** INBOUND NOTIFICATION : FAX RECEIVED SUCCESSFULLY **

TIME RECEIVED	REMOTE CSID	DURATION	PAGES	STATUS
November 2, 2015 11:14:57 AM CST	860 633 7448	317	11	Received

Nov 02 15 01:08p BizWiz Print Copy & Signs 860-633-7448 p.1

DePaul University
Department of Online Learning
Main Fax: (866) 715 - 0863
Alt Fax: (312) 362 - 5327
Email: mjackson@cdm.depaul.edu

Exam Instruction Sheet

To:	Debbie Martin Librarian - Head, Adult Services	**From:**	Miles Jackson Manager of Online Learning
Fax:	deb.martin@glastonbury-ct.gov	**Pages:**	(Including cover page)
Phone:	860-652-7730	**Date:**	November 01, 2015
Re:	**Midterm for Jasmine Dumas**	**Phone:**	(312) 362-5286

Exam Date: **November 02, 2015**	Course: **CSC-455-710 (Database Processing for Large-Scale Analytics)**	Instructor:**Dragos Visan**
Time Limit:**3.00**	**Open Book: True** **Notes: Unlimited**	
Other: Only paper notes and books allowed. No electronic devices of any kind allowed.	Calculator: **No**	
Additional Instructions:		

Return Instructions:

Please fax the completed exam toll free to (866) 715-0863 before mailing. Please include this cover sheet.

Call me at 312-362-5286 if you have any questions. Thank you!

Please return the completed exam to:

Miles Jackson
DePaul CDM, 4th Floor
243 South Wabash Ave
Chicago, IL 60604-2301

CSC 455 Database Processing for Analytics **Section 701 (Loop Campus, Tuesdays)**
DePaul University College of CDM **Fall 2015**

Midterm Exam

You have three hours to complete this exam, which has seven questions (some of which have multiple parts) and is worth a total of 100 points. You may consult the 8.5"x11" sheet of original notes that you brought with you to the exam, but you may not use any other sources of information. The use of calculators or other electronic devices is also not permitted.

Write all of your answers on the exam sheets, and be sure that your final answer to each question is clearly indicated.

Good luck!

Name: Jasmine Dumas

(The exam has a total of 10 pages, including this cover sheet.)

80/100

6/

1. (9 points) Please give a "True" or a "False" rating to each statement below.

a. A schema that is in Second Normal Form (2NF) must also be in First Normal Form (1NF).

Answer: True ✓

b. You can test for NULL using the following SQL query:

SELECT * FROM Students WHERE SSN = NULL. (IS NULL)

Answer: False ✓

c. A RIGHT OUTER JOIN can be replicated with a LEFT OUTER JOIN. (same output re-created)

?

Answer: False ✓

d. A table can have two primary keys as long as these keys do not share any attributes.

Answer: True ✗ -1.5

e. Functional dependency AB→CD can be decomposed into two functional dependencies AB→C and AB→D.

Answer: True ✓

e. By default (without ORDER BY), the output of a SQL query is sorted by the primary key.

Answer: True ✗ -1.5

7.75

2. (12 points, 3 parts)

A. Given the following data table Order:

OrderID	Name	Date	PurchaseAmt
2	Jack Daniels	11/11/2011	200.20
3	Jane Whiskey	10/11/2011	101.20
4	Maria Rum	12/01/2012	85.00
5	Aaron Bacardi	11/11/2011	90.00

Name Date

Using the table above, what values will the query

```
SELECT Name, Date
FROM Order
WHERE PurchaseAmt =(SELECT MAX(PurchaseAmt) FROM Order);
```

200.20 = 200.20

return?

Answer:

Name	Date
Jack Daniels	11/11/2011

✓

B. Write a column definition for the PurchaseAmt (just for one column, don't worry about the rest of the table) that ensures that the purchase amount must be between 0.00 and 9,999.99

1234 4,2

PurchaseAmt Number(6,2) NOT NULL

CHECK (PurchaseAmt >= 0.00 AND PurchaseAmt <= 9,999.99)

−0.25

C. Explain what are update anomalies (either with an example or in your own words). This question is not directly related to the schema in parts A/B.

update anomalies are when you go to update/add a new value into the table but the SQL domain/data type differ. −4

when you update some values of redundant data and are left w/ two different values

Before we normalise database.

9/

3. (12 pts) Consider the two tables DBDESIGNER and DESIGNTASK created by the following SQL script:

```
CREATE TABLE DBDESIGNER
(
      ID     CHAR(7),
      Name   VARCHAR2(15),

      PRIMARY KEY (ID)
);

CREATE TABLE DESIGNTASK
(
      ID           CHAR(5),
      DesignerID   CHAR(7),
      Priority     VARCHAR(20),

      PRIMARY KEY (ID),

      FOREIGN KEY (DesignerID)
            REFERENCES DBDESIGNER(ID)
);

INSERT INTO DBDESIGNER VALUES('1234567', 'Eric Schwabe');
INSERT INTO DBDESIGNER VALUES('3456789', 'Janine Spears');
INSERT INTO DBDESIGNER VALUES('5678901', 'Will Marrero');
INSERT INTO DBDESIGNER VALUES('7890123', 'Cynthia Putnam');

INSERT INTO DESIGNTASK VALUES('11111', '1234567', 'Urgent');
INSERT INTO DESIGNTASK VALUES('22222', '1234567', 'Low');
INSERT INTO DESIGNTASK VALUES('33333', '5678901', 'Low');
INSERT INTO DESIGNTASK VALUES('44444', '7890123', 'High');
```

For your reference:

DBDesigner(ID, Name)

DesignTask(ID, DesignerID, Priority)

DBDESIGNER

ID	Name

Designtask

ID	DesignerID	Priority

For each of the following operations (to be performed individually on the original tables DBDESIGNER and DESIGNTASK), indicate whether it will violate domain constraints, entity integrity, or referential integrity. If it will not violate any of these constraints, write "OK".

a. INSERT INTO DESIGNTASK VALUES ('55555', '3456789', 'Medium');

d ✓
e ✓
R ✓

OK

b. INSERT INTO DBDESIGNER VALUES ('5678901', 'Bruce Wayne');

already exists in table

d ✓
e
R NA

entity integrity

c. INSERT INTO DESIGNTASK VALUES ('66666', '9012345', 'High');

doesn't reference PK in other table

d ✓
e
R

referential integrity

d. DELETE FROM DBDESIGNER WHERE ID='7890123';

d
e
R

OK referential

DESIGNTASK HAS FK REFERENCING IT

4. (25 points) The following set of relations records information about university students, courses and assigned grades. The Student relation contains information about the student, including the name (full name is stored in one column), address and the (projected) year of graduation. The Course relation records information about courses: course name (primary key), course department and the number of credits provided by the course. Finally, the Grade relation records information about the grades given; CName is the foreign key referring to the primary key of the Course relation and StudentID is the foreign key referring to the primary key of the Student relation. The grades are a numeric value given on a 4-point system.

Student(StudentID, Name, Address, GradYear)

Grade(CName, StudentID, CGrade)

Course(CName, Department, Credits)

For each part below, give an SQL query to display the requested information.

a. For every department, display the highest grade and the number of rows that were evaluated.

```
Select  Department , MAX(CGrade) , COUNT( Department)
FROM  Course ^ JOIN Grade ON  Course.CNAME = Grade.CNAME
GROUP BY Department ;
```

(INNER ~~FULL OUTER~~) — 0.5

b. Display the list of student IDs and names for the students who will be graduating within 4 years of the latest projected graduation (e.g., if latest graduation = 2016, within 4 years should include 2013, 2014, 2015 and 2016. This is an example; do not use these numbers in your answer). You can assume that GradYear is an integer.

```
Select  StudentID , Name
FROM  Student
Where  Grad Year >= ((SELECT MAX(GRadYear)
                      FROM student) - 4) ;
```

— 0.5

Ex: 2016 - 4 = 2012

c. Display student names and their corresponding course names for all students with the last name of 'Doe' or 'Smith'. For the purposes of writing this query, you may assume that name format is 'First Middle Last'. Your output should be sorted from the highest grade to the lowest grade.

```
SELECT Name, CNAME
FROM Student
LEFT JOIN Grade ON Student.StudentID = Grade.StudentID
WHERE Name LIKE '%Doe' OR Name LIKE '%Smith'
ORDER BY CGrade ASC ;
```

DESC

-0.50

d. For students who are not enrolled in any courses, list those student's addresses and the number of different graduation years associated with each address.

```
SELECT Address, DISTINCT COUNT(GradYear)
FROM Student
LEFT JOIN Grade ON Student.StudentID = Grade.StudentID
Where Grade.StudentID NOT IN (Select StudentID
                              FROM Student) ;
```

GradYear

-0.25

CORRECT

always FALSE

e. For each graduation year, return the average number of course credits that were taken by the students.

```
SELECT DISTINCT GradYear, AVG (Credits)
FROM Student
Full OUTER JOIN Grade ON Student.StudentID = Grade.StudentID
JOIN Course ON Grade.CNAME = Course.CNAME
WHERE GRADE.StudentID NOT NULL
GROUP BY GradYear ;
```

4/

5. (12 pts) Given the schema R and the following functional dependencies:

R(X, Y, Z, W, A) with XY→Z, A→W, Z→A

R(X, Y, Z, W, A)
XY → Z
A → W
Z → A

a. Does X determine Z? (X→Z?) Why or why not?

Yes, X and Y are dependent on each other for their dependency equalling Z, but can be decomposed to

XY → Z
X → Z
Y → Z

-4 you need both XY → Z

b. Does Z determine W? (Z→W?) Why or why not?

No, There is no backwards dependency or pseudo-transitivity between Z → W

Z → A
A → W
Z → W ? NO

-4 yes transitive

Z → A → W

c. Suppose that you were also given S:

S(P, Q, U, M, N)

What functional dependencies (if any) can you assume?

you can assume if the table schema is in proper 3NF form that all the (QUM) depends on the Keys (P, N)

PN → QUM

✓

14

6. (16 points) Suppose that we have a logical schema representing movies, movie critics and their movie ratings. The relationship "ratings" captures which reviews were posted by which critics with the rating (0-100) and date. The logical schema is given below:

Movie: **(Title, SubTitle, Year, Duration, Edition)**

Ratings: (RevID, Title, SubTitle, Rating, RDate)

Reviewer: (ReviewerID, Name, Address, Affiliation)

Assume that the tables Movie and Reviewer have already been created with all of the necessary attributes, domains, and keys. Fill out the SQL table below to create the table Ratings with all of the necessary attributes, domains, and keys. Assume that Reviewer ID is a 9 characters long (fixed) string and that neither title nor sub-title needs more than 20 characters and that the rating value cannot be NULL. You only need to create the Ratings table. Do not worry about creating Movie or Reviewer tables.

```
CREATE TABLE Ratings (
RevID       CHAR(9),
Title       VARCHAR2(20),
SubTitle    VARCHAR2(20),
Rating      NUMBER(3) NOT NULL
RDate       DATE,
CONSTRAINT  UNIQUE_CONSTRAINT_PK1
        PRIMARY KEY( RevID ),
CONSTRAINT UNIQUE_CONSTRAINT_FK2
        FOREIGN KEY ( Title, SubTitle )
        REFERENCES Movie (Title, SubTitle)
CONSTRAINT UNIQUE_CONSTRAINT_FK3
        FOREIGN KEY( SubTitle RevID )
        REFERENCES Movie (SubTitle) Reviewer (RevID);
```

0-100 - 3 digits

-1 CHECK BETWEEN 0-100

-0.5

-0.5

13 5/

7. (14 points, two parts) Consider a TVShows table that keeps track of different TV shows. The table stores the show name and the year to which the entry refers. Additionally, each row stores channel name, length of the show, the average cost of an episode, and the filming location's zip, city and state. Moreover, each entry contains the name of the lead actor and their salary.

The table is already in First Normal Form, and its primary key is (Show, Year).

The schema for the TVShows table is:

(Show, Year, Channel, Length, Cost, Zip, City, State, Lead, Salary)

You are given the following functional dependencies:

Show → Length only Show

Zip → City, State

Lead → Salary

Show, Year → Channel, Cost

a. Remove any existing partial dependencies to create a set of linked relational schemas in Second Normal Form.

TVShows (Show, Year, Channel, Cost, Zip, City, State, Lead, Salary)

Shows (Show, ~~Year~~, length)

-0.5

b. Remove any existing transitive dependencies to create a set of linked relational schemas in Third Normal Form.

TV Shows (Show, Year, Channel, Cost, Zip, Lead)

Shows (Show, length)

Zip code (zip, city, state)

Lead Actor (Lead, Salary)

-0.5