CS3743 Final Exam Spring 2020 Section 002 Name \_\_Darin Soeung\_\_\_\_

©Copyright 2020 by Larry W. Clark, this may not be copied to any website.

**Rename this file using your *lastnameFirstname*.docx**

I certify that I will not share Part1, Part2, and/or the Extras or any answers with other students. I will also not post those onto any website other than submitting this for grading.

Initials: \_\_\_DS\_\_\_ Date: \_\_5/12/2020\_\_ (If this is not initialized, you will receive a 0 on the Final Exam.)

Students who asked me to share my exam or answers (prior to or during) or communicated with me during the exam:

|  |  |  |
| --- | --- | --- |
| Student Name | Prior to the Exam (mark with Y) | During the Exam (mark with Y) |
|  |  |  |
|  |  |  |
|  |  |  |

Students who I asked to share their exam or answers with me prior to or during the exam:

|  |
| --- |
| Student Name |
|  |
|  |
|  |
|  |
|  |

Initials: \_\_\_DS\_\_\_\_\_ Date: \_\_\_5/12/2020\_\_\_\_\_

Part 2 Scores:

|  |  |  |
| --- | --- | --- |
| 1&2 | 50 |  |
| 3 | 20 |  |
| 4 | 40 |  |
| Total | 110 | 0.0 |
| 300pt Total | 330 | 0.0 |

(30pts) 1. Solve each of the following queries using **relational calculus** using one GET. Look at the Extras for the definition of the tables and sample data. Symbols: ∃r ∀r

a. Get the movie Id and title of R-rated movies which are rented with a due date after May 5, 2020 by a customer named Holly Wood.

**RANGE Customer c, Movie m, Rental r**

**Get w1a(m.movieId, m.title): m.rating = “R”**

**^∃r∃c(r.movieId = m.movieId ^ r.dueDate > “2020 – 5 - 5”**

**^ r.custNr = c.custNr ^ c.name = “Holly Wood”)**

b. Get the name and birthDt of customers who have rented all the movies with an X-rating.

**RANGE Customer c, Movie m, Rental r**

**Get w1b(c.name, c.birthDt): ∀m(m.rating = “X” ->**

**∃r(r.movieId = m.movieId ^ r.custNr = c.custNr))**

(20) 2. Solve #1a and #1b using **SQL**.

a.

**select m.movieId, m.title**

**from Customer c, Movie m, Rental r**

**where m.rating = “R”**

**and r.movieId = m.movieId and r.dueDate > “2020 – 5 - 5”**

**and r.custNr = c.custNr and c.name = “Holly Wood”**

b. (for partial credit, show your conversion from relational calculus to SQL)

**Conversion from relational calc:**

**∀m(m.rating = “X” ->∃r(r.movieId = m.movieId ^ r.custNr = c.custNr))**

**~~∀m(m.rating = “X” ->∃r(r.movieId = m.movieId ^ r.custNr = c.custNr))**

**~∃m(m.rating = “X” ^ ~∃r(r.movieId = m.movieId ^ r.custNr = c.custNr))**

**SQL Answer:**

**select c.name, c.birthDt**

**from Customer c**

**where not exists(**

**Select True**

**From Movie m**

**where not m.rating = “X” and not exists(**

**select True**

**from Rental r**

**where r.movieId = m.movieId and r.custNr = c.custNr));**

(20pts) 3. Create the Java **printMovieIdsForGenre** method which is passed a genre (e.g., ROMANCE). For the Movie table (see Extras):

* use JDBC **prepareStatement** with a substitution parameter for genre to retrieve a result set
* use appropriate methods associated with the resultSet to obtain the movie Ids and then print the movie Ids (which are **String** values) for a given genre. Note that there are many **movies** with the same genre.

You may assume:

* the instance variable **connect** has already been used to connect to our database.
* the instance variables **preparedStatemen**t and **resultSet** have already been declared as was done in program#3.

You may not use my printUtility.

void **printMovieIdsForGenre** (String genre)

{

preparedStatement = connect.prepareStatement("SELECT \* from Movie where genre = ?");

preparedStatement.setString(1, genre);

resultSet = preparedStatement.executeQuery();

while (resultSet.next())

{

String genreStr = resultSet.getString(“genre”); //can be null

String movieIdStr = resultSet.getString(“movieId”); //cannot be null

If(genreStr == null)

genreStr = “---”;

System.out.printf(“%-13s %-10s \n”, genreStr, movieIdStr);

}

System.out.printf(“\n”);

}

(40pts) **4.** Watson Hospital wants you to create the Relational Data Model for a new Watson Trauma Facility System (WTFS). WTFS has three subsystems. See the **Extras** for the complete description.

**State your assumptions.**

Use the **Relational Model** to show each relation (minimizing redundancy). For each relation, show:

* Relation Name
* Each of the attributes with the key underlined
* FDs
* MVDs

You will not need any subclasses in solving this problem.

**Answer** (remember to include your FDs or MVDs for each relation)

**Doctor(doctorId, name, medicSpecial, medIssues)**

**doctorId -> name, medicSpecialty, medIssues**

**Patient(patientId, name, birthDt, gender, room, bed, habits)**

**patientId -> name, birthDt, gender, room, bed, habits**

**primaryInsCar(patientId, carrier, insuranceId, expDt)**

**patientId -> carrier, insuranceId, expDt**

**habits(patientId, frequency, amount, type)**

**patientId -> frequency, amount, type**

**pMedIssues(patientId, medIssues)**

**mvd patientId ->> medIssues**

**dMedIssues(tMedIssues ,doctorId)**

**mvd tMedIssues ->> doctorId**

**tests(testDt, reqDoct, type, result)**

**testDt, reqDoct -> type, result**

**medReq(doctorId, patientId, drugNSN, amount, frequency, endDt)**

**doctorId, patientId, drugNSN -> amount, frequency, endDt**

**medGiven(doctorId, patientId, giveDt, giveTm, drugNSN, dosage)**

**doctorId, patientId -> giveDt, giveTm, drugNSN, dosage**