**Mini python project:**

**Instructions for Project Submission:**

**Code Implementation:**

* Implement the code following the specifications provided in the statement of the project.
* Make sure you have properly separated the database connection code into a separate database.py file.

**Using git:**

* Initialize a local Git repository in your project directory.
* Make regular commits to save your changes locally.
* When you have completed a feature or part of the project, push to the branch (usually main or master).
* Push your code to your personal Git repository on the branch (usually main or master).

**Part1**

**Managing a list of objects:**

* Create a PeopleList class which contains a list of objects of type Person.
* Add an add\_person (name, age) method to add a new person to the list and save this information in a "People" table in the MySQL database.
* Add a show\_people () method to display the details of all people in the list by retrieving the data from the "People" table of the MySQL database.

**Search in a list of objects:**

* Add a search\_person(name) method to the PeopleList class which searches for a person by name in the "Persons" table in the database MySQL data and displays its details if found.

**Filtering people by age:**

* Add a method filter\_people\_by\_age(min\_age, max\_age) to the PeopleList class which retrieves the details of people whose age is between min\_age and max\_age from the "People" table in the MySQL database and displays them.

**Managing a queue:**

* Create a QueueWait class to manage a queue of people.
* Add an add\_person\_waiting(name) method to add a person to the queue and save their name in a "FileAttente" table in the MySQL database.
* Add a remove\_waiting\_person() method to remove the first person in the queue by retrieving their name from the table "FileAttente" from the MySQL database and display it.

**Prioritization in the queue:**

* Modify the QueueWait class so that it can manage people priorities.
* Add a method add\_priority\_person(name) to add a priority person to the queue and save their name in the "FileAttente" table in the MySQL database.
* Modify the delete\_waiting\_person() method to delete by priority a priority person if it exists in the "FileAttente" table of the MySQL database, otherwise delete the first normal person.

**Simulation of a reservation system:**

* Create a SalleCinema class to manage reservations in a cinema movie theater.
* Add a method reserver\_place (name, place) to reserve a place for a person and save this reservation in a "Reservations" table in the MySQL database.
* Add a method display\_reserved\_places() to display the places reserved by retrieving data from the “Reservations” table in the MySQL database.

**Part 2:**

**Room capacity management:**

* Add a number\_places\_available() method to the SalleCinema class to display the number of available seats by consulting the "Reservations" table in the MySQL database and calculate the available seats.
* Add a check in the reserver\_place(name, place) method to ensure that there are still places available in the room by consulting the “Reservations” table in the MySQL database before reserving.

**Filtering reservations by person:**

* Add a filter\_reservations\_by\_person(name) method to the class SalleCinema to display reservations made by a specific person by retrieving data from the “Reservations” table in the database MySQL and filtering by name.

**Cancellation of reservations:**

* Add a cancel\_reservation(name) method to the SalleCinema class to cancel all reservations made by a specific person by deleting the corresponding data in the "Reservations" table in the MySQL database.

**Management of special places:**

* Modify the SalleCinema class so that it can manage special seats for disabled people.
* Add a method reserver\_place\_speciale(name) to reserve a special place for a disabled person and record this reservation in the "Reservations" table of the MySQL database.