24303 DATABASES
LAB01-02: DQL & PL/SQL

Goals:
The lab will cover the following aspects:

a. Implement **queries in SQL** (Data Manipulation Language – DQL)
b. Thinking of database improvements
c. Implement **PL/SQL requirements**: Procedures, Triggers and Events

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

Note: The University has provided the required software to develop this lab, 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).

Methodology:

*****IT IS MANDATORY TO HAVE FINISHED LAB01-01 BEFORE STARTING LAB01-02; OTHERWISE YOU MAY NOT BE ABLE TO WORK IN THIS SECOND PART *****

This LAB01-02 is divided into three class sessions. The statement of the lab must be read in order, following the instructions given step by step.

<u>Session 1:</u>

- Create a new SQL file for queries (**music_festival_db_queries.sql**) and write your own queries based on the lab statement.

<u>Session 2:</u>

- Provide improvements to the database with a new Relational Model and deliver it in a pdf file: **music_festival_db_new_models.pdf** with explanations and file: **music_sql_improvments.sql** with structure code improvements.

- Start working on the list of new requirements for the database that will consist in creating procedures, trigger and events structures. You should create an SQL file for each: (**req01_music_festival.sql**, **req02_music_festival.sql [...] reqNN_music_festival.sql**). One for every new requirement.

<u>Session 3:</u>

- Continue working on the implementation of the new requirements (procedures, triggers and events as asked).

Submission:

<u>You must implement and submit:</u>

1. A pdf document with your improvements to the **Relational Model** and the argumentation of the changes (**music_festival_db_new_models.pdf**).
2. Multiple SQL scripts:
    a. Solution for the proposed queries (**music_festival_db_queries.sql**).
    b. Solutions for DB requirements (**req01_music_festival.sql, ...**).

This lab must be done in the same groups of TWO or THREE people as LAB01-01. The lab will be developed in three sessions and the teacher will assist you if you have technical problems for its development. **THE TEACHER WILL NOT PROVIDE ANY SOLUTION**.

Each group will submit the following files:

1. **music_festival_db_queries.sql**
2. **music_festival_db_new_models.pdf**
3. **reqN_music_festival.sql** ***(one for every requirement)***

All in a ZIP file. The ZIP file should be called with the NIA of each member of the group (for example, NIA1_NIA2.ZIP for groups of 2 people).

ONLY A MEMBER OF THE GROUP WILL BE RESPONSIBLE FOR RETURNING THE WORK and always after the second session of LAB01-02.

EACH WORKING GROUP WILL DELIVER * ONE VERSION *** OF THE LAB**

It is really important to follow each point of the submission instructions. Otherwise, the lab 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 this document.
- 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. Introduction

This LAB01-02 is an evolution of the work done in LAB01-01 so you need your relational model and data that every group presented in order to be able to solve this delivery.

Please read the lab statement in order and work through the exercises in order.

The **first step** is where your work starts by developing the solutions for the asked queries.

The **second step** is to analyse the database and find errors or optimizations and how to correct those errors only in the relational model.

Third and last step, will be to work on the new requirements in developing the PL/SQL code for some Procedures, Triggers and Events.

2. Queries (40 points)

In this phase you will need to create a new sql file (**music_festival_db_queries.sql**). You are asked to implement SQL queries in order to provide some required information. For each query you deliver, create an associated view. Everything should be informed into the view sql code: filters, operations, joins, etc. Also add a comment with the resulting rowcount.

For instance, for the first query:

```
DROP VIEW IF EXISTS query_1;

CREATE VIEW query_1 AS
      SELECT whatever
       FROM dummy
       WHERE lorep = 'ipsum';

-- Total: 128 rows
```

1. Because there is no planet B, the festival's organisers don't want to produce more plastic glasses than necessary. They are also studying other materials different from plastic such as soya, walnut and pine nut shells. To produce a glass from pine nut shells, they need 200 shells and an average squirrel eats 400 pine nuts per day discarding the shell.
   How many squirrels do they need in order to have enough pine nut shells to produce a glass for each festivalgoer who doesn't own one yet and in just one day?

2. The manager of the company wants to know the profile of the festival attendants. Provide the different nationalities among festivalgoers and the volume of people coming from each country. Order the result alphabetically by the country.

3. As you may know, one of the sensible aspects of the music festivals are the toilets. Since it is hard and expensive to clean them, we want to identify the festivalgoers who cannot eat spicy food but they did it anyway and now they feel dizzy because of that.
   Show the personal and festivalgoer information for all of them sort by person id ascending.

4. Security guards are going mad because they have seen some festivalgoers walking around without the bracelet on their wrists and they started to think some of them cheated to get in.
   Is there any festivalgoer who attended a music festival without having purchased a ticket? Show its complete personal information sorted by id_person in order to fine them.

5. Is there any repeated band name? If so, show them including its complete information to be able to differentiate them. First identify the repeated ones and after that, show its complete information from the band table.

6. Every year and at every music festival, after hours of music and party, there are some festivalgoers sleeping on their feet. This can be dangerous if they don't feel well because they could fall and get hurt and we don't want that. Also we have been reported that there is a pattern that some of them say that they have some kind of food intolerance. For all the festival editions, localise the festivalgoers who don't eat gluten and don't drink alcohol but, for unknown reasons, they feel wasted.
   We cannot know where they are in real time, but we can know in which concert stages they have been to check if they are sleeping on the floor.
   Retrieve its complete personal information ordered by the person id and the complete information of the stages they have been.

7. Show the personal information of the community managers who work as a freelance and manage the social media accounts for the Creamfields festival. Consider only the accounts with at least 500k followers and under 700k.
   Show also the platform name and the followers. Order the results by community manager id asc.

8. Find the beermans who have sold more litres of beer among all the Primavera Sound editions. Consider each beerman' sell as one beer of 0.33 litres. Show the id_beerman and the amount of litres sorted descending by the litres. **EXTRA**: Can you also provide the beerman name, surname, nationality and birthdate and the amount of litres sold?

9. Find how many songs have been sung by Rosalia among all the festivals. Return the detailed information regarding the festival name and edition, the stage details, the title of the song and the ordinality (the order of the songs being sung in the show)
   EXTRA: Can you now calculate how many seconds Rosalia has sung among all her songs?

10. Find the complete information of the providers who have served some veggie products and which have been consumed in any of the editions of the Tomorrowland music festival. Show only one time each provider and order them by provider id asc.

11. Make a list of distinct shows ordered descending by its duration coming from the played songs.

12. For every stage, return a list of which percentage of capacity has been achieved. Consider the more optimistic scenario where every festivalgoer will go to all the stages for all the festival editions for which they have a ticket.

13. Return a list of names and surnames of the festivalgoers who attended all the Primavera Sound editions *(**hint:** subquery should be necessary)*.

14. We would like to check unemployment for every staff type. Consider that a beerman has never worked if has not sold anything, a bartender is not related to a bar, a security member is not related to any festival stage and a community manager is not managing any account. The output must have two columns: staff type and count of unemployed.
    Order results from more unemployment to less. *(**hint:** subquery should be necessary)*.

15. We want to know how many (*a number*) of our staff workers (community managers, beermans and security's but not bartenders) have been working at any of the editions of Primavera Sound (***hint:*** *subquery should be necessary)*.

16. For a specific festivalgoer (filter in where clause a festivalgoer ID of your desire); return his or her spendings on festivals including, price tickets, beers bought and consumptions in bars. Consider that beers sold by beermans have a price of 3 US dollars.

2.1. Correction criteria

The correct implementation of all queries will be evaluated **with 40 points**. **Each correct query will give you 2.5 points.**

In this context, 3 scenarios are possible:

1. The query is correct, it shows the expected results and it has the proper rowcount. You get the maximum qualification for the query.

2. The query is not totally correct but it does not return any error. That is, the output is close to the expected result. In this case, you get half the maximum qualification.
   For instance:
   2.1. The rowcount is not as expected due to duplicated values and a partial join
   2.2. Some fields are missing.
   2.3. The sorting is not correct.

3. If the query presents syntax errors or is not able to be executed, you do not get any mark for the query.

3. Analysing the database (10 points)

You may have already identified in the Relational Model some aspects that could be improved.

Following the instructions given at the LAB00 statement about the description of the music festival scenario, the resulting database model may not represent the information of product sales associated with each music festival edition. So, we cannot know which bars are present at each music festival and, therefore, we don't know which products have been sold to which festivalgoer in each festival.

The managers of the music label want to track that.

As a professional database designer, if you've identified other aspects to be improved apart from that, include them (justified) in your proposal.

Using draw.io, **apply** the needed **modifications** to your delivered **Relational Model** at LAB01-01 and propose a solution design able to handle the information regarding which products have been sold to who and at which music festival edition.

You must create a PDF file (**music_festival_db_new_models.pdf**) including:

1. **Explanations and justifications** about your correction proposals
2. New version of the **Relational Model scheme** including the changes (and the rest of the schema also).

The SQL script (**music_sql_improvments.sql**) needed to apply the design changes to the actual implemented database from LAB01-01. You may use ALTER to modify a current table or create new ones. *(You do not need to apply your proposed changes to your database)*

3.1. Correction criteria

This part will be evaluated **with 10 points**. On this terms:

- Justifications and explanations of proposed changes (including extras) - **4 points**
- New Relational Model design - **3 points**
- SQL scripts to apply the proposed changes - **3 points**

4. New requirements for the database (50 points)

On this part you will be asked to develop new database objects: **Procedures**, **Triggers** and **Events**. For all the following requirements, remember to use:

```
DROP db_object IF EXISTS db_object_name;

CREATE db_object IF NOT EXISTS db_object_name ...
```

For the **events** to work, remember to activate the event scheduler.

Deliver all needed PL/SQL sentences of every requirement on a new sql file using the following naming convention: **reqNN_music_festival.sql** *(being NN the number of the requirement)*.

Note: Submitted scripts have to be able to be launched several times and deliver always the same result.

4.1. Requirement 1 (6 points)

Create a new procedure able to populate and update the information of **years_experience** contained in the **Staff table**.
Use the database server system date YEAR(CURRENT_DATE()) and subtract hire date information for each staff in order to calculate the years of experience.

The procedure should check all the staffs inside the Staff table and update only the field years_experience but only for the ones which have NULL value or wrong information on this field; avoid doing extra data modifications where the information is already correct.

The goal is that this procedure is able to maintain the data updated relative to the current date.

Include an example of a procedure CALL in your delivery.

Extra: What do you think about the field years_experience? Which kind of field is it? Does it respond to a great database design?

4.2. Requirement 2 (2 points)

Once the **previous requirement 1 is done and correct**, **create a new database object** which will check the years of experience for each staff member every first day of the month at 08:00:00 h with no expiration date *(**hint:** use a CALL to the procedure created in the previous requirement)*.

Hint: Refer to official documentation to see how to create an event: https://dev.mysql.com/doc/refman/8.0/en/create-event.html

4.3. Requirement 3 (8 points)

Every single time a new bartender is **inserted in the music festival database**, if it is not assigned to any specific bar (the FK from bartender to bar table has null value), assign her/him to one of the bars with less assigned bartenders *(you must find first which bars have less bartenders assigned)*.
Do the same every time a **new security member** is inserted to the database without assigning her/him to any stage. Assign her/him to the stage with less assigned security members.

Develop a proper solution for that to happen automatically before any new bartender or security member is inserted. *(**hint:** You may need to create two new database objects).*

4.4. Requirement 4 (8 points)

Create a new **procedure** which will check whereas there is **some band** with an **assigned music type for which there is no existing song assigned to it** in the database.

If it is the case, set the music type of the band **to the value of the music type which has less** bands assigned to it.

Include an example of a procedure CALL in your delivery.

4.5. Requirement 5 (10 points)

All the monetary prices in the database are specified with US Dollar currency. Since the festival's management is international, we were asked to **add more currencies to the system**.

Looking at the database tables there are columns with product prices: products provided and products in a bar which inform about the base price for each product in USD.

Do the following **DDL actions for each table**:

1. Rename unit_price to UnitPriceUSD *(without losing the data in it)*
2. Create a new field called UnitPriceEUR with the same data type as UnitPriceUSD
3. Create a new field called UnitPriceGBP with same data type as previous
4. Create a new field called UnitPriceJPY with same data type as previous

Create a new **stored procedure** which will populate those new fields using an approximation of the actual currency exchange ratios:

1 USD - 0.93 EUR
1 USD - 0.82 GBP
1 USD - 151.51 JPY

The procedure has to have an **input parameter** which will be used in order to specify if it is required to recalculate the foreign currency values for all the rows inside necessary tables (input parameter equals 0), or only for a single product identified with its id_product (input parameter > 0 and presumably will match some of the actual id_product value).

If the value of the input parameter is bigger than 0 but it doesn't match any of the id_product available, the procedure will do nothing but print a Warning message informing that there is no Product with this identifier.

Note that some rows may not have their price informed or with NULL value. The procedure has to control that and avoid execution errors. In this situation, set all the not informed prices to value 0.

Include an example of a procedure CALL in your delivery.

Extra: Create a new table in the database to store currency exchanges and use it in your procedure. Think about the best possible solution regarding the structure of the new table (columns, data types, PK, FK ?) and how to use it properly. If done, include all the needed scripts in your delivery.

4.6. Requirement 6 (6 points)

It is necessary to automate the calculation of the prices as per all the currencies in order to maintain them up to date.

Every single time a product price is changed, check that it matches the exchange rates specified in requirement 2. If not, use the **procedure created for requirement 2 to modify the row** which will take USD price as base price and update the other ones**.**

4.7. Requirement 7 (2 points)

Schedule an event that executes the **procedure created at requirement 2** every day starting on 15th November 2023 8.00 h am and ending at 31st january 2024 by midnight.
The execution should update all prices for all currencies.

4.8. Requirement 8 (8 points)

Create a **new procedure** which **will export a report** (.csv file) **including a list of festival name and edition and its count of festivalgoers** which purchased the ticket. Order descending the result list by attendance.

See below the formalisation of the action of exporting a file:

```
SELECT column1, column2, columnN
INTO OUTFILE 'FileName.csv'
FIELDS TERMINATED BY ';'
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM Table1 T1
     JOIN Table2 T2 ON T1.column1 = T2.column1
     …
WHERE T1.columnN = 'lorem ipsum'
     AND …
GROUP BY …
ORDER BY …;
```

This will export a file into the @@datadir directory: `SELECT @@datadir;`

Note1: If you are facing issues regarding --secure-file-priv, refer to Aula Global for further solutions.

Note2: Remember that having .csv files into the *datadir* folder will not allow you to DROP the database. If there is an existing file in the folder with the same name, the procedure will raise an error.

Extra: Make the name of the report dynamic according to the date it is being generated (Example: report_2023_11_13.csv)

Include an example of a procedure CALL in your delivery.

4.9. Correction criteria

This part will be evaluated **with 50 points**, on this terms:

- Each requirement has its own points specified.
- Each requirement will be evaluated on those terms:
    - **Zero points** if the requirement is not done, have compilation errors or does not deliver what is asked.
    - **Half** the requirement **points** if it compiles and delivers nearly all of what it is asked.
    - **Full** requirement points if it compiles and delivers all of what it is asked.
- Each extra task will add 1 extra point to the total *(the final mark of this part cannot exceed 50 points).*