University of Glasgow | School of Computing Science

Assessed Coursework

Course Name	**Database Systems 3**			
Coursework Number	**1**			
Deadline	**Time:**	**05:00 pm**	**Date:**	**19th FEBRUARY 2016**
% Contribution to final course mark	**10%**			
Solo or Group ✓	**Solo**	✓	**Group**	
Anticipated Hours	**10**			
Submission Instructions	**See details in assignment and on the Moodle page.**			
Please Note: This Coursework cannot be Re-Assessed				

Code of Assessment Rules for Coursework Submission

Deadlines for the submission of coursework which is to be formally assessed will be published in course documentation, and work which is submitted later than the deadline will be subject to penalty as set out below.

The primary grade and secondary band awarded for coursework which is submitted after the published deadline will be calculated as follows:

(i) in respect of work submitted not more than five working days after the deadline
   a. the work will be assessed in the usual way;
   b. the primary grade and secondary band so determined will then be reduced by two secondary bands for each working day (or part of a working day) the work was submitted late.

(ii) work submitted more than five working days after the deadline will be awarded Grade H.

Penalties for late submission of coursework will not be imposed if good cause is established for the late submission. You should submit documents supporting good cause via MyCampus.

Penalty for non-adherence to Submission Instructions is 2 bands

You must complete an "Own Work" form via https://studentltc.dcs.gla.ac.uk/ for all coursework

UNLESS submitted via Moodle

First Assessed Coursework

DEADLINE 19 FEB 2016; 05:00pm

1. Introduction

You will need to work **individually** on this exercise. You should be able to complete the totality of the exercise in about **8-10 hours**, including thinking and revision time.

[**This work is formally assessed and will constitute 10% of the marks for the course**]

Publish: Friday 5 February 2016, **Deadline**: Friday 19 February 2016

2. Database Scheme & Description

Let us assume a database that contains information about bands, their members and their releases over the years. The following schema has been proposed for this database:

Band (bid, name, country, webpage)

Release (rid, bid, title, year, type, rating)

Song (title, rid, cdbonus)

Member (mid, name, stillalive)

MemberOf (mid, bid, startyear, endyear, instrument)

The schema includes the following information, where [type] indicates that the associated attribute is of type [type]:

- **Bands**, including
  - their names [VARCHAR(50)],
  - country of origin if known [VARCHAR(20)],
  - their official webpage if available [VARCHAR(120)] and
  - a unique INTEGER as identifier in the database (i.e. **bid**).

- Different **releases** of the **bands**, including
  - the title of each release [VARCHAR(120)],
  - the year of the release [INTEGER],
  - the type of the release (e.g., “album”, “live”, “mini” or “single”) [VARCHAR(10)],
  - and, if available, a rating by a specialist critic (an INTEGER out of 10).
  - Every release has a unique INTEGER as identifier in the database (i.e. **rid**) and refers to a band (i.e., **bid** field).

- **Songs**, including
  - their titles [VARCHAR(120)],
  - which release they belong to (i.e., **rid** field),
  - and whether they are a CD bonus or not (i.e. “Y” for Yes, “N” for No) [CHAR(1)].

- **Members**, including
  - their names [VARCHAR(50)],
  - whether they are still alive or not (i.e. "Y" for Yes, "N" for No) [CHAR(1)]
  - and a unique INTEGER as identifier in the database (i.e., **mid**).

- **Relationships** between **Members** and **Bands**, stating
  - whether a particular member served in a specific band,
  - with a start year and an end year (INTEGERs),
  - and their role in the band (i.e. instrument with values "vocals", "guitar", "bass", "keyboards" and "drums") [VARCHAR (15)].

In addition, the designer of the database has chosen the following *assumptions* for the data Input:

(i) **EndYear** will have **NULL** as value if the corresponding member is still a member of the band.

(ii) **StartYear** will have **NULL** as value if the corresponding member has been in the band since its creation.

3. Assessed Work

3.1 Notes

Note 1: When writing your solutions, you must ***not*** use views in any of your solutions. You must ***not*** use operators that are not part of the SQL3 standard (e.g. LIMIT, NVL), and you should *not* use SQL3 procedures (e.g. COALESCE, CASE, IF ... ELSE statements); using the standard SQL to the fullest extent possible makes your queries portable from a DBMS to another.

All your SQL solutions **must** be readable and **must** be preceded by a **short description** in the form of a comment that briefly explains how your solution works.

Note 2: The table contents from the CSV files found in the file: **FIRST-ASSIGNMENT-FILES.zip** on the Moodle. Each file name matches the table name and includes tuples with attributes in the exact order given in the provided database scheme, so please create your tables and import the data properly.

You can also refer to the pgAdminIII introduction sheet to see an example of how to populate a table from a CSV file. Since some tables (e.g. Song) contain Latin characters, you have to choose the appropriate encoding types (UTF8).

3.2 SQL Statements over the 'Bands Database'

- STATEMENT: **Provide the CREATE TABLE statements, and justify the choice of any keys or constraints you declared.**

  [Points: 25]

- QUERY 1: **Find all the members called 'Tim', who are still alive.**

  [Points: 5]

- QUERY 2: **Get the names of all Iron Maiden's current band members.**

  [Points: 10]

- QUERY 3: **Find out if any band has the same name as any member of any band.**

  [Points: 10]

- QUERY 4: **Show how many members in the database have played bass.**
  The column should be renamed to as **Bass_Player_Count**

[Points: 10]

- QUERY 5: **Display the discography of each drummer.**

  Columns should be renamed to as **Drummer_Name**, **Release_Name**, **Year**
  All rows for the same drummer should appear together and sorted according to release year.

[Points: 10]

- QUERY 6: **Write an SQL query to display the number of bonus tracks that <u>each</u> band has recorded, whose corresponding release was rated over 5 or there is no rating.**

  Columns should be renamed to as **Band_Name**, **Bonus_Num** and sorted by**: Band_Name**

[Points: 10]

- QUERY 7: **Find the names of the bands whose each release's year is after 1999.**

[Points: 10]

- QUERY 8: **Find the names of the bands with releases since 2011.**

[Points: 10]

Total Points: [100]

4. Submission Instructions

Submissions should be made through Moodle.

Each student must complete the online Declaration of Originality Form from:

https://webapps.dcs.gla.ac.uk/ETHICS/index.cfm

By the stated deadline, you must submit the following:

1. A PDF report that clearly indicates your name and matriculation number.
2. A .txt file containing the CREATE TABLE statements.
3. A .txt file containing the SQL SELECT statements **for each** QUERY <u>including</u> comments <u>and</u> short description.

This is an individual exercise. Your SQL solutions must be your own. If you receive any assistance from anyone other than the DB3 lecturer, you must declare this assistance.

When writing SQL queries, you should only use operators that are part of the SQL3 standard.