

Homework Assignment #1

Database Systems – Semester A 2025/2026

Due 04/12/2025

This assignment consists of two parts:

- **Part A** – This part is technical, and you need to follow the instructions in order to set up the environment on which you'll run the database.
- **Part B** – Contains eight questions that you are required to answer and submit on Lemida.

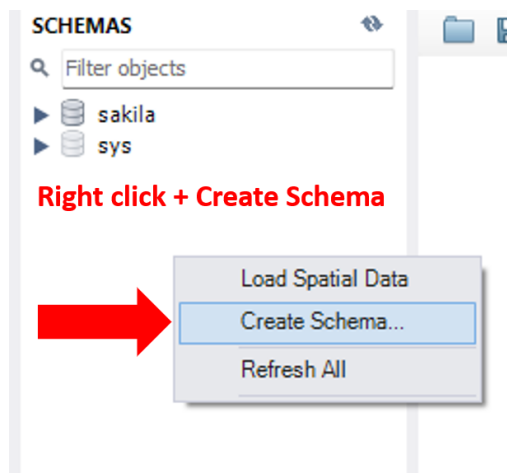
Part A - Installation

Before you start

- Make sure you followed Exercise 0 before this exercise
- Download the files from Lemida.

Then, you can either follow this YouTube tutorial, or follow these instructions, both are doing the same import in MySQL Workbench:

1. Load your SQL workbench using docker as we learned in class.
2. Create a new schema and call it "f1_data" in your workbench.



Name: Specify the name of the schema here. You can use any combination of ANSI letters, numbers and the underscore character for names that don't require quoting. For more flexibility you can use the entire Unicode Basic Multilingual Plane (BMP), but you will have to quote the name later.

Character Set: Collation: Refactor model, changing all references found in view, triggers, stored procedures and functions from the old schema name to the new one.

The character set and its collation selected here will be used when no other character/collation is set for a database object (it uses the DEFAULT value then). Setting DEFAULT here will make the schema to use the server defaults.

Click apply afterward

Apply Revert

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Review the SQL Script to be Applied on the Database

Online DDL

Algorithm: Lock Type:

```

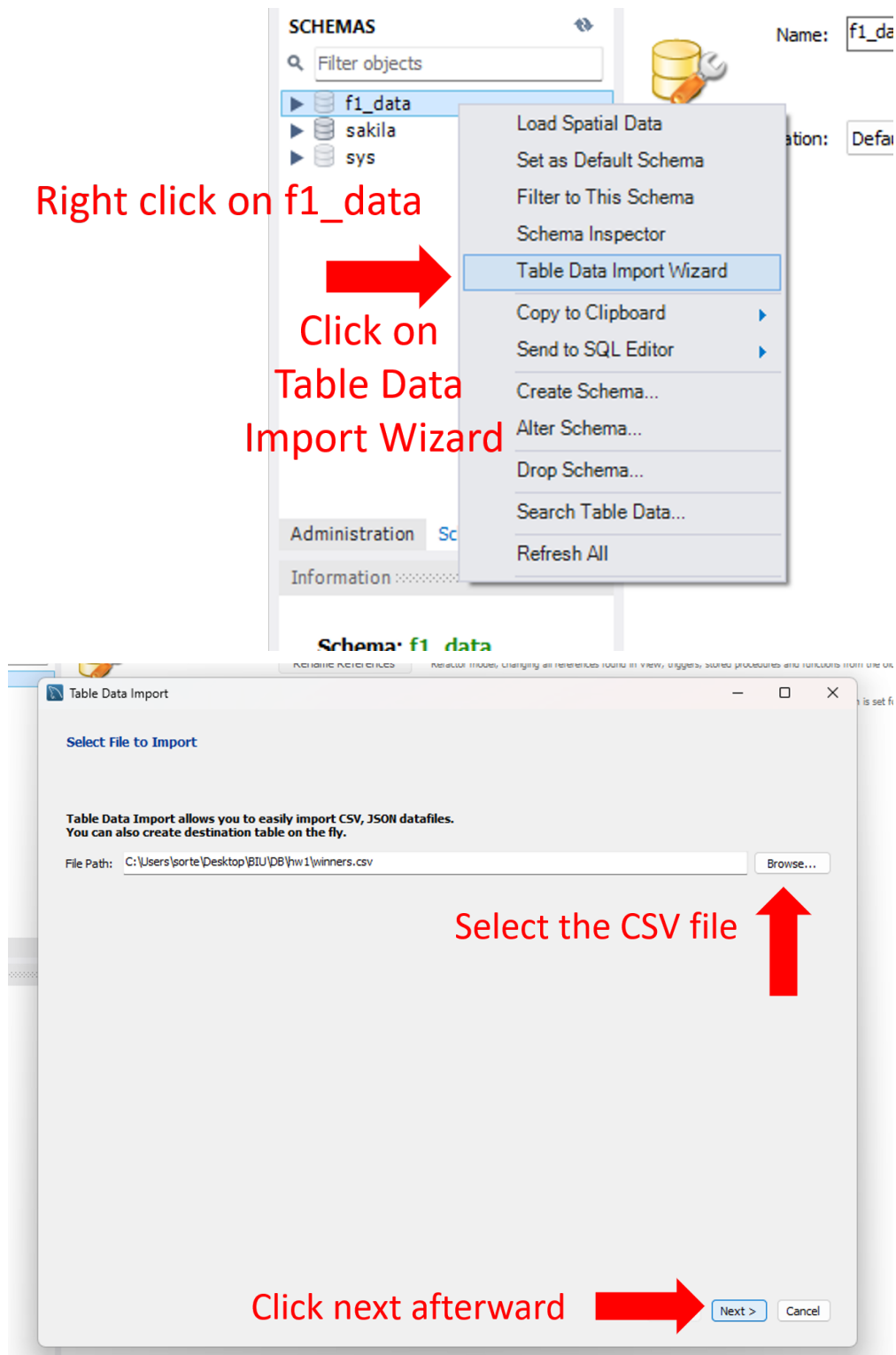
1 CREATE SCHEMA `f1_data` ;
2

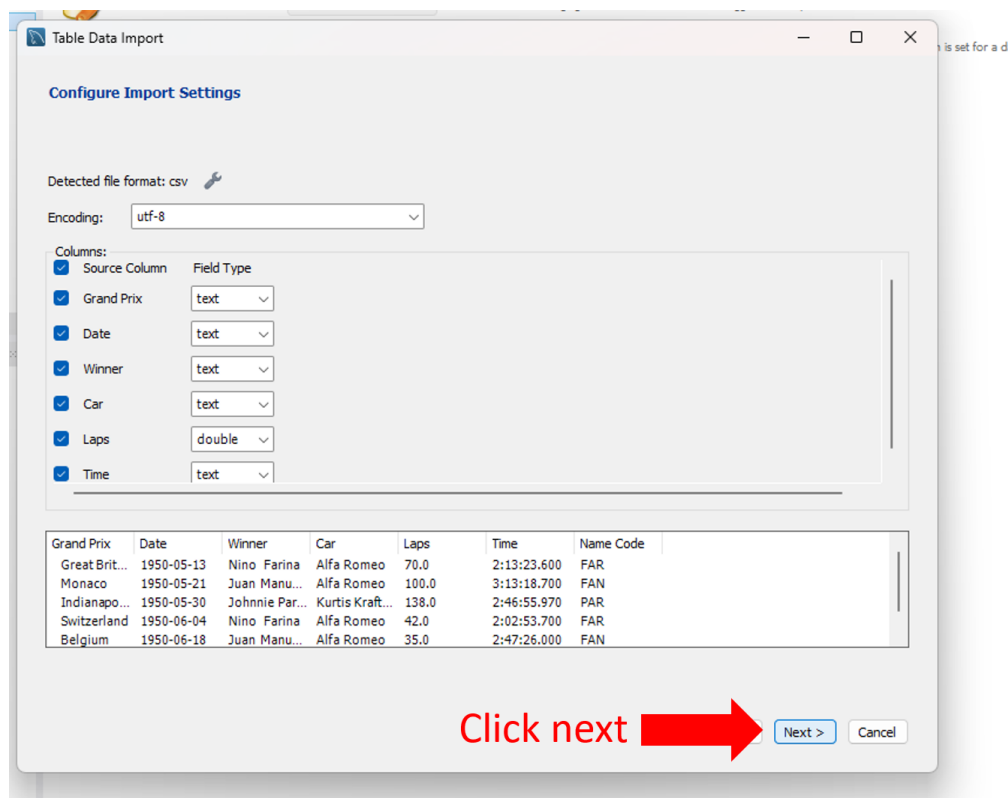
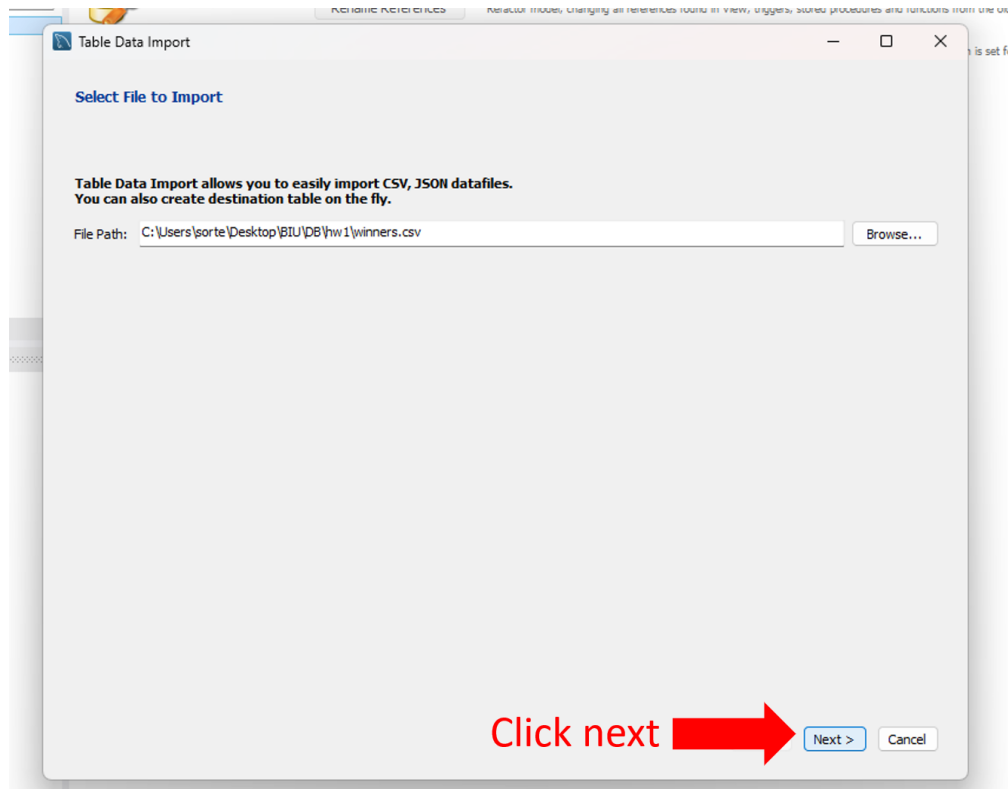
```

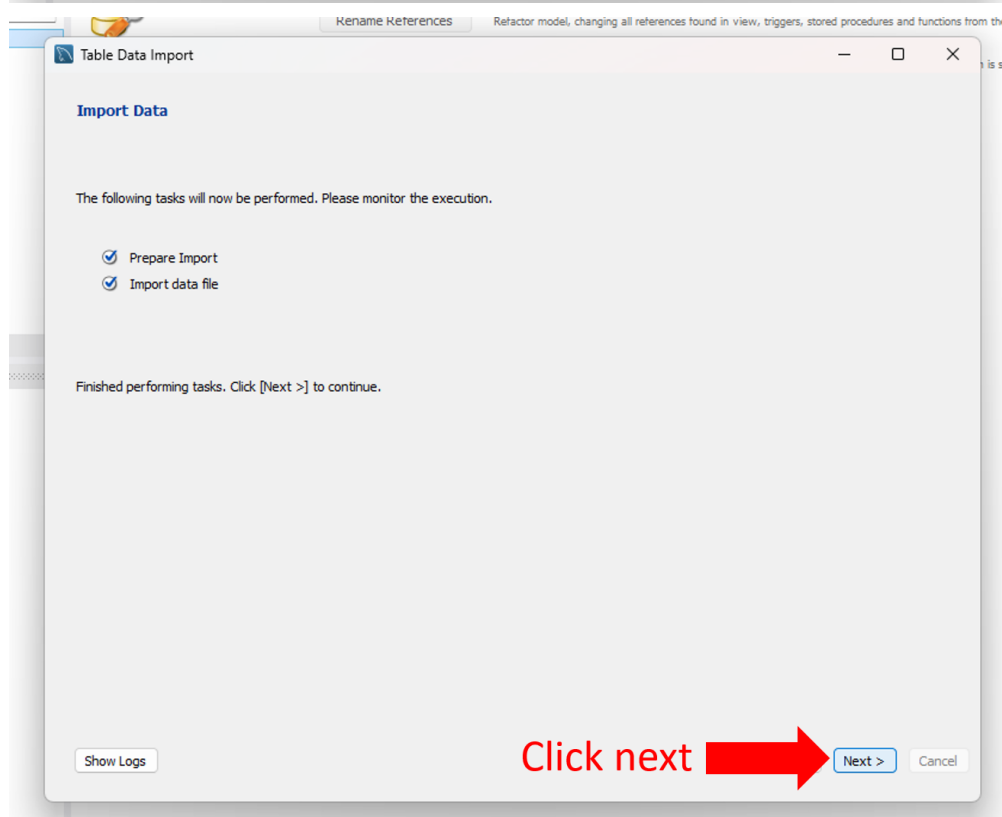
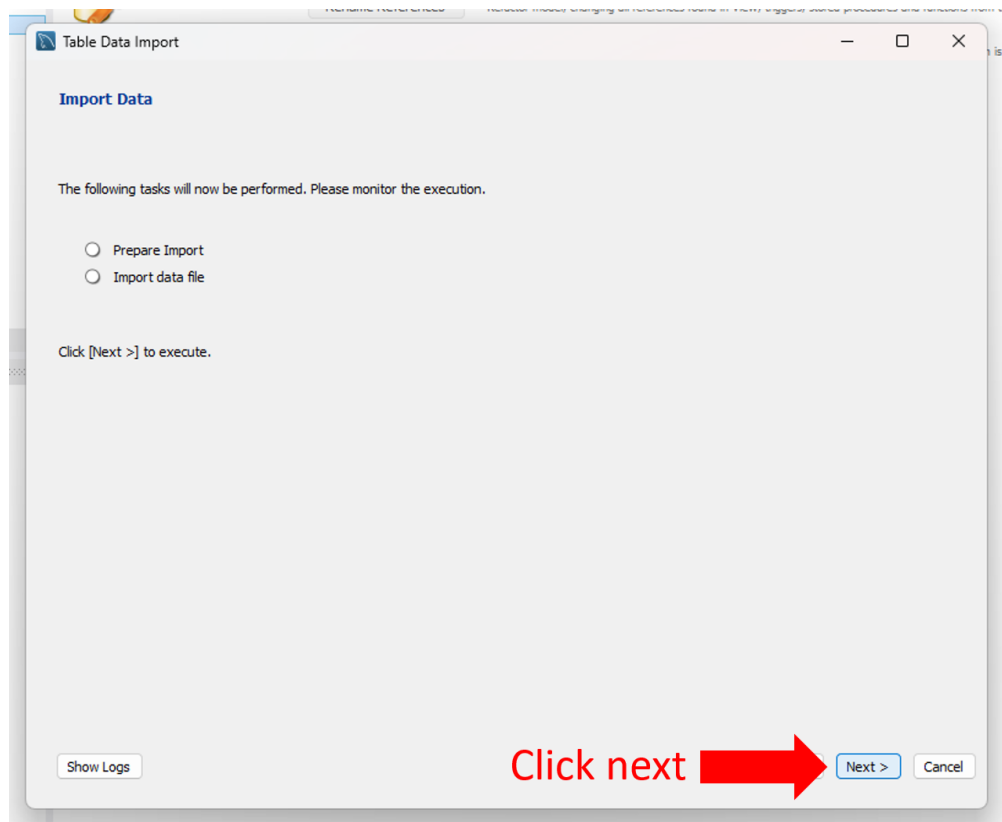
Click apply

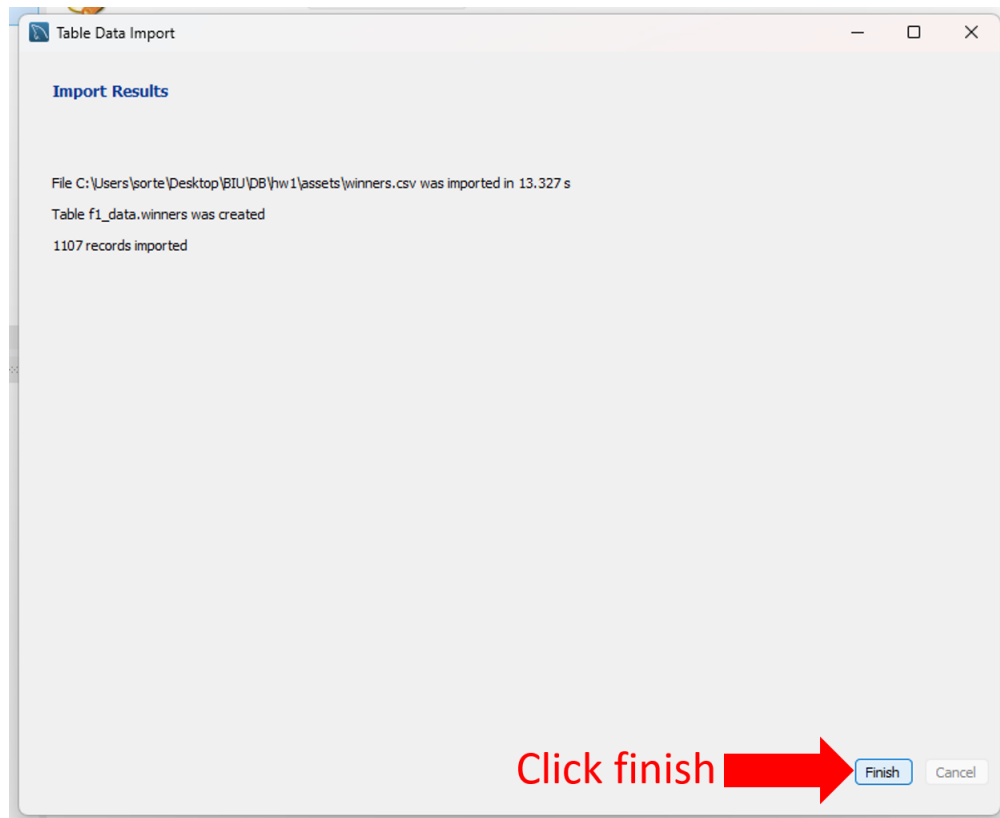
Apply Cancel

3. Load each of the CSV tables to the schema.



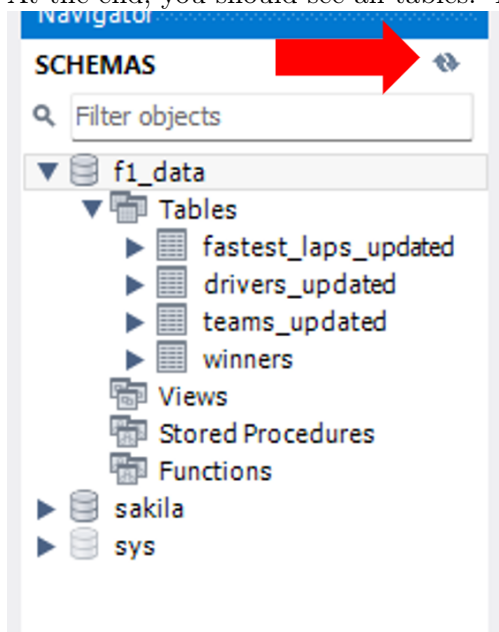






Do this again for each csv file in Lemida.

At the end, you should see all tables. You might have to click the refresh button to see them.



Credit: the data's original source is Kaggle.

Part B – SQL Queries

- Your final submission should contain a zip file containing all submission requirements and a details.txt file that should contain the ID and name of each submitting student, such as:

```
first_name last_name 123456789
first_name last_name 987654321
```

- The zip file should be named with your ID, such as "123456789.zip". If multiple students are submitting, then separate the ID numbers with underscore, such as "123456789_987654321.zip".
- In this assignment, your zip file needs to contain 8 .py files, and the details.txt file. Make sure you don't add any other unnecessary files to the zip.
- Submission is through the course website and can be either individually or in pairs – if you submit in pairs only one of the two should submit and that student will also be the one to receive feedback. A submission for three people is not permitted, if you are unable to find a pair then you should submit by yourself (it is not necessary to ask for permissions to submit by yourself).

Objective

Understanding and experiencing writing and executing SQL queries.

Data

All the questions in this part will assume the data in Part A.

Requirements

For each question you are required to submit a single file called **q{n}.py** (for example q1.py, q2.py etc.) that will include the following:

- The SQL query you used.
- Documentation - **mandatory**. Use '#' to add comments.
- Each of the python files need to be in this format:

```
import mysql.connector

if __name__ == '__main__':
    mydb = mysql.connector.connect (
        host="localhost",
        user="root",
        password="root",
        database="f1_data",
        port="3307",
    )

    cursor = mydb.cursor()
```

```

cursor.execute("""
    PUT_YOUR_QUERY
""")
print(', '.join(str(row) for row in cursor.fetchall()))
cursor.close()
mydb.close()

```

Important Notes

- The format is important because the testing is automatic. If the automatic script fails because you changed the format, you will not get any points for the question.
- Do not use commands that have not yet been learned, or will not be learned at all - a query that uses such commands, **all points will be deducted**.
- Please make sure that the queries are well formatted (use tabs and newlines, parentheses etc.) to make them readable (See the example format).
- You should assume that the data we are using is different from yours, do not make any assumptions on the data.
 - For example, if we ask who was the winner in 2023, do not write a query which contains the value “Max Verstaphen”, because you already know he won. We might use an alternated version of the data, where the winner is someone else.

Example Format (SQL)

```

SELECT film_id, title
FROM film
WHERE length > 10
ORDER BY title ASC;

```

Tips

- A car and a team are **the same**.
- Some columns have a space in their names. For example, ‘Grand Prix’. You need to use backticks ‘ when referencing these columns. Examples:

```

SELECT `Grand Prix`
FROM winners;

```

```

SELECT w.`Grand Prix`
FROM winners AS w;

```

- In some questions you should use the function YEAR, which returns the YEAR out of a date column.
- See the hints for the different questions

Questions

1. List all Formula 1 drivers from Brazil (BRA) without duplications.
This query should return a single column with the drivers names.
2. List all Formula 1 drivers from Italy (ITA) without duplications.
This query should return a single column with the drivers names.
3. Identify the driver who **won at least one race in the year 2000** and had the highest total number of laps completed in that year. Return the driver's name and his **best (minimum) fastest lap time of that year**, labeled as **min_time**.
This query should return two columns, driver name and min_time.
4. Find out how many **races were won in 2001** by the team (**Car column**) that had the most victories in the year 1999.
This query should return a single numerical value.
Hint: you can use LIMIT for simplicity.
5. Compute the **average points per car**, but only for cars where the **fastest lap time was recorded below 2 minutes (not included)**. Sort the output by **average points in descending order** and return the car name along with the average points as **avg_pts**.
This query should return two columns, car name and avg_pts.
Hint1: Use the teams_updated table and not the drivers_updated table
Hint2: Use the following SQL function to transform the fastest_laps_updated Time column to minute format:

```
MINUTE (STR_TO_DATE (fastest_laps_updated.Time, '%i:%s.%f'));
```
6. Retrieve all pairs of different **Grand Prix events** that had the **same number of laps**, as long as the number of laps was **greater than 80 (included)**. Ensure that each pair is returned **only once** and that **GP1 appears before GP2 alphabetically (for example, <Apple, Banana> and not <Banana, Apple>)**.
This query should return three columns: **GP1**, **GP2**, and **Laps**.
7. List all drivers who either **won a race while driving for 'Ferrari'** (exact naming) or have a **nationality listed as Argentina (ARG)**. Ensure that results are **sorted alphabetically** and do not contain duplicates.
This query should return a single column labeled "driver".
8. Calculate the **difference in total accumulated points** between the **Ferrari** and **Maserati** teams.
This query should return a single numerical value labeled diff.
9. Write a query that retrieves, for each driver **nationality**:
 - a. **The average number of points** earned by drivers of that nationality.
 - b. **The minimum recorded fastest lap time** for that nationality.
 - c. **The most recent date** a driver from that nationality won a race.

This query should return 4 columns: Nationality, avg_pts (average points), min_time (minimum fastest lap), and latest (most recent win date).

Hint: This query should use WITH (3 times)

Running Python Programs

To start the program, running the following line from the command line should work: work:

```
> python q1.py
```

External Python Packages

Your code should run only with the MySQL Connector/Python package. If you want to use any other library, you first need to get approval by writing a message in the Lemida course forum. Use only the Lemida course forum, so we can approve or reject it once for all the students.