Technical University of Denmark

DTU Compute
Department of Applied Mathematics and Computer Science

02170 Mandatory Group Project
Spring 2020

©Anne Haxthausen

Contents

- Practical information (on groups, hand-in and evaluation).
- The project tasks and requirements.

Practical Information

- Mandatory
  - It is mandatory to hand in a mandatory group project and get it approved in order to participate in the final written examination!
  - The final mark for this course has been planned to be the mark from the written examination. **NOTE that the exam form might change if DTU decides to cancel written exams due to Corona, and in that case it could happen that the project results will be taken into account when giving the final grade.**

- Groups
  - **Groups must already have been registered**!
  - Students who did not register have been allocated a project group by the TAs!
  - The groups and their group ids can be found in a file in the Project folder in the course File Sharing on DTU Inside. Each group id is formed by a TA name and a number, like Vittorio4.
  - The members in a group must contribute equally to the project. Only persons who have contributed are allowed to appear as report authors.

- Hand-In: What, Where and When
  - Each group should upload the following files to the Course Assignments folder on DTU Inside and set the group name to their group id (see above), not later than **Saturday April 4th** at 18:00:
    1. The **group project report** in a .pdf file named *id*_02170GroupReport2019.pdf, where *id* is your group id (e.g. Vittorio4). It must have the sections explained on one of the following pages.
    2. An **SQL script** in a .sql file named *id*_02170DatabaseScript1_2019.sql , where *id* is your group id (e.g. Vittorio4). It must contain
       (1) the statements used to create the database, its tables and views (in section 5 of the report)
       (2) the statements used to populate the tables (in section 6)
    3. An **SQL script** in a .sql file named *id*_02170DatabaseScript2_2019.sql , where *id* is your group id (e.g. Vittorio4). It must contain
       (1) the queries made (in section 7)
       (2) the delete/update statements used to change the tables (in section 8), and
       (3) the statements used to create and apply functions, procedures, triggers, and events (in section 9)

    It is a requirement that there are no run-time errors when running the scripts under MariaDB.
- The Results of the Group Project Evaluation
  - Will be communicated via DTU Inside.

Task, Objectives, and Scope

- Task:
  - is to develop and document a database of your own choice.
- Objective:
  - is to get practical experience with data modelling and database design.
- Scope:
  - Only SQL programming of the database is requested, no application logic and no user interface. MariaDB must be used for the implementation.

Examples of Databases

- Each group must choose their own example (not same as other groups) and make the database development independently. It is not legal to copy a database made by somebody else (avoid plagiarism).
- For inspiration, examples of databases developed by past 02170 students:
  - Hospital Doctor & Patient Relations Database
  - Bank Database
  - Retail Shop Database
  - Superliga - Soccer Matches & Results Database

Mandatory Report Requirements

Mandatory Sections	Tasks and Contents of Mandatory Sections
Title Page	Course Name & No, Group No, Project Title, Student Names and IDs, Date.
1. Statement of Requirements	Describe in plain words the part of the real world being modelled.
2. Conceptual Design	Show an Entity-Relationship Diagram for the domain of your database using the Textbook Adapted UML Notation. Explain. Discuss choices made.
3. Logical Design	Convert your conceptual design into a logical design (relation schemas) and discuss any choices made. Show a database schema diagram in the Textbook notation.
4. Normalization	For each relation schema in your logical design, state its normal form and explain what makes it be in the given normal form. Ensure that all schemas become in at least 3NF.
5. Implementation	Create a MariaDB database with tables and views implementing the logical design.
6. Database Instance	Populate the tables with data, and list data for all tables and views.
7. SQL Data Queries	Give at least three examples of typical select SQL statements with order by, group by and joins etc. For each query explain informally what it asks about. Show also the output of the queries.
8. SQL Table Modifications	Give examples of typical SQL table update and delete statements. Show the results of the statements.
9. SQL Programming	Give examples of functions, procedures, transactions, triggers, and events, and explain what they do. Show illustrative usage examples.

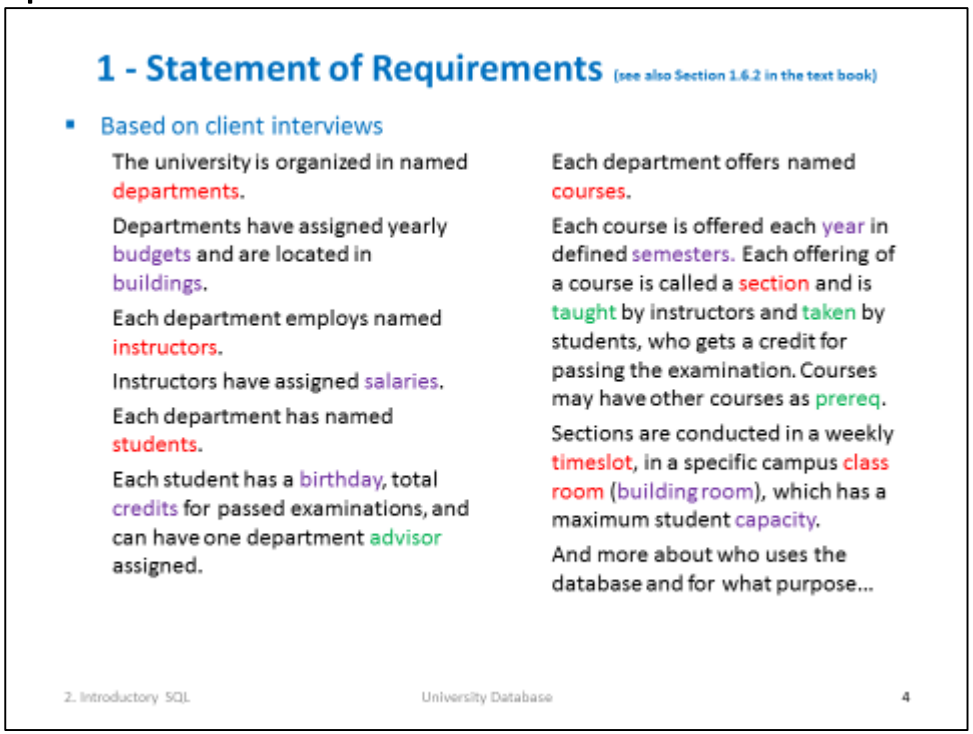
Title Page and Report Format

- Title Page
  - Make the Title Page inviting and interesting
    - It gives the reader a first good impression
  - Include
    - Course name & number
    - Project title
    - Group number (as found in the Project folder of the file sharing on DTU Inside)
    - Student names, IDs, and pictures (the last is optional)
    - Date

- Report Format
  - Include page numbers.
  - Include a table of contents.
  - Include pictures and drawings to clarify text.
  - Use readable fonts for various text elements and picture captions.
  - Include an appendix for additional material, if needed.
  - Include a bibliography, if needed.

1. Statement of Requirements

- Describe in plain words the part of the real world being modelled.
  - Example from a slide :



1 - Statement of Requirements (see also Section 1.6.2 in the text book)

- Based on client interviews

The university is organized in named departments.

Departments have assigned yearly budgets and are located in buildings.

Each department employs named instructors.

Instructors have assigned salaries.

Each department has named students.

Each student has a birthday, total credits for passed examinations, and can have one department advisor assigned.

Each department offers named courses.

Each course is offered each year in defined semesters. Each offering of a course is called a section and is taught by instructors and taken by students, who gets a credit for passing the examination. Courses may have other courses as prereq.

Sections are conducted in a weekly timeslot, in a specific campus class room (building room), which has a maximum student capacity.

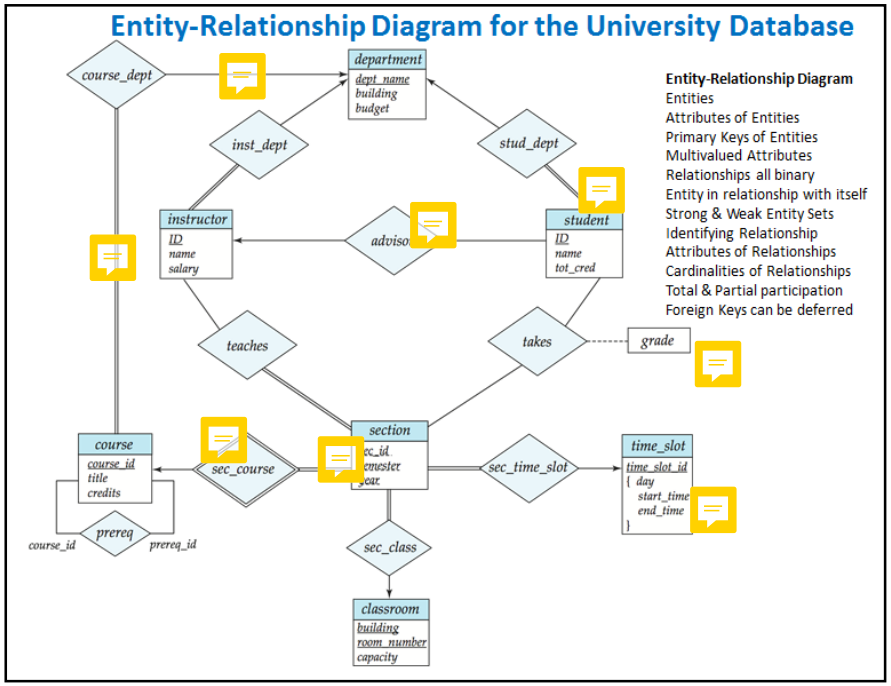
And more about who uses the database and for what purpose...

2. Introductory SQL University Database 4

However, make the text in readable and complete paragraphs and sentences!

2. Conceptual Design

- Show an Entity-Relationship Diagram for the domain of your database using the Textbook Adapted UML Notation.
  - Use the Textbook Adapted UML Notation and follow the strict rules to show (1) strong & weak entity sets with their names, attributes and primary keys, and (2) relationship sets with their names, attributes, primary keys, cardinalities, and total/partial participation.
  - Example:



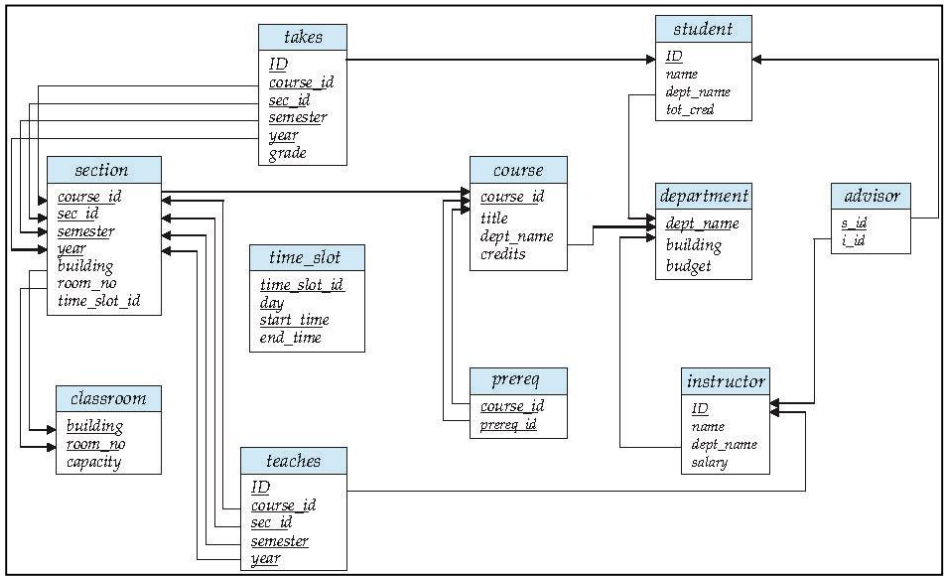
- Explain. Explain the meaning of entities, relationships, and their attributes.
- Discuss any choices made. E.g. why the cardinalities and participation constraints are chosen as they are etc.

3. Logical Design

- Convert your conceptual design into a logical design (relation schemas inclusive specification of foreign keys) and discuss any choices made. You must follow the method described in the course slides and book (Chapter 7).
  - Example: Conversion of the diagram shown on previous slides gives

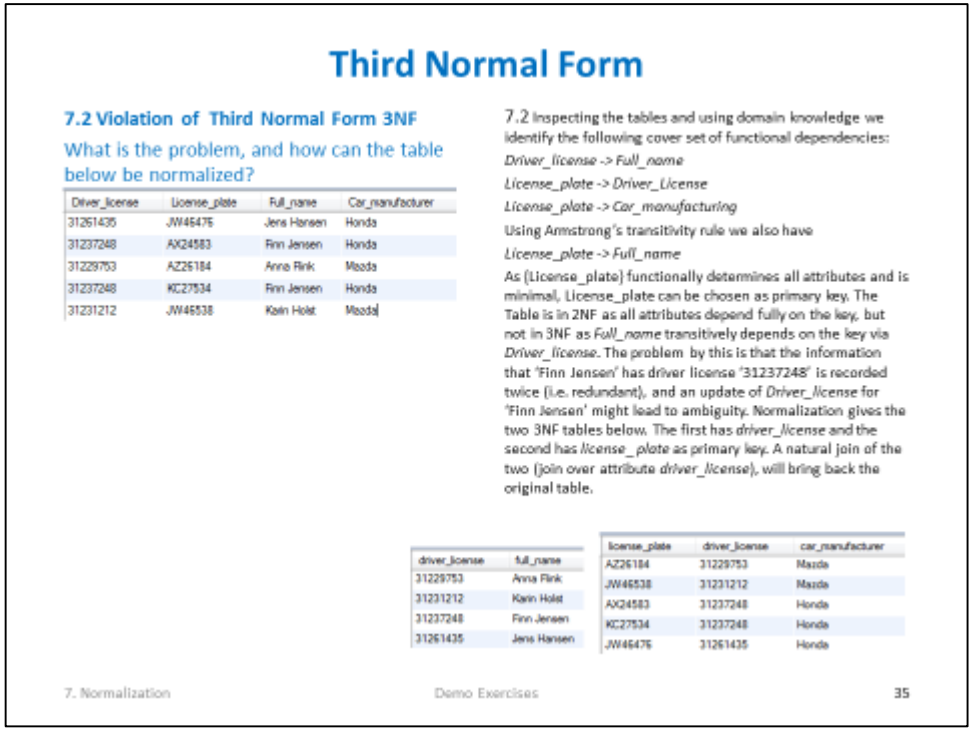
    Instructor(<u>InstID</u>, InstName, DeptName, Salary) **foreign key** (DeptName) **references** Department(DeptName)
    Department(<u>DeptName</u>, Building, Budget)
    ….
- Show a database schema diagram for the relation schemas of the logical design in the Textbook notation. Example of a database schema diagram for the University DB:

Textbook notation:



4. Normalization

- For each relation schema in your logical design, state its normal form and explain what makes it be in the given normal form.
- If some tables are not at least in 3NF: Then check whether this is due to some problems in your conceptual model or its conversion to a logical design. If so fix the problems, otherwise normalize the tables directly to 3NF.
  - Example:



Third Normal Form

7.2 Violation of Third Normal Form 3NF

What is the problem, and how can the table below be normalized?

Driver_license	License_plate	Full_name	Car_manufacturer
31261435	JW46476	Jens Hansen	Honda
31237248	AX24583	Finn Jensen	Honda
31229753	AZ26184	Anna Fink	Mazda
31237248	KC27534	Finn Jensen	Honda
31231212	JW46538	Kain Holst	Mazda

7.2 Inspecting the tables and using domain knowledge we identify the following cover set of functional dependencies:

Driver_license -> Full_name

License_plate -> Driver_License

License_plate -> Car_manufacturing

Using Armstrong's transitivity rule we also have

License_plate -> Full_name

As {License_plate} functionally determines all attributes and is minimal, License_plate can be chosen as primary key. The Table is in 2NF as all attributes depend fully on the key, but not in 3NF as *Full_name* transitively depends on the key via *Driver_license*. The problem by this is that the information that 'Finn Jensen' has driver license '31237248' is recorded twice (i.e. redundant), and an update of *Driver_license* for 'Finn Jensen' might lead to ambiguity. Normalization gives the two 3NF tables below. The first has *driver_license* and the second has *license_plate* as primary key. A natural join of the two (join over attribute *driver_license*), will bring back the original table.

driver_license	full_name
31229753	Anna Fink
31231212	Kain Holst
31237248	Finn Jensen
31261435	Jens Hansen

license_plate	driver_license	car_manufacturer
AZ26184	31229753	Mazda
JW46538	31231212	Mazda
AX24583	31237248	Honda
KC27534	31237248	Honda
JW46476	31261435	Honda

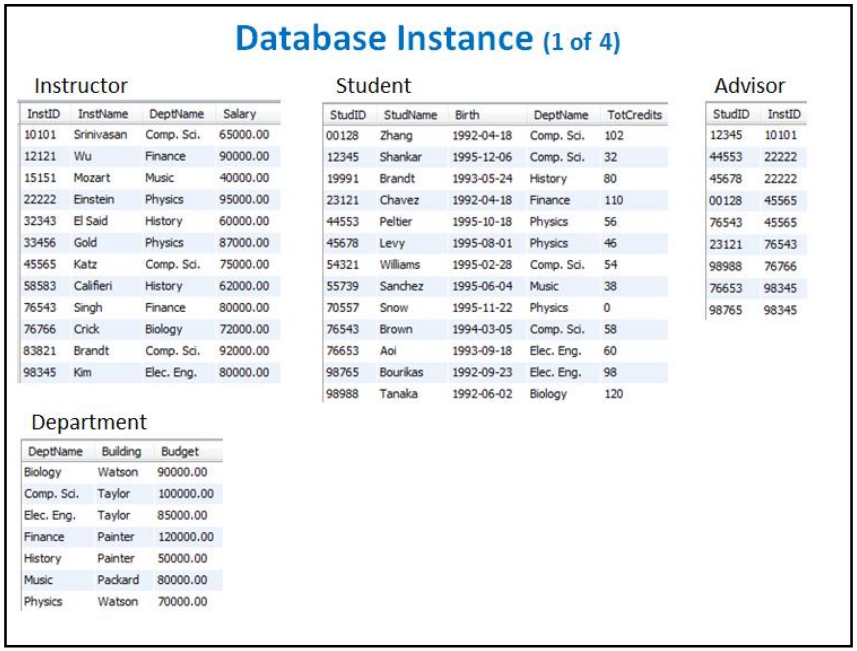
7. Normalization Demo Exercises 35

5. Implementation

- Create a MariaDB database with tables and views implementing the logical design (as achieved after possible revisions in step 4):
  - Use SQL statements CREATE DATABASE, CREATE TABLE and CREATE VIEW.
  - Show the statements in the report.

6. Database Instance

- Populate the tables with data, and list data for all tables and views.
  1. Use SQL **INSERT** to populate the tables.
  2. Use SQL **SELECT * FROM** *table* to list instances of all tables and views.
  3. Show the output of step 2 in the report.
- Example of output, from one of the course slides:



Database Instance (1 of 4)

Instructor

InstID	InstName	DeptName	Salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

Student

StudID	StudName	Birth	DeptName	TotCredits
00128	Zhang	1992-04-18	Comp. Sci.	102
12345	Shankar	1995-12-06	Comp. Sci.	32
19991	Brandt	1993-05-24	History	80
23121	Chavez	1992-04-18	Finance	110
44553	Peltier	1995-10-18	Physics	56
45678	Levy	1995-08-01	Physics	46
54321	Williams	1995-02-28	Comp. Sci.	54
55739	Sanchez	1995-06-04	Music	38
70557	Snow	1995-11-22	Physics	0
76543	Brown	1994-03-05	Comp. Sci.	58
76653	Aoi	1993-09-18	Elec. Eng.	60
98765	Bourikas	1992-09-23	Elec. Eng.	98
98988	Tanaka	1992-06-02	Biology	120

Advisor

StudID	InstID
12345	10101
44553	22222
45678	22222
00128	45565
76543	45565
23121	76543
98988	76766
76653	98345
98765	98345

Department

DeptName	Building	Budget
Biology	Watson	90000.00
Comp. Sci.	Taylor	100000.00
Elec. Eng.	Taylor	85000.00
Finance	Painter	120000.00
History	Painter	50000.00
Music	Packard	80000.00
Physics	Watson	70000.00

7. SQL Data Queries

- Give at least 3 examples of typical select SQL statements with order by, group by and joins etc. For each query explain informally what it asks about. Show also the output of the queries.
  - Example illustrating group by:

- Find the name and average salary for those departments whose average salary is greater than 42000.

```
SELECT DeptName, AVG(Salary) FROM Instructor
GROUP BY DeptName HAVING AVG(Salary) > 65000;
```

DeptName	AVG(Salary)
Biology	72000.000000
Comp. Sci.	77333.333333
Elec. Eng.	80000.000000
Finance	85000.000000
Physics	91000.000000

8. SQL Table Modifications

- Give examples of typical SQL table update and delete statements.
  - Show with illustrative examples how you do table modifications using the SQL commands UPDATE and DELETE.
- Example:

Example of UPDATE Statement

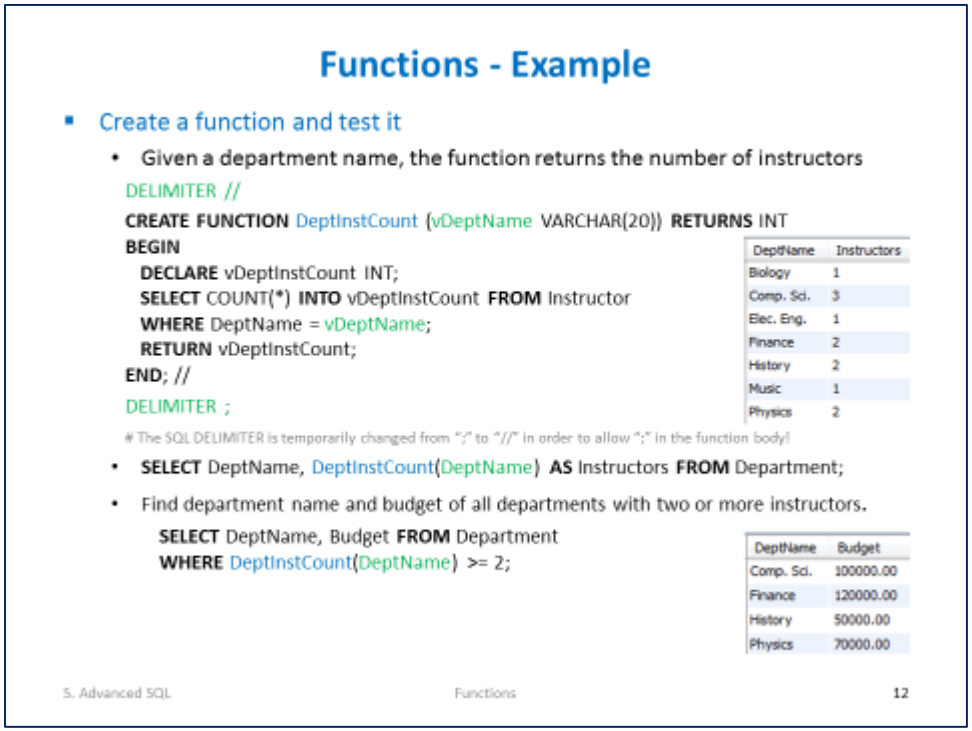
- The following statement increases salaries of instructors whose salary is over 80000 by 3%, and all others with a 5% raise.

```
UPDATE Instructor SET Salary =
  CASE
  WHEN Salary<=80000
  THEN Salary*1.05
  ELSE Salary*1.03
  END;
```

Also show the contents of the table after the update/delete.

9. SQL Programming

- Give examples of functions, procedures, transactions, triggers, and events, and explain what they do. Give one example of each.
  - Remember also to show illustrative usage examples of how they work.
- Example:



Functions - Example

- Create a function and test it
  - Given a department name, the function returns the number of instructors

```
DELIMITER //
CREATE FUNCTION DeptInstCount (vDeptName VARCHAR(20)) RETURNS INT
BEGIN
  DECLARE vDeptInstCount INT;
  SELECT COUNT(*) INTO vDeptInstCount FROM Instructor
  WHERE DeptName = vDeptName;
  RETURN vDeptInstCount;
END; //
DELIMITER ;
```

DeptName	Instructors
Biology	1
Comp. Sci.	3
Elec. Eng.	1
Finance	2
History	2
Music	1
Physics	2

The SQL DELIMITER is temporarily changed from ";" to "//" in order to allow ";" in the function body!

  - `SELECT DeptName, DeptInstCount(DeptName) AS Instructors FROM Department;`
  - Find department name and budget of all departments with two or more instructors.

```
SELECT DeptName, Budget FROM Department
WHERE DeptInstCount(DeptName) >= 2;
```

DeptName	Budget
Comp. Sci.	100000.00
Finance	120000.00
History	50000.00
Physics	70000.00

5. Advanced SQL Functions 12