24303 DATABASES
LAB01-01: DB design, DDL, DML & DQL

Goals:
The lab will cover the following aspects in the **exact** following **order**:

1. Complete the **Conceptual Model** of the Music Festival scenario as described in LAB00.
2. **Design Relational Model** matching the previous Conceptual Model.
3. Implement SQL commands to **create the actual DB in MySQL** (Data Definition Language – DDL).
4. **Insert data** coming from .csv files in the tables using SQL commands (Data Manipulation Language – DML).

Probably, you have to use additional study time to complete this practice. Students are encouraged to check all the material in detail before submission.

Note: The University has provided the required software to develop this practice, which is properly installed in the computers of the laboratory room classes. You will always have the option of working on the university computers. Furthermore, the professors have provided installation guides and manuals to facilitate the installation in your personal computers and for diverse OS systems (Windows, MacOS and other Unix-Like systems).

Submission: The statement of the project must be read in order, following the instructions given step by step.

After the second session of LAB01-01, you must finish and submit:

1. A **pdf document** with the designs of the **Conceptual** and **Relational Model** and the explanation of the justification of assumptions and decisions made. Please take care of the presentation, export the designs properly using the export option in the in draw.io menu, add the name of all the participants in the group, watch you writing, etc. Be professional. The file will be called **music_festival_db_models.pdf**

2. Three **SQL scripts**:
   2.1. Creation of the database called **music_festival_db_deploy.sql**
   2.2. Insertion data called **music_festival_db_population.sql**
   2.3. Database export including data called **music_festival_db_export.sql** *(you can use MySQL Workbench export wizard or mysql dump commands in the terminal)*

This lab will be done in teams of three people (exceptionally two). The project will be developed in two class sessions and the teacher will assist you if you have technical problems for its development. Teachers will guide you and provide hints and tips but not any kind of solution.

Each group will submit the **music_festival_db_models.pdf**, **music_festival_db_deploy.sql, music_festival_db_population.sql** and **music_festival_db_export.sql** files (all in a ZIP file). The ZIP file should be called with the NIA of each member of the group (for instance, NIA1_NIA2_NIA3.zip).

ONLY A MEMBER OF THE GROUP WILL BE RESPONSIBLE FOR RETURNING THE WORK and always after the second session of LAB01-01.

It is really important to follow each point of the submission instructions. Otherwise, the practice will not be properly qualified, that is, it will be evaluated with a 0. **Please, before your submission check the following requirements:**

- All your scripts work BEFORE your submission
- You have defined your tables and fields using the names specified in LAB00
- The file scripts must include the UTF-8 codification
- The files scripts must follow the names specified in this document

Deadline: See Aula Global.

1. Creation of the Conceptual Model (25 points)

Create an accurate database Conceptual Model design able to handle the music festival scenario as described in LAB00 using the proper figures from draw.io IDE.

1.1. Correction criteria

Every design will start from the maximum mark (15 points) and there will be subtractions in the following cases:

- Poor effort or quality on the delivered document, (eg. model difficult to understand) (-5)
- Missing entity or non related to the rest of the model (-2)
- Missing PK on an entity (-1)
- A wrong weak entity identification (-1)
- Missing generalisation or bad execution of it (-1)
- Loss of provided information, (eg. non existing attribute for an entity) (-1)
- Wrong relationship definition, (eg. wrong cardinality) (-1)

2. Creation of the Relational Model (25 points)

Starting from the Conceptual Model created on LAB00, create the matching Relational Model according to it. Since you may have more knowledge now, check if there are any changes needed to be done.

Prepare a pdf document (**music_festival_db_models.pdf**) including exported designs of both **Conceptual Model** and **Relational Model** and also you may want to **expose some assumptions and decisions** you have done. There is not only one correct solution. Do not trust other teams' criteria, because two different solutions can be both great or bad. Just follow your path.

The delivered document has to contain the name of all the group participants on a front page and the designs have to be explained and defended with few lines.

Note: It may be interesting to check out the data that is intended to be inserted later on in the database to ensure that your design meets all the requirements.

2.1. Correction criteria

Every design will start from the maximum mark (10 points) and there will be subtractions in the following cases where the Relational Model doesn't match the Conceptual Model:

- Poor effort or quality on the delivered document (-5)
- Missing table (-2)
- Missing PK on a table (-1)
- Missing column from an existing attribute (-1)
- Wrong translation or missing FK (-1)
- Non matching table or column name (-1)

3. Creating the database (25 points)

Following the database design you proposed, you will have to decide the data types, domains of the fields (columns), primary and foreign keys for each table. Then, you have to implement the SQL commands to create the full database.

The sequence of the SQL commands must respect the constraints imposed by the table's foreign keys (FK). In other words, the order of the table's creation matters. You can also decide to create all the tables with PK but without FK and later on do ALTER tables to add the FKs in the proper order.

Compose all the SQL creation commands in a file called **music_festival_db_deploy.sql**.

This file should begin with the following three instructions:

```
DROP SCHEMA IF EXISTS music_festival;
CREATE DATABASE IF NOT EXISTS music_festival
DEFAULT CHARACTER SET 'utf8mb4'
DEFAULT COLLATE 'utf8mb4_general_ci';
USE music_festival;
```

The execution of all the commands of this file should not cause any errors. If it contains errors, it will not be possible to evaluate the rest of the lab.
The use of the DROP TABLE IF EXISTS and CREATE TABLE IF NOT EXISTS statements for each table should be taken into account to avoid errors in case the table already exists before creation. After giving it a try, it is normal to have part of the database created and you may need to DROP it before trying again from scratch.

The resulting script has to have the property of being executed several times and always deliver the same result so it is reusable and does not raise any errors.

The name of the database must be **music_festival**.

At this stage it is really important that you **check out the provided data on the .csv files** to be inserted in the database later on in order to match the columns' data type definition for each table.

Take into account the data types modificators such as UNSIGNED for integer columns in case of avoiding negative values, AUTO_INCREMENT, NULL or NOT NULL in case a field is accepting NULL value or not , DEFAULT value, ENUMS, VARCHAR(XX), BIT, TINYINT, MEDIUMINT, INT, etc.

3.1. Correction criteria

Every delivery will start from 25 points on this phase and there might be subtractions in case of:

- For every significant difference between the previous proposed database design vs the final implemented database (-10)
- Missing table (-2)
- Wrong field data type or wrong data type modifications (e.g. UNSIGNED) (-1)
- A field is missing from a table (-1)
- A PK is missing or bad definition of it (-1)
- An FK is missing or bad definition of it (watch out the composed ones!) (-1)
- Wrong table or column name (-1)
- The sql script does not provide always the same result after several runs (-1)

Regarding the reuse of the file, pay attention to the table creation order to resolve all dependency conflicts. The creation script has to have the property of being launched many times and always delivers the same result.

4. Populating the database (25 points)

On the Aula Global you will find a file called **music_festival_db_data.zip**, which contains the information to be inserted into the database. **Disclaimer:** any similitude between provided data and the real world are just due to random facts and no animal has been harmed but some beers were drunk.

You can create a sequence of INSERTs commands to properly fill your database tables using the data contained in the provided excel file. You can generate the inserts manually or use the CONCATENATE function in Excel.

You can also use the LOAD DATA statement to load the data into temporary tables **(if you create extra temporary tables, include them in your delivery)** and after that INSERT the structured data into the final tables.

Note: Even though you may need to split the information from some of the data files, **it is prohibited to modify or delete any data from the provided zip file under no circumstance.** All the provided data has to be inserted.

The INSERT and/or LOAD DATA statements must be submitted in a file called **inserts_nba_db.sql**.

LOAD DATA is an all-in-one SQL instruction that allows you to load a full datafile from the hard drive to a database table.
You can find more info about LOAD DATA statement here: https://dev.mysql.com/doc/refman/8.0/en/load-data.html

Formalization:

```
LOAD DATA INFILE "file.csv" -- Place your data files into @@datadir path
INTO TABLE MYTABLE -- Destination table
COLUMNS TERMINATED BY ',' -- The special character used to separate cols
OPTIONALLY ENCLOSED BY '"' -- In case of VARCHAR enclosed between “”
LINES TERMINATED BY '\n' -- How the rows are separated
IGNORE 1 LINES; -- To avoid the header on first line
```

Sometimes it may be necessary to use LOAD DATA *LOCAL* INFILE... instead of LOAD DATA INFILE…

There are several modificators for the LOAD DATA instruction such as choosing which table columns to be inserted, data modifications in load time, etc.

> **Note:** There might arise some errors depending on the configuration of your local database server such as *--secure-file-priv*
> Refer to Aula Global to find a solution for that.

As commented before, when your SQL file is executed, it should not raise any error. **If it contains errors, it will not be possible to evaluate the rest of the lab.**

Important: Consider the order of the insertions, otherwise this is a frequent pain in the ass like a warm beer because: primary keys and foreign keys violation. For instance, it makes no sense to insert the data regarding bar sales first if we have not previously inserted data of the bars, the products and the festivalgoers. Since each sale is associated with a bar, a product and a person, we need to load all these three before inserting the sales.

Remember to also prepare the database export and deliver it under the name **music_festival_db_export.sql. This is mandatory in order to be evaluated.**

4.1. Correction criteria

Every delivery will start from 20 points in this phase and there might be subtractions in case of:

- Modification of the original data (-20)
- Some data is missing on the resulting database (-5)
- A one row INSERT statement is not properly formatted or it cannot be executed (-1)
- A multirow INSERT statement is not properly formatted or it cannot be executed (-5)
- A LOAD DATA statement is not properly formatted or it cannot be executed (-5)
- Data redundancy on a table (-5)
- The sql script does not provide always the same result after several runs (-1)

Regarding the reuse of the file, pay attention to the insertion order to resolve all dependency conflicts due to the integrity laws and truncate or delete table data before inserting it.

In order to do double checks, query your tables to know its rowcount after insert.