

**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	4
1.	Real-life Scenario	4
2.	Database Applications Description:	5
3.	Applications to be Supported:	6
II.	Conceptual Design	6
1.	Entity set	6
2.	Relationships:	7
3.	ERD Diagram	8
III.	Logical Design	8
IV.	Normalization	9
V.	Database Implementation	14
1.	Table Student	14
2.	Table Student_Emergency_Contact	15
3.	Table Dormitory_Staff	16
4.	Table Dormitory_Staff_PhoneNumber	17
5.	Table Dormitory Manager	18
6.	Table Guard	19
7.	Table Office Staff	20
8.	Table Dormitory	21
9.	Table Room	22
10.	Table Bill	23
11.	Table Request	24
12.	Table Facility	25
13.	Table Live	26
14.	Table Send	27
15.	Table Include	28
16.	Table Contract	29
VI.	Queries	30
1.	Simple Query	30
2.	Complicated Query	43

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

- o 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)

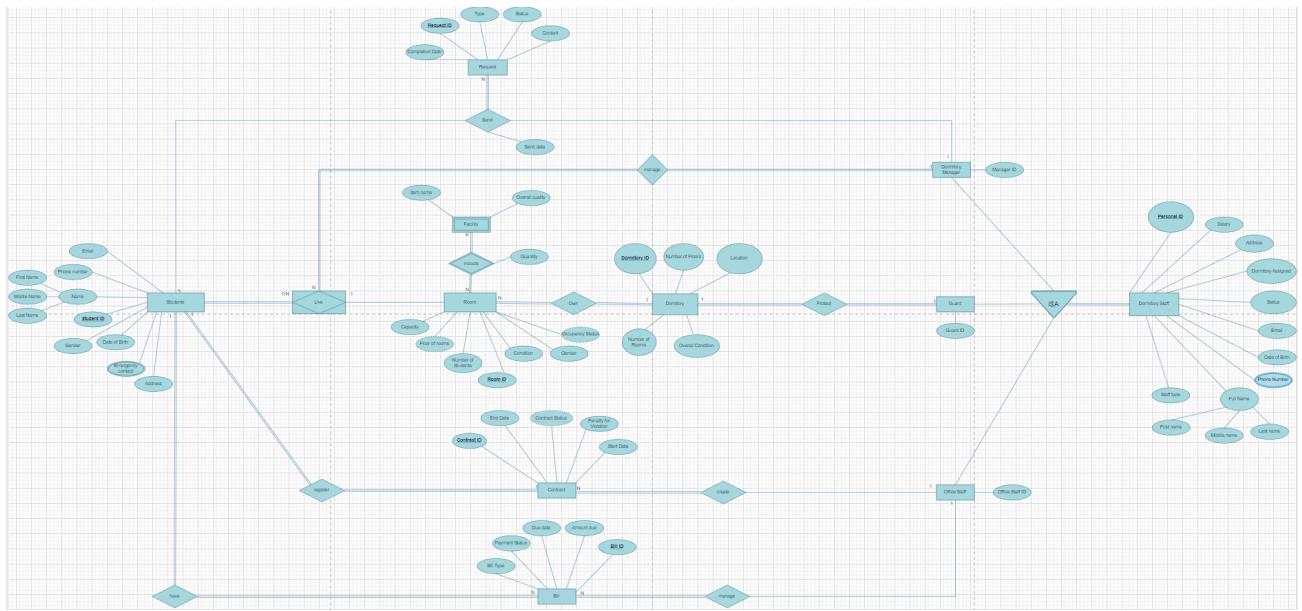
- Dormitory Manager Department staff (**Personal ID**, first name, middle name, last name, salary, phone number, email address, address and status, assigned dormitory, staff type). Each of the staff will be split to either three types below:
 - o Guard (**Guard id**)
 - o Office (**Office id**)
 - o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

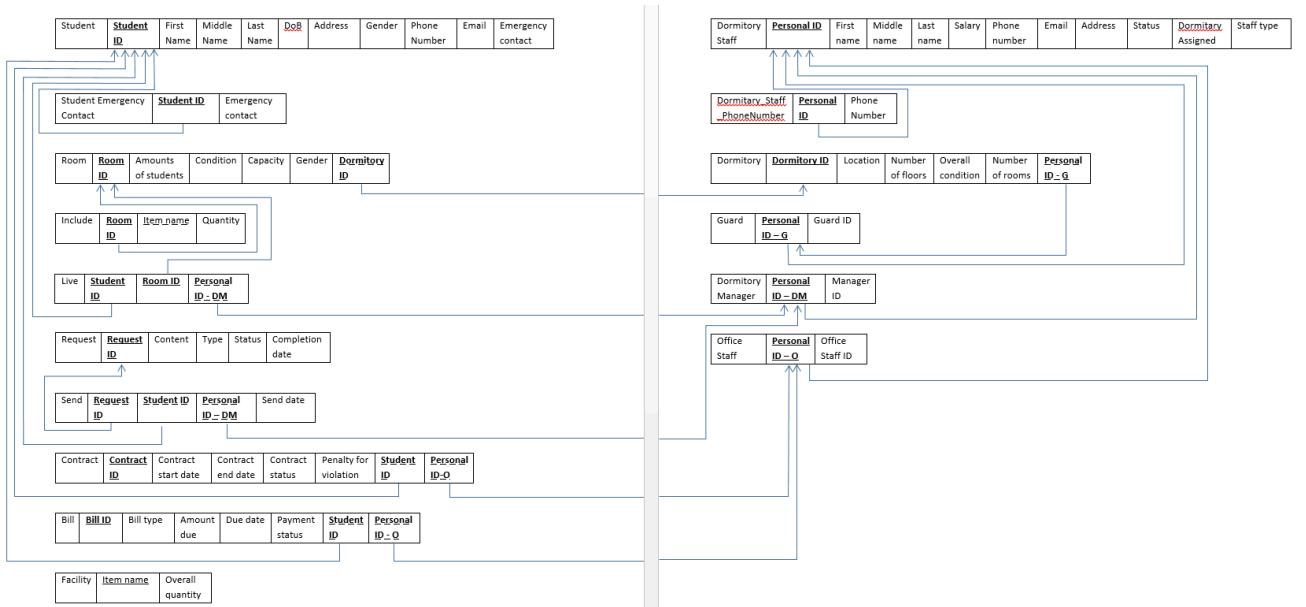
Request (**Request ID**, content, type, status, completion date)

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_Emergency_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 ID → 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

```
1   CREATE TABLE student (
2       StudentID varchar(55) NOT NULL PRIMARY KEY,
3       firstName varchar(55) NOT NULL,
4       middleName varchar(55) NOT NULL,
5       lastName varchar(55) NOT NULL,
6       Date_of_Birth DATE NOT NULL,
7       Address varchar(55) NOT NULL,
8       Gender varchar(55) NOT NULL,
9       phoneNumber varchar(55) NOT NULL,
10      EmailAddress varchar(55) NOT NULL
11  );
```

student x

1 • SELECT * FROM university_dormitory.student;

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

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	DuyQ@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	KhaiQ@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

student 1 x

Apply | Revert

2. Table Student_Emergency_Contact

```

1   CREATE TABLE university_dormitory.Student_Emergency_Contact(
2       Emergency_Contact VARCHAR(55) NOT NULL,
3       StudentID VARCHAR(55) NOT NULL,
4       FOREIGN KEY (StudentID) REFERENCES student(StudentID)
5   );

```

The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query editor contains the SQL command:

```
1 • SELECT * FROM university_dormitory.student_emergency_contact;
```

The results grid displays the data from the table, which has two columns: Emergency_Contact and StudentID. The data consists of 15 rows, each containing a unique emergency contact number and a student ID. The student IDs are all 'SV001' through 'SV009'. The results grid includes various toolbar icons and a sidebar with tabs for Result Grid, Form Editor, Field Types, and Query Stats.

Emergency_Contact	StudentID
0912345678	SV001
0987654321	SV001
0901234567	SV002
0912345679	SV003
0987654332	SV003
0901234598	SV004
0912345645	SV004
0987654328	SV005
0901234587	SV005
0912345698	SV006
0901234548	SV006
0912345671	SV007
0987654367	SV007
0901234595	SV008
0912345617	SV008
0901234578	SV009
0912345600	SV009

3. Table Dormitory_Staff

```

1   CREATE TABLE university_dormitory.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  );

```

The screenshot shows the MySQL Workbench interface with a query window titled "dormitory_staff". The query is:

```
1 • SELECT * FROM university_dormitory.dormitory_staff;
```

The results are displayed in a "Result Grid" table:

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
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

4. Table Dormitory_Staff_PhoneNumber

```

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

```

The screenshot shows a MySQL Workbench interface with a query editor and a result grid. The query editor contains the SQL command:

```
1 • | SELECT * FROM university_dormitory.dormitory_staff_phonenumber;
```

The result grid displays the following data:

PersonalID	phoneNumber
000001	0123456789
000001	0923456789
000002	0987654321
000002	0787654321
000003	0987654312
000004	0987654123
000005	0198273645
000005	0698273645
000006	0912854367
000007	0928198482
000007	0928108882
000008	0182946284
000009	0917826543
000010	0768965142
000010	0768965192
000011	0858262492
000011	0958212402

5. Table Dormitory Manager

```

1   CREATE TABLE dormitory_manager (
2       PersonalID_DM varchar(55) NOT NULL,
3       ManagerID varchar(55) NOT NULL,
4       PRIMARY KEY (PersonalID_DM),
5       FOREIGN KEY (PersonalID_DM) REFERENCES dormitory_staff (PersonalID)
6   );

```

The screenshot shows the MySQL Workbench interface with two tabs open: 'dormitory_manager' and 'dormy_manager 1'. The 'dormitory_manager' tab contains a SQL query window with the command 'SELECT * FROM university_dormitory.dormitory_managers;'. Below it is a result grid showing three rows of data:

PersonalID_DM	ManagerID
000004	M01
000005	M02
000006	M03
*	NULL

The right side of the interface features a vertical toolbar with icons for Result Grid, Form Editor, Field Types, and Query Stats.

6. Table Guard

```

1  CREATE TABLE guard (
2      GuardID varchar(55) NOT NULL,
3      PersonalID_G varchar(55) NOT NULL,
4      PRIMARY KEY (PersonalID_G),
5      FOREIGN KEY (PersonalID_G) REFERENCES dormitory_staff (PersonalID)
6  );

```

The screenshot shows the MySQL Workbench interface with a query editor window titled 'guard'. The query 'SELECT * FROM university_dormitory.guard;' is run, and the results are displayed in a grid. The grid has two columns: 'GuardID' and 'PersonalID_G'. The data rows are: G01, 000001; G02, 000002; G03, 000003; and a blank row with NULL values. The right side of the interface features a vertical toolbar with icons for Result Grid, Form Editor, Field Types, and Query Stats.

GuardID	PersonalID_G
G01	000001
G02	000002
G03	000003
NULL	NULL

7. Table Office Staff

```

1   CREATE TABLE office_staff (
2       StaffID varchar(55) NOT NULL,
3       PersonalID_0 varchar(55) NOT NULL,
4       PRIMARY KEY (PersonalID_0),
5       FOREIGN KEY (PersonalID_0) REFERENCES dormitory_staff (PersonalID)
6   );

```

The screenshot shows the MySQL Workbench interface with a query editor window titled 'office_staff'. The query is:

```
1 • | SELECT * FROM university_dormitory.office_staff;
```

The results are displayed in a 'Result Grid' table:

StaffID	PersonalID_O
O01	000007
O02	000008
O03	000009
O04	000010
O05	000011
O06	000012
*	NULL
*	NULL

On the right side of the interface, there is a vertical toolbar with icons for 'Result Grid', 'Form Editor', 'Field Types', and 'Query Stats'.

8. Table Dormitory

```

1   CREATE TABLE dormitory (
2       Dormitory_ID varchar(55) NOT NULL,
3       Location varchar(55) NOT NULL,
4       Number_of_Floors int NOT NULL,
5       Number_of_Rooms int NOT NULL,
6       Overall_Condition varchar(55) NOT NULL,
7       PersonalID_G varchar(55) DEFAULT NULL,
8       PRIMARY KEY (Dormitory_ID),
9       KEY PersonalID_G (PersonalID_G),
10      FOREIGN KEY (PersonalID_G) REFERENCES guard (PersonalID_G)
11      )

```

The screenshot shows the MySQL Workbench interface with a query editor window titled 'dormitory'. The query is:

```
1 • | SELECT * FROM university_dormitory.dormitory;
```

The results grid displays the following data:

Dormitory_ID	Location	Number_of_Floors	Number_of_Rooms	Overall_Condition	PersonalID_G
A	Phía Đông Học Viện	3	9	Mới	000001
B	Phía Nam Học Viện	5	10	Cũ	000002
C	Phía Bắc Học Viện	2	7	Ôn định	000003
*	NULL	NULL	NULL	NULL	NULL

On the right side of the interface, there is a sidebar with several tabs: Result Grid (selected), Form Editor, Field Types, and Query Stats.

9. Table Room

```

1   CREATE TABLE room (
2       Room_ID varchar(55) NOT NULL,
3       number_of_Students int NOT NULL,
4       Size_of_room varchar(55) NOT NULL,
5       Capacity int NOT NULL,
6       Gender VARCHAR(55) NOT NULL,
7       Dormitory_ID varchar(55) NOT NULL,
8       PRIMARY KEY (Room_ID),
9       FOREIGN KEY (Dormitory_ID) REFERENCES Dormitory (Dormitory_ID)
10      );

```

The screenshot shows the MySQL Workbench interface with a query window titled "room". The query is:

```
1 • SELECT * FROM university_dormitory.room;
```

The results are displayed in a "Result Grid" table:

Room_ID	number_of_Students	Size_of_room	Capacity	Gender	Dormitory_ID
101-A	5	Lớn	7	Nam	A
101-B	3	Nhỏ	3	Nam	B
101-C	3	Lớn	7	Nữ	C
102-A	4	Lớn	7	Nam	A
102-B	0	Nhỏ	3	Nữ	B
102-C	2	Trung	5	Nam	C
103-C	1	Nhỏ	3	Nữ	C
201-A	3	Trung	5	Nữ	A
201-B	2	Nhỏ	3	Nữ	B
201-C	2	Trung	5	Nam	C
202-A	2	Trung	5	Nữ	A
202-B	1	Nhỏ	3	Nam	B
202-C	3	Trung	5	Nữ	C
203-A	3	Trung	5	Nam	A
203-C	1	Nhỏ	3	Nam	C
204-C	1	Nhỏ	3	Nữ	C

10. Table Bill

```

1   CREATE TABLE bill (
2       Bill_ID varchar(55) NOT NULL,
3       Bill_Type varchar(55) NOT NULL,
4       Amount_Due int NOT NULL,
5       Due_Date varchar(55) NOT NULL,
6       Payment_Status varchar(55) NOT NULL,
7       StudentID varchar(55) NOT NULL,
8       PersonalID_0 varchar(55) NOT NULL,
9       PRIMARY KEY (Bill_ID),
10      KEY StudentID (StudentID),
11      KEY PersonalID_0 (PersonalID_0),
12      FOREIGN KEY (StudentID) REFERENCES student (StudentID),
13      FOREIGN KEY (PersonalID_0) REFERENCES office_staff (PersonalID_0)
14  )

```

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query editor contains the SQL command: `SELECT * FROM university_dormitory.bill;`. The result grid displays 17 rows of data from the 'bill' table, which tracks various bills for students. The columns are: Bill_ID, Bill_Type, Amount_Due, Due_Date, Payment_Status, StudentID, and PersonalID_O.

Bill_ID	Bill_Type	Amount_Due	Due_Date	Payment_Status	StudentID	PersonalID_O
B001	Điện	500000	2024-12-15	Chưa thanh toán	SV001	000007
B002	Nước	200000	2024-12-20	Đã thanh toán	SV002	000008
B003	Sử dụng Internet	300000	2024-12-10	Chưa thanh toán	SV003	000009
B004	Bảo trì	150000	2024-12-05	Đã thanh toán	SV004	000010
B005	Phạt	100000	2024-12-15	Đã thanh toán	SV005	000011
B006	Tiền phòng	2000000	2024-12-20	Chưa thanh toán	SV006	000012
B007	Điện	600000	2024-12-10	Chưa thanh toán	SV007	000007
B008	Nước	250000	2024-12-05	Đã thanh toán	SV008	000008
B009	Sử dụng Internet	350000	2024-12-15	Chưa thanh toán	SV009	000009
B010	Bảo trì	200000	2024-12-20	Đã thanh toán	SV010	000010
B011	Phạt	120000	2024-12-10	Chưa thanh toán	SV011	000011
B012	Tiền phòng	1800000	2024-12-05	Đã thanh toán	SV012	000012
B013	Điện	550000	2024-12-15	Chưa thanh toán	SV013	000007
B014	Nước	220000	2024-12-20	Đã thanh toán	SV014	000008
B015	Sử dụng Internet	320000	2024-12-10	Chưa thanh toán	SV015	000009
B016	Bảo trì	180000	2024-12-05	Đã thanh toán	SV016	000010
B017	Phạt	80000	2024-12-15	Đã thanh toán	SV017	000011

11. Table Request

```

1   CREATE TABLE request (
2       RequestID varchar(55) NOT NULL,
3       Content varchar(55) NOT NULL,
4       Type_of_Request varchar(55) NOT NULL,
5       Status_of_Request varchar(55) NOT NULL,
6       Completion_Date varchar(55) NOT NULL,
7       PRIMARY KEY (`RequestID`)
8   );

```

The screenshot shows a MySQL Workbench interface with a query editor window titled 'request'. The query is:

```
1 • | SELECT * FROM university_dormitory.request;
```

The results are displayed in a 'Result Grid' table:

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 bi hong	Thiet bi	Dang xu ly	none
RQ012	Sua cuu so bi hong	Phong	Hoan thanh	2024-11-03
RQ013	Sua chieu sang trong phong	Phong	Dang xu ly	none
RQ014	Sua canh cuu	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
RQ017	Sua den bi hong	Bao tri	Dang xu ly	----

12. Table Facility

```

1   CREATE TABLE facility (
2       Room_ID varchar(55) NOT NULL,
3       itemName varchar(55) NOT NULL,
4       Overall_Quality varchar(55) NOT NULL,
5       PRIMARY KEY (Room_ID, itemName), -- Composite key
6       FOREIGN KEY (Room_ID) REFERENCES room (Room_ID) -- Foreign key reference to the room
7   );

```

The screenshot shows the MySQL Workbench interface with a query editor window titled 'facility'. The query is:

```
1 •  SELECT * FROM university_dormitory.facility;
```

The results are displayed in a grid:

Room_ID	itemName	Overall_Quality
101-A	Điều hòa	Hồng
101-A	Ghế	Tốt
101-A	Giường	Tốt
101-A	Quạt	Tốt
101-B	Điều hòa	Tốt
101-B	Giường	Khá
101-B	Tủ	Khá
101-C	Điều hòa	Tốt
101-C	Ghế	Khá
101-C	Giường	Tốt
102-A	Điều hòa	Tốt
102-A	Ghế	Khá
102-A	Giường	Tốt
102-A	Quạt	Hồng
102-A	Tủ	Tốt
102-B	Điều hòa	Khá
102-B	Ghế	Khá

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   );

```

The screenshot shows the MySQL Workbench interface with a query editor window titled 'facility' containing the query: 'SELECT * FROM university_dormitory.live;'. Below the query is a 'Result Grid' showing the following data:

StudentID	Room_ID	PersonalID_DM
SV001	101-C	000006
SV002	101-C	000006
SV003	101-C	000006
SV004	102-B	000005
SV005	103-C	000006
SV006	201-A	000004
SV007	201-A	000004
SV008	201-A	000004
SV009	201-B	000005
SV010	201-B	000005
SV011	202-A	000004
SV012	202-A	000004
SV013	202-C	000006
SV014	202-C	000006
SV015	202-C	000006
SV016	204-C	000004
SV017	201-A	000004

The right sidebar of the interface includes icons for Result Grid, Form Editor, Field Types, and Query Stats, with 'Result Grid' currently selected. A 'Read Only' status indicator is also present.

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  )

```

The screenshot shows a MySQL Workbench interface with a query editor window titled 'send'. The query is:

```
1 •  SELECT * FROM university_dormitory.send;
```

The results are displayed in a 'Result Grid' table:

StudentID	PersonalID_DM	RequestID	Send_Date
SV001	000004	RQ001	2024-09-22
SV003	000005	RQ002	2024-09-23
SV003	000006	RQ003	2024-09-24
SV005	000004	RQ004	2024-09-25
SV005	000005	RQ005	2024-09-26
SV006	000006	RQ006	2024-09-27
SV007	000004	RQ007	2024-09-28
SV008	000005	RQ008	2024-09-29
SV009	000006	RQ009	2024-09-30
SV009	000004	RQ010	2024-10-01
SV011	000005	RQ011	2024-10-02
SV012	000006	RQ012	2024-10-03
SV013	000004	RQ013	2024-10-04
SV014	000005	RQ014	2024-10-05
SV016	000006	RQ015	2024-10-06
SV016	000004	RQ016	2024-10-07
SV016	000005	RQ017	2024-10-07

15. Table Include

```

1  CREATE TABLE include (
2      Room_ID varchar(55) NOT NULL,
3      itemName varchar(55) NOT NULL,
4      Quantity int NOT NULL,
5      PRIMARY KEY (Room_ID, itemName), -- Composite primary key
6      FOREIGN KEY (Room_ID) REFERENCES room (Room_ID), -- Foreign key to room table
7      FOREIGN KEY (Room_ID, itemName) REFERENCES facility (Room_ID, itemName) -- Foreign key to facility table
8  );

```

The screenshot shows the MySQL Workbench interface with a query editor window titled 'include'. The query is:

```
1 •  SELECT * FROM university_dormitory.include;
```

The results are displayed in a grid:

Room_ID	itemName	Quantity
101-A	Điều hòa	2
101-A	Ghế	5
101-A	Giường	5
101-A	Quạt	3
101-B	Điều hòa	1
101-B	Giường	2
101-B	Tủ	1
101-C	Điều hòa	2
101-C	Ghế	3
101-C	Giường	3
102-A	Điều hòa	2
102-A	Ghế	4
102-A	Giường	4
102-A	Quạt	2
102-A	Tủ	2
102-B	Điều hòa	1
102-B	Ghế	4

16. Table Contract

```

1   CREATE TABLE contract (
2       ContractID varchar(55) NOT NULL,
3       Contract_Start_Date varchar(55) NOT NULL,
4       Contract_End_Date varchar(55) NOT NULL,
5       Contract_Status varchar(55) NOT NULL,
6       Penalty_for_Violation varchar(55) NOT NULL,
7       PersonalID_0 varchar(55) NOT NULL,
8       StudentID varchar(55) NOT NULL,
9       PRIMARY KEY (ContractID),
10      KEY PersonalID_0 (PersonalID_0),
11      KEY StudentID (StudentID),
12      FOREIGN KEY (PersonalID_0) REFERENCES office_staff (PersonalID_0),
13      FOREIGN KEY (StudentID) REFERENCES student (StudentID)
14  );

```

contract

```
1 • SELECT * FROM university_dormitory.contract;
```

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

ContractID	Contract_Start_Date	Contract_End_Date	Contract_Status	Penalty_for_Violation	PersonalID_O	StudentID
CT001	2024-01-01	2024-12-31	Hoat Dong	Phat Tien	000007	SV001
CT002	2024-02-01	2024-12-31	Hoat Dong	Canh Cao	000008	SV002
CT003	2024-03-01	2024-12-31	Tam Ngung	Phat Tien	000009	SV003
CT004	2024-04-01	2024-12-31	Hoat Dong	Dinh Chi	000010	SV004
CT005	2024-05-01	2024-12-31	Hoat Dong	Canh Cao	000011	SV005
CT006	2024-06-01	2024-12-31	Hoat Dong	Phat Tien	000012	SV006
CT007	2024-01-10	2024-12-31	Hoat Dong	Dinh Chi	000007	SV007
CT008	2024-02-15	2024-12-31	Hoat Dong	Phat Tien	000008	SV008
CT009	2024-03-25	2024-12-31	Hoat Dong	Canh Cao	000009	SV009
CT010	2024-04-05	2024-12-31	Tam Ngung	Dinh Chi	000010	SV010
CT011	2024-05-10	2024-12-31	Hoat Dong	Phat Tien	000011	SV011
CT012	2024-06-15	2024-12-31	Hoat Dong	Phat Tien	000012	SV012
CT013	2024-07-01	2024-12-31	Hoat Dong	Canh Cao	000007	SV013
CT014	2024-08-01	2024-12-31	Hoat Dong	Phat Tien	000008	SV014
CT015	2024-09-01	2024-12-31	Hoat Dong	Dinh Chi	000009	SV015
CT016	2024-10-01	2024-12-31	Hoat Dong	Phat Tien	000010	SV016
CT017	2024-11-01	2024-12-31	Hoat Dong	Canh Cao	000011	SV017
CT018	2024-12-01	2024-12-31	Hoat Dong	Phat Tien	000012	SV018
CT019	2024-01-01	2024-06-30	Tam Ngung	Dinh Chi	000007	SV019
CT020	2024-02-01	2024-06-30	Hoat Dong	Phat Tien	000008	SV020
CT021	2024-03-01	2024-06-30	Hoat Dong	Canh Cao	000009	SV021
CT022	2024-04-01	2024-06-30	Hoat Dong	Dinh Chi	000010	SV022
CT023	2024-05-01	2024-06-30	Hoat Dong	Phat Tien	000011	SV023
CT024	2024-06-01	2024-06-30	Hoat Dong	Canh Cao	000012	SV024
CT025	2024-07-01	2024-12-31	Hoat Dong	Phat Tien	000007	SV025
CT026	2024-08-01	2024-12-31	Hoat Dong	Dinh Chi	000008	SV026
CT027	2024-09-01	2024-12-31	Hoat Dong	Canh Cao	000009	SV027

contract 1

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan | Apply | Revert

VI. Queries

1. Simple Query

- Bill:

- o 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)

bill

```
1 • SELECT
2     b.Bill_ID,
3     b.Bill_Type,
4     b.Amount_Due,
5     s.StudentID,
6     s.LastName,
7     s.EmailAddress AS StudentEmail,
8     s.phoneNumber AS StudentPhoneNumber
9   FROM
10      bill b
11   JOIN
12      student s ON b.StudentID = s.StudentID
13   WHERE
14      b.Payment_Status <> 'Đã thanh toán'
15      AND b.Due_Date > CURDATE();
```

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

Bill_ID	Bill_Type	Amount_Due	StudentID	Lastname	StudentEmail	StudentPhoneNumber
B001	Điện	500000	SV001	Lan	LanNT@stu.ptit.edu.vn	0987654321
B003	Sử dụng Internet	300000	SV003	Thanh	ThanhLT@stu.ptit.edu.vn	0987654323
B006	Tiền phòng	2000000	SV006	Mai	MaiVT@stu.ptit.edu.vn	0987654326
B007	Điện	600000	SV007	Nhung	NhungDT@stu.ptit.edu.vn	0987654327
B009	Sử dụng Internet	350000	SV009	My	MyNT@stu.ptit.edu.vn	0987654329
B011	Phát	120000	SV011	Kim	KimTT@stu.ptit.edu.vn	0987654331
B013	Điện	550000	SV013	Huyen	HuyenPT@stu.ptit.edu.vn	0987654333
B015	Sử dụng Internet	320000	SV015	Thu	ThuLT@stu.ptit.edu.vn	0987654335
B018	Tiền phòng	200000	SV018	Chi	ChiNT@stu.ptit.edu.vn	0987654338
B020	Nước	190000	SV020	Kim	KimTT@stu.ptit.edu.vn	0987654340
B022	Bảo trì	150000	SV022	Hoang	HoangNV@stu.ptit.edu.vn	0987654326
B024	Tiền phòng	2500000	SV024	Son	SonTT@stu.ptit.edu.vn	0987654328
B026	Nước	230000	SV026	Duy	DuyLV@stu.ptit.edu.vn	0987654330
B028	Bảo trì	200000	SV028	Tuan	TuanHV@stu.ptit.edu.vn	0987654332
B030	Tiền phòng	2200000	SV030	An	AnNH@stu.ptit.edu.vn	0987654334

bill 1

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan | Read Only

- o 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.

The screenshot shows a database interface with a query editor and a result grid. The query editor contains the following SQL code:

```

1 • SELECT
2     b.Bill_ID,
3     os.StaffID,
4     ds.PersonalID,
5     ds.FirstName,
6     ds.LastName,
7     ds.Salary,
8     ds.StaffType
9
10    FROM
11        bill b
12    JOIN
13        office_staff os ON b.PersonalID_0 = os.PersonalID_0
14    JOIN
15        dormitory_staff ds ON os.PersonalID_0 = ds.PersonalID
16    WHERE
17        b.Bill_ID = 'B005';
18

```

The result grid displays the following data:

Bill_ID	StaffID	PersonalID	FirstName	LastName	Salary	StaffType
B005	O05	000011	Tran	Hung	15000000	Office Staff

The interface includes a toolbar at the top, a sidebar on the right with icons for Result Grid, Form Editor, and Field Types, and a status bar at the bottom indicating 'Read Only'.

- o 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

bill

```

1 • SELECT
2     b.Bill_ID,
3     b.Bill_Type,
4     b.Amount_Due,
5     b.Payment_Status,
6     b.Due_Date
7
8     FROM
9     bill b
10    WHERE
11        b.StudentID = 'SV009';

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor | Field Types | Apply

Bill_ID	Bill_Type	Amount_Due	Payment_Status	Due_Date
B009	Sử dụng Internet	350000	Chưa thanh toán	2024-12-15
B064	Bảo trì	50000	Đã thanh toán	2024-12-15
B114	Phát	150000	Đã thanh toán	2025-02-04
*	HULL	HULL	HULL	HULL

bill 3

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

bill

```

1 • SELECT
2     SUM(Amount_Due) as TotalRevenue,
3     SUM(CASE WHEN Payment_Status = 'Đã thanh toán' THEN Amount_Due ELSE 0 END) AS ActualRevenue
4     FROM bills;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Result Grid | Form Editor | Field Types | Read Only

TotalRevenue	ActualRevenue
58175000	28785000

Result 4

- Contract:

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

contract

```

1 SELECT ContractID, Contract_Start_Date, Contract_End_Date, Penalty_for_Violation, PersonalID_O, StudentID FROM Contract
2 WHERE Contract_Status = 'Hoat Dong';

```

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

ContractID	Contract_Start_Date	Contract_End_Date	Penalty_for_Violation	PersonalID_O	StudentID
CT001	2024-01-01	2024-12-31	Phat Tien	000007	SV001
CT002	2024-02-01	2024-12-31	Canh Cao	000008	SV002
CT004	2024-04-01	2024-12-31	Dinh Chi	000010	SV004
CT005	2024-05-01	2024-12-31	Canh Cao	000011	SV005
CT006	2024-06-01	2024-12-31	Phat Tien	000012	SV006
CT007	2024-01-10	2024-12-31	Dinh Chi	000007	SV007
CT008	2024-02-15	2024-12-31	Phat Tien	000008	SV008
CT009	2024-03-25	2024-12-31	Canh Cao	000009	SV009
CT011	2024-05-10	2024-12-31	Phat Tien	000011	SV011
CT012	2024-06-15	2024-12-31	Phat Tien	000012	SV012
CT013	2024-07-01	2024-12-31	Canh Cao	000007	SV013
CT014	2024-08-01	2024-12-31	Phat Tien	000008	SV014
CT015	2024-09-01	2024-12-31	Dinh Chi	000009	SV015
CT016	2024-10-01	2024-12-31	Phat Tien	000010	SV016
CT017	2024-11-01	2024-12-31	Canh Cao	000011	SV017
CT018	2024-12-01	2024-12-31	Phat Tien	000012	SV018
CT020	2024-02-01	2024-06-30	Phat Tien	000008	SV020
CT021	2024-03-01	2024-06-30	Canh Cao	000009	SV021
CT022	2024-04-01	2024-06-30	Dinh Chi	000010	SV022
CT023	2024-05-01	2024-06-30	Phat Tien	000011	SV023
CT024	2024-06-01	2024-06-30	Canh Cao	000012	SV024
CT025	2024-07-01	2024-12-31	Phat Tien	000007	SV025
CT026	2024-08-01	2024-12-31	Dinh Chi	000008	SV026
CT027	2024-09-01	2024-12-31	Canh Cao	000009	SV027
CT028	2024-10-01	2024-12-31	Phat Tien	000010	SV028
CT029	2024-11-01	2024-12-31	Dinh Chi	000011	SV029
CT030	2024-12-01	2024-12-31	Phat Tien	000012	SV030

Contract 2 | Apply | Revert

- o 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.

contract

```

1 • SELECT Student.StudentID, Student.firstName, Student.middleName, Student.lastname, Student.Date_of_Birth, Student.Address, Student.Gender, Student.PhoneNumber, Student.EmailAddress
2 FROM Student
3 JOIN contract
4 ON Student.StudentID = contract.StudentID
5 WHERE contract.ContractID = 'CT004';

```

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

StudentID	firstName	middleName	lastname	Date_of_Birth	Address	Gender	PhoneNumber	EmailAddress
SV004	Phan	Thi	Anh	2002-05-15	Danang	Nữ	0987654324	AnhPT@stu.ptit.edu.vn

Result 3 | Read Only

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

contract x

```
1 •  SELECT ContractID,Contract_Start_Date,Contract_End_Date,Contract_Status,Penalty_for_Violation,PersonalID_O, StudentID FROM Contract
2 WHERE Contract_Start_Date BETWEEN '2024-01-01' AND '2024-03-31';
```

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

ContractID	Contract_Start_Date	Contract_End_Date	Contract_Status	Penalty_for_Violation	PersonalID_O	StudentID
CT001	2024-01-01	2024-12-31	Hoat Dong	Phat Tien	000007	SV001
CT002	2024-02-01	2024-12-31	Hoat Dong	Canh Cao	000008	SV002
CT003	2024-03-01	2024-12-31	Tam Ngung	Phat Tien	000009	SV003
CT007	2024-01-10	2024-12-31	Hoat Dong	Dinh Chi	000007	SV007
CT008	2024-02-15	2024-12-31	Hoat Dong	Phat Tien	000008	SV008
CT009	2024-03-25	2024-12-31	Hoat Dong	Canh Cao	000009	SV009
CT019	2024-01-01	2024-06-30	Tam Ngung	Dinh Chi	000007	SV019
CT020	2024-02-01	2024-06-30	Hoat Dong	Phat Tien	000008	SV020
CT021	2024-03-01	2024-06-30	Hoat Dong	Canh Cao	000009	SV021
CT031	2024-01-01	2024-12-31	Hoat Dong	Canh Cao	000007	SV031
CT032	2024-02-01	2024-12-31	Hoat Dong	Phat Tien	000008	SV032
CT033	2024-03-01	2024-12-31	Hoat Dong	Dinh Chi	000009	SV033
CT043	2024-01-01	2024-06-30	Hoat Dong	Canh Cao	000007	SV043
CT044	2024-02-01	2024-06-30	Hoat Dong	Phat Tien	000008	SV044
CT045	2024-03-01	2024-06-30	Hoat Dong	Dinh Chi	000009	SV045
*	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Contract 4 x Apply Revert

- **Dormitory:**

- Query 1: Calculate the total number of rooms.

dormitory x

```
1 •  SELECT SUM(Number_of_Rooms) AS Total_Rooms
2 FROM dormitory;
```

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

Total_Rooms
26

Result 2 x Read Only

- o 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).

The screenshot shows a database interface with two main sections: a query editor at the top and a results grid below it.

Query Editor:

```

dormitory x
1 • SELECT PersonalID,firstName,lastName,Salary,Email,Address,State,Dormitory_Assigned,staffType from dormitory_staff
2 JOIN dormitory
3 ON PersonalID = dormitory.PersonalID_G
4 WHERE dormitory.Dormitory_ID = 'A';
  
```

Results Grid:

PersonalID	firstName	lastName	Salary	Email	Address	State	Dormitory_Assigned	staffType
000001	Nguyen	Bao	10000000	bao.nguyen@gmail.com	123 Duong A, Ha Noi	Dang lam viec	A	Guard

On the right side of the interface, there is a vertical toolbar with icons for different functions: Result Grid (selected), Form Editor, Field Types, and Query Stats.

- **Dormitory Staff:**

- o 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

dormitory_staff

```

1  SELECT
2      ds.PersonalID,
3      ds.firstName,
4      ds.middleName,
5      ds.lastName,
6      ds.salary,
7      ds.email,
8      ds.address,
9      ds.state,
10     ds.Dormitory_Assigned,
11     GROUP_CONCAT(DISTINCT dsp.phoneNumber) AS phoneNumbers
12  FROM dormitory_staff ds
13  JOIN dormitory_staff_phonenumber dsp ON ds.PersonalID = dsp.PersonalID
14  WHERE ds.Dormitory_Assigned = 'A'
15  GROUP BY ds.PersonalID;

```

Result Grid | Filter Rows: Export: Wrap Cell Content: Result Grid Form Editor Field Types Query Stats Read Only

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	0987654321
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

Result 2 | Filter Rows: Export: Wrap Cell Content: Result Grid Form Editor Field Types Query Stats Read Only

- o 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

dormitory_staff

```

1  SELECT
2      ds.PersonalID,
3      ds.firstName,
4      ds.middleName,
5      ds.lastName,
6      ds.salary,
7      ds.email,
8      ds.address,
9      ds.state,
10     ds.Dormitory_Assigned,
11     GROUP_CONCAT(DISTINCT dsp.phoneNumber) AS phoneNumbers
12  FROM dormitory_staff ds
13  JOIN dormitory_staff_phonenumber dsp ON ds.PersonalID = dsp.PersonalID
14  WHERE ds.staffType = 'Office Staff'
15  GROUP BY ds.PersonalID;

```

Result Grid | Filter Rows: Export: Wrap Cell Content: Result Grid Form Editor Field Types Query Stats Read Only

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

Result 3 | Filter Rows: Export: Wrap Cell Content: Result Grid Form Editor Field Types Query Stats Read Only

● Facility:

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

facility

```

1 SELECT * FROM Facility
2 WHERE Overall_Quality = 'Hồng';
3

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Result Grid | Form Editor | Field Types | Query Stats | Apply | Revert

Room_ID	itemName	Overall_Quality
101-A	Điều hòa	Hồng
102-A	Quạt	Hồng
201-B	Quạt	Hồng
301-A	Điều hòa	Hồng
NULL	NULL	NULL

Facility 2 x

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

facility

```

1 • WITH ItemCount AS (
2     SELECT Room_ID, COUNT(itemName) AS item_count
3     FROM facility
4     GROUP BY Room_ID
5 ),
6 • MaxItemRooms AS (
7     SELECT Room_ID, item_count
8     FROM ItemCount
9     WHERE item_count = (SELECT MAX(item_count) FROM ItemCount)
10 )
11     SELECT f.Room_ID, f.itemName, f.Overall_Quality, m.item_count AS max_items
12     FROM facility f
13     JOIN MaxItemRooms m ON f.Room_ID = m.Room_ID;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Result Grid | Form Editor | Field Types | Query Stats | Read Only

Room_ID	itemName	Overall_Quality	max_items
102-A	Điều hòa	Tốt	5
102-A	Ghế	Khá	5
102-A	Giường	Tốt	5
102-A	Quạt	Hồng	5
102-A	Tủ	Tốt	5
201-A	Điều hòa	Tốt	5
201-A	Ghế	Tốt	5
201-A	Giường	Khá	5
201-A	Quạt	Khá	5
201-A	Tủ	Tốt	5

Result 3 x

- **Include:**

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

The screenshot shows a database interface with a query editor and a results grid.

Query Editor (include):

```

1  SELECT r.Room_ID, i.itemName, i.Quantity, f.Overall_Quality
2  FROM room r
3  JOIN include i ON r.Room_ID = i.Room_ID
4  JOIN facility f ON i.Room_ID = f.Room_ID AND i.itemName = f.itemName
5  ORDER BY r.Room_ID;
    
```

Result Grid:

Room_ID	itemName	Quantity	Overall_Quality
101-A	Điều hòa	2	Hồng
101-A	Ghế	5	Tốt
101-A	Gường	5	Tốt
101-A	Quạt	3	Tốt
101-B	Điều hòa	1	Tốt
101-B	Gường	2	Khá
101-B	Tủ	1	Khá
101-C	Điều hòa	2	Tốt
101-C	Ghế	3	Khá
101-C	Gường	3	Tốt
102-A	Điều hòa	2	Tốt
102-A	Ghế	4	Khá
102-A	Gường	4	Tốt
102-A	Quạt	2	Hồng
102-A	Tủ	2	Tốt
102-B	Điều hòa	1	Khá
102-B	Ghế	1	Khá
102-B	Gường	1	Tốt
102-B	Tủ	1	Khá
102-C	Điều hòa	1	Tốt
102-C	Ghế	2	Tốt
102-C	Gường	2	Khá
102-C	Tủ	1	Khá
103-C	Điều hòa	1	Khá
103-C	Gường	1	Tốt
201-A	Điều hòa	1	Tốt
201-A	Ghế	3	Tốt

Right Panel:

- Result Grid (selected)
- Form Editor
- Field Types
- Query Stats
- Execution Plan

Bottom Right: Read Only

- **Live:**

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

The screenshot shows a database interface with a query editor and a results grid. The query is:

```

1 SELECT StudentID,Room_ID FROM Live
2 WHERE Room_ID LIKE '101%';

```

The results grid displays the following data:

StudentID	Room_ID
SV022	101-A
SV023	101-A
SV024	101-A
SV025	101-A
SV026	101-A
SV027	101-B
SV028	101-B
SV001	101-C
SV002	101-C
SV003	101-C

The interface includes a toolbar with various icons, a status bar at the bottom, and a sidebar on the right with icons for Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan.

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

The screenshot shows a database interface with a query editor and a results grid. The query is:

```

1 SELECT StudentID,Room_ID FROM Live
2 WHERE PersonalID_DM = '000004';

```

The results grid displays the following data:

StudentID	Room_ID
SV006	201-A
SV007	201-A
SV008	201-A
SV011	202-A
SV012	202-A
SV016	204-C
SV017	301-A
SV019	302-A
SV022	101-A
SV023	101-A
SV024	101-A
SV025	101-A
SV026	101-A
SV029	102-A
SV030	102-A
SV031	102-A
SV032	102-A
SV038	203-A
SV039	203-A
SV040	203-A
SV042	303-A
SV043	303-A
SV044	304-A
SV045	304-A

The interface includes a toolbar with various icons, a status bar at the bottom, and a sidebar on the right with icons for Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan.

- Request:

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

request x

```

1 SELECT *
2 FROM request
3 WHERE Status_of_Request != 'Hoan thanh';

```

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

RequestID	Content	Type_of_Request	Status_of_Request	Completion_Date
RQ002	Sua den bi hong	Thiet bi	Dang xu ly	none
RQ003	Doi phong do may lanh bi hong	Phong	Dang xu ly	none
RQ006	Sua dieu hoa	Thiet bi	Dang xu ly	none
RQ007	Doi guong trong phong tam	Phong	Dang xu ly	none
RQ009	Doi den trong phong hoc	Phong	Dang xu ly	none
RQ011	Sua may bi hong	Thiet bi	Dang xu ly	none
RQ013	Sua chieu sang trong phong	Phong	Dang xu ly	none
RQ015	Bao tri tu lanh	Thiet bi	Dang xu ly	none
RQ017	Bao tri binh nong lanh	Bao tri	Dang xu ly	none
RQ018	Bao tri dieu hoa	Bao tri	Dang xu ly	none
RQ020	Bao tri quat	Bao tri	Dang xu ly	none
...	NULL	NULL	NULL	NULL

request 2 x

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

Form Editor

Field Types

Query Stats

Result Grid

Apply | Revert

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

request x

```

1 • SELECT
2     s.StudentID,
3     CONCAT(s.firstName, ' ', s.middleName, ' ', s.lastName) AS StudentName,
4     r.*
5   FROM request r
6   JOIN send se ON se.RequestID = r.RequestID
7   JOIN student s ON s.StudentID = se.StudentID
8 ORDER BY r.RequestID;

```

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

StudentID	Studentname	RequestID	Content	Type_of_Request	Status_of_Request	Completion_Date
SV001	Nguyen Thi Lan	RQ001	Sua quat hong	Thiet bi	Hoan thanh	2024-11-01
SV003	Le Thi Thanh	RQ002	Sua den bi hong	Thiet bi	Dang xu ly	none
SV003	Le Thi Thanh	RQ003	Doi phong do may lanh bi hong	Phong	Dang xu ly	none
SV005	Huong Thi Linh	RQ004	Bao tri co so ha tang	Bao tri	Hoan thanh	2024-11-02
SV005	Huong Thi Linh	RQ005	Sua may vi trinh	Thiet bi	Hoan thanh	2024-11-03
SV006	Vu Thi Mai	RQ006	Sua dieu hoa	Thiet bi	Dang xu ly	none
SV007	Do Thi Nhung	RQ007	Doi guong trong phong tam	Phong	Dang xu ly	none
SV008	Bui Thi Thanh	RQ008	Bao tri binh nong lanh	Bao tri	Hoan thanh	2024-11-04
SV009	Nguyen Thi My	RQ009	Doi den trong phong hoc	Phong	Dang xu ly	none
SV009	Nguyen Thi My	RQ010	Bao tri dong ho	Bao tri	Hoan thanh	2024-11-01
SV011	Trinh Thi Kim	RQ011	Sua may bi hong	Thiet bi	Dang xu ly	none
SV012	Nguyen Thi Thao	RQ012	Sua cua so bi hong	Phong	Hoan thanh	2024-11-03
SV013	Pham Thi Huyen	RQ013	Sua chieu sang trong phong	Phong	Dang xu ly	none
SV014	Do Thi Quyen	RQ014	Sua canh sua	Bao tri	Hoan thanh	2024-11-02
SV016	Pham Thi Thanh	RQ015	Bao tri tu lanh	Thiet bi	Dang xu ly	none
SV016	Pham Thi Thanh	RQ016	Sua dieu hoa trong phong	Phong	Hoan thanh	2024-11-01
SV016	Pham Thi Thanh	RQ017	Bao tri binh nong lanh	Bao tri	Dang xu ly	none
SV018	Nguyen Thi Chi	RQ018	Bao tri dieu hoa	Bao tri	Dang xu ly	none
SV019	Huong Thi Hoa	RQ019	Sua loi may giat	Thiet bi	Hoan thanh	2024-11-04
SV021	Le Thi Thao	RQ020	Bao tri quat	Bao tri	Dang xu ly	none

Result 3 x

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

Form Editor

Field Types

Query Stats

Execution Plan

Result Grid

Read Only

- Room

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

The screenshot shows a database interface with two tabs: 'room' and 'room 2'. The 'room' tab is active, displaying a SQL query and its execution results.

SQL Query:

```

1  SELECT *
2  FROM room
3  WHERE Gender = 'Nam'
4  AND Size_of_room = 'Lớn'
5  AND number_of_Students < Capacity;
    
```

Result Grid:

Room_ID	number_of_Students	Size_of_room	Capacity	Gender	Dormitory_ID
101-A	5	Lớn	7	Nam	A
102-A	4	Lớn	7	Nam	A
*	HULL	HULL	HULL	HULL	HULL

Toolbars and Sidebar:

- Top toolbar: Includes icons for file operations, search, and export.
- Right sidebar: Contains buttons for 'Result Grid', 'Form Editor', 'Field Types', 'Query Stats', and 'Execution Plan'.
- Bottom buttons: 'Apply' and 'Revert'.

- Send

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

send

```

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

```

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

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 xuly	none	SV003	000005	2024-09-23
RQ003	Do phong do may lanh bi hong	Phong	Dang xuly	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 xuly	none	SV006	000006	2024-09-27
RQ007	Do guong trong phong tam	Phong	Dang xuly	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	Do den trong phong hoc	Phong	Dang xuly	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 bi hong	Thiet bi	Dang xuly	none	SV011	000005	2024-10-02
RQ012	Sua cuu so bi hong	Phong	Hoan thanh	2024-11-03	SV012	000006	2024-10-03
RQ013	Sua chieu sang trong phong	Phong	Dang xuly	none	SV013	000004	2024-10-04
RQ014	Sua canh cuu	Bao tri	Hoan thanh	2024-11-02	SV014	000005	2024-10-05
RQ015	Bao tri tu lanh	Thiet bi	Dang xuly	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 xuly	none	SV016	000005	2024-10-08
RQ018	Bao tri dieu hoa	Bao tri	Dang xuly	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 xuly	none	SV021	000005	2024-10-11

Result 2 × Read Only

- Student

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

student

```

1 SELECT * FROM student
2 WHERE Gender = 'N' AND Address = 'Hanoi' AND lastName >= 'N';

```

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

StudentID	firstname	middleName	lastName	Date_of_Birth	Address	Gender	phoneNumber	EmailAddress
SV007	Do	Thi	Nhung	2000-04-14	Hano [?]	Female	0987654327	NhungDT@stu.ptit.edu.vn
SV012	Nguyen	Thi	Thao	2001-03-10	Hano [?]	Female	0987654332	ThaoNT@stu.ptit.edu.vn
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

student 2 × Apply Revert

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

The screenshot shows a database interface with a query editor and a results grid. The query is:

```

1 • SELECT
2     s.StudentID,
3     CONCAT(s.firstName, ' ', s.middleName, ' ', s.lastName) AS fullName,
4     GROUP_CONCAT(ec.Emergency_Contact ORDER BY ec.Emergency_Contact ASC SEPARATOR ', ') AS Emergency_Contacts
5   FROM
6     Student_Emergency_Contact ec
7   JOIN
8     student s ON ec.StudentID = s.StudentID
9   GROUP BY
10    s.StudentID, s.firstName, s.middleName, s.lastName;

```

The results grid displays 21 rows of data:

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	0901234546, 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	Phuoc 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).

The screenshot shows a database interface with a query editor. The query is:

```

1 • WITH BillCounts AS (
2     SELECT
3         StudentID,
4         COUNT(DISTINCT Bill_ID) as NumberOfBills,
5         SUM(COALESCE(Amount_Due, 0)) as Total_Amount,
6         SUM(CASE
7             WHEN Payment_Status = 'Chua thanh toan'
8                 THEN COALESCE(Amount_Due, 0)
9             ELSE 0
10        END) as Unpaid_Amount
11     FROM bill
12     GROUP BY StudentID
13 )
14
15     SELECT
16         s.StudentID,
17         CONCAT(COALESCE(s.firstName, ''), ' ',
18             COALESCE(s.middleName, ''), ' ',
19             COALESCE(s.lastName, '')) AS fullName,
20         l.Room_ID,
21         c.ContractID,
22         bc.NumberOfBills,
23         bc.Total_Amount,
24         bc.Unpaid_Amount,
25         l.PersonalID_DM AS Manager_ID,
26         CASE
27             WHEN b.Bill_Status = 'Y'
28                 THEN 'Yes'
29             ELSE 'No'
30        END AS Request_Status
31     FROM BillCounts bc
32     JOIN student s ON bc.StudentID = s.StudentID
33     JOIN room l ON s.Room_ID = l.Room_ID
34     JOIN contract c ON s.StudentID = c.StudentID
35     JOIN bill b ON bc.Bill_ID = b.Bill_ID
36     WHERE bc.StudentID = l.StudentID
37     AND bc.StudentID = c.StudentID
38     AND bc.StudentID = b.StudentID
39
40

```

```

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: | Result Grid | Form Editor | Field Types | Query Stats | Read Only

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
SV014	Phu Thi Quynh	202-C	CT014	2	1270000	0	000004	1

- 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 | Filter Rows: | Export: | Wrap Cell Content: | Result Grid | Form Editor | Field Types | Query Stats | Read Only

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
000004	Phu Thi Khanh	A	0987654123	SV027, 101-A	Trinh Linh Khanh	0987654331	0901234551,0912345600

- 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)

student X

```

1 •  SELECT DISTINCT
2     r.Room_ID,
3     r.number_of_Students,
4     r.Size_of_Room,
5     r.Capacity,
6     r.Gender,
7     r.Dormitory_ID,
8
9     CASE
10        WHEN f.itemName IS NULL
11            AND (i.itemName IS NULL OR i.itemName != 'Quạt' OR i.Quantity < r.number_of_Students)
12        THEN 'Thiếu'
13        ELSE 'Đủ'
14    END AS cooling_status,
15
16    CASE
17        WHEN r.number_of_Students = 0 THEN 'Trống'
18        WHEN r.Capacity > number_of_Students THEN 'Còn chỗ'
19        ELSE 'Đầy'
20    END AS occupancy_status
21
22 FROM
23     Room r
24 LEFT JOIN
25     include i ON r.Room_ID = i.Room_ID
26 LEFT JOIN
27     facility f ON r.Room_ID = f.Room_ID AND f.itemName = 'Điều hòa'
28 ORDER BY Room_ID;

```

Result Grid | Filter Rows: Export: Wrap Cell Content: Result Grid Form Editor Read Only

Room_ID	number_of_Students	Size_of_Room	Capacity	Gender	Dormitory_ID	cooling_status	occupancy_status
101-A	5	Lớn	7	Nam	A	Đủ	Còn chỗ
101-B	3	Nhỏ	3	Nam	B	Đủ	Đầy
101-C	3	Lớn	7	Nữ	C	Đủ	Còn chỗ
102-A	4	Lớn	7	Nam	A	Đủ	Còn chỗ
102-B	0	Nhỏ	3	Nữ	B	Đủ	Trống
102-C	2	Trung	5	Nam	C	Đủ	Còn chỗ

Result 6 x