**Exercise 2 (334013968)**

**Link to github:** <https://github.com/MarmerMax/Databasase-Exercise2>

**Part a:**

package ex2;

import java.sql.CallableStatement;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class Main {

public static void main(String[] args) {

try{

Class.forName("com.mysql.jdbc.Driver");

try(Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/hospital", "root", "root")){

Statement stmt = con.createStatement();

ResultSet rs = showListOfPationtsToDoctor(999, stmt); //first function

//updateAppointmentTimeOfPatient(999, 111, con); //second function

//rs = stmt.executeQuery("SELECT \* FROM queue");

int numOfColumns = rs.getMetaData().getColumnCount();

while (rs.next()){

for (int col = 1; col <= numOfColumns; col++){

System.out.print(rs.getString(col) + " ");

}

System.out.println();

}

}} catch (Exception ex){

System.err.println("Connection fails...");

}

}

}

**1.**

public static ResultSet showListOfPationtsToDoctor(int doctorId, Statement stmt) throws SQLException {

String num = "" + doctorId;

ResultSet rs = stmt.executeQuery(

"select a.patient\_id, patient\_name, appointment\_time " + "from appointment as a join patients as p on a.patient\_id = p.patient\_id "

+ "where doctor\_id = " + num + " " + "order by appointment\_time;");

return rs;

}

**2.**

public static void updateAppointmentTimeOfPatient(int doctor\_id, int patient\_id, Connection con) throws SQLException {

String query = "call update\_appointment\_time(?, ?)";

CallableStatement stmt = con.prepareCall(query);

stmt.setInt(1, doctor\_id);

stmt.setInt(2, patient\_id);

stmt.executeQuery();

}

# ---------------------------------------------------------------------- #

# PROCEDURE: Update to actual time, uses from java#

# ---------------------------------------------------------------------- #

DELIMITER $$

CREATE PROCEDURE `update\_appointment\_time`(in d\_id int, in p\_id int)

BEGIN

update queue

set actual\_time = NOW()

where appointment\_id = (

select appointment\_id

from appointment as a

where ( (a.doctor\_id = d\_id) and (a.patient\_id = p\_id) )

);

END $$

DELIMITER ;

**3.**

# ---------------------------------------------------------------------- #

# VIEW: View table #

# ---------------------------------------------------------------------- #

create view largest\_waiting

as select patient\_name

from appointment as a inner join queue as q on a.appointment\_id = q.appointment\_id inner join patients as p on a.patient\_id = p.patient\_id

order by actual\_time - appointment\_time limit 10;

**4.**

# ---------------------------------------------------------------------------------------------- #

# TRIGGER: After insert appointment to queue update queue summary #

# ---------------------------------------------------------------------------------------------- #

DELIMITER $$

CREATE TRIGGER queue\_summary\_after\_insert\_to\_queue

AFTER INSERT ON queue

FOR EACH ROW

BEGIN

declare d\_id int;

declare new\_date date;

select a.doctor\_id into d\_id from appointment as a inner join doctors as d on a.doctor\_id = d.doctor\_id order by a.appointment\_id desc limit 1;

select date(new.actual\_time) into new\_date from queue limit 1;

if (d\_id in (select doctor\_id from queue\_summary as q where q.date = new\_date)) then

update queue\_summary

set num\_of\_patients = num\_of\_patients + 1

where (doctor\_id = d\_id and `date` = new\_date);

else

insert into queue\_summary values (date(new\_date), d\_id, 1);

end if;

set d\_id = null;

set new\_date = null;

END$$

DELIMITER ;

# ------------------------------------------------------------------------------------------------------ #

# TRIGGER: After delete appointment from queue update queue summary #

# ------------------------------------------------------------------------------------------------------ #

DELIMITER $$

CREATE TRIGGER queue\_after\_delete\_from\_queue

AFTER DELETE ON queue

FOR EACH ROW

BEGIN

declare removed\_doctor int;

declare removed\_date datetime;

declare count\_of\_patients int;

select doctor\_id into removed\_doctor

from appointment as a left join queue as q on a.appointment\_id = q.appointment\_id

where actual\_time is null limit 1;

select date(appointment\_time) into removed\_date

from appointment as a left join queue as q on a.appointment\_id = q.appointment\_id

where actual\_time is null limit 1;

select num\_of\_patients into count\_of\_patients

from queue\_summary

where (doctor\_id = removed\_doctor and date(`date`) = removed\_date) limit 1;

if ( count\_of\_patients = 1) then

delete from queue\_summary where doctor\_id = removed\_doctor and date(`date`) = removed\_date;

elseif ( count\_of\_patients > 1) then

update queue\_summary

set num\_of\_patients = num\_of\_patients - 1

where doctor\_id = removed\_doctor and date(`date`) = removed\_date;

end if;

set removed\_doctor = null;

set removed\_date = null;

set count\_of\_patients = null;

END$$

DELIMITER ;

**Part b:**

**select** doctor\_name, salary

**from** doctors **as** d **inner** **join** queue\_summary **as** q **on** d.doctor\_id = q.doctor\_id

**where** num\_of\_patients >= 5 **and** date(`date`) = '2019-03-19';

∏(doctor\_name, salary)

(**σ**(num\_of\_patients >= 5 **∧** date(`date`) = ’2019-03-19’)

(**ρ**d(doctors)(d.doctor\_id=q.doctor\_id)**ρ**q(queue\_summary)))



**Part c:**

מפתחות: {UYZ}, {ZXU}, {WXU}, {UY}

רמת נירמול: NF3