATE PROCEDURE DisplayDoctorSchedules( in did INT)

BEGIN

select \* from schedules as sc where sc.d\_id = did;

END$$

DELIMITER ;

call DisplayDoctorSchedules(341270111);

**Manager Queries:**

**# 1 - display the doctor that diagnosed the youngest patient with diabetes**

select doctor.\*from Session, DOS, doctor,

(select TMP2.p\_id

From (select min(patient.p\_age) as p\_age from DOS, Session, patient

where DOS.s\_id = Session.s\_id and Session.p\_id = patient.p\_id and DOS.diag\_id = 104) as TMP1,

(select patient.\* from DOS, Session, patient where DOS.s\_id = Session.s\_id and Session.p\_id = patient.p\_id and DOS.diag\_id = 104) as TMP2

Where TMP2.p\_age = TMP1.p\_age) as TMP1

Where Session.p\_id = TMP1.p\_id and Session.s\_id = DOS.s\_id and DOS.diag\_id = 104 and doctor.d\_id = Session.d\_id;

**# 2 - Display all the male Patients that are older than 45 , sorted by asc age**

select pa.\* from Patient as pa

where pa.d\_gender = 0

and pa.d\_age >= 45

order by d\_age asc;

**# 3 - Display all female doctors in age btween 35 and 59 , sorted by asc age**

select doc.\*

from Doctor as doc

where doc.d\_gender = 1 and d\_dob BETWEEN '1963-12-25' AND '1987-12-31' order by d\_age asc;

**# 4 - Display all doctors with more than 10 years of experience sorted by years of exp desc**

select doc.d\_f\_name,doc.d\_l\_name,doc.d\_years\_of\_exp

from Doctor as doc

where d\_years\_of\_exp >9 order by d\_years\_of\_exp desc;

**# 5 - Display all the doctors specific last name**

DELIMITER $$

CREATE PROCEDURE GetAllDoctorWithThisLastName( in last\_name varchar(255))

BEGIN

SELECT d\_f\_name,d\_l\_name FROM Clinic\_2022.doctor

where d\_l\_name=last\_name;

END$$

DELIMITER ;

call GetAllDoctorWithThisLastName('Geller');

**# 6 - Display all Prescription(including lines) written by a specific doctor**

DELIMITER $$

CREATE PROCEDURE GetPrcPerDoc5( in last\_name varchar(255))

BEGINselect line.prsc\_id,Medicine.m\_name,line.lip\_amount

from Lines\_In\_Prescription as line,(select pre.prsc\_id

From Prescription as pre,Session as si,Doctor as doc

where doc.d\_id = si.d\_id and pre.s\_id = si.s\_id and doc.d\_l\_name =last\_name) as tmp,Medicine

where tmp.prsc\_id = line.prsc\_id and line.m\_id =Medicine.m\_id;

END$$

DELIMITER ;

call GetPrcPerDoc5('Geller’);

**# 7 - Display all Prescription that been given in a specific day**

DELIMITER $$

CREATE PROCEDURE GetPrscInDate1( in in\_date1 varchar(255))BEGINselect line.prsc\_id,Medicine.m\_name,line.lip\_amount as amount

from Lines\_In\_Prescription as line,(select pre.prsc\_id from Prescription as pre,Session as si,Patient as pai

where pai.p\_id = si.p\_id and pre.s\_id = si.s\_id and si.s\_date = in\_date1) as tmp,Medicine

where tmp.prsc\_id = line.prsc\_id and line.m\_id =Medicine.m\_id;

END$$

DELIMITER ;

call GetPrscInDate1("2022-1-4");

**# 8 - Display Doctor information who wrote the most diagnostics**

select \*fromdoctor,

(select TMP1.d\_id, TMP1.diag\_count

From (select Session.d\_id, count(Session.d\_id) as diag\_count from Session, DOS

where Session.s\_id = DOS.s\_id group by Session.d\_id) as TMP1,

(select max(TMP1.diagnostic\_count) as max\_diag\_count from

(select count(Session.d\_id) as diagnostic\_count from Session, DOS

where Session.s\_id = DOS.s\_id group by Session.d\_id) as TMP1) as TMP2

Where TMP1.diag\_count = TMP2.max\_diag\_count) as TMP

where doctor.d\_id = TMP.d\_id;

**# 9 Display the first name of all patients who had HGB > 16**

select patient.p\_f\_name from patient, Session, analysis

where analysis.s\_id = session.s\_id and analysis.a\_HGB > 16 and patient.p\_id = Session.p\_id;

**# 10 - Display all the Patients who ever had a session in a specific room**

select patient.\*from patient,

(select \* from Session where Session.r\_num = 3) as TMP

where patient.p\_id = TMP.p\_id;

**commands to be executed by the user:**

**Insert:**

DELIMITER $$

CREATE PROCEDURE AddPatientProcedure( in pid INT, pfname CHAR(25), plname CHAR(25), pgender BOOL, pDOB DATE, patientage FLOAT )

BEGIN

insert into Patient values

(pid, pfname, plname, pgender, pDOB,patientage);

END$$

DELIMITER ;

call AddPatientProcedure(316559837, "Rick", "Sanchez", 0, '1969-1-1', 52);

call AddPatientProcedure(316559838, "Morty", "Smith", 0, '2000-1-1', 22);

**Update:**

DELIMITER $$

CREATE PROCEDURE UpdatePatientFirstNameProcedure( in pid INT, pfname CHAR(25))

BEGIN

UPDATE Patient SET Patient.p\_f\_name = pfname

WHERE Patient.p\_id = pid;

END$$

DELIMITER ;

call UpdatePatientFirstNameProcedure(316559837, "Rick1");

call UpdatePatientFirstNameProcedure(316559838, "Morty1");

**Delete:**

DELIMITER $$

CREATE PROCEDURE DeletePatientProcedure( in pid INT)

BEGIN

DELETE FROM Patient WHERE Patient.p\_id = pid;

END$$

DELIMITER ;

call DeletePatientProcedure(316559837);

call DeletePatientProcedure(316559838);