Helpdesk website backend part

We want to store the following data in Redis and manage with Python:

The employees of a company reports problems (or tasks) onto a helpdesk. The tasks can be performed by the employees (not the same one as who reported it).

Create a class, which has the following methods:

1. store an employee in the system (email)
2. delete an employee from the system (email)
3. list the registered employees
4. choose a manager (email): there is a manager in the helpdesk system who can change the priority of the tasks. This method sets who is the manager out of the employees (only an employee can be the manager).
5. gives back who is the manager
6. an employee records a new task (email, task name, description): the method generates and returns the task_id. The Redis should store the task_id, the recording time, and the employee who records it. Every new task gets 0 priority, which is also stored in Redis. The method should examine the employee email, only the registered employees can record a task.
7. task details (task_id): gives back the name and the description of a task.
8. Change priority (email, task_id, priority): only the manager can change the priority. The methods sets the priority of a task.
9. assign an employee to a task (task_id, assigner_employee_email, assigned_employee_email): the method assigns an employee to a task by another (or even the same) employee. The system stores both employees to a task with their roles in the assignment.
10. perform a task (email, task_id): it writes to the screen (or it can be an insert in SQL or a JSON with which the method can store the data in an other DBMS, but now it is enough to write it to the screen) the followings: performed task_id, task name, description, recording time, priority, email of the recorder employee, email of the employee who performed it, performing time, the email of the manager. It deletes the task from the Redis with all information that belong to it.
11. gives back a list with the task_ids ordered by their recording times (the first task should be the oldest one)
12. gives back a list with the task_ids ordered by their priorities (the first task should be the task with the highest priority)
13. gives back a list with task_ids assigned to an employee (email). The list should be ordered by priority, the first should have the highest priority.
14. gives back the newest task_id
15. gives back the priority of a task (task_id)