**Final Exam**

Class Room: Online

Assignment Points: 15 points

Thursday 12/3/2020

**Name: Felipe Lima**

**Student ID: 109290055**

**Exam rules:**

* You MUST submit this final exam by **12/5/2020, 11:59 pm.** There will not be any extension or late submission.
* Submit your assignment in PDF format in Canvas. You can use word, excel or similar tools and convert into pdf.
* This is open book exam and any kind of resource materials are allowed.
* Collaboration and consultation is NOT allowed. Do your own work.

**Section 1: 3 points**

Normalize the following form into **3NF.** Only your 3rd NF will be graded.

|  |  |  |
| --- | --- | --- |
| **University Departments Sample Form** | | |
|  |  |  |
| Dept Name | ……………………….. |  |
| Building Num | ……………………….. |  |
| Phone 1 | xxx-xxx-xxxx |  |
| Phone 2 | xxx-xxx-xxxx |  |
| Phone 3 | xxx-xxx-xxxx |  |
|  |  |  |
| Instructor Name | Subject | Gender |
| ……….. ……………. | …………….. | X |
| ……….. ……………. | …………….. | X |
| ……….. ……………. | …………….. | X |
| ……….. ……………. | …………….. | X |

This is the University departments sample form used by many departments.

If there is no concatenated key or many to many relationship, many times you can put directly into 3rd NF, i.e., do not carried away with unnecessary normalization. There is not always 1st, 2nd and 3rd NF needed.

**Normalize**, as you did in HW-1 Normalization.

Hints: List all attributes.

****Identify the repeating group of attributes.

Create entities and keys e.g. PK/FK.

You don't need more than 3 entities in your 3 NF.

**Section 2: 3 points**

Create **ERD design** for following scenario:

Your data model design (ERD) should include relationships between tables with primary keys, foreign keys, optionality and cardinality relationships. Captions are NOT required.

**Scenario:** There are 3 tables with 2 columns in each table:

**Department (** Dept ID, Department Name )

**Employee (** Employee ID, Employee Name )

**Activity (** Activity ID, Activity Name )

Each Employee must belongsto ONLY ONE Department.

Department may have ZERO, ONE OR MORE Employees, i.e. Department may exists without any employee.

Each Employee may participate in ZERO, ONE OR MORE Activities

****Each Activity may be performed by ZERO, ONE OR MORE Employees.

**Section 3: 2 points**

1. Create table **T1** with following columns and constraints.

Note: DO NOT use alter table, list all constraints while creating table.

C1 INT (10) Primary key

C2 INT (10)

C3 INT (10)

C4 VARCHAR (40)

**Constraints:**

C3 NON-ZERO

C2 greater than C3

C4 default value of 'HR'

DROP TABLE IF EXISTS T1;

CREATE TABLE T1

(

C1 INT (10) PRIMARY KEY,

C2 INT (10),

C3 INT (10) NOT NULL,

C4 VARCHAR (40) DEFAULT 'Data Analyst',

CHECK (C2>C3),

CHECK (C3 != 0)

);

1. Create table **T2** with following columns and Foreign Key.

Note: DO NOT use alter table, create FK while creating table.

C5 INT (10) Primary key

C6 INT (10)

FK on C6 column referencing to C1 column in table T1 above.

DROP TABLE IF EXISTS T2;

CREATE TABLE T2

(

C5 INT (10) PRIMARY KEY,

C6 INT (10),

FOREIGN KEY (C6) REFERENCES T1(C1)

);

1. Explain, in short, the meaning and importance of Referential Integrity (RI).

Referential Integrity (RI) is a Foreign Key Constraint used to prevent that changes in one part of the system don't lead to unanticipated problems elsewhere. For example, it prevents inserting a row into a child table where the parent key value is missing, and deletions and updates are also restricted. This makes it so that connected fields (keys that reference other keys in other tables) update together and not separately.

**Section 4: 4 points**

All questions are based on below **Employees table**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EmpId** | **ManagerId** | **Name** | **Department** | **Salary** | **City** |
| 1 | 0 | Alex Smith | Admin | $90,000 | Boulder |
| 2 | 1 | Amy Mars | Admin | $50,000 | Longmont |
| 3 | 1 | Logan Mars | Admin | $70,000 | Longmont |
| 4 | 1 | James Mont | Marketing | $55,000 |  |
| 5 | 6 | John Smith | Marketing | $60,000 | Boulder |
| 6 | 1 | Lily Mars | Marketing | $95,000 |  |
| 7 | 6 | Ravi Grace | Database | $75,000 | Longmont |
| 8 | 6 | Tara Frank | Database | $80,000 | Longmont |
| 9 | 6 | Tom Ford | Database | $65,000 |  |
| 10 | 6 | William Cruze | Database | $85,000 | Longmont |

1. Write a SQL statement to find the Name and Salary who has **5th HIGHEST** Salary in the entire Employee table.

**SELECT Name, Salary FROM Employees ORDER BY `Salary` DESC LIMIT 4,1;**

1. Write a SQL statement to find the Department and their count whose count is more than 3.

**SELECT Department, COUNT(Department) AS Count**

**FROM Employees**

**GROUP BY Department**

**HAVING Count > 3;**

1. Write a SQL statement to show Name, Department and City.

However, if City is NULL, then display 'Broomfield' otherwise display City itself.

**SELECT Name, Department, IF (City IS NULL, "Broomfield", City) AS City**

**FROM Employees;**

1. Write a SQL statement to find distinct employee Name who is also a Manager

**SELECT e.Name**

**FROM employees e**

**INNER JOIN employees m**

**ON (e.EmpId = m.ManagerId)**

**GROUP BY e.Name;**

1. Write a SQL statement to find Maximum, Minimum and Average Salary from the entire Employee table.

**SELECT MAX(Salary)AS "Max", MIN(Salary) AS "Min", AVG(Salary) AS "Avg"**

**FROM Employees;**

1. Write a SQL statement to show Name, Department and Salary who earn MORE THAN the Average Salary in **THEIR department**. You must use sub-query.

**SELECT e.Namee, e.Department, e.Salary**

**FROM Employees e**

**INNER JOIN (SELECT Department, AVG(Salary) AS Av FROM Employees GROUP BY Department) avgs**

**ON e.Department = avgs.Department**

**WHERE Salary > avgs.Av**

**GROUP BY e.Namee, e.Department, e.Salary;**

1. Write a SQL statement to show Name, Department, Salary and their Rank **WITHIN Department** from highest to lowest salary.

i.e, Salary rank must reset and re-rank start from 1 for EACH Department.

**SELECT Name, Department, Salary, RANK() OVER (**

**PARTITION BY Department**

**ORDER BY salary DESC) salary\_rank**

**FROM employees**

**GROUP BY Name, Salary, Department;**

1. Write a SQL statement to find HIGHEST paying employee’s Name and Salary from the entire Employee table. You must use sub-query.

**SELECT Namee, Salary FROM(SELECT Namee, Salary,**

**RANK () OVER (**

**ORDER BY Salary DESC) ranking**

**FROM Employees)t1**

**WHERE ranking = 1;**

**Section 5: 3 points**

Create a Retail Sales Company **Data Warehouse design** using **STAR schema** from following info. Make sure to indicate proper \_DIM and \_Fact tables and their PKs/FKs. You need to join those tables using JUST straight lines (optionality and cardinality relationships are NOT required).

Date, Month, Year, SupplierName, SupplierCountry, ProductName, ProductCategory, CustomerName, CustomerCity, SalesItem, SalesPrice, SalesQty, SalesAmount

Note: You may use MySQL workbench or just handwritten to create STAR schema Data Warehouse design.



☺ ☺ ☺ The END ☺ ☺ ☺