CMMS User Guide

1. **Introduction**

1.1 Purpose of this manual

This manual aims to introduce the Campus Maintenance and Management System (CMMS), including its goal, function, commands and view, which helps administrators and operators organize and track different types of activities. The manual will mainly focus on using methods, and the internal implementation details are excluded.

1.2 What is Campus Maintenance and Management System?

Campus Maintenance and Management System, also called CMMS in short, is a campus maintenance and management system that provides both a command-line interface and a graphical user interface.

Users can create and manage maintenance, cleaning, and repair activities through terminal commands, while the optional GUI mode offers a visual and organized way to browse records and perform operations. For more expert users, the SQL query function is also provided.

The system manages several core elements on campus, including employees, temporary workers, managers, buildings, rooms, and scheduled activities. Users can assign people to tasks, link activities to specific locations, track supervision relationships, manage office occupancy, and generate built-in reports for administrative analysis.

2. **Getting start**

2.1 System Requirements

Before running the CMMS, please ensure the following requirements are met.

2.1.1 Operation requirements

Windows 10/11, Mac OS 12+(recommended)

Linux (Alternative)

2.1.2 Interpreter and dependency

- Python 3.8+ (3.12 recommended)
- My SQL server 8.0+ installed locally
- Required Python package: mysql-connector-python (If you haven't got this, run "pip

install mysql-connector-python" in your terminal.)

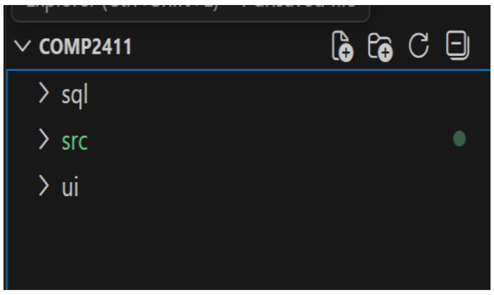
- (Optional) GUI environment for running the Tkinter interface.

2.2 Start CMMS

CMMS can launch either in command line mode or GUI mode. Two modes have the same functions.

2.2.1 Installation

You can download or clone the folder comp2411/ to your own computer, and the downloaded folder should cover the structure below:

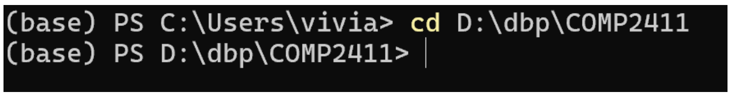


Make sure the project is complete before using it.

2.2.2 Running CMMS program

In this part, we will run in PowerShell as an example. If you are a more advanced user with IDE on your computer, we also recommend you run in IDE integrated terminal by clicking the running button on main.py and cli.py.

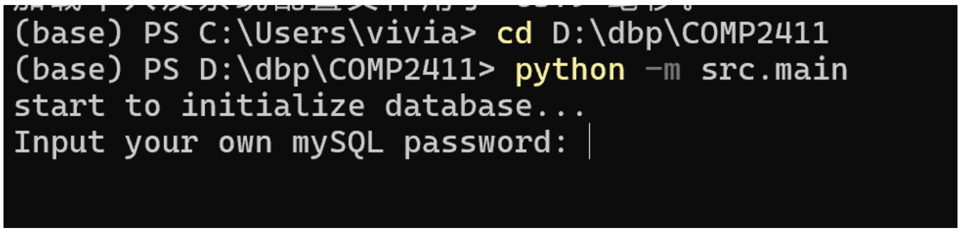
Step1. Open the root directory where the file is located. Run "cd .\COMP2411" in your terminal.



```
(base) PS C:\Users\vivia> cd D:\dbp\COMP2411
(base) PS D:\dbp\COMP2411>
```

Step2. Run in terminal

Input "python -m src.main" and click Enter, the following message will show on your terminal if you run successfully:

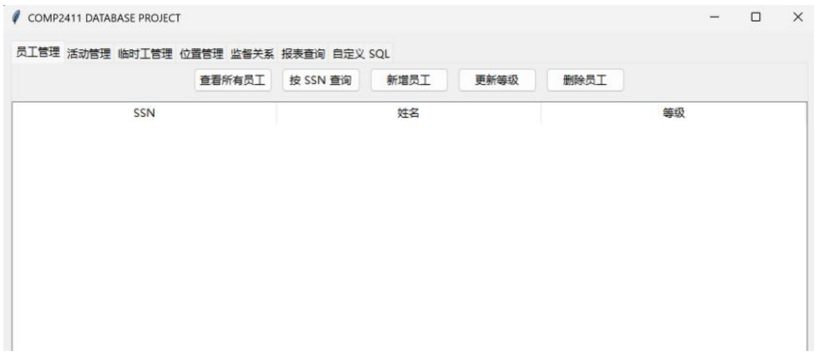


```
(base) PS C:\Users\vivia> cd D:\dbp\COMP2411
(base) PS D:\dbp\COMP2411> python -m src.main
start to initialize database...
Input your own mySQL password:
```

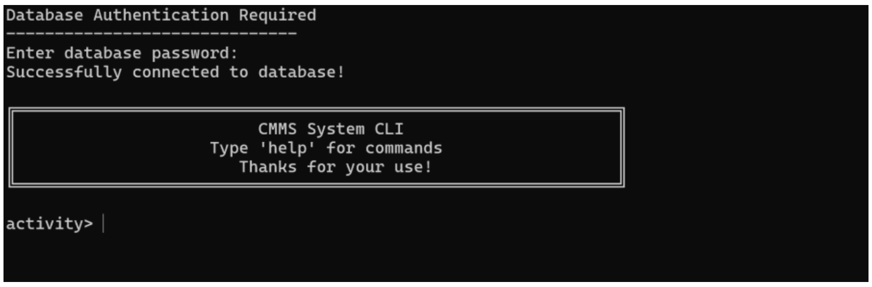
Alternatively, if you need to run with command line mode, please run "python -m ui.cli". The success message is the same as above.

If this step fails, please check if you have installed mysql-connector-python in your environment.

Then, enter your own MySQL password, and you will see the GUI of the program.



And this is the command line mode:



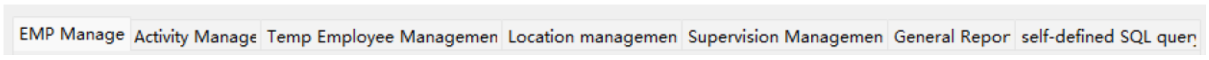
```
Database Authentication Required
------------------------------
Enter database password:
Successfully connected to database!

              CMMS System CLI
          Type 'help' for commands
             Thanks for your use!

activity>
```

3. Command Overview

Because the operation of GUI mode and command line mode has distinct differences, this part will introduce their using methods separately.

3.1 GUI Mode

In GUI mode, you will see the functions are divided into 7 pages, including employee management, activity management, temporary employee management, location management, supervising relationships, report query and custom SQL query. In this part, we will introduce the above functions one by one.

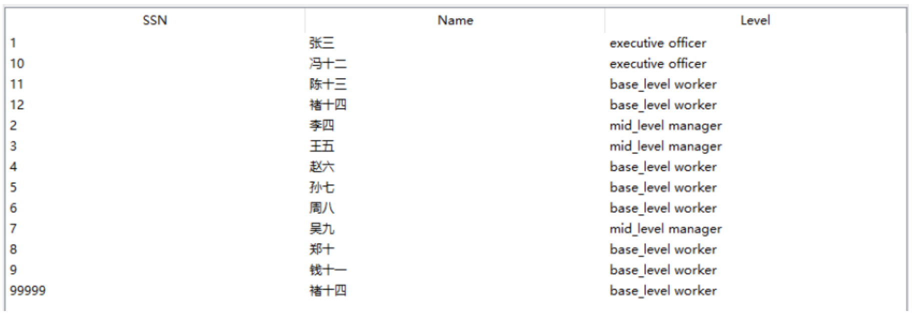


3.1.1 Employee operations

You will enter the employee management operation page as soon as you enter the GUI mode. In this page, there are 5 buttons to help you complete the following operations:

- View all employees

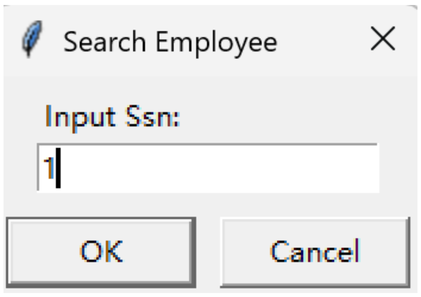
  Click "Search All", and you will get:



SSN	Name	Level
1	张三	executive officer
10	冯十二	executive officer
11	陈十三	base_level worker
12	褚十四	base_level worker
2	李四	mid_level manager
3	王五	mid_level manager
4	赵六	base_level worker
5	孙七	base_level worker
6	周八	base_level worker
7	吴九	mid_level manager
8	郑十	base_level worker
9	钱十一	base_level worker
99999	褚十四	base_level worker

- Find employee through SSN

  Click "Search by SSN":

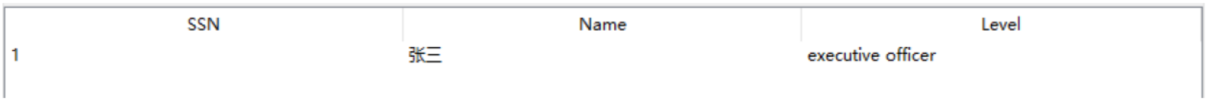


Search Employee

Input Ssn:

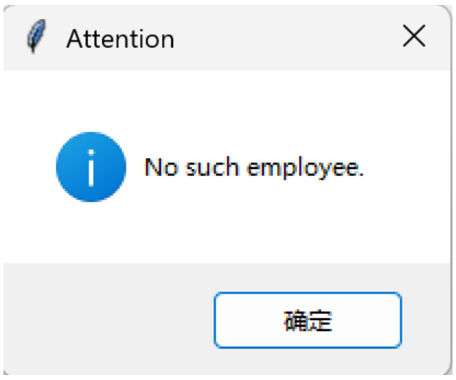
1

Enter a valid SSN and you can successfully find the information.



SSN	Name	Level
1	张三	executive officer

If the SSN is invalid, you will receive an Infor message:



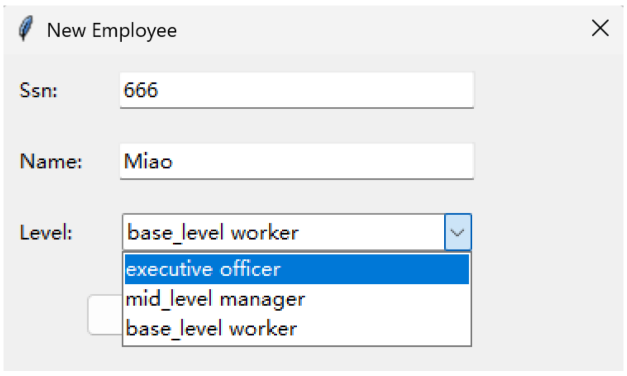
Attention

No such employee.

- Add new employees

  Click "Insert EMP".

  Then you need to enter the information of the new employee manually and select the level for it.



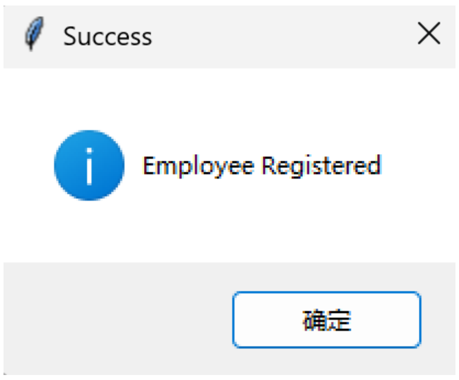
New Employee

Ssn: 666

Name: Miao

Level: base_level worker

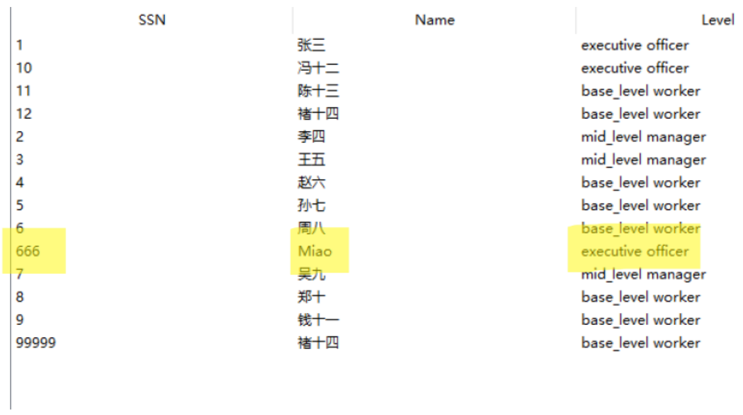
You would get a message if the addition succussed:



Success

Employee Registered

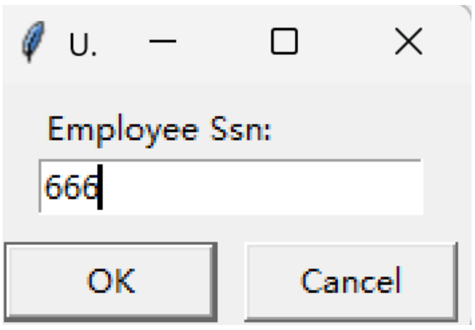
Then you can see it in your employee list:



SSN	Name	Level
1	张三	executive officer
10	冯十二	executive officer
11	陈十三	base_level worker
12	褚十四	base_level worker
2	李四	mid_level manager
3	王五	mid_level manager
4	赵六	base_level worker
5	孙七	base_level worker
6	周八	base_level worker
666	Miao	executive officer
7	吴九	mid_level manager
8	郑十	base_level worker
9	钱十一	base_level worker
99999	褚十四	base_level worker

- Update employees' level

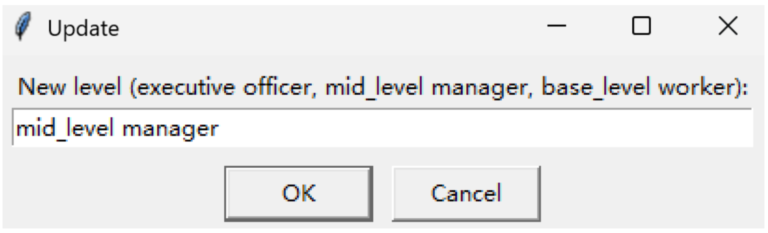
Click "Update level".



U.

Employee Ssn: 666

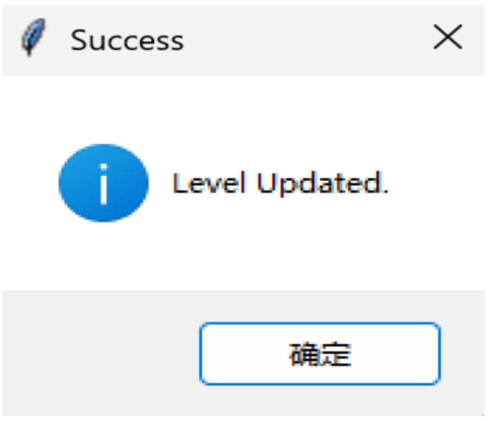
Click "OK" and you need to type the new level manually.


Update

New level (executive officer, mid_level manager, base_level worker):

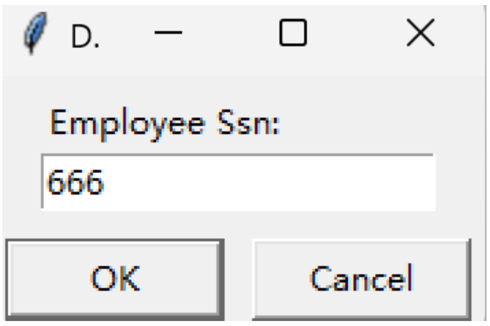
mid_level manager

You/will get an Infor message if the update succussed.


Success

Level Updated.

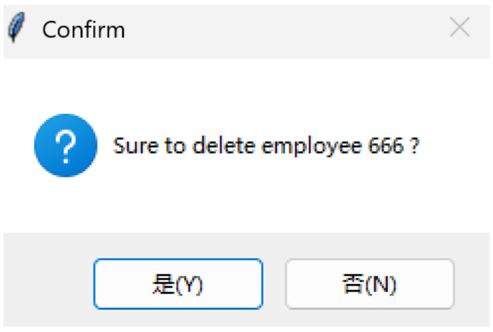
- Delete employee
- Click “Delete EMP”.


D.

Employee Ssn:

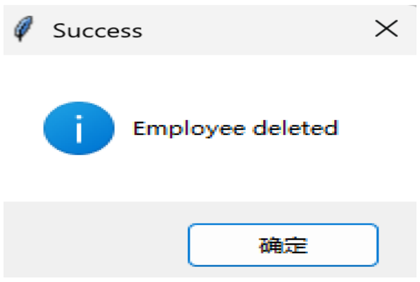
666

Click OK, and you will be required to check twice to prevent deleting by mistake.


Confirm

Sure to delete employee 666 ?

Click Y and you will delete successfully:

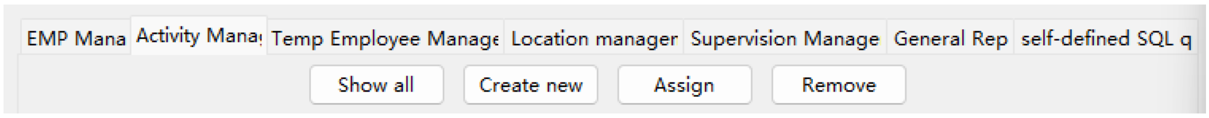

Success

Employee deleted

And you can see the deleted employee will no longer appear in the employee list.

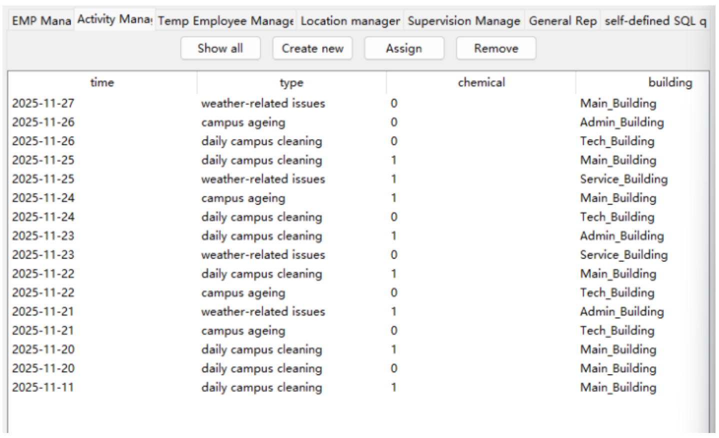
3.1.2 Activity operations

Click “Activity Manage” button and you will see all the operations you can do to Activity.



- Show all activities

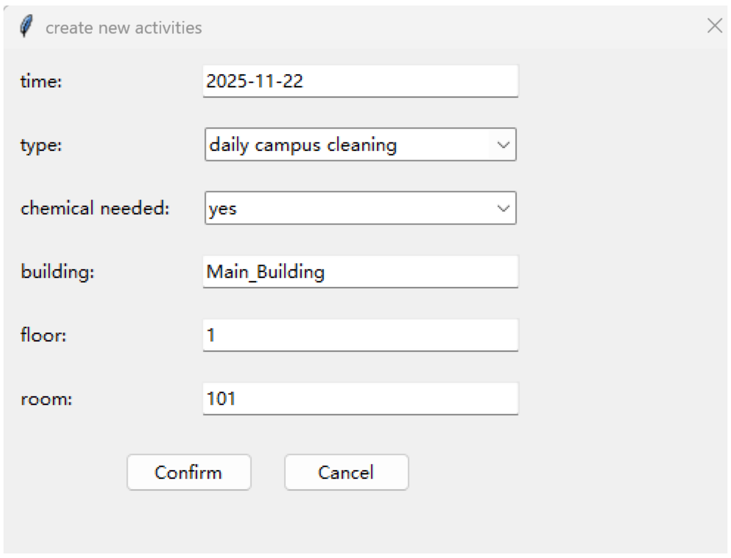
  Click “Show all” to see all activities



time	type	chemical	building
2025-11-27	weather-related issues	0	Main_Building
2025-11-26	campus ageing	0	Admin_Building
2025-11-26	daily campus cleaning	0	Tech_Building
2025-11-25	daily campus cleaning	1	Main_Building
2025-11-25	weather-related issues	1	Service_Building
2025-11-24	campus ageing	1	Main_Building
2025-11-24	daily campus cleaning	0	Tech_Building
2025-11-23	daily campus cleaning	1	Admin_Building
2025-11-23	weather-related issues	0	Service_Building
2025-11-22	daily campus cleaning	1	Main_Building
2025-11-22	campus ageing	0	Tech_Building
2025-11-21	weather-related issues	1	Admin_Building
2025-11-21	campus ageing	0	Tech_Building
2025-11-20	daily campus cleaning	1	Main_Building
2025-11-20	daily campus cleaning	0	Main_Building
2025-11-11	daily campus cleaning	1	Main_Building

- Create New Activity

  Click “Create New” to enter create interface. In this interface, you need to input the details about the new activity manually. The time should in “YYYY-MM-DD" format, and the building must exist.



time: 2025-11-22

type: daily campus cleaning

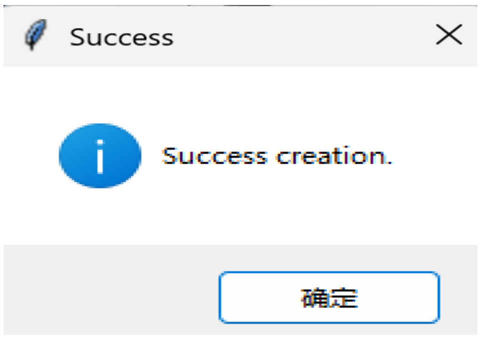
chemical needed: yes

building: Main_Building

floor: 1

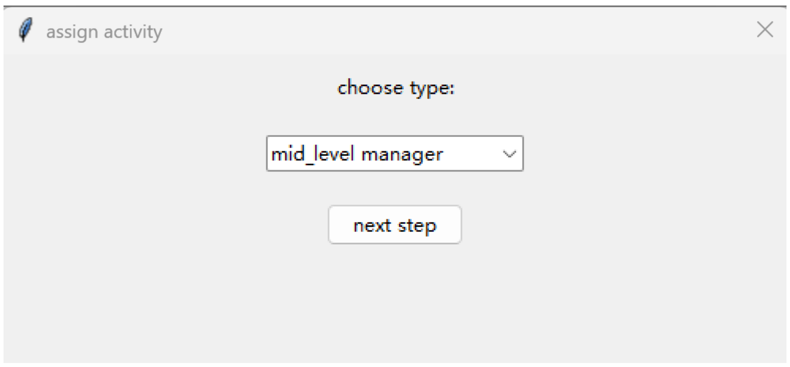
room: 101

You will get the Infor message if you successfully create it.



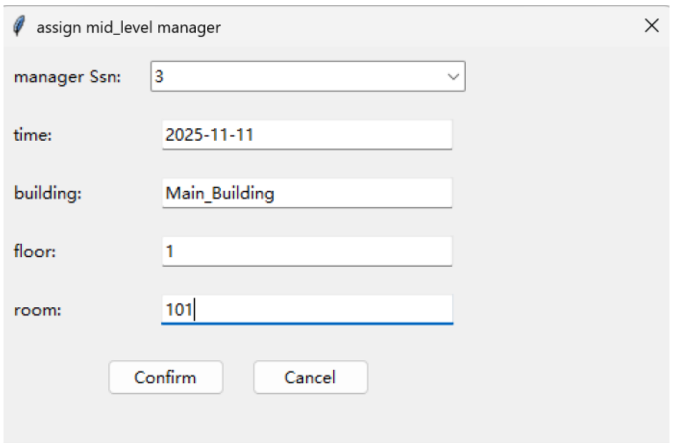
Success creation.

- Assign the acticity to an employee



choose type: mid_level manager

Then you need to input the target activity manually:


assign mid_level manager

manager Ssn: 3

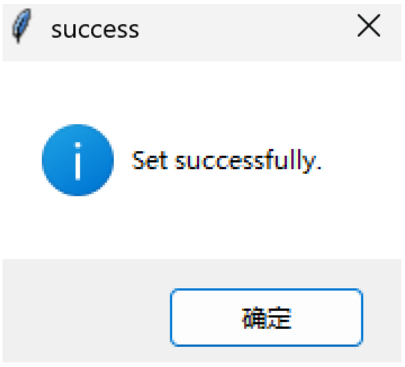
time: 2025-11-11

building: Main_Building

floor: 1

room: 101

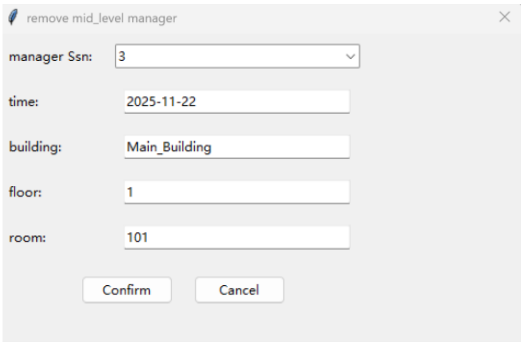
Click confirm and you will get the Infor message if your assignment completed.


success

Set successfully.

- Remove a manage relationship

  Click “Remove” and start to remove a manage relation. You need to input the correct manage information manually.


remove mid_level manager

manager Ssn: 3

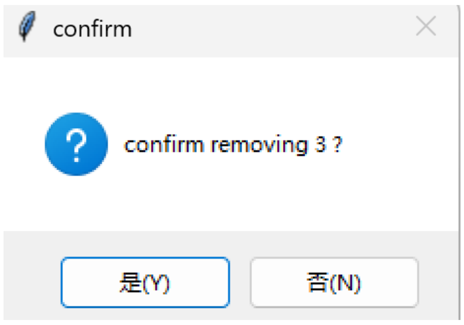
time: 2025-11-22

building: Main_Building

floor: 1

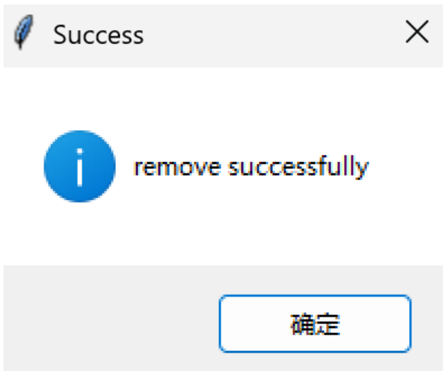
room: 101

After entering correct message, you can confirm and remove.


confirm

confirm removing 3 ?

After removing, you will get a success message.

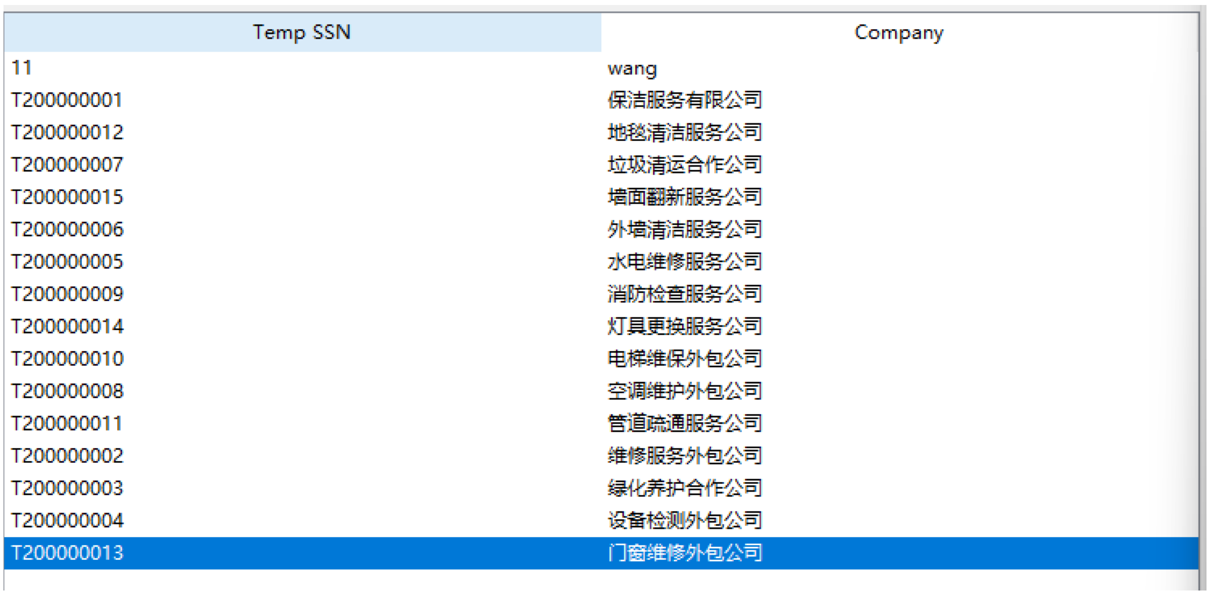

Success

remove successfully

3.1.3 Temporary employee operations

- Show All Temp Employees

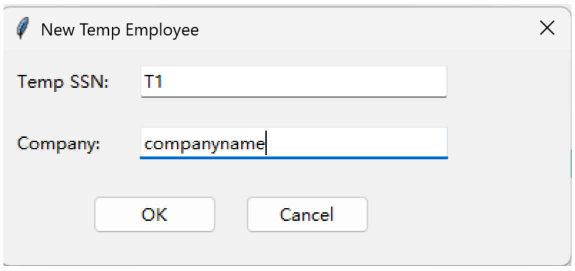
  Click "Show All Temp Employees" and you can see the all the temp employees' SSN and Company.



Temp SSN	Company
11	wang
T200000001	保洁服务有限公司
T200000012	地毯清洁服务公司
T200000007	垃圾清运合作公司
T200000015	墙面翻新服务公司
T200000006	外墙清洁服务公司
T200000005	水电维修服务公司
T200000009	消防检查服务公司
T200000014	灯具更换服务公司
T200000010	电梯维保外包公司
T200000008	空调维护外包公司
T200000011	管道疏通服务公司
T200000002	维修服务外包公司
T200000003	绿化养护合作公司
T200000004	设备检测外包公司
T200000013	门窗维修外包公司

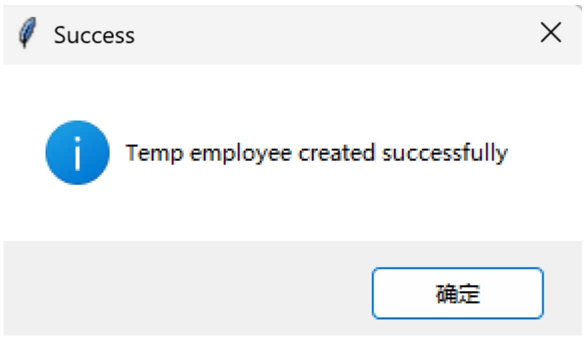
- Create New Temp Employee

  You can create new temp employees following these steps:

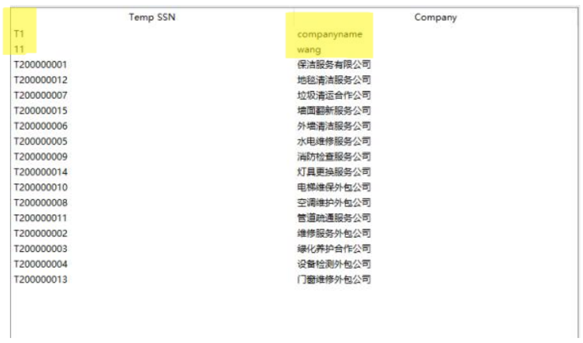


Temp SSN: T1

Company: companyname

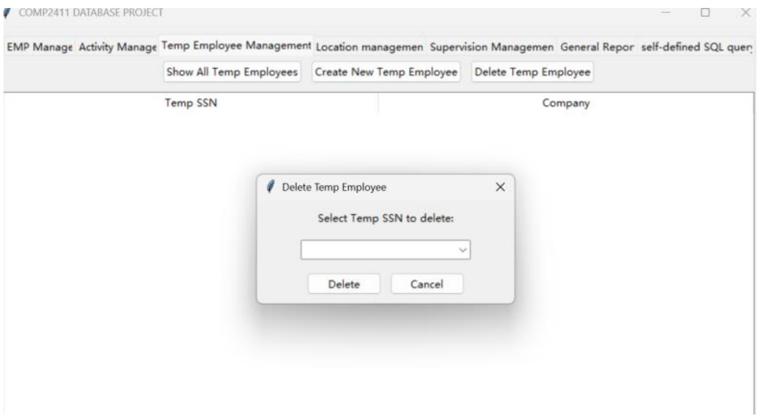


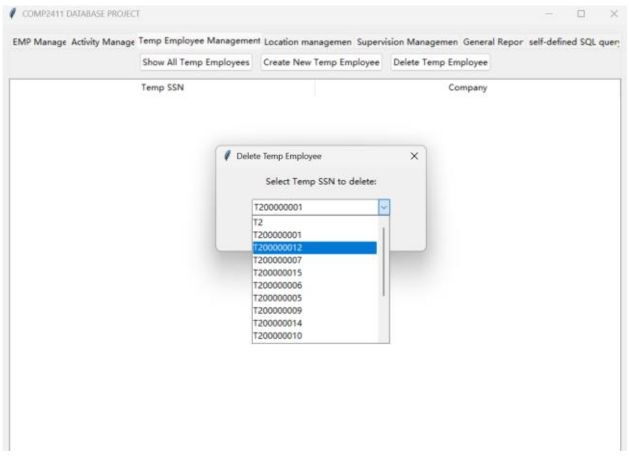
Temp employee created successfully



Temp SSN	Company
T1	companyname
11	wang
T200000001	保洁服务有限公司
T200000012	地毯清洁服务公司
T200000007	垃圾清运合作公司
T200000015	墙面翻新服务公司
T200000006	外墙清洁服务公司
T200000005	水电维修服务公司
T200000009	消防检查服务公司
T200000014	灯具更换服务公司
T200000010	电梯维保外包公司
T200000008	空调维护外包公司
T200000011	管道疏通服务公司
T200000002	维修服务外包公司
T200000003	绿化养护合作公司
T200000004	设备检测外包公司
T200000013	门窗维修外包公司

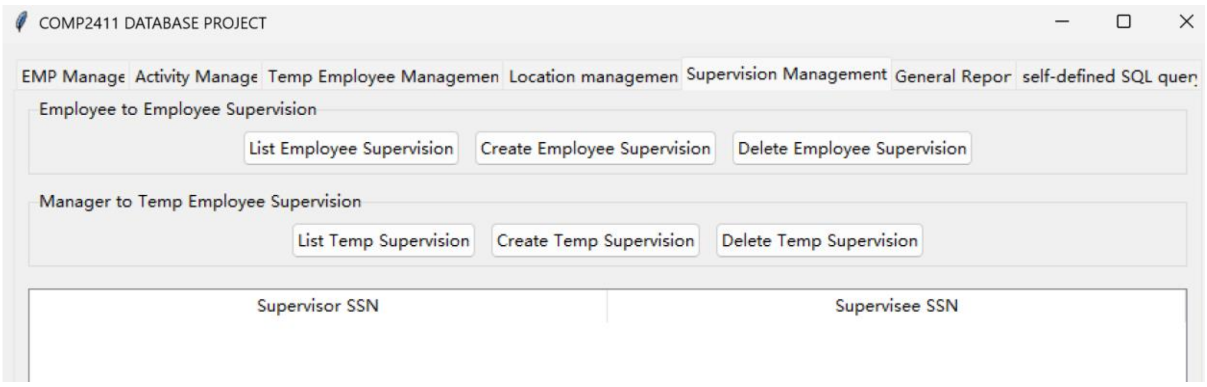
- Delete Temp Employee





3.1.4 Supervision Management

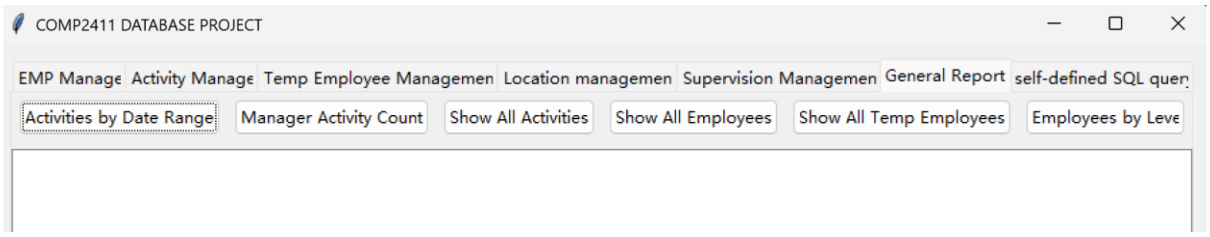
We have View All Locations, Add New Location, View Vacant Officers, View All Officers, Filter by Building, Assign Office, Vacate Office functions in this page. You can follow the instructions on your interface and use them smoothly. The detailed things are similar to the functions above, which you can easily refer to.



3.1.5 General Report

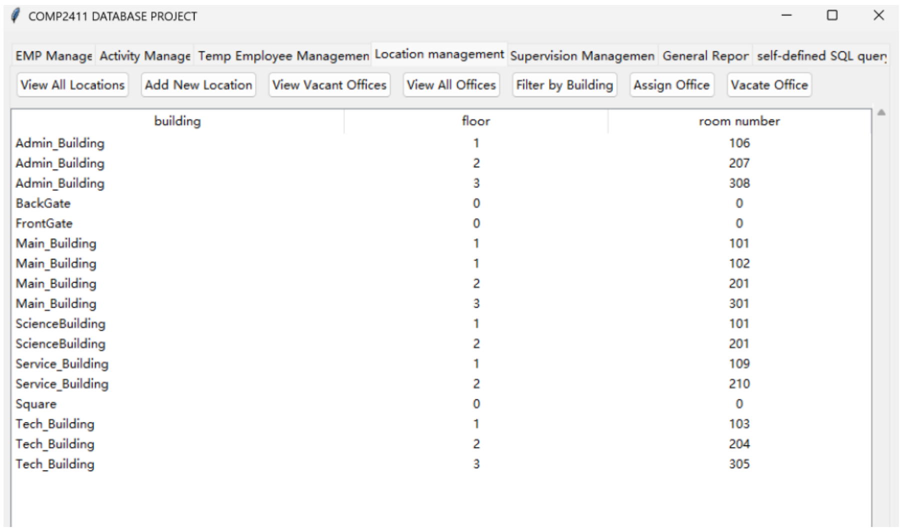
The General Report tab provides quick access to commonly used statistical summaries.
You can generate activity reports, employee lists, manager workload counts, and other aggregated data by selecting the corresponding button at the top.

All results are displayed in the table below automatically.
Some reports require user input (e.g., date range or employee level), while others load instantly with a single click.



3.1.6 Location management

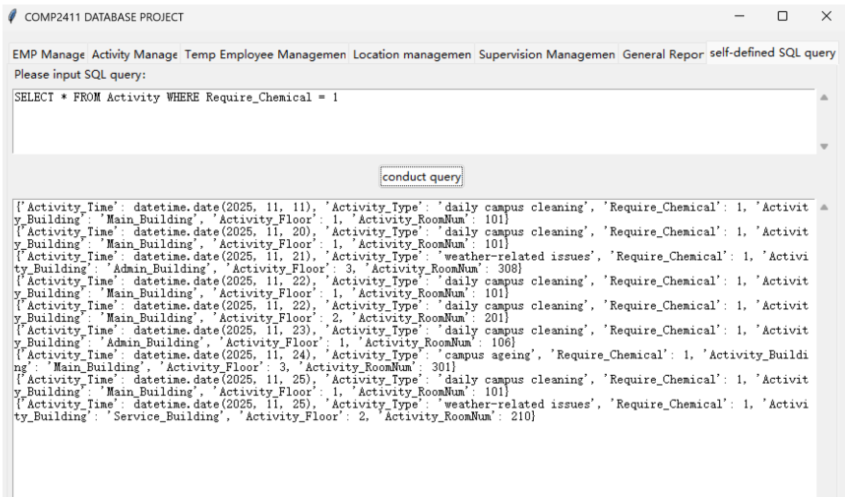
- This tab allows users to manage all office and location records on campus.
  You can view all locations, filter rooms by building, add new locations, and inspect which offices are currently vacant.
- Office assignment and removal operations can also be performed here.
  By selecting Assign Office or Vacate Office, users can update occupancy information quickly through guided input dialogs.



building	floor	room number
Admin_Building	1	106
Admin_Building	2	207
Admin_Building	3	308
BackGate	0	0
FrontGate	0	0
Main_Building	1	101
Main_Building	1	102
Main_Building	2	201
Main_Building	3	301
ScienceBuilding	1	101
ScienceBuilding	2	201
Service_Building	1	109
Service_Building	2	210
Square	0	0
Tech_Building	1	103
Tech_Building	2	204
Tech_Building	3	305

3.1.7 Custom SQL query

You can input the SQL query directly in the box, then click ”conduct query” to run.



```
SELECT * FROM Activity WHERE Require_Chemical = 1
```

```
{'Activity_Time': datetime.date(2025, 11, 11), 'Activity_Type': 'daily campus cleaning', 'Require_Chemical': 1, 'Activit
y_Building': 'Main_Building', 'Activity_Floor': 1, 'Activity_RoomNum': 101}
{'Activity_Time': datetime.date(2025, 11, 20), 'Activity_Type': 'daily campus cleaning', 'Require_Chemical': 1, 'Activit
y_Building': 'Main_Building', 'Activity_Floor': 1, 'Activity_RoomNum': 101}
{'Activity_Time': datetime.date(2025, 11, 21), 'Activity_Type': 'weather-related issues', 'Require_Chemical': 1, 'Activi
ty_Building': 'Admin_Building', 'Activity_Floor': 3, 'Activity_RoomNum': 308}
{'Activity_Time': datetime.date(2025, 11, 22), 'Activity_Type': 'daily campus cleaning', 'Require_Chemical': 1, 'Activit
y_Building': 'Main_Building', 'Activity_Floor': 1, 'Activity_RoomNum': 101}
{'Activity_Time': datetime.date(2025, 11, 22), 'Activity_Type': 'daily campus cleaning', 'Require_Chemical': 1, 'Activit
y_Building': 'Main_Building', 'Activity_Floor': 2, 'Activity_RoomNum': 201}
{'Activity_Time': datetime.date(2025, 11, 23), 'Activity_Type': 'daily campus cleaning', 'Require_Chemical': 1, 'Activit
y_Building': 'Admin_Building', 'Activity_Floor': 1, 'Activity_RoomNum': 106}
{'Activity_Time': datetime.date(2025, 11, 24), 'Activity_Type': 'campus ageing', 'Require_Chemical': 1, 'Activity_Buildi
ng': 'Main_Building', 'Activity_Floor': 3, 'Activity_RoomNum': 301}
{'Activity_Time': datetime.date(2025, 11, 25), 'Activity_Type': 'daily campus cleaning', 'Require_Chemical': 1, 'Activit
y_Building': 'Main_Building', 'Activity_Floor': 1, 'Activity_RoomNum': 101}
{'Activity_Time': datetime.date(2025, 11, 25), 'Activity_Type': 'weather-related issues', 'Require_Chemical': 1, 'Activi
ty_Building': 'Service_Building', 'Activity_Floor': 2, 'Activity_RoomNum': 210}
```

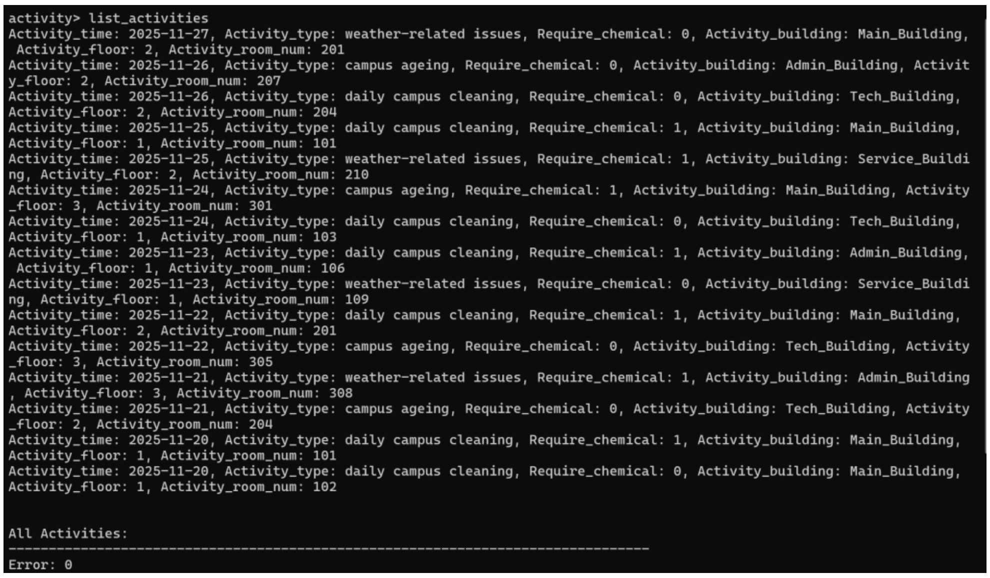
3.2 Command Line Mode

3.2.1 Activity commands

Command	Description
list_activities	List all activities in the system.
create_activity "time" "type" "chemical" "building" "floor" "room"	Create a new activity.
get_activity "time" "building" "floor" "room"	View details of a specific activity.

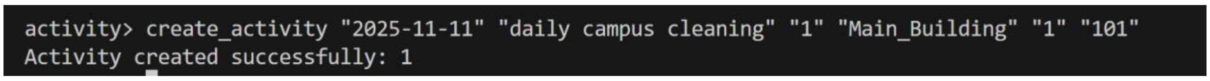
Use examples:

- list_activities:



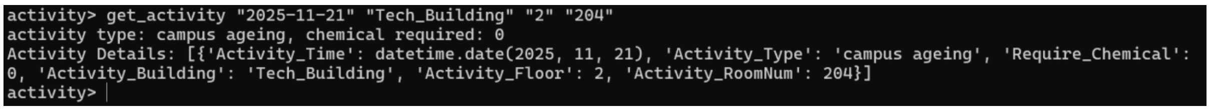
```
activity> list_activities
Activity_time: 2025-11-27, Activity_type: weather-related issues, Require_chemical: 0, Activity_building: Main_Building,
 Activity_floor: 2, Activity_room_num: 201
Activity_time: 2025-11-26, Activity_type: campus ageing, Require_chemical: 0, Activity_building: Admin_Building, Activit
y_floor: 2, Activity_room_num: 207
Activity_time: 2025-11-26, Activity_type: daily campus cleaning, Require_chemical: 0, Activity_building: Tech_Building,
Activity_floor: 2, Activity_room_num: 204
Activity_time: 2025-11-25, Activity_type: daily campus cleaning, Require_chemical: 1, Activity_building: Main_Building,
Activity_floor: 1, Activity_room_num: 101
Activity_time: 2025-11-25, Activity_type: weather-related issues, Require_chemical: 1, Activity_building: Service_Buildi
ng, Activity_floor: 2, Activity_room_num: 210
Activity_time: 2025-11-24, Activity_type: campus ageing, Require_chemical: 1, Activity_building: Main_Building, Activity
_floor: 3, Activity_room_num: 301
Activity_time: 2025-11-24, Activity_type: daily campus cleaning, Require_chemical: 0, Activity_building: Tech_Building,
Activity_floor: 1, Activity_room_num: 103
Activity_time: 2025-11-23, Activity_type: daily campus cleaning, Require_chemical: 1, Activity_building: Admin_Building,
 Activity_floor: 1, Activity_room_num: 106
Activity_time: 2025-11-23, Activity_type: weather-related issues, Require_chemical: 0, Activity_building: Service_Buildi
ng, Activity_floor: 1, Activity_room_num: 109
Activity_time: 2025-11-22, Activity_type: daily campus cleaning, Require_chemical: 1, Activity_building: Main_Building,
Activity_floor: 2, Activity_room_num: 201
Activity_time: 2025-11-22, Activity_type: campus ageing, Require_chemical: 0, Activity_building: Tech_Building, Activity
_floor: 3, Activity_room_num: 305
Activity_time: 2025-11-21, Activity_type: weather-related issues, Require_chemical: 1, Activity_building: Admin_Building
, Activity_floor: 3, Activity_room_num: 308
Activity_time: 2025-11-21, Activity_type: campus ageing, Require_chemical: 0, Activity_building: Tech_Building, Activity
_floor: 2, Activity_room_num: 204
Activity_time: 2025-11-20, Activity_type: daily campus cleaning, Require_chemical: 1, Activity_building: Main_Building,
Activity_floor: 1, Activity_room_num: 101
Activity_time: 2025-11-20, Activity_type: daily campus cleaning, Require_chemical: 0, Activity_building: Main_Building,
Activity_floor: 1, Activity_room_num: 102


All Activities:
--------------------------------------------------------------------------------
Error: 0
```

- create_activity "time" "type" "chemical" "building" "floor" "room"



```
activity> create_activity "2025-11-11" "daily campus cleaning" "1" "Main_Building" "1" "101"
Activity created successfully: 1
```

- get_activity "2025-11-21" "Tech_Building" "2" "204"



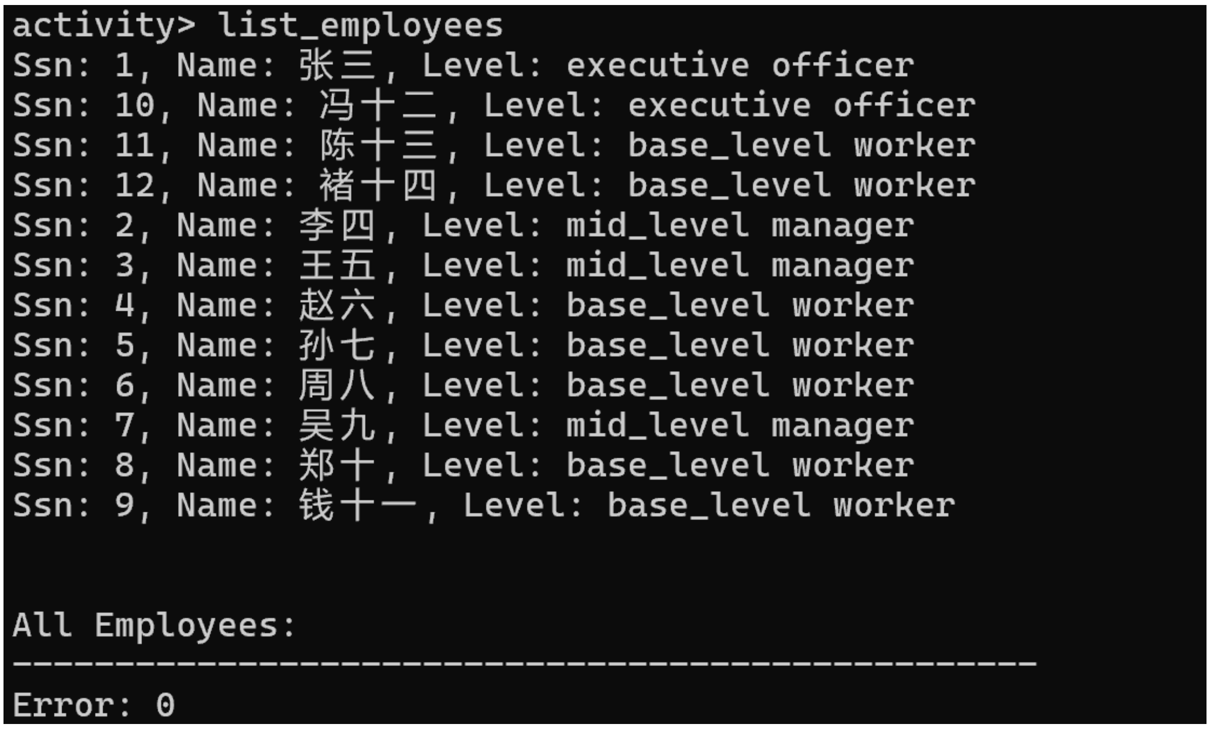
```
activity> get_activity "2025-11-21" "Tech_Building" "2" "204"
activity type: campus ageing, chemical required: 0
Activity Details: [{'Activity_Time': datetime.date(2025, 11, 21), 'Activity_Type': 'campus ageing', 'Require_Chemical':
0, 'Activity_Building': 'Tech_Building', 'Activity_Floor': 2, 'Activity_RoomNum': 204}]
activity>
```

3.2.2 Employee commands

Command	Description
list_employees	Display all employees.
add_employee "ssn" "name" "level"	Add a new employee. Level must be one of: executive officer, mid_level manager, base_level worker.

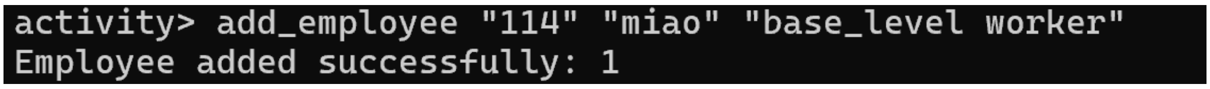
Use examples:

- list_employees



```
activity> list_employees
Ssn: 1, Name: 张三, Level: executive officer
Ssn: 10, Name: 冯十二, Level: executive officer
Ssn: 11, Name: 陈十三, Level: base_level worker
Ssn: 12, Name: 褚十四, Level: base_level worker
Ssn: 2, Name: 李四, Level: mid_level manager
Ssn: 3, Name: 王五, Level: mid_level manager
Ssn: 4, Name: 赵六, Level: base_level worker
Ssn: 5, Name: 孙七, Level: base_level worker
Ssn: 6, Name: 周八, Level: base_level worker
Ssn: 7, Name: 吴九, Level: mid_level manager
Ssn: 8, Name: 郑十, Level: base_level worker
Ssn: 9, Name: 钱十一, Level: base_level worker


All Employees:
--------------------------------------------------
Error: 0
```

- add_employee "114" "miao" "base_level worker"



```
activity> add_employee "114" "miao" "base_level worker"
Employee added successfully: 1
```

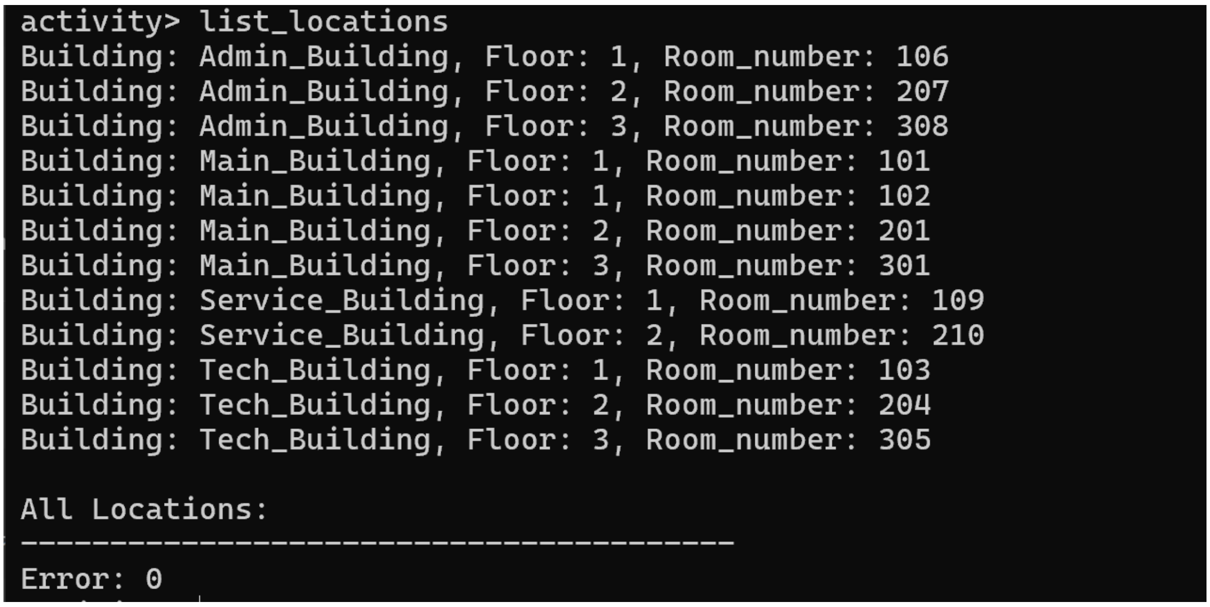
3.2.3 Location commands

Command	Description
list_locations	Show all registered locations.

create_location "building" "floor" "room"	Add a new campus location.

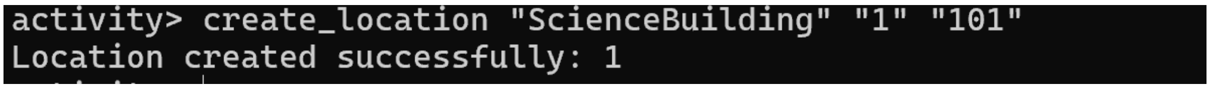
Use examples:

- list_locations



```
activity> list_locations
Building: Admin_Building, Floor: 1, Room_number: 106
Building: Admin_Building, Floor: 2, Room_number: 207
Building: Admin_Building, Floor: 3, Room_number: 308
Building: Main_Building, Floor: 1, Room_number: 101
Building: Main_Building, Floor: 1, Room_number: 102
Building: Main_Building, Floor: 2, Room_number: 201
Building: Main_Building, Floor: 3, Room_number: 301
Building: Service_Building, Floor: 1, Room_number: 109
Building: Service_Building, Floor: 2, Room_number: 210
Building: Tech_Building, Floor: 1, Room_number: 103
Building: Tech_Building, Floor: 2, Room_number: 204
Building: Tech_Building, Floor: 3, Room_number: 305

All Locations:
----------------------------------------
Error: 0
```

- create_location "ScienceBuilding" "1" "101"



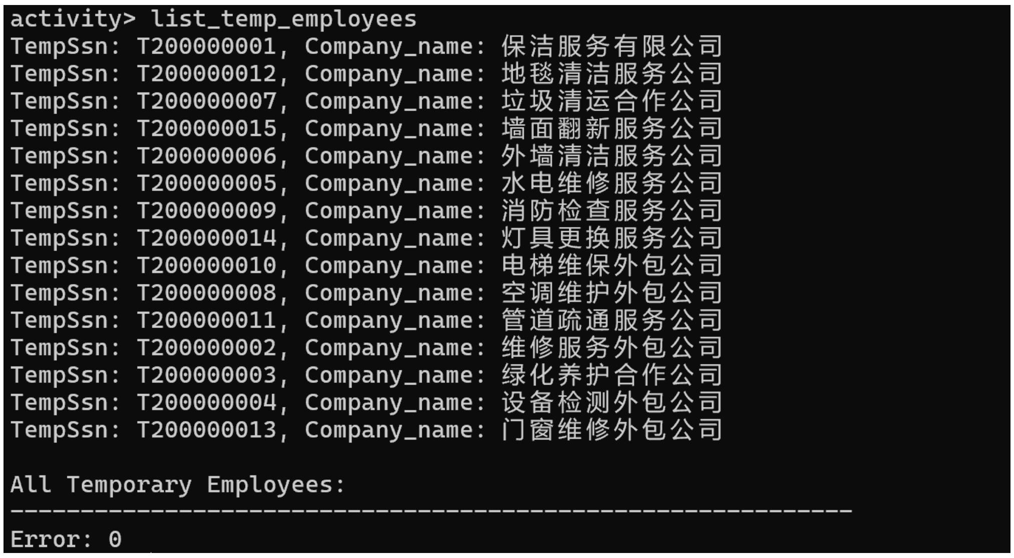
```
activity> create_location "ScienceBuilding" "1" "101"
Location created successfully: 1
```

3.2.4 Temporary employee commands

Command	Description
list_temp_employees	List all temporary workers.
create_temp_employee "ssn" "company"	Add a temporary employee and assign its contractor company.

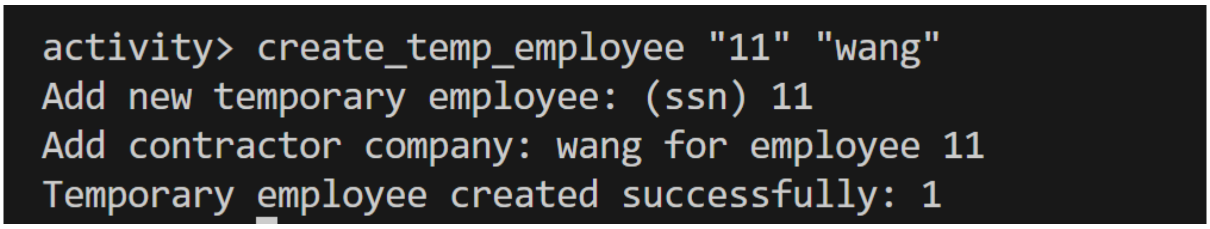
Use examples:

- List_temp_employees



```
activity> list_temp_employees
TempSsn: T200000001, Company_name: 保洁服务有限公司
TempSsn: T200000012, Company_name: 地毯清洁服务公司
TempSsn: T200000007, Company_name: 垃圾清运合作公司
TempSsn: T200000015, Company_name: 墙面翻新服务公司
TempSsn: T200000006, Company_name: 外墙清洁服务公司
TempSsn: T200000005, Company_name: 水电维修服务公司
TempSsn: T200000009, Company_name: 消防检查服务公司
TempSsn: T200000014, Company_name: 灯具更换服务公司
TempSsn: T200000010, Company_name: 电梯维保外包公司
TempSsn: T200000008, Company_name: 空调维护外包公司
TempSsn: T200000011, Company_name: 管道疏通服务公司
TempSsn: T200000002, Company_name: 维修服务外包公司
TempSsn: T200000003, Company_name: 绿化养护合作公司
TempSsn: T200000004, Company_name: 设备检测外包公司
TempSsn: T200000013, Company_name: 门窗维修外包公司

All Temporary Employees:
------------------------------------------------------------
Error: 0
```

- create_temp_employee "ssn" "company"



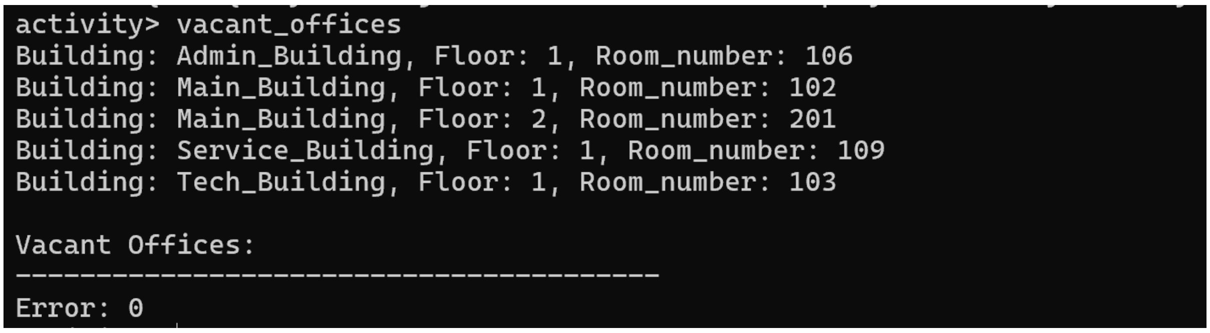
```
activity> create_temp_employee "11" "wang"
Add new temporary employee: (ssn) 11
Add contractor company: wang for employee 11
Temporary employee created successfully: 1
```

3.2.5 Report commands

Command	Description
vacant_offices	List all currently unoccupied offices.

Use examples:

- vacant_offices



```
activity> vacant_offices
Building: Admin_Building, Floor: 1, Room_number: 106
Building: Main_Building, Floor: 1, Room_number: 102
Building: Main_Building, Floor: 2, Room_number: 201
Building: Service_Building, Floor: 1, Room_number: 109
Building: Tech_Building, Floor: 1, Room_number: 103

Vacant Offices:
----------------------------------------
Error: 0
```

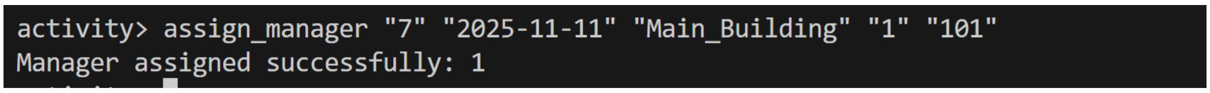
3.2.6 Assignment commands

Command	Description

assign_manager "manager_ssn" "time" "building" "floor" "room"	Assign a manager to an activity. PS; You can use this command only after create a new activity without supervisor.
assign_employee "time" "building" "floor" "room" "worker_ssn"	Assign a worker or temporary worker to an activity.

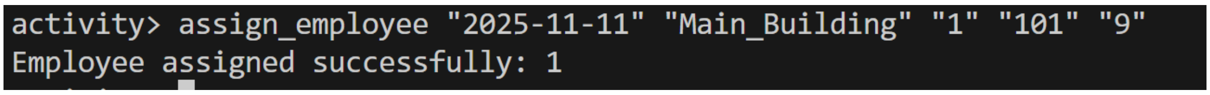
Use examples:

- assign_manager "7" "2025-11-11" "Main_Building" "1" "101"



```
activity> assign_manager "7" "2025-11-11" "Main_Building" "1" "101"
Manager assigned successfully: 1
```

- assign_employee "2025-11-11" "Main_Building" "1" "101" "9"



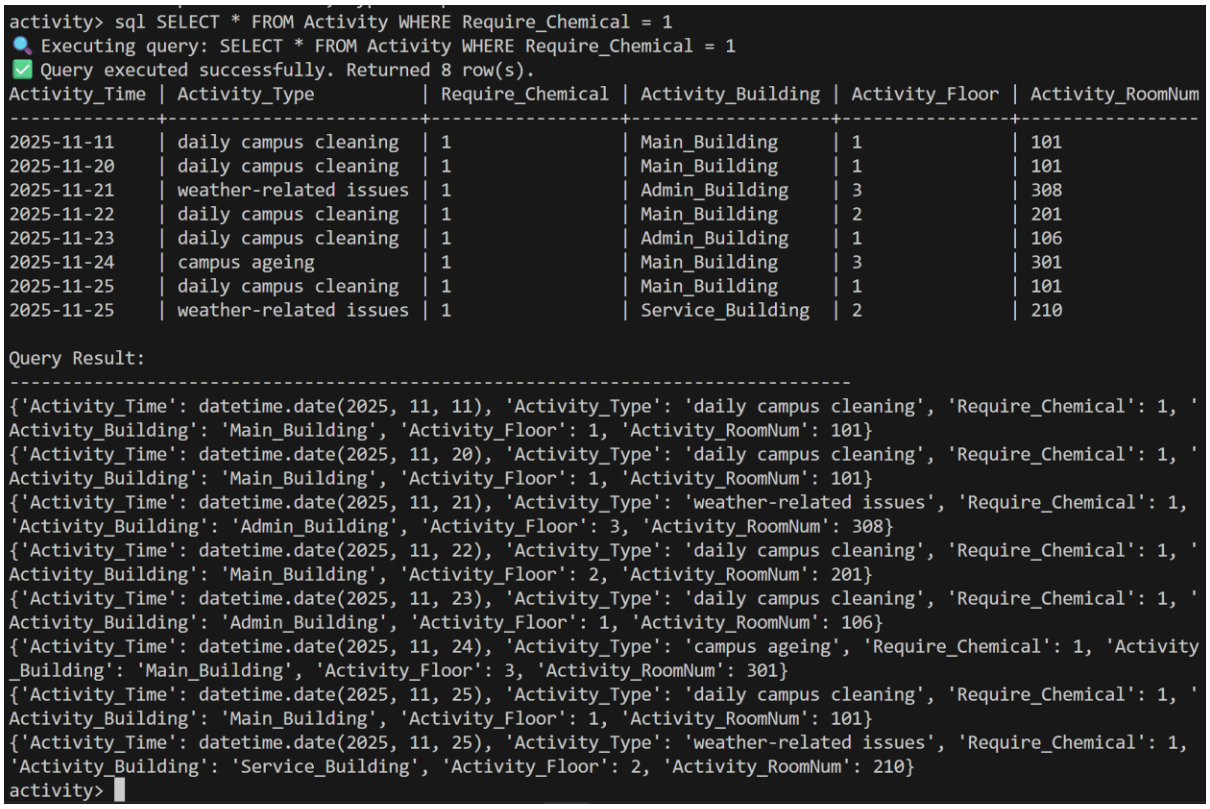
```
activity> assign_employee "2025-11-11" "Main_Building" "1" "101" "9"
Employee assigned successfully: 1
```

3.2.7 SQL queries

Command	Description
sql SQL_STATEMENT	Execute a custom SQL query directly on the database. Example: sql SELECT * FROM Employee WHERE Emp_Level='base_level worker'.

Use examples:

- sql SELECT * FROM Activity WHERE Require_Chemical = 1



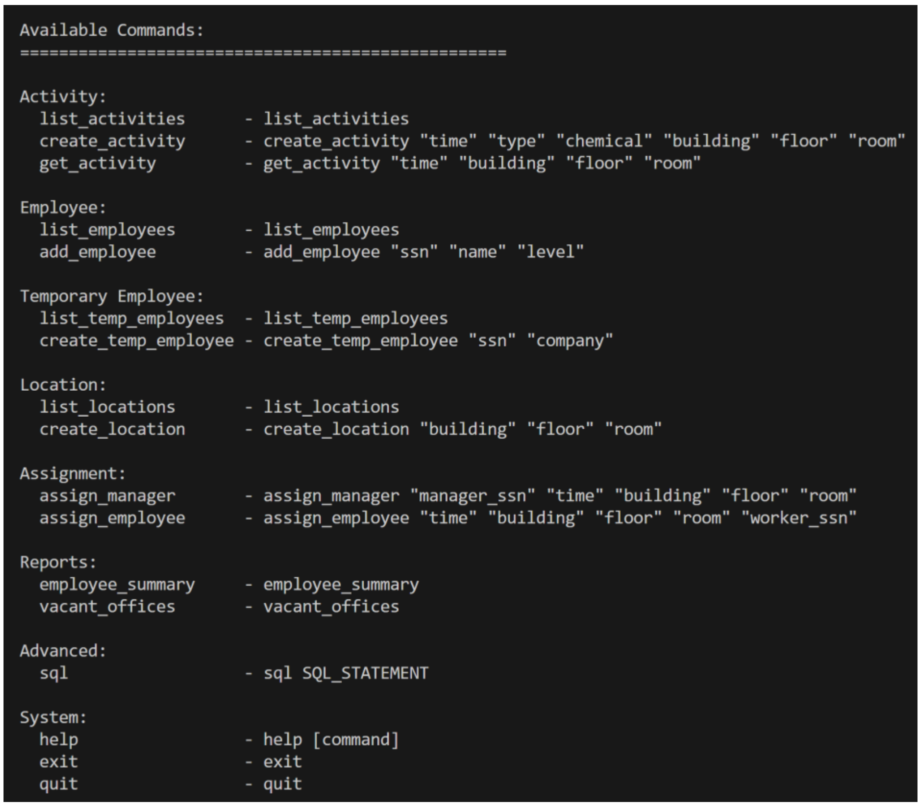
```
activity> sql SELECT * FROM Activity WHERE Require_Chemical = 1
🔍 Executing query: SELECT * FROM Activity WHERE Require_Chemical = 1
✅ Query executed successfully. Returned 8 row(s).
Activity_Time | Activity_Type          | Require_Chemical | Activity_Building | Activity_Floor | Activity_RoomNum
--------------+------------------------+------------------+-------------------+----------------+-----------------
2025-11-11    | daily campus cleaning  | 1                | Main_Building     | 1              | 101
2025-11-20    | daily campus cleaning  | 1                | Main_Building     | 1              | 101
2025-11-21    | weather-related issues | 1                | Admin_Building    | 3              | 308
2025-11-22    | daily campus cleaning  | 1                | Main_Building     | 2              | 201
2025-11-23    | daily campus cleaning  | 1                | Admin_Building    | 1              | 106
2025-11-24    | campus ageing          | 1                | Main_Building     | 3              | 301
2025-11-25    | daily campus cleaning  | 1                | Main_Building     | 1              | 101
2025-11-25    | weather-related issues | 1                | Service_Building  | 2              | 210

Query Result:
--------------------------------------------------------------------------------
{'Activity_Time': datetime.date(2025, 11, 11), 'Activity_Type': 'daily campus cleaning', 'Require_Chemical': 1, '
Activity_Building': 'Main_Building', 'Activity_Floor': 1, 'Activity_RoomNum': 101}
{'Activity_Time': datetime.date(2025, 11, 20), 'Activity_Type': 'daily campus cleaning', 'Require_Chemical': 1, '
Activity_Building': 'Main_Building', 'Activity_Floor': 1, 'Activity_RoomNum': 101}
{'Activity_Time': datetime.date(2025, 11, 21), 'Activity_Type': 'weather-related issues', 'Require_Chemical': 1,
'Activity_Building': 'Admin_Building', 'Activity_Floor': 3, 'Activity_RoomNum': 308}
{'Activity_Time': datetime.date(2025, 11, 22), 'Activity_Type': 'daily campus cleaning', 'Require_Chemical': 1, '
Activity_Building': 'Main_Building', 'Activity_Floor': 2, 'Activity_RoomNum': 201}
{'Activity_Time': datetime.date(2025, 11, 23), 'Activity_Type': 'daily campus cleaning', 'Require_Chemical': 1, '
Activity_Building': 'Admin_Building', 'Activity_Floor': 1, 'Activity_RoomNum': 106}
{'Activity_Time': datetime.date(2025, 11, 24), 'Activity_Type': 'campus ageing', 'Require_Chemical': 1, 'Activity
_Building': 'Main_Building', 'Activity_Floor': 3, 'Activity_RoomNum': 301}
{'Activity_Time': datetime.date(2025, 11, 25), 'Activity_Type': 'daily campus cleaning', 'Require_Chemical': 1, '
Activity_Building': 'Main_Building', 'Activity_Floor': 1, 'Activity_RoomNum': 101}
{'Activity_Time': datetime.date(2025, 11, 25), 'Activity_Type': 'weather-related issues', 'Require_Chemical': 1,
'Activity_Building': 'Service_Building', 'Activity_Floor': 2, 'Activity_RoomNum': 210}
activity>
```

3.2.8 System commands

Command	Description
help	Display all available commands with short explanations.
exit / quit / EOF	Exit the CLI program.

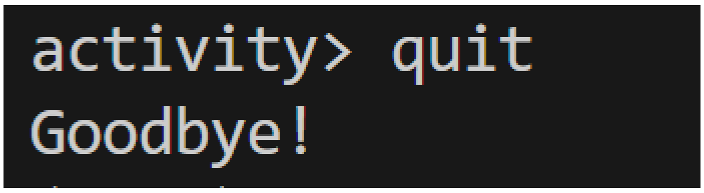
Use examples:

- help



```
Available Commands:
==================================================

Activity:
  list_activities      - list_activities
  create_activity      - create_activity "time" "type" "chemical" "building" "floor" "room"
  get_activity         - get_activity "time" "building" "floor" "room"

Employee:
  list_employees       - list_employees
  add_employee         - add_employee "ssn" "name" "level"

Temporary Employee:
  list_temp_employees  - list_temp_employees
  create_temp_employee - create_temp_employee "ssn" "company"

Location:
  list_locations       - list_locations
  create_location      - create_location "building" "floor" "room"

Assignment:
  assign_manager       - assign_manager "manager_ssn" "time" "building" "floor" "room"
  assign_employee      - assign_employee "time" "building" "floor" "room" "worker_ssn"

Reports:
  employee_summary     - employee_summary
  vacant_offices       - vacant_offices

Advanced:
  sql                  - sql SQL_STATEMENT

System:
  help                 - help [command]
  exit                 - exit
  quit                 - quit
```

- quit



```
activity> quit
Goodbye!
```

4. Troubleshooting

Because the way that cli and GUI throwing error message are different, this section will separately describe the error issues in two modes.

4.1 Command line mode

4.1.1 Startup and Connection

Message: Failed to initialize service: {e}
Cause: The system cannot connect to the database (MySQL not running, wrong password, or database not initialized).
Solutions:

- Ensure MySQL Server is running on the configured host/port (default: localhost:3306).
- Enter the correct database password when prompted.

- If this is the first run, make sure the schema has been created (the system can auto-initialize from schema.sql).

4.1.2 Command Errors

Error Message:

Usage: create activity "time" "type" "chemical" "building" "floor" "room"

Cause:
The command was entered with the wrong number of arguments.

Solution:
Re-enter the command exactly as shown in the usage message, using proper quoting and six arguments.

Error Message:

- Usage: get_activity "time" "building" "floor" "room"
  Usage: add_employee "ssn" "name" "level"
  Usage: create_location "building" "floor" "room"
  Usage: create_temp_employee "ssn" "company"
  Usage: assign_manager ... / assign_employee ...
  Usage: sql SQL_STATEMENT

Cause:
The user provided too few or too many parameters for the command.

Solution:
Follow the exact syntax shown in the “Command Overview” section and ensure the correct number of parameters.

4.1.3 Generic Runtime Errors

Error Message: Error: {e}

Cause:
A general exception occurred during execution. Common reasons include:

- Referencing a non-existing employee, activity, location, or temp worker
- Invalid input format
- Validation failure in the underlying data layer
- Database constraint errors
- Internal SQL errors

Solution:

- Check that the referenced records exist (e.g., using list commands)
- Verify that all parameters match the required formats (date, SSN, level, floor, room)
- Re-run the command with corrected input

4.1.4 Data Not Found Messages

Error Messages:

- No activities found.
- No employees found.
- No locations found.
- No temporary employees found.
- No summary data available.
- No vacant offices found.

Cause: The database contains no records matching the query.

Solution:

- Create the required records before querying
- Check spelling and arguments (e.g., building, time, SSN)
- Verify that filters (time, room, floor) match existing data

4.1.5 SQL Execution Errors

Error Message: SQL Error: {e}

Cause:
The SQL statement provided by the user is invalid.
Possible issues include:

- SQL syntax errors
- Incorrect column or table names
- Missing quotes around string values
- Querying non-existing tables

Solution:

- Check the SQL syntax carefully
- Match field names with the database schema (schema.sql)
- Use only SELECT queries for safety

Error Message: Query executed successfully (no results returned).

Cause:
The SQL statement is valid but returns zero rows.

Solution:

- Check the filtering conditions
- Ensure that target data actually exists
- Correct WHERE conditions or table names

Error Message:

Usage: sql SQL_STATEMENT

Cause:
The sql command was used without providing a SQL string.

Solution:
Provide a complete SQL command, for example: sql SELECT * FROM Employee;

4.1.6 Assignment & Relationship Errors

Error Message: Error: ... (during assign_manager or assign_employee)

Cause:

- Employee or manager SSN does not exist
- Activity does not exist
- Invalid data formats (floor/room/time)
- Business constraints violated (for example: invalid level)

Solution:

- Ensure the activity exists before assignment
- Verify the SSN using list_employees
- Use correct date/time and location formats

4.2 GUI Mode

4.2.1 The GUI window does not appear

Possible Causes

- Python environment is not correctly configured.
- Tkinter is not installed (for some Linux distributions).
- MySQL connection failed during initialization.

Solutions

- Ensure you are running the program with Python 3.9+.
- Check your database password and confirm MySQL is running.

4.2.2 "System Failed" or "SQL Error" pop-ups

Possible Causes

- The SQL query generated by the GUI violates constraints.
- The MySQL password is incorrect.
- The database schema does not match the Stage 1 design.

Solutions

- Double-check foreign key constraints (e.g., building/floor/room must exist).
- Restart MySQL and verify your login credentials.
- Re-import the provided SQL schema.

4.2.3 A dialog window closes without doing anything

Possible Causes

- Some input fields were left empty.
- The user pressed "Cancel".
- The input did not pass the validator and the GUI blocked submission.

Solutions

- Make sure all required fields are filled.
- Check error messages for hints such as:
  "All blanks need to be filled",
  "Empty SSN",
  "Input Error".

4.2.4 A list (TreeView) shows no results

Possible Causes

- No matching records exist.
- The database is empty.
- Filters (e.g., building name) were misspelled.

Solutions

- Try Show All buttons first to verify data exists.
- Ensure the spelling of building names and SSN is correct.
- Check that the database was created successfully.

4.2.5 The GUI freezes or becomes unresponsive

Possible Causes

- Running a heavy SQL query (especially in the SQL tab).
- Too many windows opened at once.
- Your MySQL server is slow or overloaded.

Solutions

- Close unnecessary dialog windows.
- Avoid long-running custom SQL queries.
- Restart MySQL service.

4.2.6 Cannot assign/remove an employee from an activity

Possible Causes

- SSN does not exist in database.
- Activity does not exist.
- The building/floor/room combination is invalid.

Solutions

- Use “Show all Employees” and “Show all Activities” to confirm the record exists.
- Ensure that floor and room numbers are integers.
- Make sure the corresponding location exists in the Location tab.

4.2.7 SQL Query tab returns nothing

Possible Causes

- SQL syntax is wrong.
- Query returns an empty result set.
- Trying to run multi-line SQL incorrectly.

Solutions

- Test with simple queries, e.g.: SELECT * FROM Employee;
- Ensure each query ends properly (no trailing semicolons required).
- Avoid multiple statements in one submission.

4.2.8 Date-related errors (Activity creation/filters)

Possible Causes

- Incorrect date format (system expects YYYY-MM-DD).
- Activity time not matching MySQL DATE type.

Solutions

- Always enter dates like: 2024-11-21
- Check that the Activity_Time column is a DATE type, not VARCHAR.

4.2.9 GUI closes unexpectedly

Possible Causes

- quick_query.close() failed when exiting.
- Python crashed due to an unhandled SQL exception.

Solutions

- Try running from terminal to see error logs.
- Ensure your DAO layer has the correct close() implementation.
- Restart and try again.

5. Frequently Asked Questions (FAQ)

Q1. Why does the system ask for my database password every time?

The CLI does not store credentials for security reasons.
You must enter the MySQL password every time you start up.

Q2. What format should activity time follow?

The system requires:

YYYY-MM-DD

If the format is wrong, activity creation and assignment will fail.

Q3. Why does my query return unexpected empty results?

Most empty results come from:

- Case-sensitive values ('B1' ≠ 'b1')
- Exact match requirements for building, floor, or room
- Missing quotes in SQL queries
- Leading/trailing spaces in input

Q4. Why can't I update a record after creating it?

Even if a record is created, some operations have validation dependencies:

- Assigning a manager requires the manager to be mid-level or above
- Assigning employees requires the activity to already exist
- Some operations block if some related clash happens.

Q5. Can I use partial matching or fuzzy search?

No.
All system lookups are exact-match because they rely on SQL primary keys, for example: "201" ≠ "0201".

Q6. Why does the GUI show empty tables even after I inserted data?

This usually happens when the database connection is valid, but the inserted data does not match the required schema constraints.
For example, an activity may fail to insert because the building, floor, or room does not exist in the Location table, causing the action to roll back silently.

How to fix it

- Use View All Locations and confirm that the location exists before adding activities.
- Re-insert the data and watch for error pop-ups such as "Set fail" or "Input Error".
- Make sure MySQL is running with the correct schema imported.

Q7. Why can't I assign employees or temporary workers to an activity?

Assignment dialogs rely entirely on existing data. If a dropdown list (ComboBox) appears empty, it means the required employee type is missing in the database.

Possible causes

- No mid-level managers or base-level workers have been created.
- Temp employees exist, but the company information was not inserted correctly.
- The activity you want to assign does not exist or the time/building/floor/room combination is incorrect.

How to fix it

- Create employees using Insert EMP first.
- Check activities using Show all under Activity Manage.
- Ensure floor and room values are integers and match actual locations.

6. Additional Resources

If you have questions about the tools used in this project, please refer to their official documents. This manual will not elaborate further.

- **MySQL Documentation**
  https://dev.mysql.com/doc/
  Reference for SQL syntax, constraints, data types, and troubleshooting server issues.

- **Python MySQL Connector**
  https://dev.mysql.com/doc/connector-python/en/
  Details regarding connection errors, cursor usage, and query execution.

- **Tkinter GUI Reference (if GUI included)**
  https://docs.python.org/3/library/tkinter.html
  Standard library documentation for GUI components such as frames, input fields, and message dialogs.

If you have any questions about SQL queries itself, please refer to:

MySQL 8.0 Reference Manual https://dev.mysql.com/doc/refman/8.0/en/select.html

7. Appendix: Terminology

Activity

A scheduled task that takes place at a specific time and location. Activities include cleaning, maintenance, and repairs. Each activity is uniquely identified by its timestamp and location.

Employee

A full-time campus staff member with one of the following levels:

Executive Officer

Mid-level Manager

Base-level Worker

Employees can be assigned to activities and managed through the system.

Temporary Employee (Temp Worker)

A non-permanent worker hired through an external contractor company. Temp workers can participate in activities but have separate records from employees.

Supervisor / Supervision Relationship

A hierarchical relationship where a mid-level manager supervises an employee or temporary worker. Used for tracking organizational structure and responsibilities.

Location

A specific place on campus, represented by building, floor, and room.
Locations are required when creating activities or assigning offices.

Office / Vacant Office

A room currently allocated to an employee.
A vacant office is a room not assigned to any staff member, retrievable through built-in queries.

DAO (Data Access Object)

A layer in the system that handles all SQL operations such as inserting, querying, updating, and deleting data. It separates business logic from database operations.

Service Layer

The main logic controller that connects the GUI/CLI to the DAO.
It processes user actions, validates inputs, and performs operations.

GUI (Graphical User Interface)

A Tkinter-based interface with multiple tabs, allowing users to perform CMMS operations visually without typing commands.

CLI (Command Line Interface)

A text-based interface that accepts typed commands. Recommended for advanced users or batch operations.

SQL (Structured Query Language)

The language used to interact with the database. CMMS supports custom SQL queries through the sql command in the CLI.

Schema

The structure of the database, including tables, fields, and constraints, defined in schema.sql.

SSN (Social Security Number)

The unique identifier for employees and temporary workers in the system.

Validation

The process of checking input correctness (e.g., time format, valid employee level, legal supervision). Invalid inputs will be rejected with corresponding warnings.