POSTS AND TELECOMMUNICATIONS INSTITUTE OF TECHNOLOGY

Faculty of Information Technology

TERM PROJECT

COURSE: DATABASE

Lecturer:	Dr. Nguyen Dinh Hoa
Class:	E22CQCN03-B
Group:	2
Topics:	Design a database system for a University Dormitory
Member of group:	Duong Minh Quang Dang Tuan Phong Bui Minh Tung

Ha Noi – 2024

TABLE OF CONTENT

I. Requirement Analysis

1. Real-life Scenario

Each year, there are many students enrolled in a university. In order to satisfy their needs of accommodation, universities usually set up 1 or more buildings as dormitories for them. To manage and control the information of such a high number of people, a well-developed database is essentially required. In this case, let assume that the university that we are working with has 3 dormitories called A, B and C. Here is what it will look like:

Firstly, we need Dormitory Manager Department staff to manage all the work, the Dormitory Manager Department staff will be split into 3 categories: office staff, guard and dormitory manager. They will have personal id, full name, salary, phone number (a staff can have multiple phone numbers), email address (only using one email for work), address, date of birth and status (still working or not), assigned dormitory and staff type.

Next, we need information about the dormitories in that university. It will consist of dormitory ID, location, number of floors, number of rooms and overall condition (new, old, stable, not stable)

After moving in, students will live in the assigned room which is managed by all the staff of the Dormitory Manager Department. The room details are Room ID (for example, 204-B, this means that room number is 04 and the room is located in floor 2 of dormitory B), amounts of students, condition, capacity and occupancy status. Each room will have a list of facilities, which has: item name, quantity and quality.

Students can also send requests to the manager and after receiving it, they can consider that. In request, we have request ID, content, type, status, completion date.

To register in a dormitory, a student must create a contract with the Dormitory Manager Department office staff. This contract consists of contract ID, contract start date, contract end date, contract status and penalty for violation.

After successfully registering into a room in the dormitory, student's information is obligated to update in the database system. Students will have: Student ID, student name (First Name, Middle Name, Last Name), date of birth, address, gender, email (provided by university), phone number(one student can only have one phone number), emergency contact.

After moving in, students will live in the assigned room which is managed by all the staff of the Dormitory Manager Department. The room details are Room ID (for example, 204-B, this means that room number is 04 and the room is located in floor 2 of dormitory B), amounts of students, condition, capacity and occupancy status. Each room will have a list of facilities, which has: item name, quantity and quality. Students can also send requests to the manager and after receiving it, they can consider that. In request, we have request id, type request, status request, content request and completion date.

After a month or a semester, each student will have a bill for electricity, water, internet usage and maybe some maintenance or even penalty. This will be managed by the office staff. The bill will consist of bill id, bill type, amount due, due date and payment status.

2. Database Applications Description:

The database will manage various aspects of a university dormitory:

- Dormitory Management:
  - Dormitory Manager Department Staff: Managing information for office staff, guards and dormitory managers
  - Dormitories: Information about the dormitories including location, number of floors, rooms and overall condition
- Student Management:
  - Registration and Contracts: Students must create contracts with office staff before moving in
  - Student Information: Personal details, academic information and contact details
  - Room Assignments: Assigning and managing student rooms
- Room Management:
  - Room Details: Room ID, capacity, occupancy status and condition
  - Facilities: Items in the room and their conditions
- Requests:
  - Maintenance Requests: Student can send maintenance requests which are managed by dormitory staff
  - Changing Room Requests: Student can send changing room requests which are managed by dormitory staff
- Billing and Payment:
  - Billing: Managing bills for utilities, maintenance and penalties
- Reporting

- Reports: Generating reports on room occupancy, student information, billing status and maintenance activities

3. Applications to be Supported:

Loads: Adding new students, rooms, dormitories, staff, contracts, requests and bills to the database

Updates: Modifying existing records such as updating student information, room assignments, staff details, maintenance and changing room request statuses and billing information

Retrievals: Fetching specific details such as student information, room assignments, staff information, maintenance requests, changing room request

Reports: Generating statistical reports on room occupancy, student payments, monthly revenue, active contracts and room conditions

II. Conceptual Design

1. Entity set

(Key Attribute will be Bold, Underline text)

· Dormitory Manager Department staff (**Personal ID**, first name, middle name, last name, salary, phone number, email address, address and status, assigned dormitory, staff type). Each of the staff will be split to either three types below:

- Guard (**Guard id**)
- Office (**Office id**)
- Dormitory Manager (**Manger id**)

· Dormitory (**Dormitory ID**, location, number of floors, overall condition, number of rooms)

· Student (**Student ID**, Student name (First Name, Middle Name, Last Name), date of birth, address, gender, email address, phone number, emergency contact)

· Contract (**Contract ID**, contract start date, contract end date, contract status, penalty for violation)

· Room (**Room ID**, amounts of students, condition, capacity, occupancy status, gender, floor of rooms)

· Facility (Item name, quantity, overall quality)

· Request (**Request ID,** content, type, status, completion date)

· Bill (**Bill ID**, bill type, amount due, due date and payment status)

2. Relationships:

· Student <register> Contract: one student has only 1 contract, and one contract must be owned by only 1 student -> 1 to 1 (both is fully participation).

· Student <have> Bill: one student may have many bills, but one bill only belonged to one student: 1 to many (both is fully participation)

· Student <live> Rooms: one student must live in one room, but one room can be lived in by many students or no students: many to one (only student is fully participation)

· Dormitory <own> Room: one dormitory can contain lots of rooms, but one room only belongs to a dormitory -> 1 to many (both is fully participation)

· Guard <protect> Dormitory: one guard protect one dormitory, and one dormitory is protected by one guard -> 1 to 1 (both is fully participation)

· Room <include> Facility: one room can have many types of facilities, but each type facility is unique in each room -> 1 to many (both is fully participation)

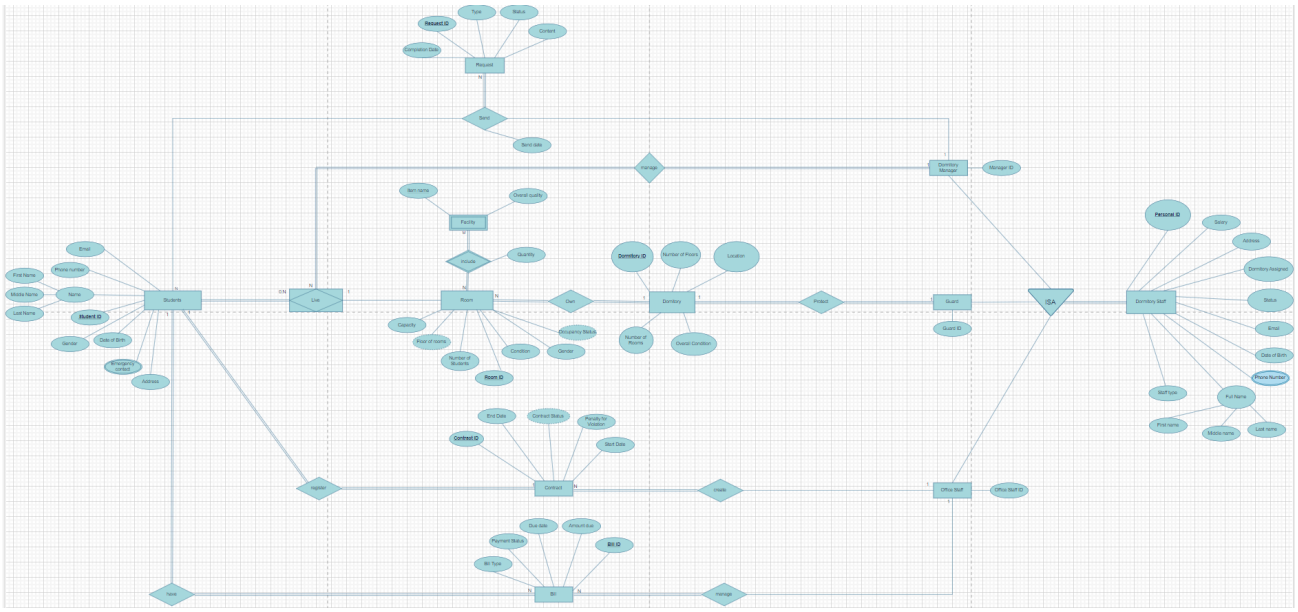
· Student <Send> Request to Manager: one student can send many requests and this will be managed by only one manager. But one manager can manage many requests of many students. (only Request is fully participation)

· Office staff <create> Contract: one staff can create multiple contracts, but a contract must be created by a staff -> 1 to many (only contract is fully participation)

· Office staff <manage> Bill: one staff can manage many bills, but one bill must be managed by a staff -> 1 to many (only bill is fully participation)

· Dormitory manager <manage> Live: one dorm manager can manage the living process of many students in many rooms, but one student living in one room can only be managed by a dorm manager -> 1 to many (both is fully participation).

3. ERD Diagram



III. Logical Design

(Noted that Primary key will be Bold, Underline text; Foreign Key will in Bold, Italic, Dashed Underline text; and Partial Identifier of weak entity set will be normal, underline text)

Relational Schema:

Student (**Student ID**, First Name, Middle Name, Last Name, date of birth, address, gender, phone number, email address, emergency contact)

Student_Emegency_Contact (**Student ID**, emergency contact)

Dormitory Staff (**Personal ID**, first name, middle name, last name, salary, phone number, email, address, status, assigned dormitory, staff type)

Dormitory_Staff_PhoneNumber (**Personal ID**, phone number)

Dormitory manager (**Personal ID – DM**, manager ID)

Guard (**Personal ID – G**, guard ID)

Office staff (**Personal ID – O**, staff ID)

Dormitory (**Dormitory ID**, location, number of floors, overall condition, number of rooms, ***Personal ID - G***)

Room (**Room ID**, amounts of students, condition, capacity, gender, ***Dormitory ID***).

Bill (**Bill ID**, bill type, amount due, due date, payment status, ***Student ID***, ***Personal ID - O***)

Contract (**Contract ID**, contract start date, contract end date, contract status, penalty for violation, ***Personal ID - O***, ***Student ID***)

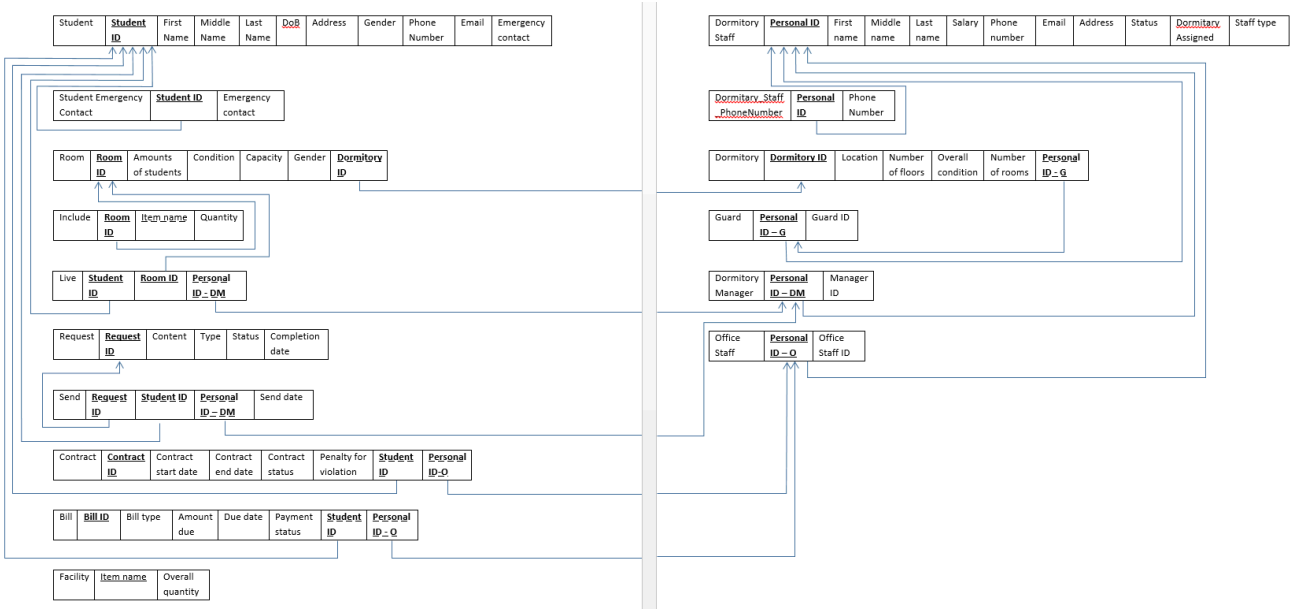
Request (**Request ID**, content, type, status, completion date)

Facility (Item name, overall quality)

Live (***Student ID***, ***Room ID***, ***Personal ID – DM***)

Send (***Student ID***, ***Personal ID - DM***, ***Request ID***, send date)

Include (**Room ID**, Item name, quantity)



IV. Normalization

- R1 = Student (**Student ID**, First Name, Middle Name, Last Name, date of birth, address, gender, phone number, email address)

F1 = {Student ID → First Name; Student ID → Middle Name; Student ID → Last Name; Student ID → Date of Birth; Student ID → Address; Student ID → gender; Student ID → phone number; Student ID → email address; Email Address → Student ID; Phone number→ Student ID} - Min cover

The relational schema includes atomic attributes => The schema is in 1NF.

Student ID, phone number and email address are the candidate keys, they are all single-attribute keys => The schema is in 2NF. (because all other non-prime attributes will fully depend on them)

No non-prime attribute is transitively dependent on any key of R => The schema is in 3NF.

- R2 = Student_Emegency_Contact (**Student ID**; emergency contact)

F2 = {Student ID → emergency contact; emergency contact → Student ID} - Min cover

The relational schema includes atomic attributes => The schema is in 1NF.

emergency contact and Student ID are both candidate keys=> The schema is in 3NF

- R3 = Dormitory Staff (**Personal ID**; first name; middle name; last name; salary; email address; staff type; status; assigned dormitory)

F3 = {Personal ID → First Name; Personal ID → Middle Name; Personal → Last Name; Personal ID → salary; Personal ID → Email Address; Personal ID → Status; Personal ID → Assigned Dormitory; Email Address → Personal ID; Personal ID → Staff type} - Min cover

The relational schema includes atomic attributes => The schema is in 1NF.

Personal ID and email address are the candidate keys, they are all single-attribute keys => The schema is in 2NF.

No non-prime attribute is transitively dependent on any key of R => The schema is in 3NF.

- R4 = Dormitory_Staff_PhoneNumber (**Personal ID**, phone number)

F4 = {Personal ID → Phone number; phone number → Personal ID} - Min cover

The relational schema includes atomic attributes => The schema is in 1NF

Personal ID and phone number are both candidate keys=> The schema is in 3NF

- R5 = Dormitory manager (**Personal ID – DM**, manager ID)

F5 = {Personal ID- DM → Manager ID, Manager ID → Personal ID-DM} - Min cover

The relational schema includes atomic attributes => The schema is in 1NF.

Personal ID-DM and Manager ID are candidate keys=> The schema is in 3NF

- R6 = Guard (**Personal ID – G**, guard ID)

F6 = {Personal ID-G - > guard ID, guard ID → Personal ID-G} - Min cover

The relational schema includes atomic attributes => The schema is in 1NF.

Personal ID- and Guard ID are candidate keys=> The schema is in 3NF

- R7 = Office staff (**Personal ID – O**, staff ID)

F7 = {Personal ID-O - > staff ID, staff ID → Personal ID-O} - Min cover

The relational schema includes atomic attributes => The schema is in 1NF.

Personal ID-O and Staff ID are candidate keys=> The schema is in 3NF

- R8 = (**Dormitory ID**, location, number of floors, overall condition, number of rooms, *Personal ID - G*)

F8 = {Dormitory ID → location, Dormitory ID → number of floors, Dormitory ID → overall condition, Dormitory ID → number of rooms, Dormitory ID →Personal ID - G, Personal ID - G → Dormitory ID} - Min cover

The relational schema includes atomic attributes => The schema is in 1NF.

Dormitory ID and Personal ID - G are the candidate keys, they are all single-attribute keys => The schema is in 2NF.

No non-prime attribute is transitively dependent on Dormitory ID => The schema is in 3NF.

- R9 = Room (**Room ID**, amounts of students, condition, capacity, gender, *Dormitory ID*). (occupancy status and floor of room are derived attributes)

F9 = {Room ID → amounts of students, Room ID → condition, Room ID → capacity, Room ID → occupancy status, Room ID → Gender, Room ID → floor of room, Room ID → Dormitory ID, amount of students, capacity → occupancy status} - Min cover

The relational schema includes atomic attributes => The schema is in 1NF.

Room ID is the candidate key, all non-prime attributes depend fully on Room ID => The schema is in 2NF.

No non-prime attribute is transitively dependent on Room ID => The schema is in 3NF.

- R10 = Bill (**Bill ID**, bill type, amount due, due date, payment status, *Student ID*, *Personal ID - O*) - Min cover

F10 = {Bill ID → bill type, Bill ID → amount due, Bill ID →due date, Bill ID → payment status, Bill ID →Student ID, Bill ID →Personal ID - O}

The relational schema includes atomic attributes => The schema is in 1NF.

Bill ID is the only candidate key, it is a single-attribute key => The schema is in 2NF.

No non-prime attribute is transitively dependent on Bill ID => The schema is in 3NF.

- R11 = Request (**Request ID**, content, type, status, completion date)

F11 = {Request ID -> content, Request ID -> type, Request ID -> status, Request ID -> completion date) - Min cover

The relational schema includes atomic attributes => The schema is in 1NF.

Request ID is the only candidate key, it is a single-attribute key => The schema is in 2NF. .

No non-prime attribute is transitively dependent on Request ID => The schema is in 3NF.

- R12 = Facility (**Item name**, overall quality, Room ID)

F12 = {Item name, Room ID → overall quality) - Min cover

The relational schema includes atomic attributes -> The schema is in 1NF.

Room ID, Item name is the candidate key, overall quality depends fully on candidate key = > The schema is in 2NF.

No non-prime attribute is transitively dependent on the candidate key => The schema is in 3NF.

- R13 = Live (**Student ID, Room ID**, ***Personal ID – DM***)

F13 = {Student ID, Room ID → Personal ID - DM, Student ID → Room ID}- Min cover

The relational schema includes atomic attributes => The schema is in 1NF.

Student ID is the only candidate key, it is a single-attribute key => The schema is in 2NF.

No non-prime attribute is transitively dependent on Student ID => The schema is in 3NF.

- R14 = Send (**Student ID, Personal ID - DM, Request ID,** send date)

F14 = {Request ID → Student ID; Request ID, Student ID → Personal ID - DM} - Min cover

The relational schema includes atomic attributes => The schema is in 1NF.

Student ID is the only candidate key, it is a single-attribute key => The schema is in 2NF.

No non-prime attribute is transitively dependent on Student ID => The schema is in 3NF.

- R15 = Include (**Room ID,** Item name, quantity)

F15 = {Room ID, Item name →quantity} -Min Cover

The relational schema includes atomic attributes => The schema is in 1NF.

Room ID, Item name is the only candidate key, quantity is fully depend on key=> The schema is in 2NF

No non-prime attribute is transitively dependent on the candidate key => The schema is in 3NF.

- R16 = Contract (**Contract ID**, contract start date, contract end date, contract status, penalty for violation, ***Personal ID - O***, ***Student ID***)

F16 = {Contract ID -> contract start date, Contract ID -> contract end date, Contract ID -> contract status, Contract ID -> penalty for violation, Contract ID -> ***Personal ID – O***, Contract ID -> ***Student ID*** , ***Student ID*** -> Contract ID} -Min Cover

The relational schema includes atomic attributes => The schema is in 1NF.

Contract ID, Student ID are the only candidate keys, they are all single attribute keys=> The schema is in 2NF

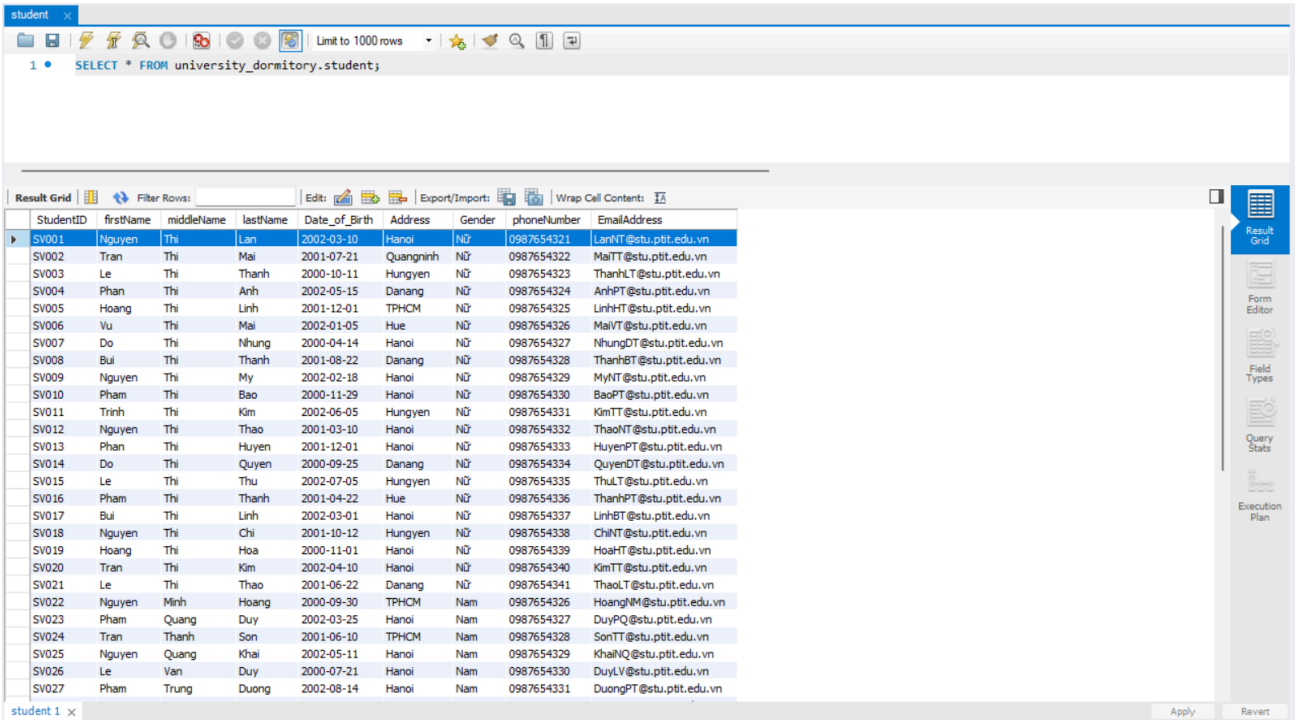
No non-prime attribute is transitively dependent on the candidate key => The schema is in 3NF.

V. Database Implementation

- ***There are 16 tables in this database***
- ***For each table, we will show 2 images: 1 image to present the SQL command to create table and 1 image to illustrate the result when we show all record in these tables.***
- ***The SQL command to create table and insert data will be presented in the attachment project.sql file when we submit the report***

1. Table **Student**

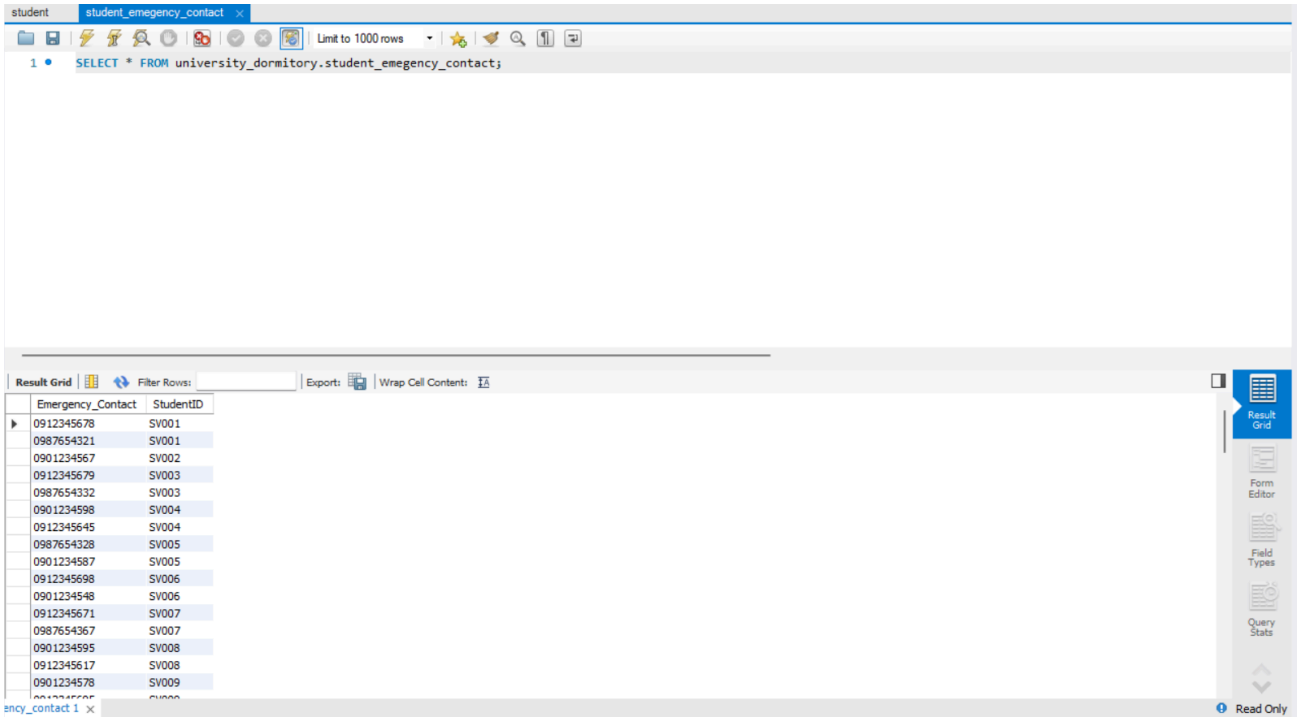
```sql
CREATE TABLE student (
  StudentID varchar(55) NOT NULL PRIMARY KEY,
  firstName varchar(55) NOT NULL,
  middleName varchar(55) NOT NULL,
  lastName varchar(55) NOT NULL,
  Date_of_Birth DATE NOT NULL,
  Address varchar(55) NOT NULL,
  Gender varchar(55) NOT NULL,
  phoneNumber varchar(55) NOT NULL,
  EmailAddress varchar(55) NOT NULL
);
```



StudentID	firstName	middleName	lastName	Date_of_Birth	Address	Gender	phoneNumber	EmailAddress
SV001	Nguyen	Thi	Lan	2002-03-10	Hanoi	Nữ	0987654321	LanNT@stu.ptit.edu.vn
SV002	Tran	Thi	Mai	2001-07-21	Quangninh	Nữ	0987654322	MaiTT@stu.ptit.edu.vn
SV003	Le	Thi	Thanh	2000-10-11	Hungyen	Nữ	0987654323	ThanhLT@stu.ptit.edu.vn
SV004	Phan	Thi	Anh	2002-05-15	Danang	Nữ	0987654324	AnhPT@stu.ptit.edu.vn
SV005	Hoang	Thi	Linh	2001-12-01	TPHCM	Nữ	0987654325	LinhHT@stu.ptit.edu.vn
SV006	Vu	Thi	Mai	2002-01-05	Hue	Nữ	0987654326	MaiVT@stu.ptit.edu.vn
SV007	Do	Thi	Nhung	2000-04-14	Hanoi	Nữ	0987654327	NhungDT@stu.ptit.edu.vn
SV008	Bui	Thi	Thanh	2001-08-22	Danang	Nữ	0987654328	ThanhBT@stu.ptit.edu.vn
SV009	Nguyen	Thi	My	2002-02-18	Hanoi	Nữ	0987654329	MyNT@stu.ptit.edu.vn
SV010	Pham	Thi	Bao	2000-11-29	Hanoi	Nữ	0987654330	BaoPT@stu.ptit.edu.vn
SV011	Trinh	Thi	Kim	2002-06-05	Hungyen	Nữ	0987654331	KimTT@stu.ptit.edu.vn
SV012	Nguyen	Thi	Thao	2001-03-10	Hanoi	Nữ	0987654332	ThaoNT@stu.ptit.edu.vn
SV013	Phan	Thi	Huyen	2001-12-01	Hanoi	Nữ	0987654333	HuyenPT@stu.ptit.edu.vn
SV014	Do	Thi	Quyen	2000-09-25	Danang	Nữ	0987654334	QuyenDT@stu.ptit.edu.vn
SV015	Le	Thi	Thu	2002-07-05	Hungyen	Nữ	0987654335	ThuLT@stu.ptit.edu.vn
SV016	Pham	Thi	Thanh	2001-04-22	Hue	Nữ	0987654336	ThanhPT@stu.ptit.edu.vn
SV017	Bui	Thi	Linh	2002-03-01	Hanoi	Nữ	0987654337	LinhBT@stu.ptit.edu.vn
SV018	Nguyen	Thi	Chi	2001-10-12	Hungyen	Nữ	0987654338	ChiNT@stu.ptit.edu.vn
SV019	Hoang	Thi	Hoa	2000-11-01	Hanoi	Nữ	0987654339	HoaHT@stu.ptit.edu.vn
SV020	Tran	Thi	Kim	2002-04-10	Hanoi	Nữ	0987654340	KimTT@stu.ptit.edu.vn
SV021	Le	Thi	Thao	2001-06-22	Danang	Nữ	0987654341	ThaoLT@stu.ptit.edu.vn
SV022	Nguyen	Minh	Hoang	2000-09-30	TPHCM	Nam	0987654326	HoangNM@stu.ptit.edu.vn
SV023	Pham	Quang	Duy	2002-03-25	Hanoi	Nam	0987654327	DuyPQ@stu.ptit.edu.vn
SV024	Tran	Thanh	Son	2001-06-10	TPHCM	Nam	0987654328	SonTT@stu.ptit.edu.vn
SV025	Nguyen	Quang	Khai	2002-05-11	Hanoi	Nam	0987654329	KhaiNQ@stu.ptit.edu.vn
SV026	Le	Van	Duy	2000-07-21	Hanoi	Nam	0987654330	DuyLV@stu.ptit.edu.vn
SV027	Pham	Trung	Duong	2002-08-14	Hanoi	Nam	0987654331	DuongPT@stu.ptit.edu.vn

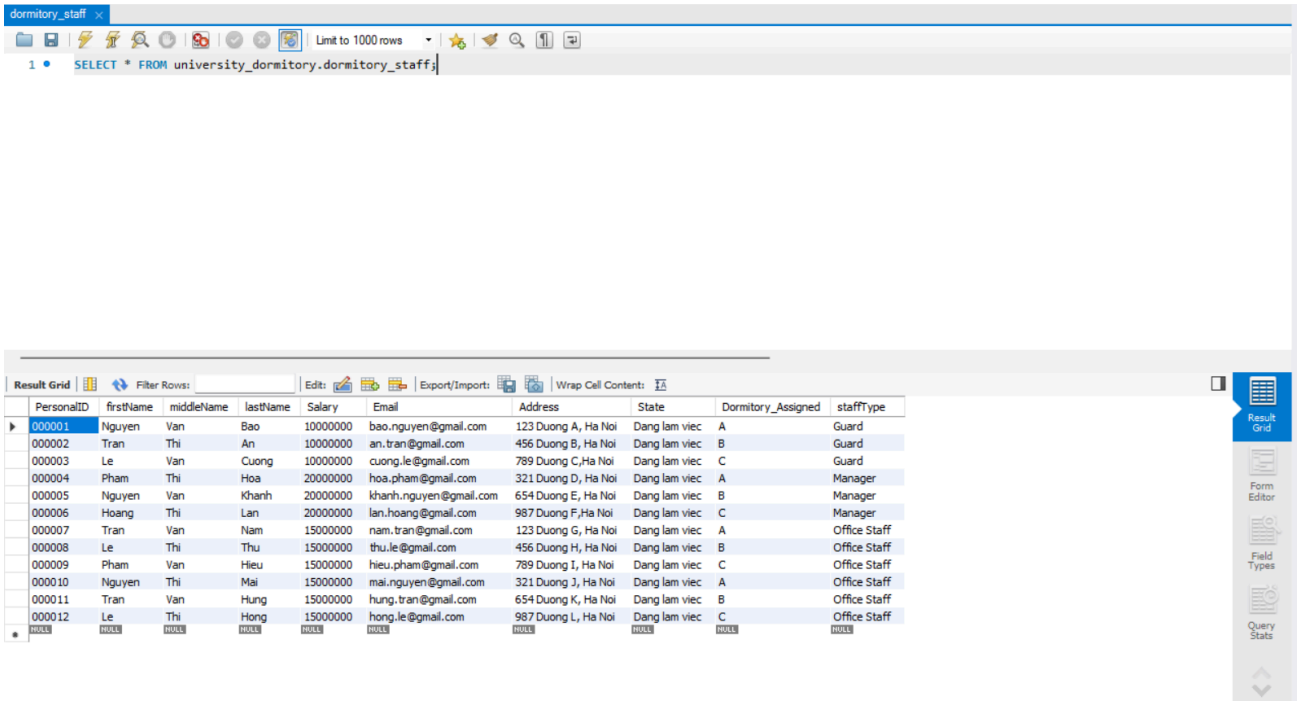
2. Table **Student_Emergency_Contact**

```
CREATE TABLE university_domitory.Student_Emegency_Contact(
    Emergency_Contact VARCHAR(55) NOT NULL,
    StudentID VARCHAR(55) NOT NULL,
    FOREIGN KEY (StudentID) REFERENCES student(StudentID)
);
```



3. Table **Dormitary_Staff**

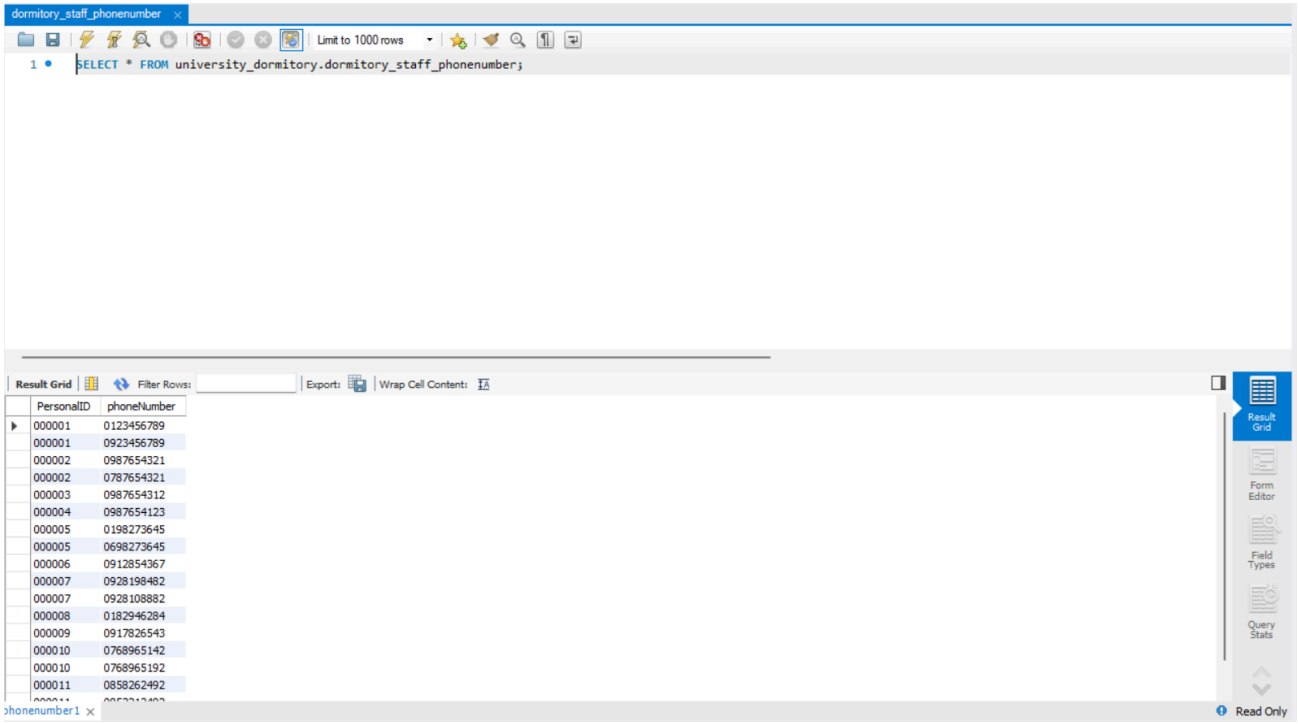
```
1   CREATE TABLE university_domitory.Dormitory_Staff(
2       PersonalID VARCHAR(55) NOT NULL,
3       firstName VARCHAR(55) NOT NULL,
4       middleName VARCHAR(55) NOT NULL,
5       lastName VARCHAR(55) NOT NULL,
6       Salary integer NOT NULL,
7       Email VARCHAR(55) NOT NULL,
8       Address VARCHAR(55) NOT NULL,
9       State VARCHAR(55) NOT NULL,
10      Dormitory_Assigned VARCHAR(55) NOT NULL,
11      staffType VARCHAR(55) NOT NULL,
12      PRIMARY KEY(PersonalID)
13  );
```



PersonalID	firstName	middleName	lastName	Salary	Email	Address	State	Dormitory_Assigned	staffType
000001	Nguyen	Van	Bao	10000000	bao.nguyen@gmail.com	123 Duong A, Ha Noi	Dang lam viec	A	Guard
000002	Tran	Thi	An	10000000	an.tran@gmail.com	456 Duong B, Ha Noi	Dang lam viec	B	Guard
000003	Le	Van	Cuong	10000000	cuong.le@gmail.com	789 Duong C,Ha Noi	Dang lam viec	C	Guard
000004	Pham	Thi	Hoa	20000000	hoa.pham@gmail.com	321 Duong D, Ha Noi	Dang lam viec	A	Manager
000005	Nguyen	Van	Khanh	20000000	khanh.nguyen@gmail.com	654 Duong E, Ha Noi	Dang lam viec	B	Manager
000006	Hoang	Thi	Lan	20000000	lan.hoang@gmail.com	987 Duong F,Ha Noi	Dang lam viec	C	Manager
000007	Tran	Van	Nam	15000000	nam.tran@gmail.com	123 Duong G, Ha Noi	Dang lam viec	A	Office Staff
000008	Le	Thi	Thu	15000000	thu.le@gmail.com	456 Duong H, Ha Noi	Dang lam viec	B	Office Staff
000009	Pham	Van	Hieu	15000000	hieu.pham@gmail.com	789 Duong I, Ha Noi	Dang lam viec	C	Office Staff
000010	Nguyen	Thi	Mai	15000000	mai.nguyen@gmail.com	321 Duong J, Ha Noi	Dang lam viec	A	Office Staff
000011	Tran	Van	Hung	15000000	hung.tran@gmail.com	654 Duong K, Ha Noi	Dang lam viec	B	Office Staff
000012	Le	Thi	Hong	15000000	hong.le@gmail.com	987 Duong L, Ha Noi	Dang lam viec	C	Office Staff

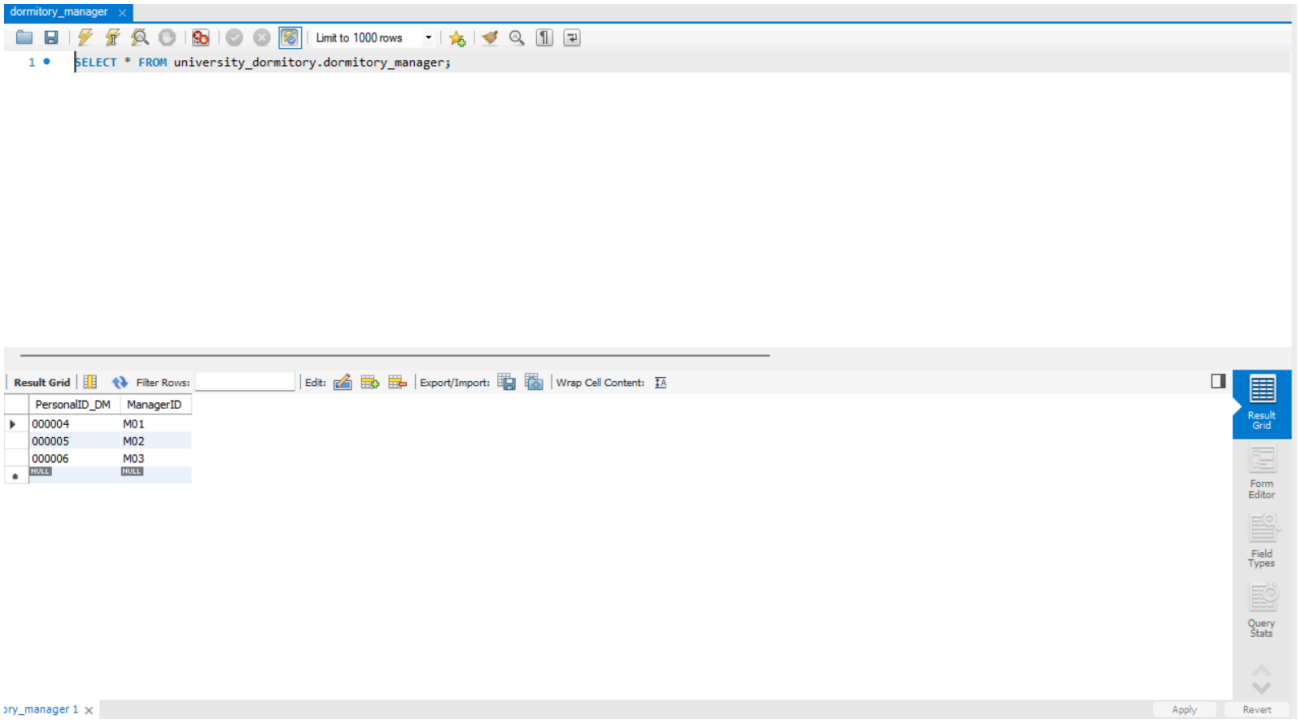
4. Table **Dormitary_Staff_PhoneNumber**

```
1   CREATE TABLE university_domitory.Dormitory_Staff_PhoneNumber(
2       PersonalID VARCHAR(55) NOT NULL,
3       phoneNumber VARCHAR(55) NOT NULL,
4       FOREIGN KEY (PersonalID) REFERENCES Dormitory_staff(PersonalID)
5   )
```



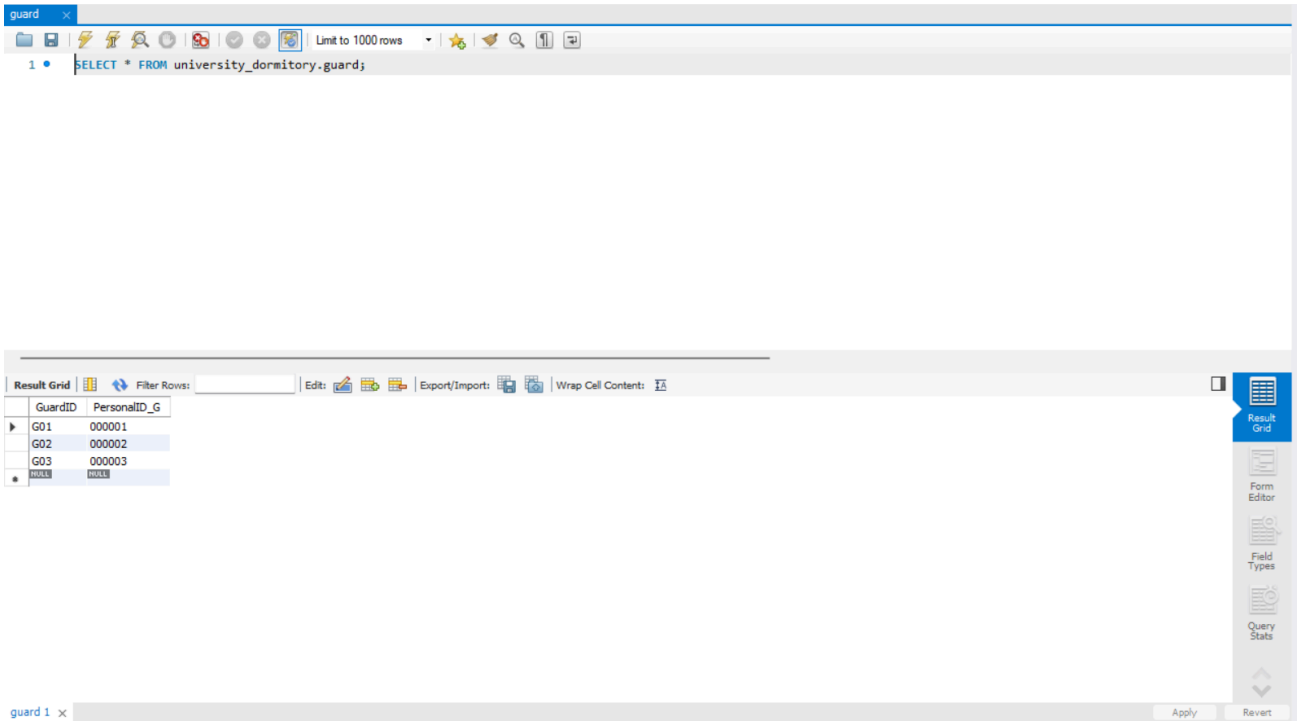
5. Table **Dormitory Manager**

```sql
CREATE TABLE dormitory_manager (
  PersonalID_DM varchar(55) NOT NULL,
  ManagerID varchar(55) NOT NULL,
  PRIMARY KEY (PersonalID_DM),
  FOREIGN KEY (PersonalID_DM) REFERENCES dormitory_staff (PersonalID)
);
```



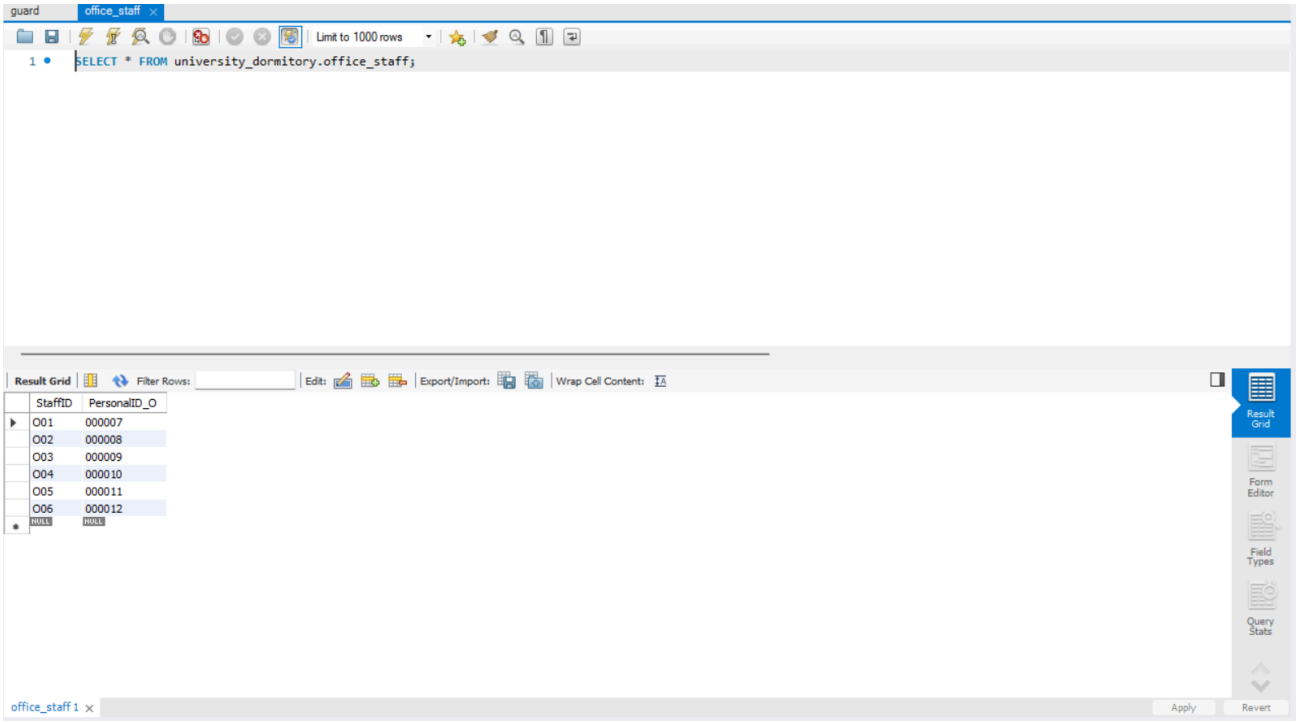
6. Table **Guard**

```
CREATE TABLE guard (
  GuardID varchar(55) NOT NULL,
  PersonalID_G varchar(55) NOT NULL,
  PRIMARY KEY (PersonalID_G),
  FOREIGN KEY (PersonalID_G) REFERENCES dormitory_staff (PersonalID)
);
```



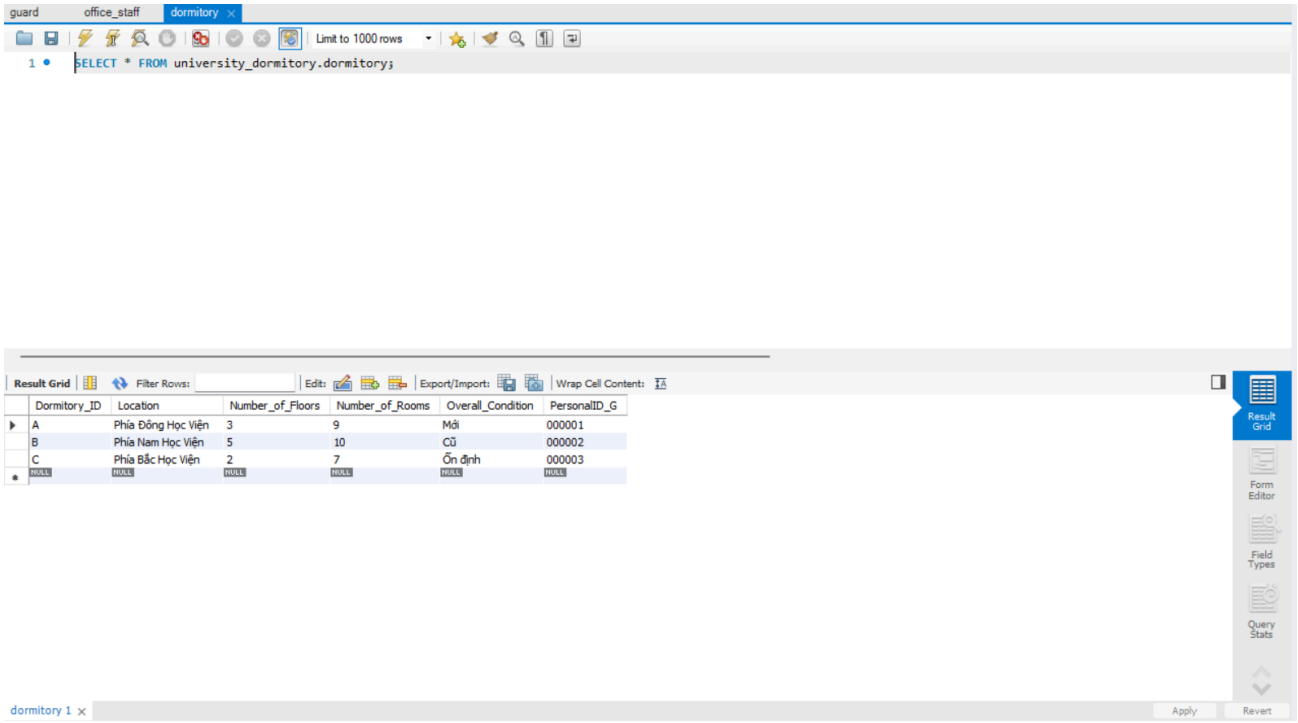
7. Table **Office Staff**

```
1   CREATE TABLE office_staff (
2     StaffID varchar(55) NOT NULL,
3     PersonalID_O varchar(55) NOT NULL,
4     PRIMARY KEY (PersonalID_O),
5     FOREIGN KEY (PersonalID_O) REFERENCES dormitory_staff (PersonalID)
6   );
```



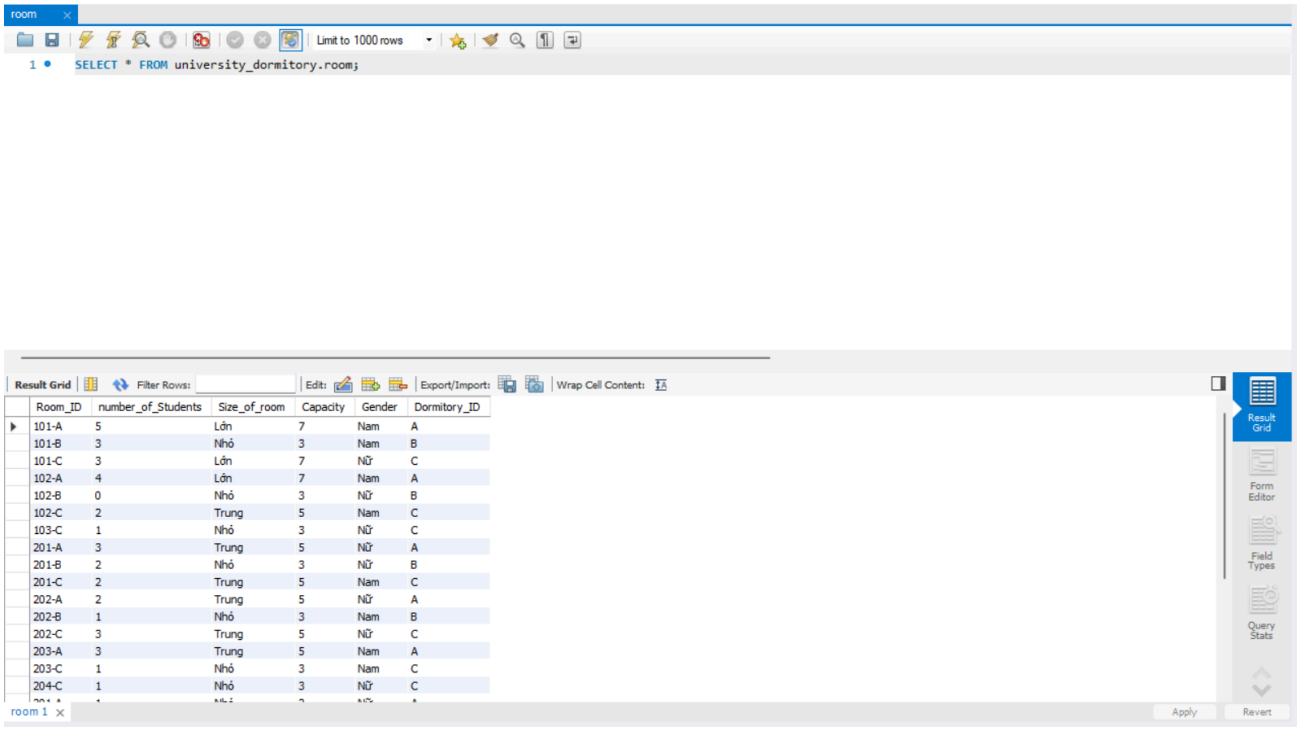
8. Table **Dormitary**

```
CREATE TABLE dormitory (
  Dormitory_ID varchar(55) NOT NULL,
  Location varchar(55) NOT NULL,
  Number_of_Floors int NOT NULL,
  Number_of_Rooms int NOT NULL,
  Overall_Condition varchar(55) NOT NULL,
  PersonalID_G varchar(55) DEFAULT NULL,
  PRIMARY KEY (Dormitory_ID),
  KEY PersonalID_G (PersonalID_G),
  FOREIGN KEY (PersonalID_G) REFERENCES guard (PersonalID_G)
)
```



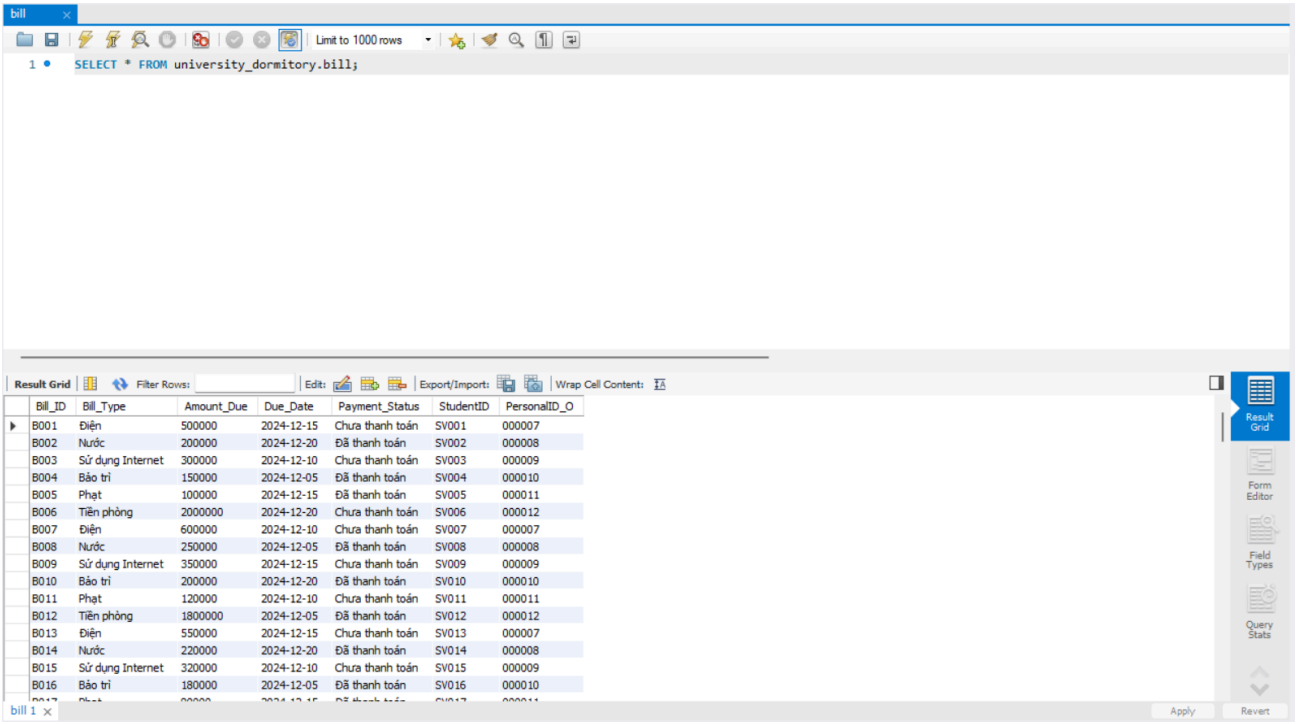
9. Table **Room**

```sql
CREATE TABLE room (
  Room_ID varchar(55) NOT NULL,
  number_of_Students int NOT NULL,
  Size_of_room varchar(55) NOT NULL,
  Capacity int NOT NULL,
  Gender VARCHAR(55) NOT NULL,
  Dormitory_ID varchar(55) NOT NULL,
  PRIMARY KEY (Room_ID),
  FOREIGN KEY (Dormitory_ID) REFERENCES Dormitory (Dormitory_ID)
);
```



10. Table **Bill**

```
CREATE TABLE bill (
  Bill_ID varchar(55) NOT NULL,
  Bill_Type varchar(55) NOT NULL,
  Amount_Due int NOT NULL,
  Due_Date varchar(55) NOT NULL,
  Payment_Status varchar(55) NOT NULL,
  StudentID varchar(55) NOT NULL,
  PersonalID_O varchar(55) NOT NULL,
  PRIMARY KEY (Bill_ID),
  KEY StudentID (StudentID),
  KEY PersonalID_O (PersonalID_O),
  FOREIGN KEY (StudentID) REFERENCES student (StudentID),
  FOREIGN KEY (PersonalID_O) REFERENCES office_staff (PersonalID_O)
)
```



11. Table **Request**

```
CREATE TABLE request (
  RequestID varchar(55) NOT NULL,
  Content varchar(55) NOT NULL,
  Type_of_Request varchar(55) NOT NULL,
  Status_of_Request varchar(55) NOT NULL,
  Completion_Date varchar(55) NOT NULL,
  PRIMARY KEY (`RequestID`)
);
```

request

Limit to 1000 rows

```
SELECT * FROM university_dormitory.request;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

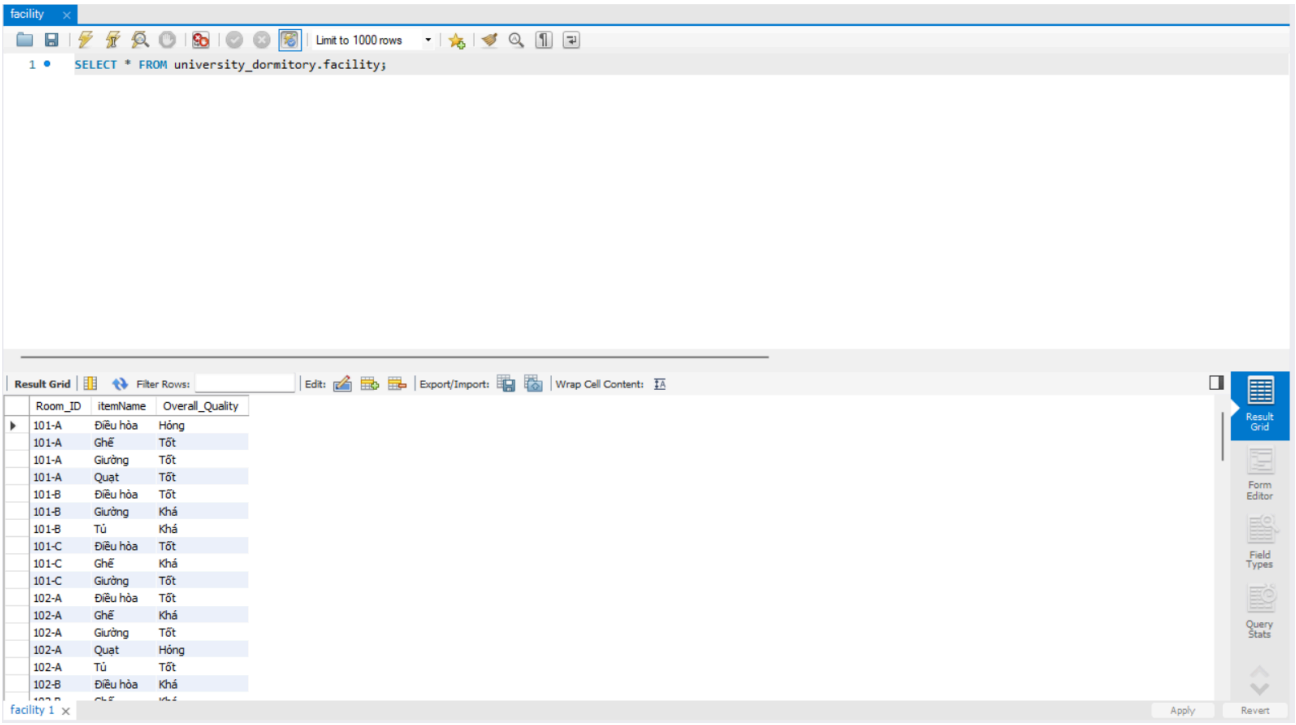
RequestID	Content	Type_of_Request	Status_of_Request	Completion_Date
RQ001	Sua quat hong	Thiet bi	Hoan thanh	2024-11-01
RQ002	Sua den bi hong	Thiet bi	Dang xu ly	none
RQ003	Doi phong do may lanh bi hong	Phong	Dang xu ly	none
RQ004	Bao tri co so ha tang	Bao tri	Hoan thanh	2024-11-02
RQ005	Sua may vi tinh	Thiet bi	Hoan thanh	2024-11-03
RQ006	Sua dieu hoa	Thiet bi	Dang xu ly	none
RQ007	Doi guong trong phong tam	Phong	Dang xu ly	none
RQ008	Bao tri binh nong lanh	Bao tri	Hoan thanh	2024-11-04
RQ009	Doi den trong phong hoc	Phong	Dang xu ly	none
RQ010	Bao tri dong ho	Bao tri	Hoan thanh	2024-11-01
RQ011	Sua may in bi hong	Thiet bi	Dang xu ly	none
RQ012	Sua cua so bi hong	Phong	Hoan thanh	2024-11-03
RQ013	Sua chieu sang trong phong	Phong	Dang xu ly	none
RQ014	Sua canh cua	Bao tri	Hoan thanh	2024-11-02
RQ015	Bao tri tu lanh	Thiet bi	Dang xu ly	none
RQ016	Sua dieu hoa trong phong	Phong	Hoan thanh	2024-11-01

Result Grid | Form Editor | Field Types | Query Stats

request 1 | Apply | Revert

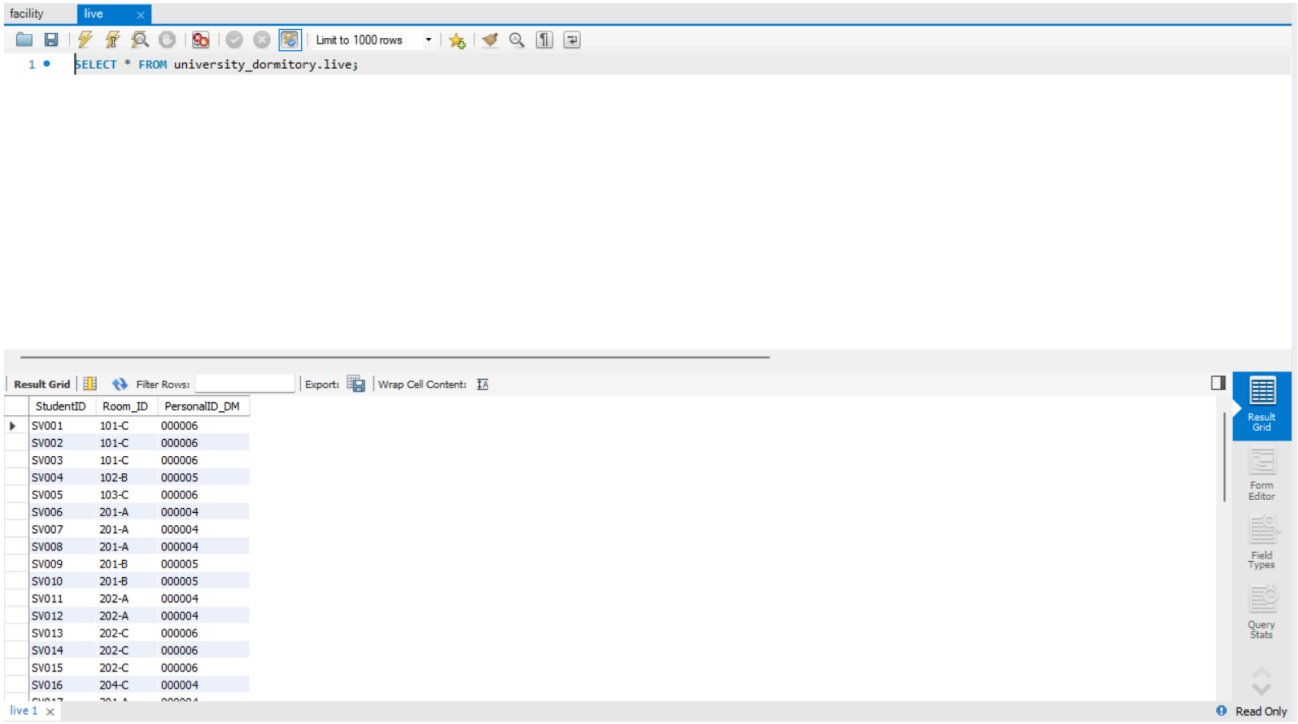
12. Table **Facility**

```sql
CREATE TABLE facility (
  Room_ID varchar(55) NOT NULL,
  itemName varchar(55) NOT NULL,
  Overall_Quality varchar(55) NOT NULL,
  PRIMARY KEY (Room_ID, itemName),  -- Composite key
  FOREIGN KEY (Room_ID) REFERENCES room (Room_ID)  -- Foreign key reference to the room
);
```



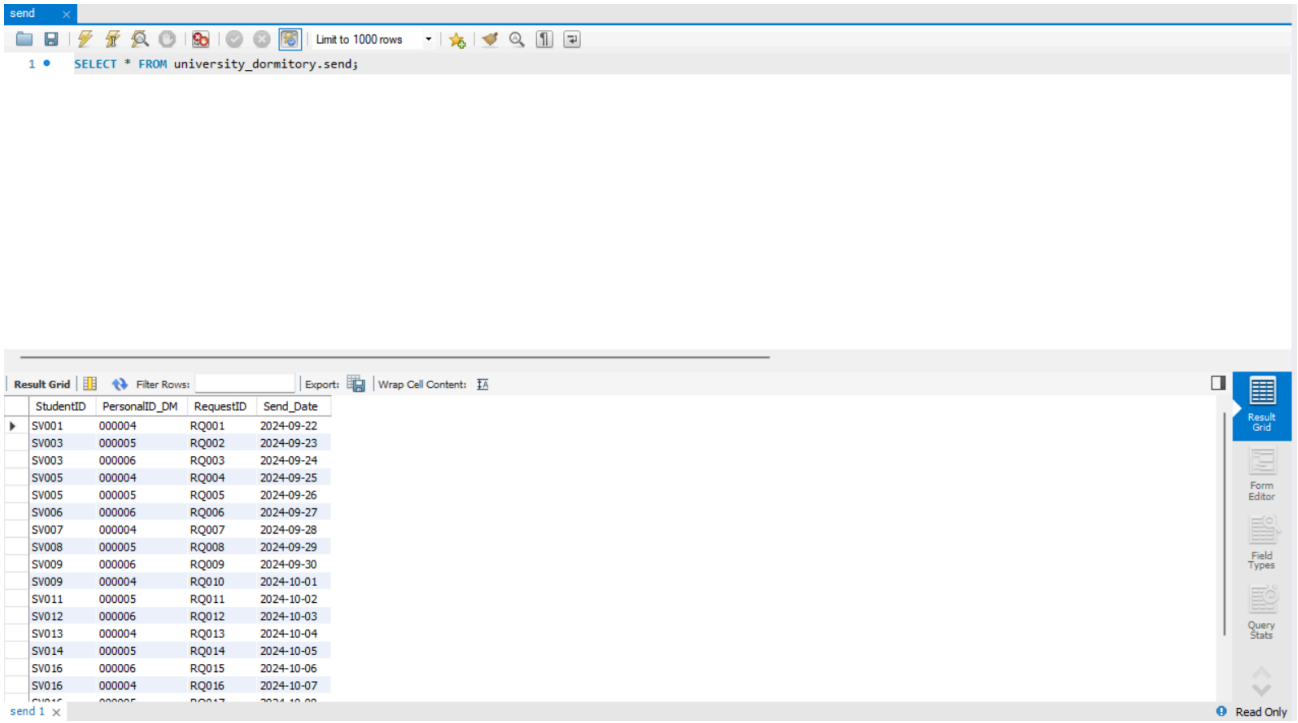
13. Table **Live**

```
1   CREATE TABLE live (
2     StudentID varchar(55) NOT NULL,
3     Room_ID varchar(55) NOT NULL,
4     PersonalID_DM varchar(55) NOT NULL,
5     FOREIGN KEY (StudentID) REFERENCES student (StudentID),
6     FOREIGN KEY (Room_ID) REFERENCES room (Room_ID),
7     FOREIGN KEY (PersonalID_DM) REFERENCES dormitory_manager (PersonalID_DM)
8   );
```



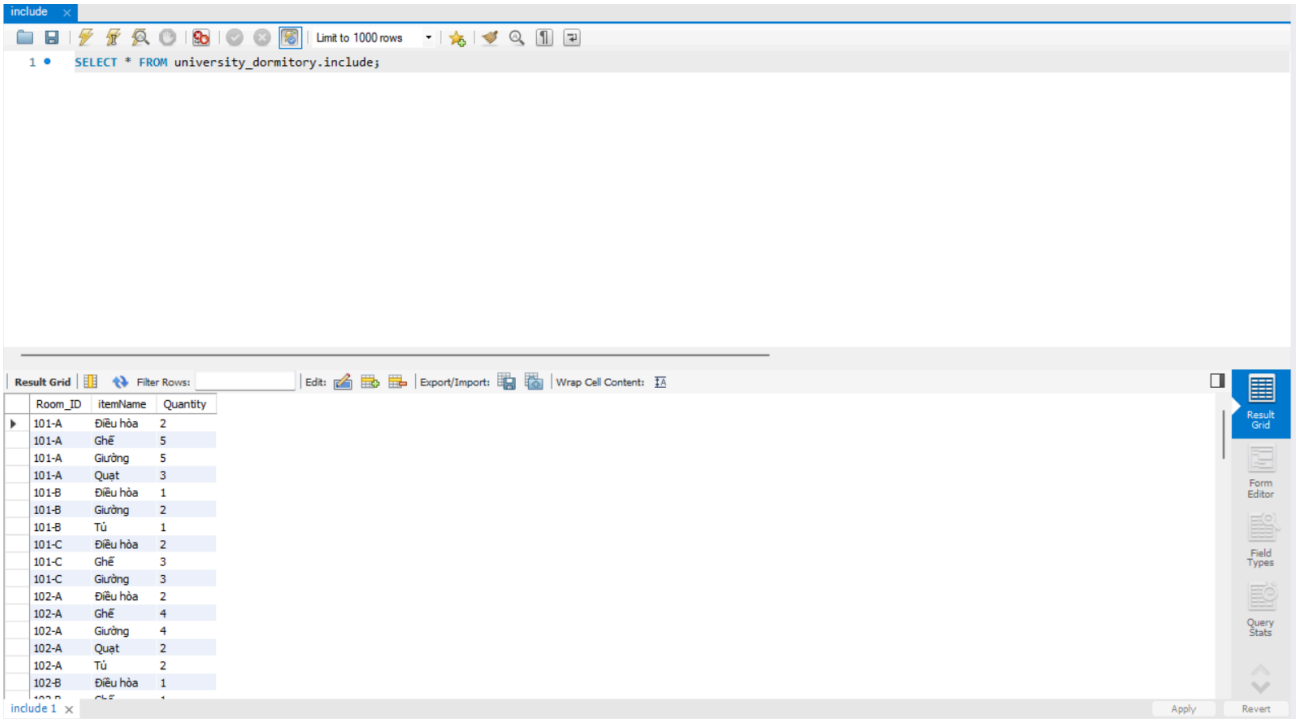
14. Table **Send**

```
1   CREATE TABLE send (
2     StudentID varchar(55) NOT NULL,
3     PersonalID_DM varchar(55) NOT NULL,
4     RequestID varchar(55) NOT NULL,
5     Send_Date varchar(55) NOT NULL,
6     FOREIGN KEY (StudentID) REFERENCES student (StudentID),
7     FOREIGN KEY (PersonalID_DM) REFERENCES dormitory_manager(PersonalID_DM),
8     FOREIGN KEY (RequestID) REFERENCES request(RequestID)
9   )
```



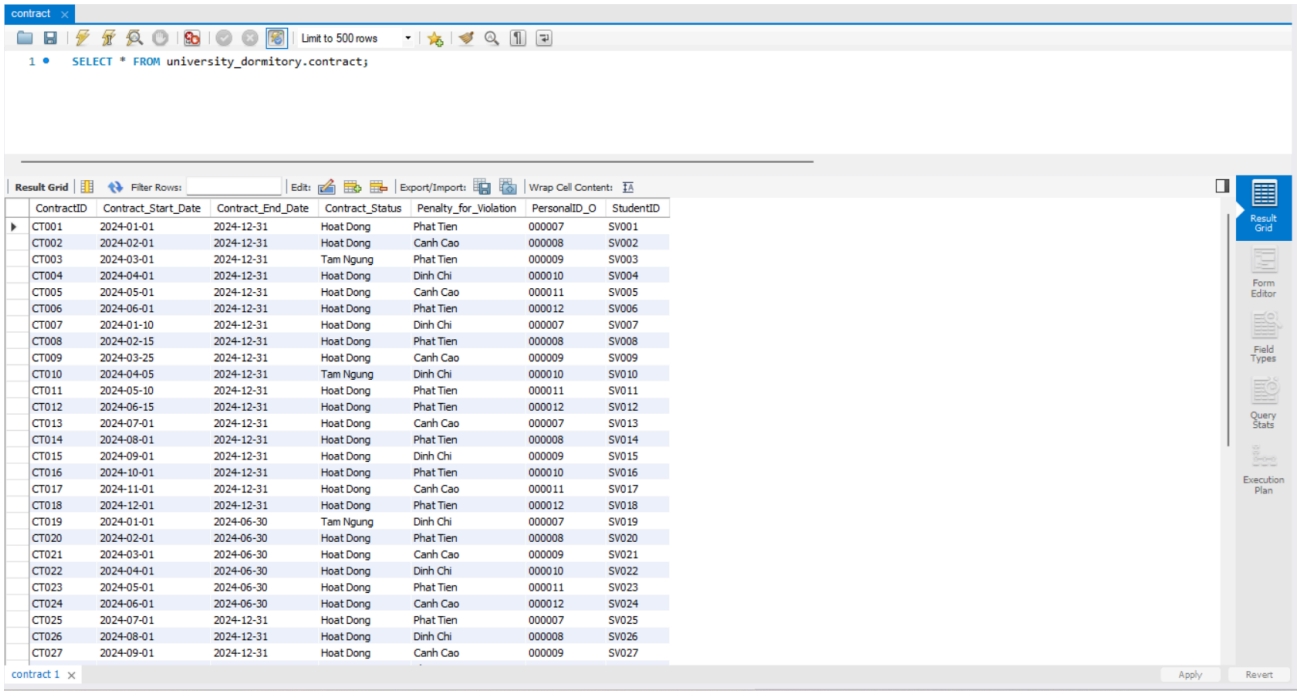
15. Table **Include**

```sql
CREATE TABLE include (
  Room_ID varchar(55) NOT NULL,
  itemName varchar(55) NOT NULL,
  Quantity int NOT NULL,
  PRIMARY KEY (Room_ID, itemName),  -- Composite primary key
  FOREIGN KEY (Room_ID) REFERENCES room (Room_ID),  -- Foreign key to room table
  FOREIGN KEY (Room_ID, itemName) REFERENCES facility (Room_ID, itemName)  -- Foreign ke
);
```



16. Table **Contract**

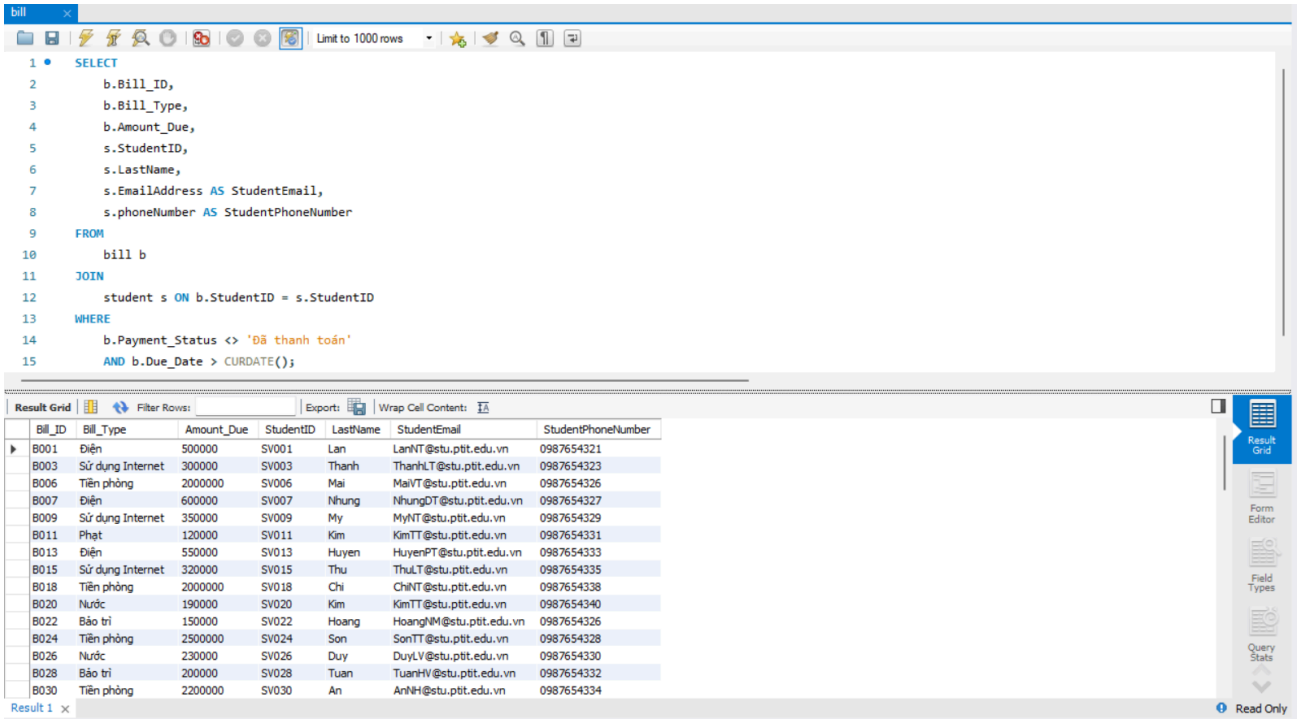
```sql
CREATE TABLE contract (
  ContractID varchar(55) NOT NULL,
  Contract_Start_Date varchar(55) NOT NULL,
  Contract_End_Date varchar(55) NOT NULL,
  Contract_Status varchar(55) NOT NULL,
  Penalty_for_Violation varchar(55) NOT NULL,
  PersonalID_O varchar(55) NOT NULL,
  StudentID varchar(55) NOT NULL,
  PRIMARY KEY (ContractID),
  KEY PersonalID_O (PersonalID_O),
  KEY StudentID (StudentID),
  FOREIGN KEY (PersonalID_O) REFERENCES office_staff (PersonalID_O),
  FOREIGN KEY (StudentID) REFERENCES student (StudentID)
);
```



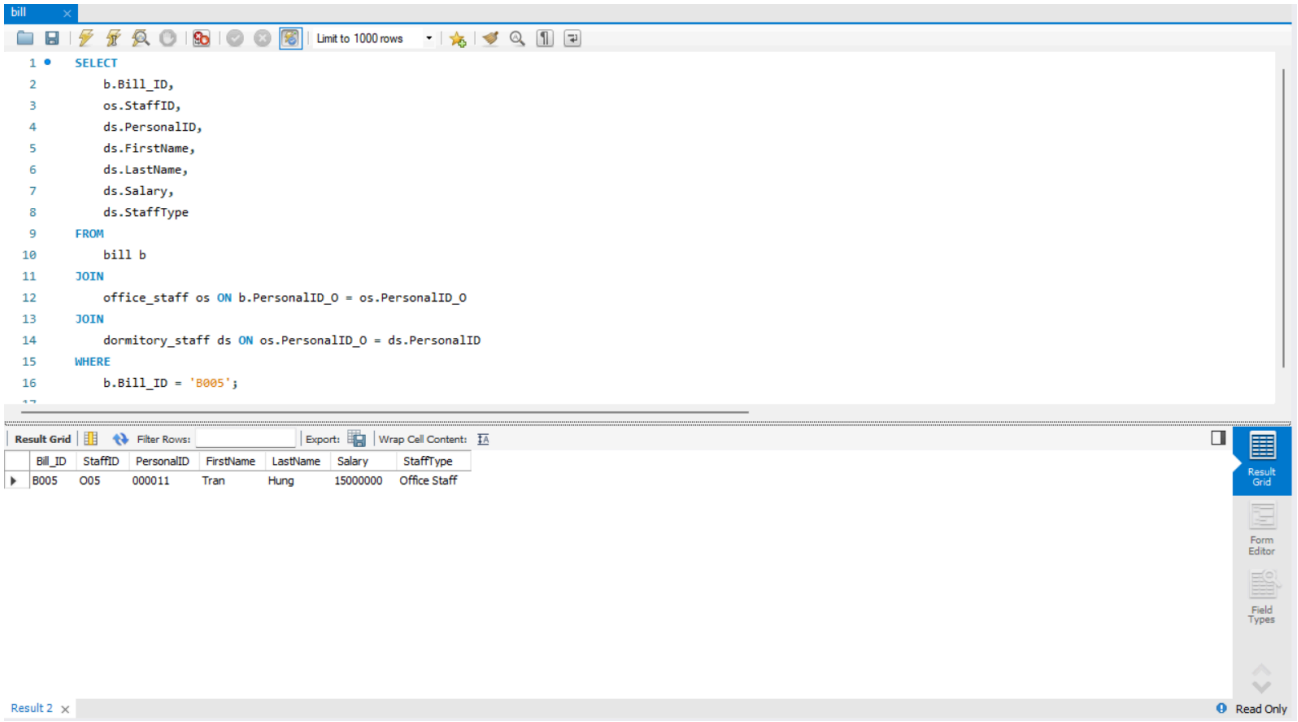
VI. Queries

1. Simple Query

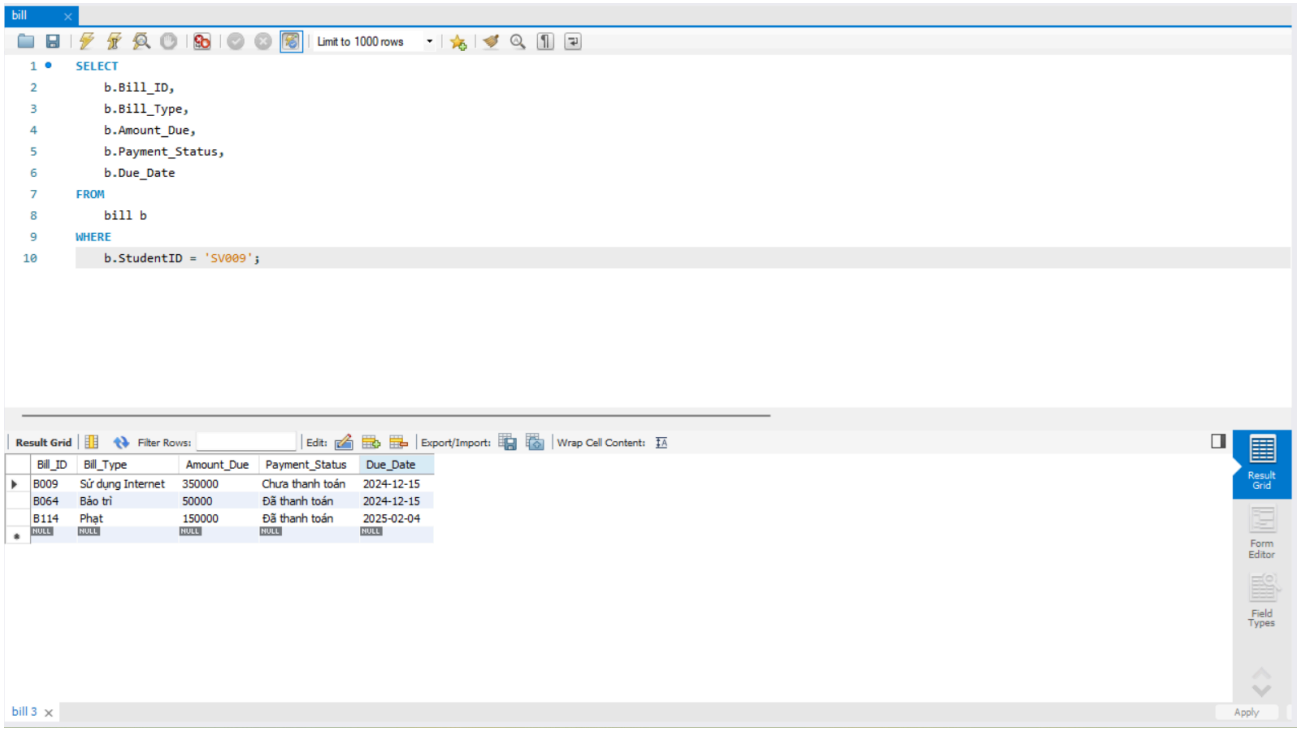
- **Bill:**
  - Query1: List all the bill that is not done and has due date > current date (the output table will have bill id, bill type, amount due, student ID, student LastName, student email address and student phone number)



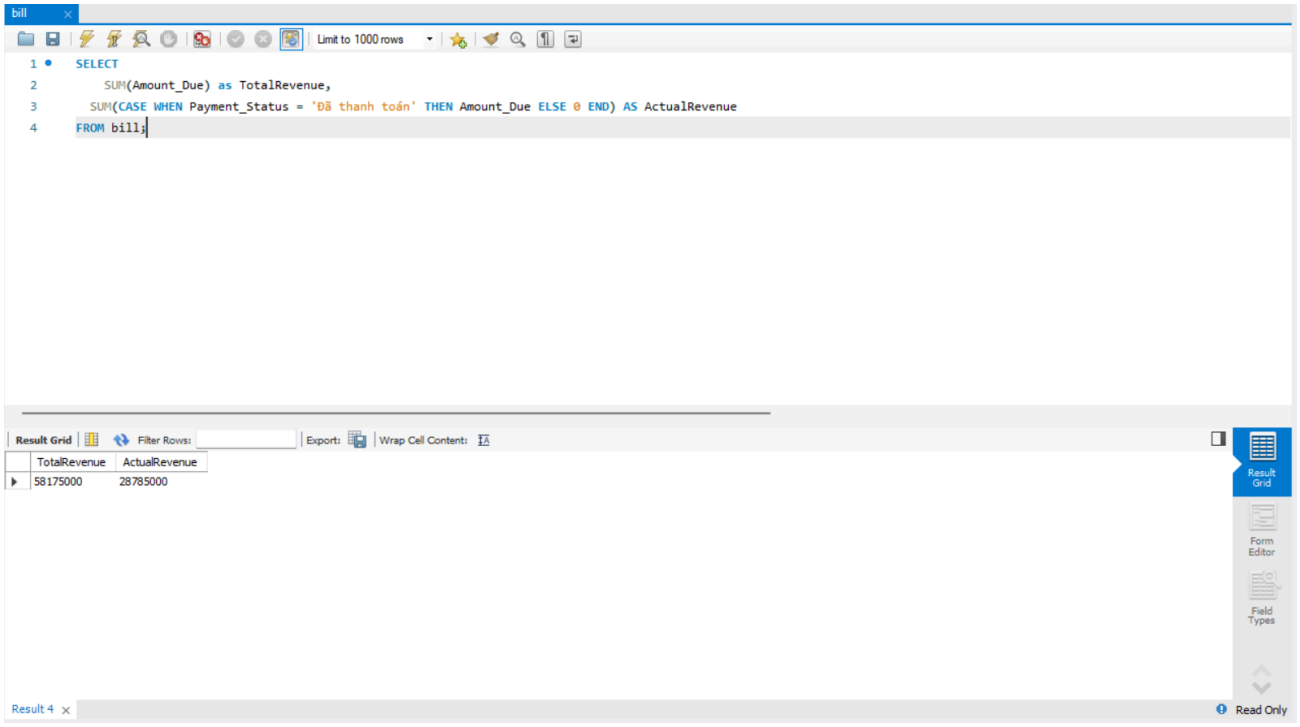
- Query 2: List all the information of the office staff that manage a bill (the output table will include billID and all the information of the office staff) example: give out the information of the staff manage bill B005.



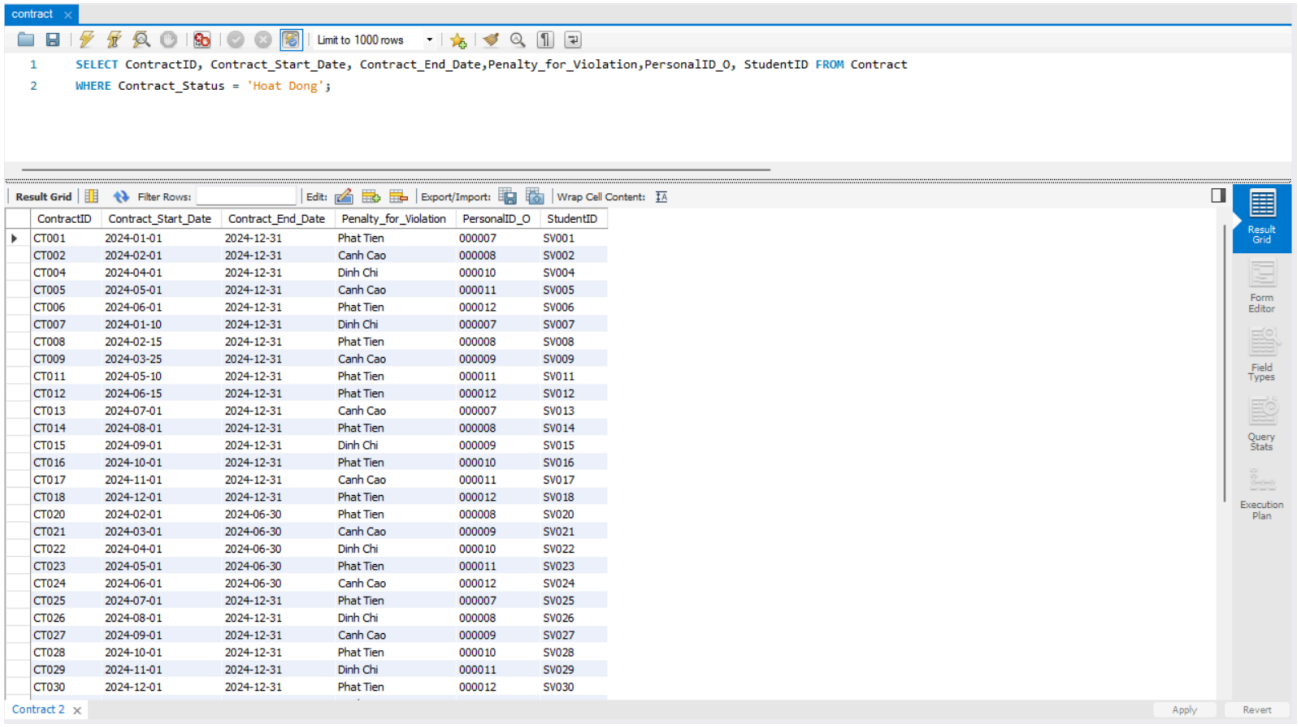
- Query 3: List all the bill that a student have(the output table will have all the bill detail except studentID): example: give out all the bill of student SV009



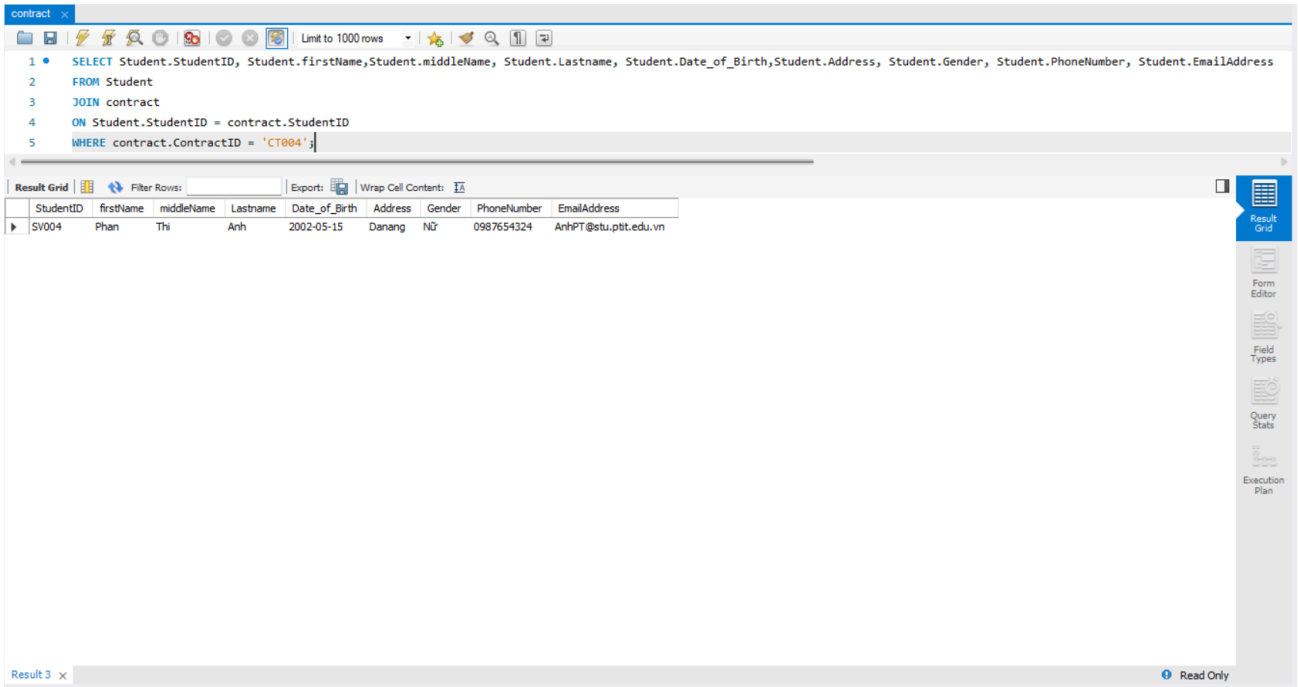
- Query 4: Calculate the total monthly revenue of the dormitory



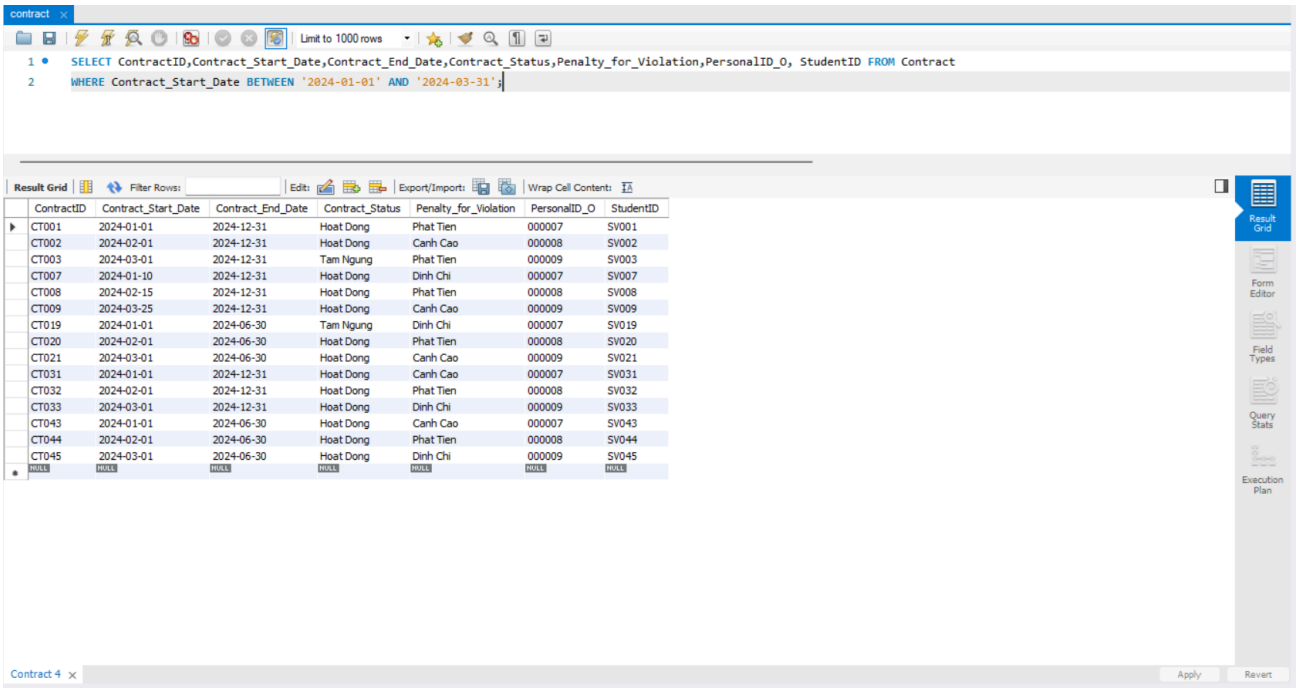
- **Contract:**
  - Query 1: List all active contracts(output: all contract information except contract_status)



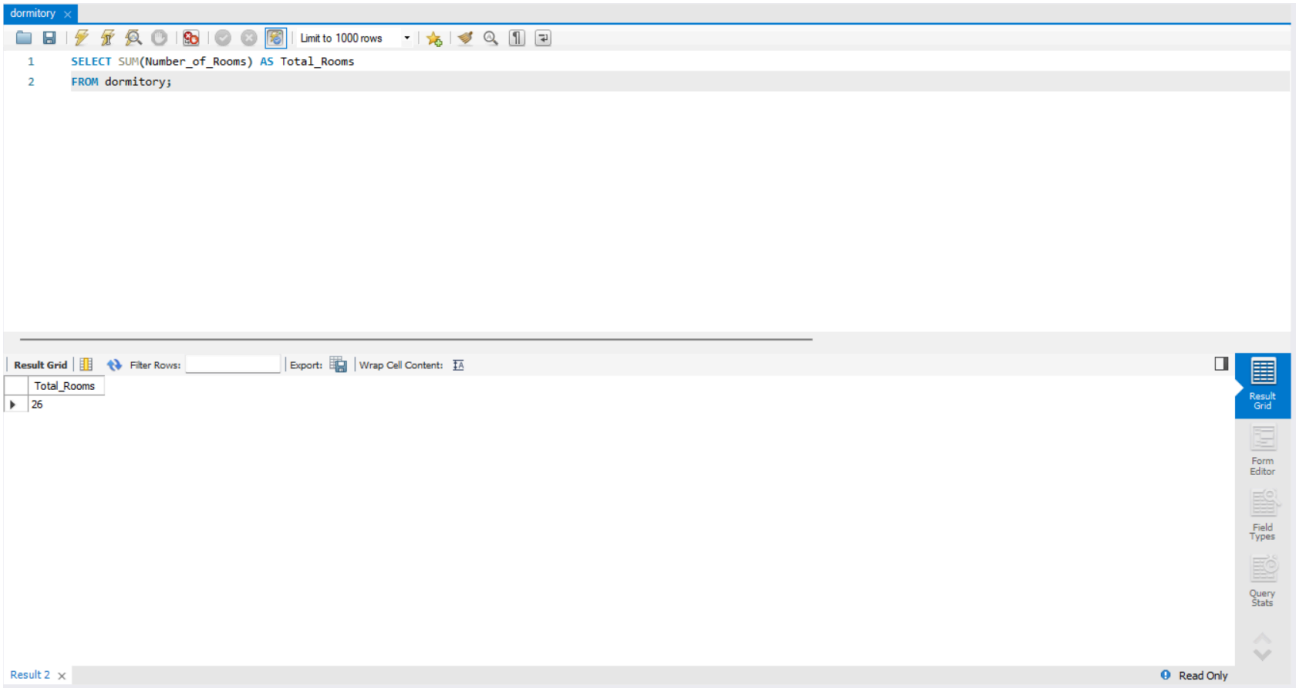
- Query 2: List all student information that has a contract(the output will consist of contractID and all student information): example: student that has contract CT014.



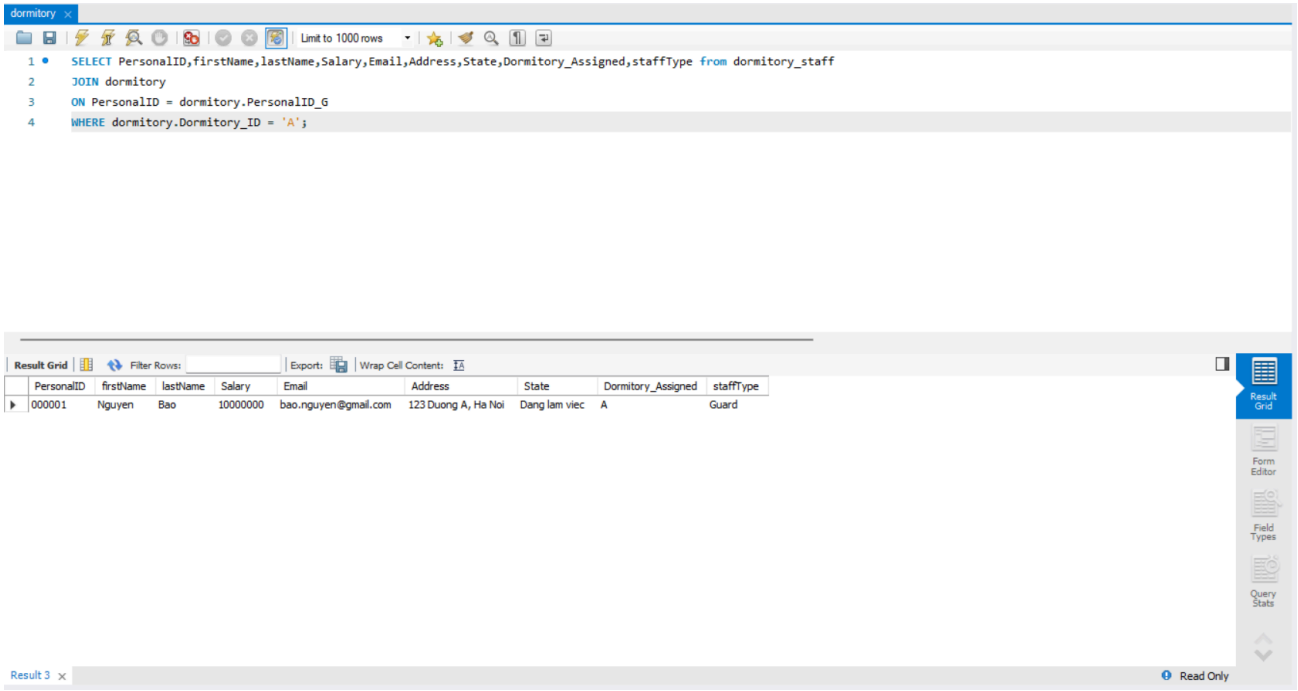
- Query 3: List all contracts with a start date that have a certain period(output: all contract information).



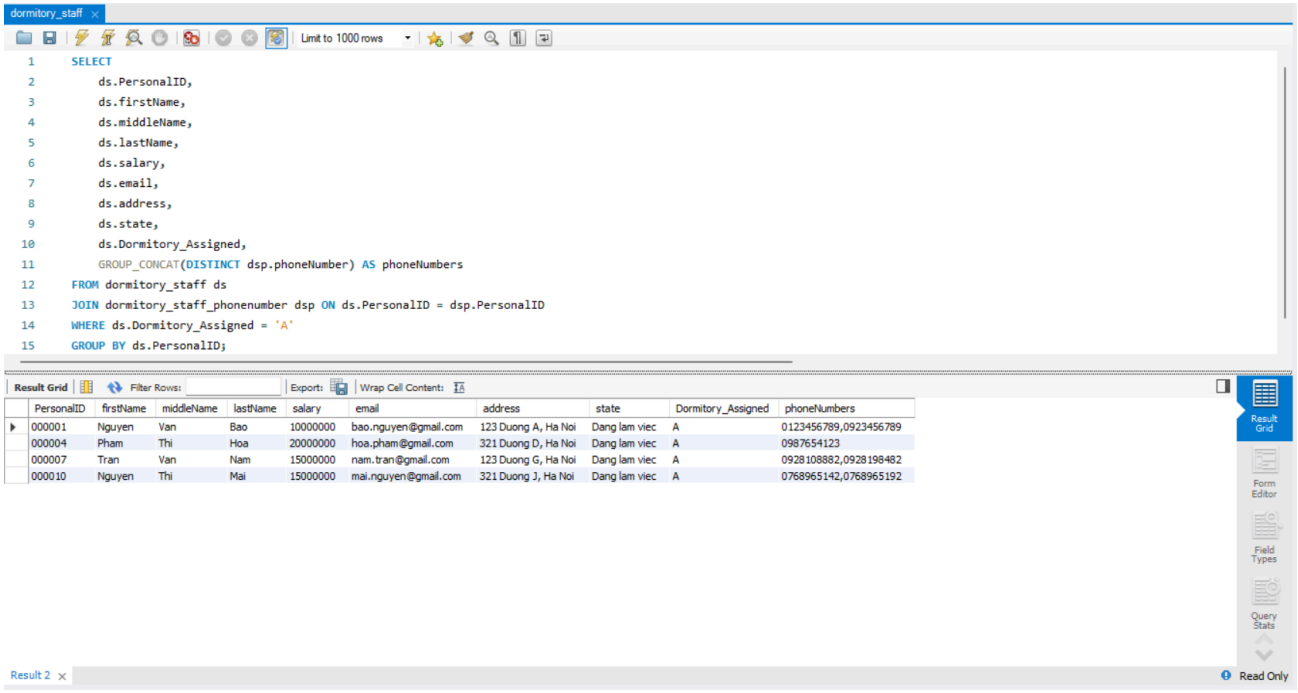
- **Dormitory:**
  - Query 1: Calculate the total number of rooms.



- Query 2: List all the information of the guard working in a dorm(Output: DormitoryID, all information of the guard. Eg: List all the information of the guards who protect A Dorm).



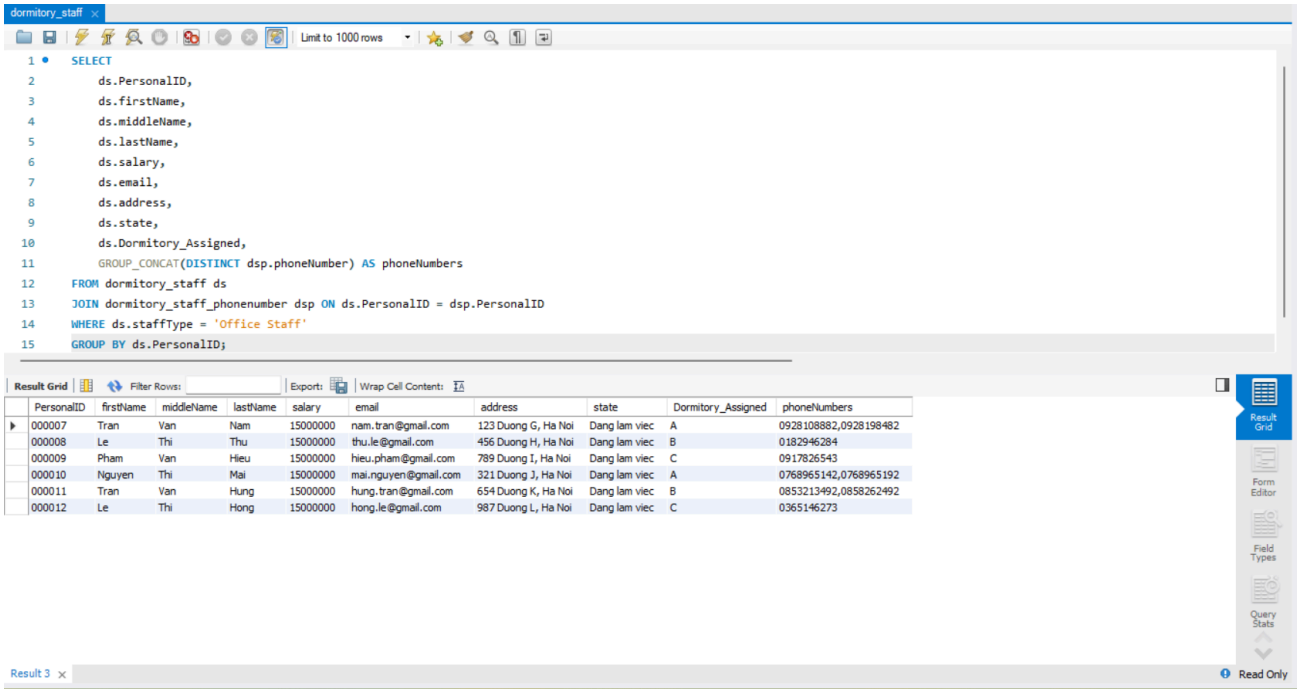
- **Dormitory Staff:**
  - Query 1: List all the information of staffs working in a certain dormitory (the output table will have all staff information except Dormitory_Assigned) example: list all the staffs working in Dormitory A



```
SELECT
    ds.PersonalID,
    ds.firstName,
    ds.middleName,
    ds.lastName,
    ds.salary,
    ds.email,
    ds.address,
    ds.state,
    ds.Dormitory_Assigned,
    GROUP_CONCAT(DISTINCT dsp.phoneNumber) AS phoneNumbers
FROM dormitory_staff ds
JOIN dormitory_staff_phonenumber dsp ON ds.PersonalID = dsp.PersonalID
WHERE ds.Dormitory_Assigned = 'A'
GROUP BY ds.PersonalID;
```

PersonalID	firstName	middleName	lastName	salary	email	address	state	Dormitory_Assigned	phoneNumbers
000001	Nguyen	Van	Bao	10000000	bao.nguyen@gmail.com	123 Duong A, Ha Noi	Dang lam viec	A	0123456789,0923456789
000004	Pham	Thi	Hoa	20000000	hoa.pham@gmail.com	321 Duong D, Ha Noi	Dang lam viec	A	0987654123
000007	Tran	Van	Nam	15000000	nam.tran@gmail.com	123 Duong G, Ha Noi	Dang lam viec	A	0928108882,0928198482
000010	Nguyen	Thi	Mai	15000000	mai.nguyen@gmail.com	321 Duong J, Ha Noi	Dang lam viec	A	0768965142,0768965192

- Query 2: List all the information of staffs have the same type (the output table will have all staff information except Staff_Type). Example: list all the office staff

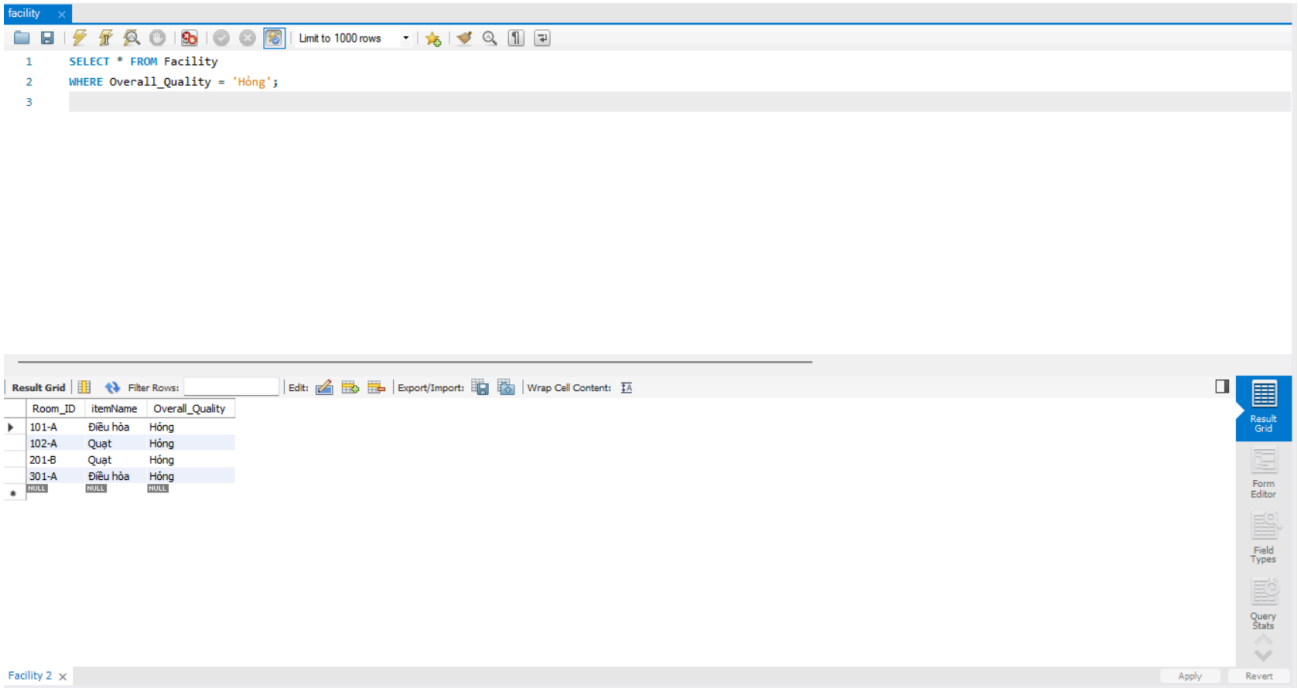


```
SELECT
    ds.PersonalID,
    ds.firstName,
    ds.middleName,
    ds.lastName,
    ds.salary,
    ds.email,
    ds.address,
    ds.state,
    ds.Dormitory_Assigned,
    GROUP_CONCAT(DISTINCT dsp.phoneNumber) AS phoneNumbers
FROM dormitory_staff ds
JOIN dormitory_staff_phonenumber dsp ON ds.PersonalID = dsp.PersonalID
WHERE ds.staffType = 'Office Staff'
GROUP BY ds.PersonalID;
```

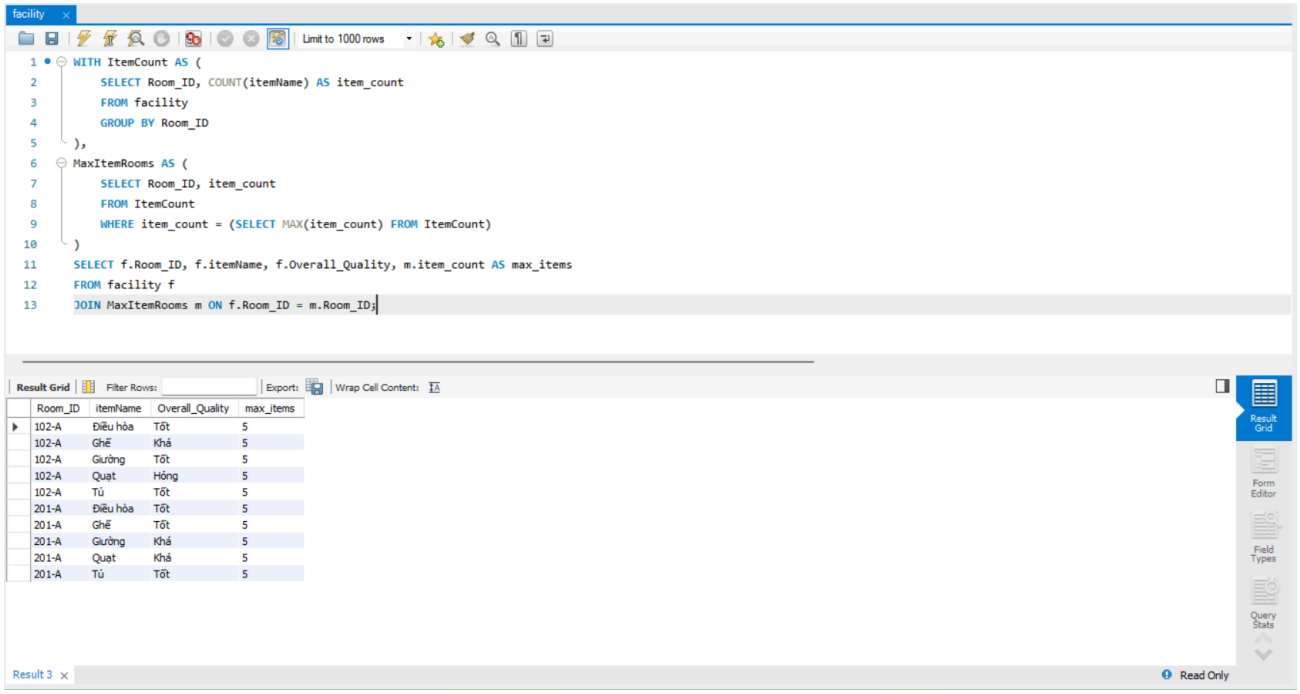
PersonalID	firstName	middleName	lastName	salary	email	address	state	Dormitory_Assigned	phoneNumbers
000007	Tran	Van	Nam	15000000	nam.tran@gmail.com	123 Duong G, Ha Noi	Dang lam viec	A	0928108882,0928198482
000008	Le	Thi	Thu	15000000	thu.le@gmail.com	456 Duong H, Ha Noi	Dang lam viec	B	0182946284
000009	Pham	Van	Hieu	15000000	hieu.pham@gmail.com	789 Duong I, Ha Noi	Dang lam viec	C	0917826543
000010	Nguyen	Thi	Mai	15000000	mai.nguyen@gmail.com	321 Duong J, Ha Noi	Dang lam viec	A	0768965142,0768965192
000011	Tran	Van	Hung	15000000	hung.tran@gmail.com	654 Duong K, Ha Noi	Dang lam viec	B	0853213492,0858262492
000012	Le	Thi	Hong	15000000	hong.le@gmail.com	987 Duong L, Ha Noi	Dang lam viec	C	0365146273

- **Facility:**

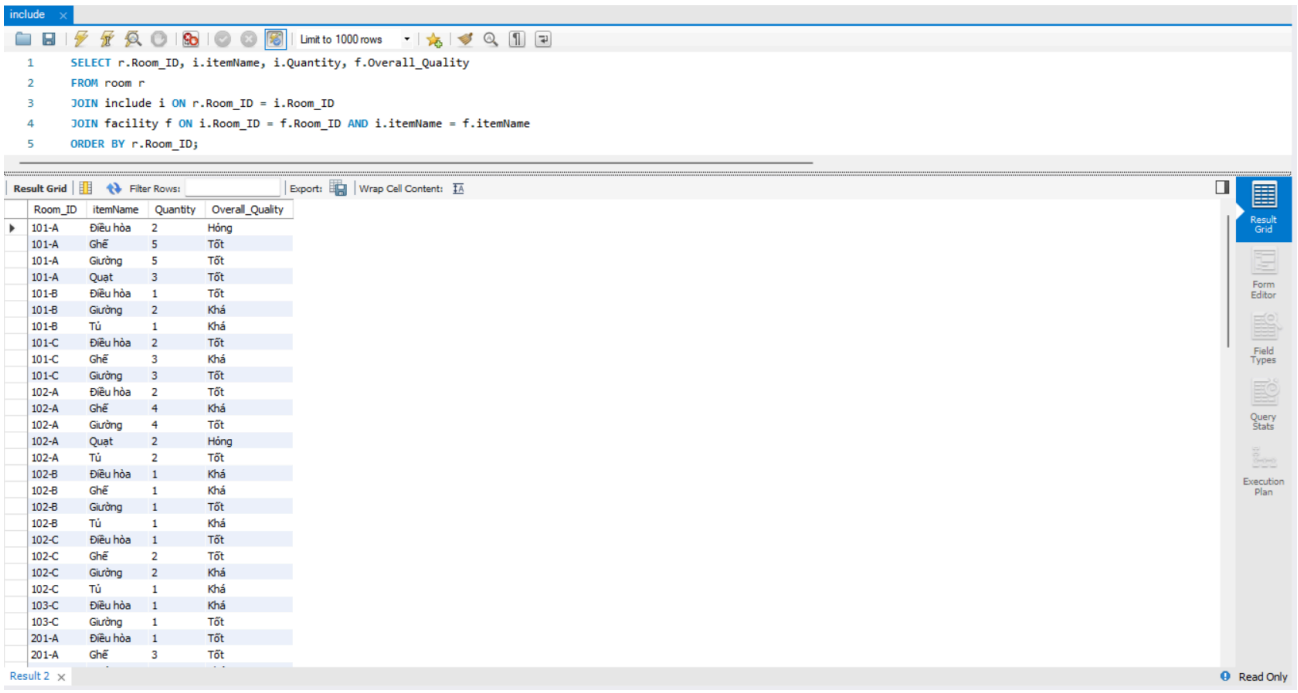
- Query 1: List all the items that have the overall_quality = ‘Hong’(output:all information in facility).



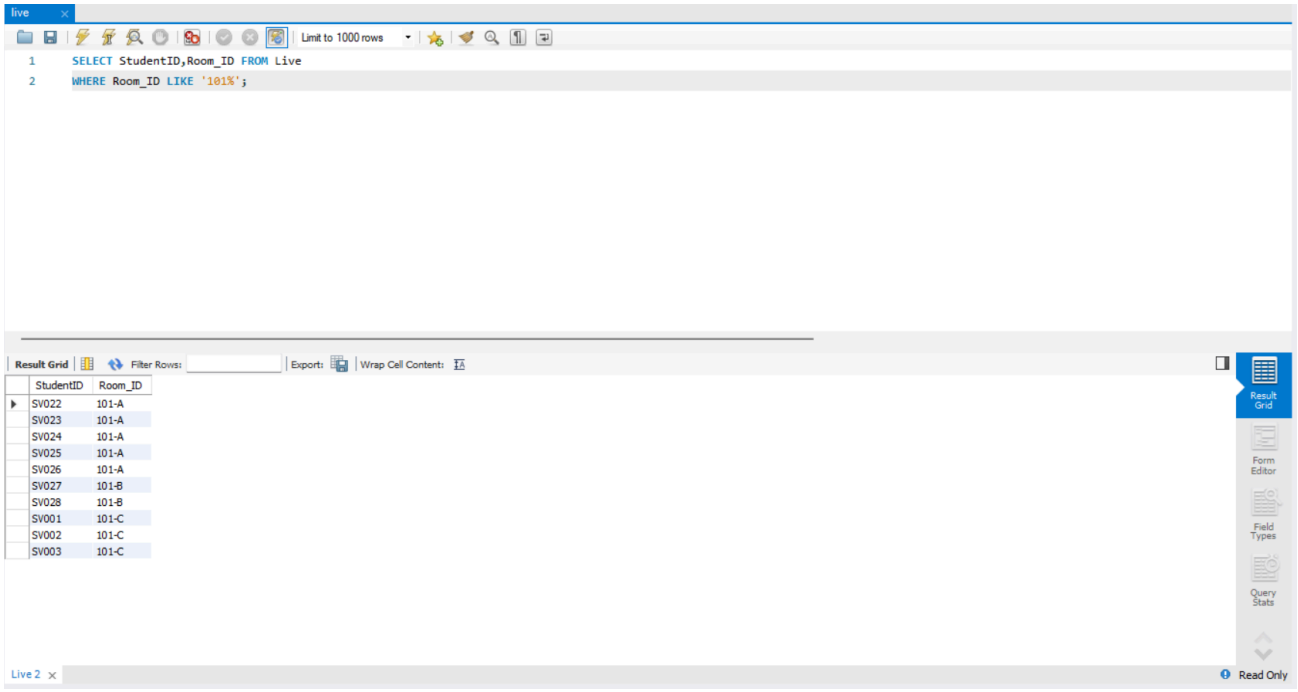
- Query 2: List all the rooms that have the maximum number of facilities (output: all information in facility).



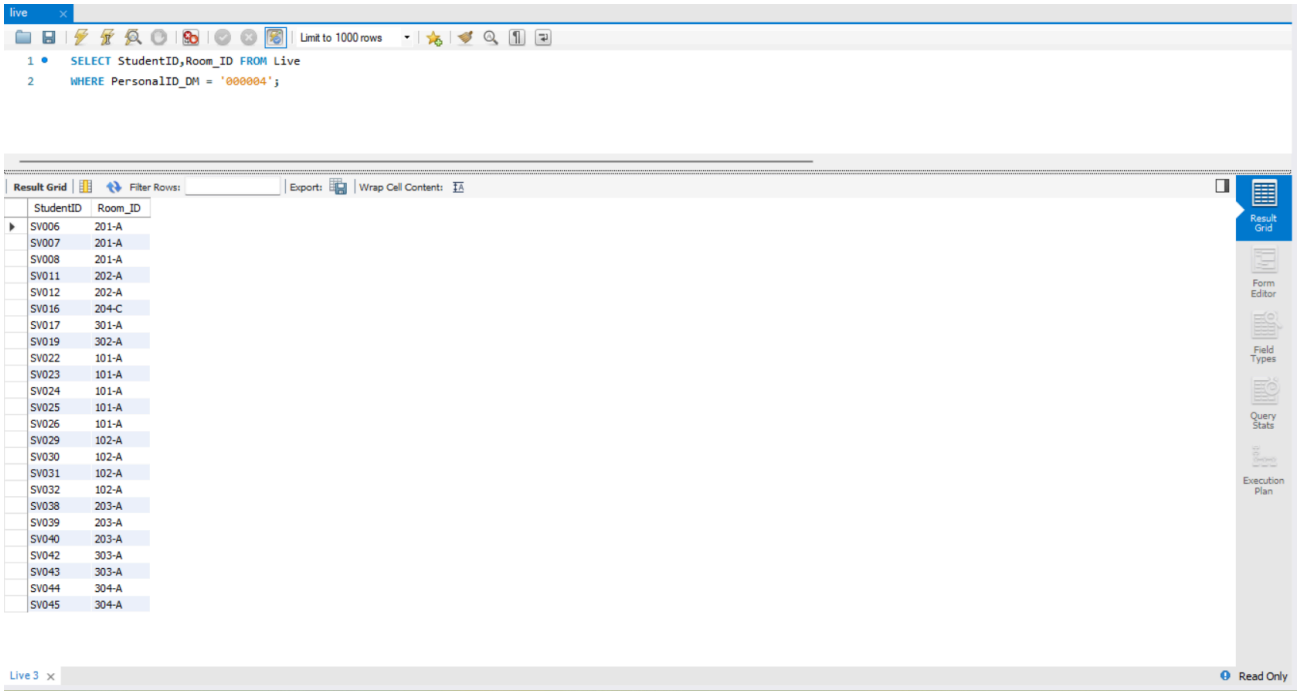
- **Include:**
    - Query 1: List all the facility in a room with its quantity(the output table will have roomID, itemName, quality and quantity)



- **Live:**
    - Query 1: List all the students in the room have the room name starts with a certain number(eg: 101) (output: studentID,roomID)

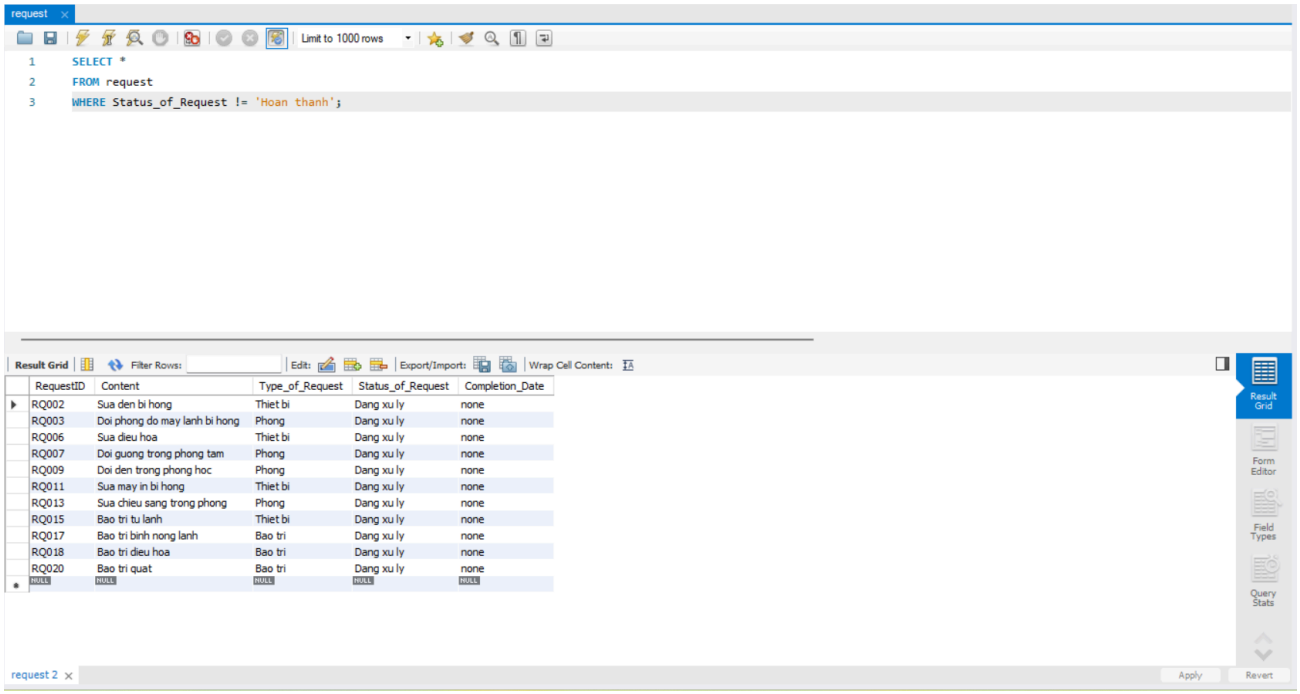


- Query 2: List all the students and rooms that have been managed by a certain manager(output: studentID, roomID).

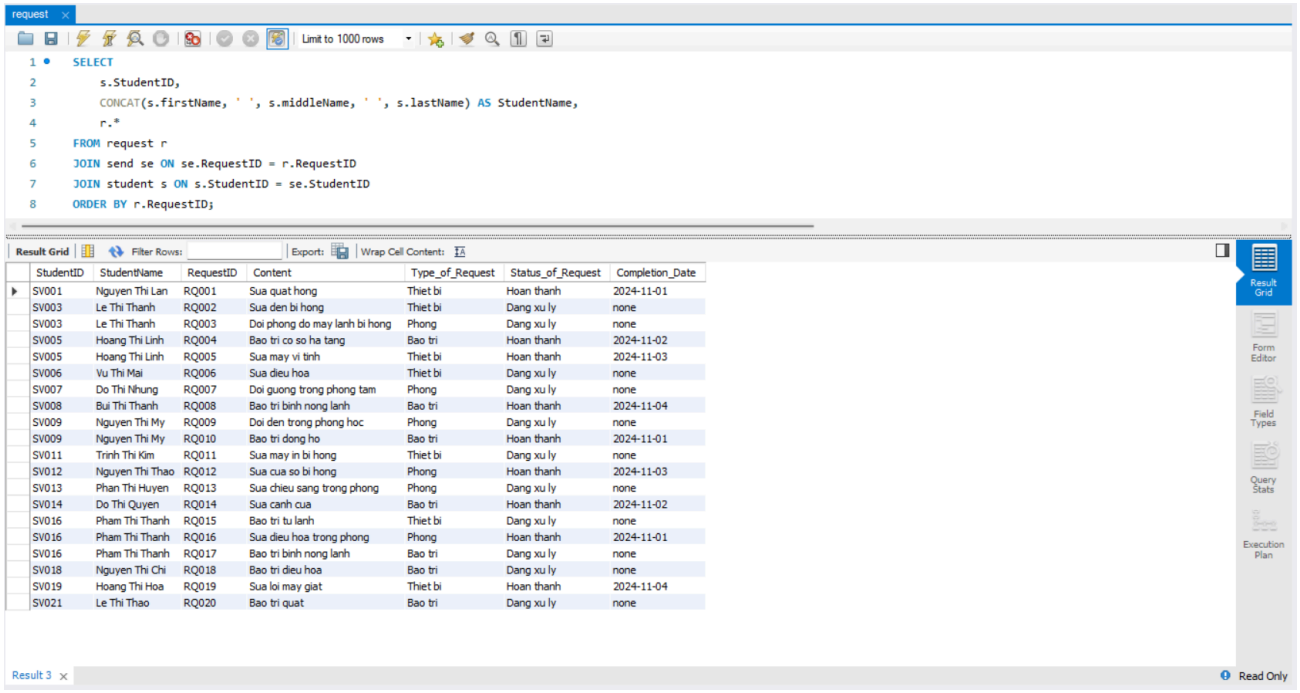


- **Request:**

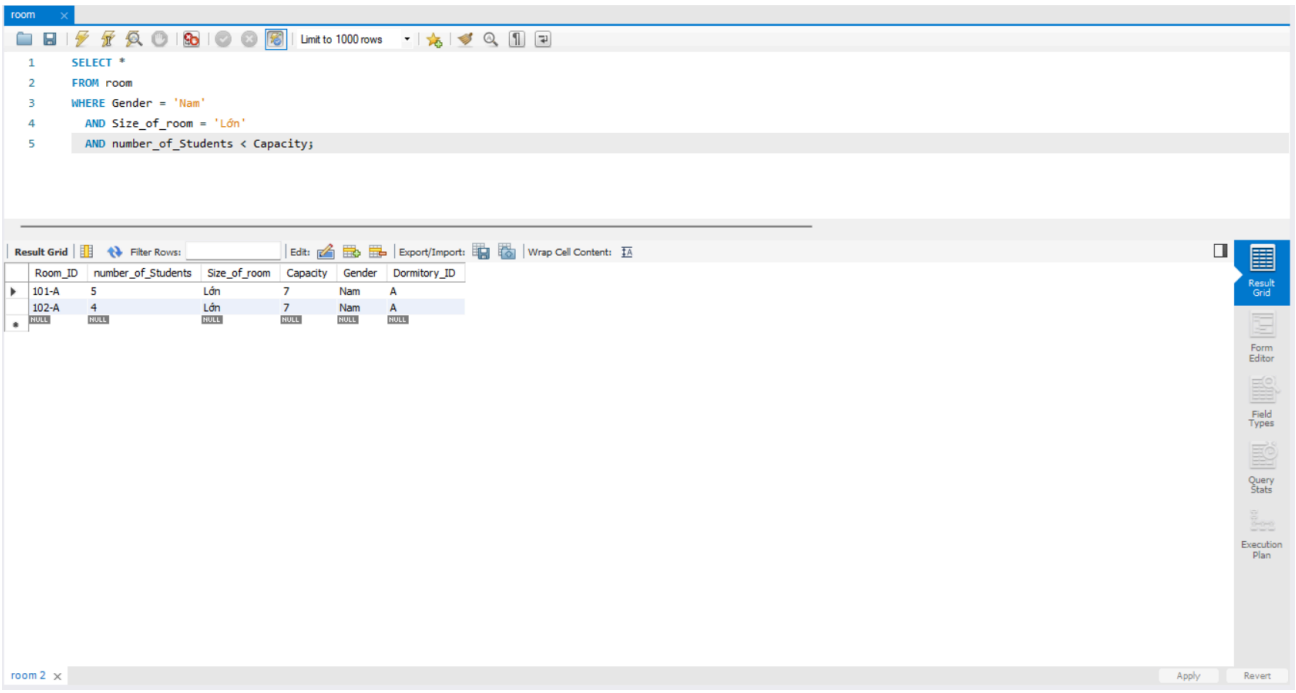
- Query 1: Find all request that is not completed(output will be all information of request)



- Query 2: List all the requests of a student(output will be all information of request plus student ID and student name)



- **Room**
    - List all rooms that only for men, is a big room and not full(output will be all information of room)



- **Send**
    - List all the information of a request with additional studentID, mangerID and send date.

send

```
SELECT r.*, s.StudentID, s.PersonalID_DM AS ManagerID, s.Send_Date
FROM Request r
JOIN Send s ON r.RequestID = s.RequestID;
```

Result Grid

RequestID	Content	Type_of_Request	Status_of_Request	Completion_Date	StudentID	ManagerID	Send_Date
RQ001	Sua quat hong	Thiet bi	Hoan thanh	2024-11-01	SV001	000004	2024-09-22
RQ002	Sua den bi hong	Thiet bi	Dang xu ly	none	SV003	000005	2024-09-23
RQ003	Doi phong do may lanh bi hong	Phong	Dang xu ly	none	SV003	000006	2024-09-24
RQ004	Bao tri co so ha tang	Bao tri	Hoan thanh	2024-11-02	SV005	000004	2024-09-25
RQ005	Sua may vi tinh	Thiet bi	Hoan thanh	2024-11-03	SV005	000005	2024-09-26
RQ006	Sua dieu hoa	Thiet bi	Dang xu ly	none	SV006	000006	2024-09-27
RQ007	Doi guong trong phong tam	Phong	Dang xu ly	none	SV007	000004	2024-09-28
RQ008	Bao tri binh nong lanh	Bao tri	Hoan thanh	2024-11-04	SV008	000005	2024-09-29
RQ009	Doi den trong phong hoc	Phong	Dang xu ly	none	SV009	000006	2024-09-30
RQ010	Bao tri dong ho	Bao tri	Hoan thanh	2024-11-01	SV009	000004	2024-10-01
RQ011	Sua may in bi hong	Thiet bi	Dang xu ly	none	SV011	000005	2024-10-02
RQ012	Sua cua so bi hong	Phong	Hoan thanh	2024-11-03	SV012	000006	2024-10-03
RQ013	Sua chieu sang trong phong	Phong	Dang xu ly	none	SV013	000004	2024-10-04
RQ014	Sua canh cua	Bao tri	Hoan thanh	2024-11-02	SV014	000005	2024-10-05
RQ015	Bao tri tu lanh	Thiet bi	Dang xu ly	none	SV016	000006	2024-10-06
RQ016	Sua dieu hoa trong phong	Phong	Hoan thanh	2024-11-01	SV016	000004	2024-10-07
RQ017	Bao tri binh nong lanh	Bao tri	Dang xu ly	none	SV016	000005	2024-10-08
RQ018	Bao tri dieu hoa	Bao tri	Dang xu ly	none	SV018	000006	2024-10-09
RQ019	Sua loi may giat	Thiet bi	Hoan thanh	2024-11-04	SV019	000004	2024-10-10
RQ020	Bao tri quat	Bao tri	Dang xu ly	none	SV021	000005	2024-10-11

Result 2 — Read Only

- **Student**
  - List all the student that are female and has their last name start from ‘N’ and come from Hanoi

student

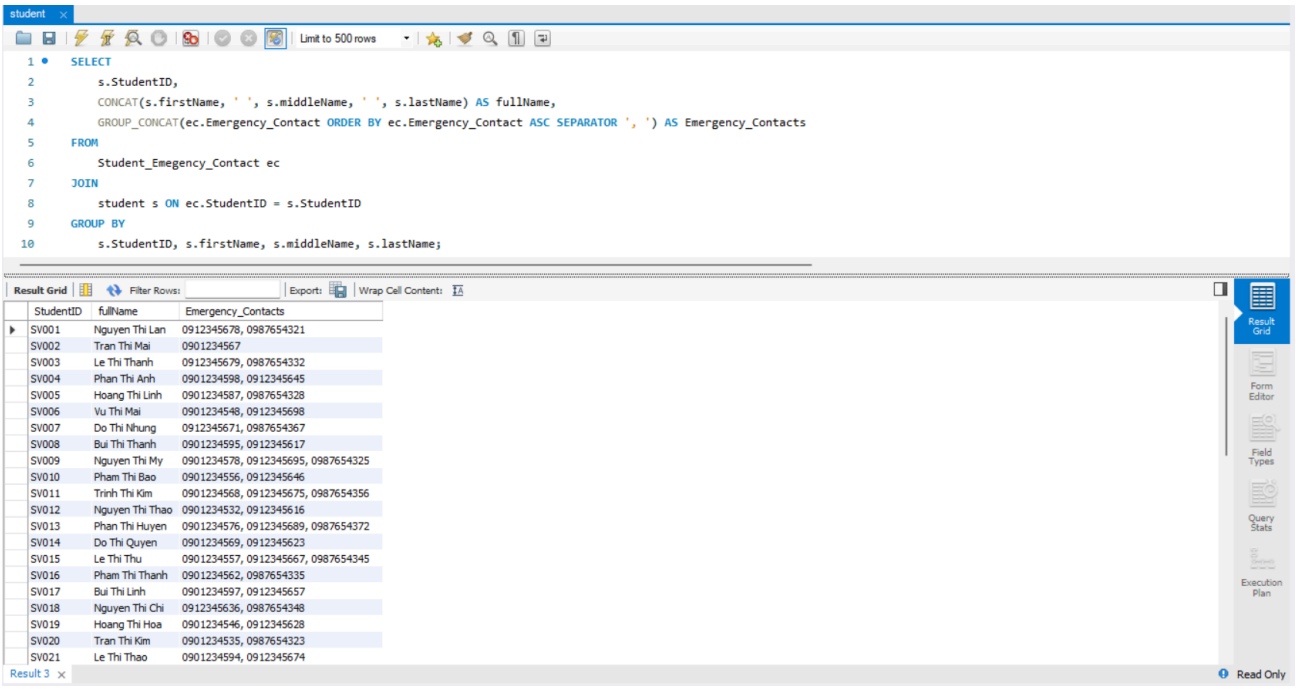
```
SELECT * FROM student
WHERE Gender = 'NỮ' AND Address = 'Hanoi' AND lastName >= 'N';
```

Result Grid

StudentID	firstName	middleName	lastName	Date_of_Birth	Address	Gender	phoneNumber	EmailAddress
SV007	Do	Thi	Nhung	2000-04-14	Hanoi	NỮ	0987654327	NhungDT@stu.ptit.edu.vn
SV012	Nguyen	Thi	Thao	2001-03-10	Hanoi	NỮ	0987654332	ThaoNT@stu.ptit.edu.vn

student 2 — Apply — Revert

  - List all of Student emergency contact (output have studentID, student name and their emergency contact)

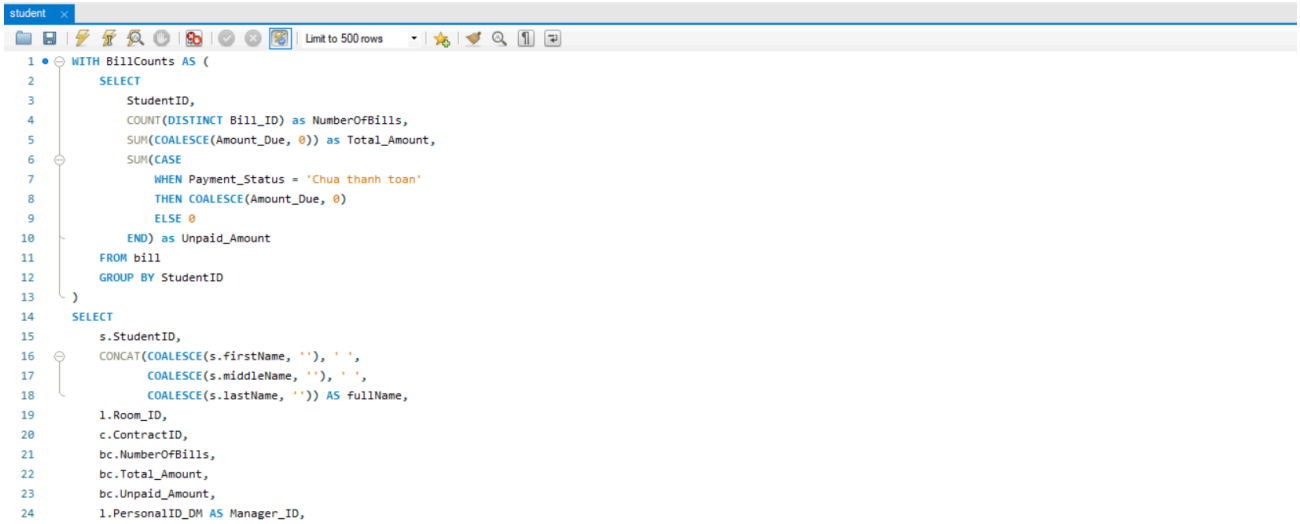


```sql
SELECT
    s.StudentID,
    CONCAT(s.firstName, ' ', s.middleName, ' ', s.lastName) AS fullName,
    GROUP_CONCAT(ec.Emergency_Contact ORDER BY ec.Emergency_Contact ASC SEPARATOR ', ') AS Emergency_Contacts
FROM
    Student_Emegency_Contact ec
JOIN
    student s ON ec.StudentID = s.StudentID
GROUP BY
    s.StudentID, s.firstName, s.middleName, s.lastName;
```

StudentID	fullName	Emergency_Contacts
SV001	Nguyen Thi Lan	0912345678, 0987654321
SV002	Tran Thi Mai	0901234567
SV003	Le Thi Thanh	0912345679, 0987654332
SV004	Phan Thi Anh	0901234598, 0912345645
SV005	Hoang Thi Linh	0901234587, 0987654328
SV006	Vu Thi Mai	0901234548, 0912345698
SV007	Do Thi Nhung	0912345671, 0987654367
SV008	Bui Thi Thanh	0901234595, 0912345617
SV009	Nguyen Thi My	0901234578, 0912345695, 0987654325
SV010	Pham Thi Bao	0901234556, 0912345646
SV011	Trinh Thi Kim	0901234568, 0912345675, 0987654356
SV012	Nguyen Thi Thao	0901234532, 0912345616
SV013	Phan Thi Huyen	0901234576, 0912345689, 0987654372
SV014	Do Thi Quyen	0901234569, 0912345623
SV015	Le Thi Thu	0901234557, 0912345667, 0987654345
SV016	Pham Thi Thanh	0901234562, 0987654335
SV017	Bui Thi Linh	0901234597, 0912345657
SV018	Nguyen Thi Chi	0912345636, 0987654348
SV019	Hoang Thi Hoa	0901234546, 0912345628
SV020	Tran Thi Kim	0901234535, 0987654323
SV021	Le Thi Thao	0901234594, 0912345674

2. Complicated Query

- List all students (student id and student name) and their roomID, contract id, total amount needed to pay, unpaid bill amount, the manager that manages their room (manager ID), if they have a request or not (Yes or No).



```sql
WITH BillCounts AS (
    SELECT
        StudentID,
        COUNT(DISTINCT Bill_ID) as NumberOfBills,
        SUM(COALESCE(Amount_Due, 0)) as Total_Amount,
        SUM(CASE
            WHEN Payment_Status = 'Chua thanh toan'
            THEN COALESCE(Amount_Due, 0)
            ELSE 0
        END) as Unpaid_Amount
    FROM bill
    GROUP BY StudentID
)
SELECT
    s.StudentID,
    CONCAT(COALESCE(s.firstName, ''), ' ',
           COALESCE(s.middleName, ''), ' ',
           COALESCE(s.lastName, '')) AS fullName,
    l.Room_ID,
    c.ContractID,
    bc.NumberOfBills,
    bc.Total_Amount,
    bc.Unpaid_Amount,
    l.PersonalID_DM AS Manager_ID,
```

```
25          (SELECT COUNT(*) FROM send se2 WHERE se2.StudentID = s.StudentID) AS NumberOfRequests
26      FROM student s
27      LEFT JOIN BillCounts bc ON bc.StudentID = s.StudentID
28      JOIN contract c ON s.StudentID = c.StudentID
29      JOIN live l ON l.StudentID = s.StudentID
30      LEFT JOIN send se ON se.StudentID = s.StudentID
31      GROUP BY
32          s.StudentID,
33          s.firstName,
34          s.middleName,
35          s.lastName,
36          l.Room_ID,
37          c.ContractID,
38          l.PersonalID_DM,
39          bc.Total_Amount,
40          bc.Unpaid_Amount,
41          bc.NumberOfBills
42      ORDER BY s.StudentID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

StudentID	fullName	Room_ID	ContractID	NumberOfBills	Total_Amount	Unpaid_Amount	Manager_ID	NumberOfRequests
SV001	Nguyen Thi Lan	101-C	CT001	4	1015000	620000	000006	1
SV002	Tran Thi Mai	101-C	CT002	4	2090000	0	000006	0
SV003	Le Thi Thanh	101-C	CT003	4	920000	780000	000006	2
SV004	Phan Thi Anh	102-B	CT004	4	780000	70000	000005	0
SV005	Hoang Thi Linh	103-C	CT005	4	1230000	180000	000006	2
SV006	Vu Thi Mai	201-A	CT006	3	2560000	2320000	000004	1
SV007	Do Thi Nhung	201-A	CT007	3	835000	685000	000004	1
SV008	Bui Thi Thanh	201-A	CT008	3	1270000	0	000004	1
SV009	Nguyen Thi My	201-B	CT009	3	550000	350000	000005	2
SV010	Pham Thi Bao	201-B	CT010	3	1440000	1240000	000005	0
SV011	Trinh Thi Kim	202-A	CT011	3	410000	290000	000004	1
SV012	Nguyen Thi Thao	202-A	CT012	3	2360000	340000	000004	1
SV013	Phan Thi Huyen	202-C	CT013	3	810000	630000	000006	1

Result 4 | Read Only

- List all dormitory staff: Manager and all the student and room that they manage(the output table will have managerID, managerName, staffDormitory, staffPhoneNumber, Manage(studentID and roomID), studentName, studentPhoneNumber and studentEmergencyContact)

student

Limit to 500 rows

```
1   SELECT
2       ds.PersonalID AS managerID,
3       CONCAT(ds.firstname,' ',ds.middleName,' ',ds.lastName) AS managerName,
4       ds.Dormitory_Assigned AS staffDormitory,
5       (SELECT GROUP_CONCAT(DISTINCT phoneNumber)
6        FROM dormitory_staff_phonenumber
7        WHERE PersonalID = ds.PersonalID) AS staffPhoneNumbers,
8       CONCAT(st.StudentID,', ',r.Room_ID) AS Manage,
9       CONCAT(st.firstname,' ',st.middleName,' ',st.lastName) AS studentName,
10      st.phoneNumber AS studentPhoneNumber,
11      (SELECT GROUP_CONCAT(DISTINCT Emergency_Contact)
12       FROM student_emegency_contact
13       WHERE StudentID = st.StudentID) AS studentEmergencyContact
14  FROM dormitory_staff ds
15  JOIN live l ON l.PersonalID_DM = ds.PersonalID
16  JOIN room r ON r.Room_ID = l.Room_ID
17  JOIN student st ON st.StudentID = l.StudentID
18  WHERE ds.staffType = 'Manager'
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

managerID	managerName	staffDormitory	staffPhoneNumbers	Manage	studentName	studentPhoneNumber	studentEmergencyContact
000004	Pham Thi Hoa	A	0987654123	SV006, 201-A	Vu Thi Mai	0987654326	0901234548,0912345698
000004	Pham Thi Hoa	A	0987654123	SV007, 201-A	Do Thi Nhung	0987654327	0912345671,0987654367
000004	Pham Thi Hoa	A	0987654123	SV008, 201-A	Bui Thi Thanh	0987654328	0901234595,0912345617
000004	Pham Thi Hoa	A	0987654123	SV011, 202-A	Trinh Thi Kim	0987654331	0901234568,0912345675,0987654356
000004	Pham Thi Hoa	A	0987654123	SV012, 202-A	Nguyen Thi Thao	0987654332	0901234532,0912345616
000004	Pham Thi Hoa	A	0987654123	SV016, 204-C	Pham Thi Thanh	0987654336	0901234562,0987654335
000004	Pham Thi Hoa	A	0987654123	SV017, 301-A	Bui Thi Linh	0987654337	0901234597,0912345657
000004	Pham Thi Hoa	A	0987654123	SV019, 302-A	Hoang Thi Hoa	0987654339	0901234546,0912345628
000004	Pham Thi Hoa	A	0987654123	SV022, 101-A	Nguyen Minh Hoang	0987654326	0912345618,0987654326
000004	Pham Thi Hoa	A	0987654123	SV023, 101-A	Pham Quang Duy	0987654327	0901234547,0912345687
000004	Pham Thi Hoa	A	0987654123	SV024, 101-A	Tran Thanh Son	0987654328	0901234579,0912345632,0987654368
000004	Pham Thi Hoa	A	0987654123	SV025, 101-A	Nguyen Quang Khai	0987654329	0901234582,0912345697,0987654333
000004	Pham Thi Hoa	A	0987654123	SV026, 101-A	Le Van Duy	0987654330	0901234593,0912345672

Result 5 | Read Only

- Check a room occupancy and if a room has sufficient cooling facility or not: example: list all the rooms with a column call status, if a room don't have a air conditioner and if in those rooms, if it doesn't have a fan or the number of fans smaller than the number of students in that room, add in column cooling_status ‘Thiếu’ , the rest are ‘Đủ’ (the output data will have all the information of the room, plus cooling_status column and occupancy column)