DBMS PROJECT REPORT

PROJECT TITLE

APARTMENT MANAGEMENT SYSTEM

TEAM DETAILS

Meghana Kudua (PES1UG22CS347)

Makarand Gokhale (PES1UG22CS325)

SECTION 'F'

OVERVIEW

Our project, the Apartment Management System, is designed to make it easier to manage apartments. It has three types of users: Owners, Tenants, and Employees. We used Flask for the backend, HTML for the frontend, and MySQL for the database.

Our project helps in apartment maintenance by allowing owners to track tenants and payments, letting tenants manage their rental agreements and complaints, and helping employees handle complaints in their assigned blocks.

Key Features:

1. **Homepage and User Login**: The system starts with a homepage where users can choose to log in as an Owner, Tenant, or Employee. After selecting their role and entering the correct login details, they are taken to their own dashboard.
2. **Owner Dashboard**: Owners can see details about their rooms, tenants, and payments. They can keep track of which tenants are living in which rooms, whether the rent has been paid, and if rental agreements are up to date.
3. **Tenant Dashboard**: Tenants can view and manage their rental agreements, make payments, renew their contracts, and raise complaints. They can also see the status of their past complaints.

4. **Employee Dashboard**: Employees, who manage different blocks, can see and resolve complaints raised by tenants in their assigned block. This ensures that tenant issues are addressed quickly.

The system uses **Flask** to handle user actions and **MySQL** to store and manage all the data, such as tenant, owner and employee information, room details, payments, and complaints.

IMPLEMENTATION

Backend (MySQL):

- **Database**: MySQL is used to store all data, such as owners, tenants, employees, rooms, payments, and complaints.
- **Data Links**: The database has connections between tables to keep data accurate, like linking rooms to owners and payments to tenants.

Backend Logic (Flask and PyMySQL):

- **Main Routing (hello.py)**: Flask runs the backend logic, handles user requests, connects to the database using PyMySQL, and updates or retrieves data.
- **User Sessions**: Sessions ensure only logged-in users can access their dashboards safely.
- **Forms**: WTForms is used to create and validate forms for user input.

Frontend (HTML Templates):

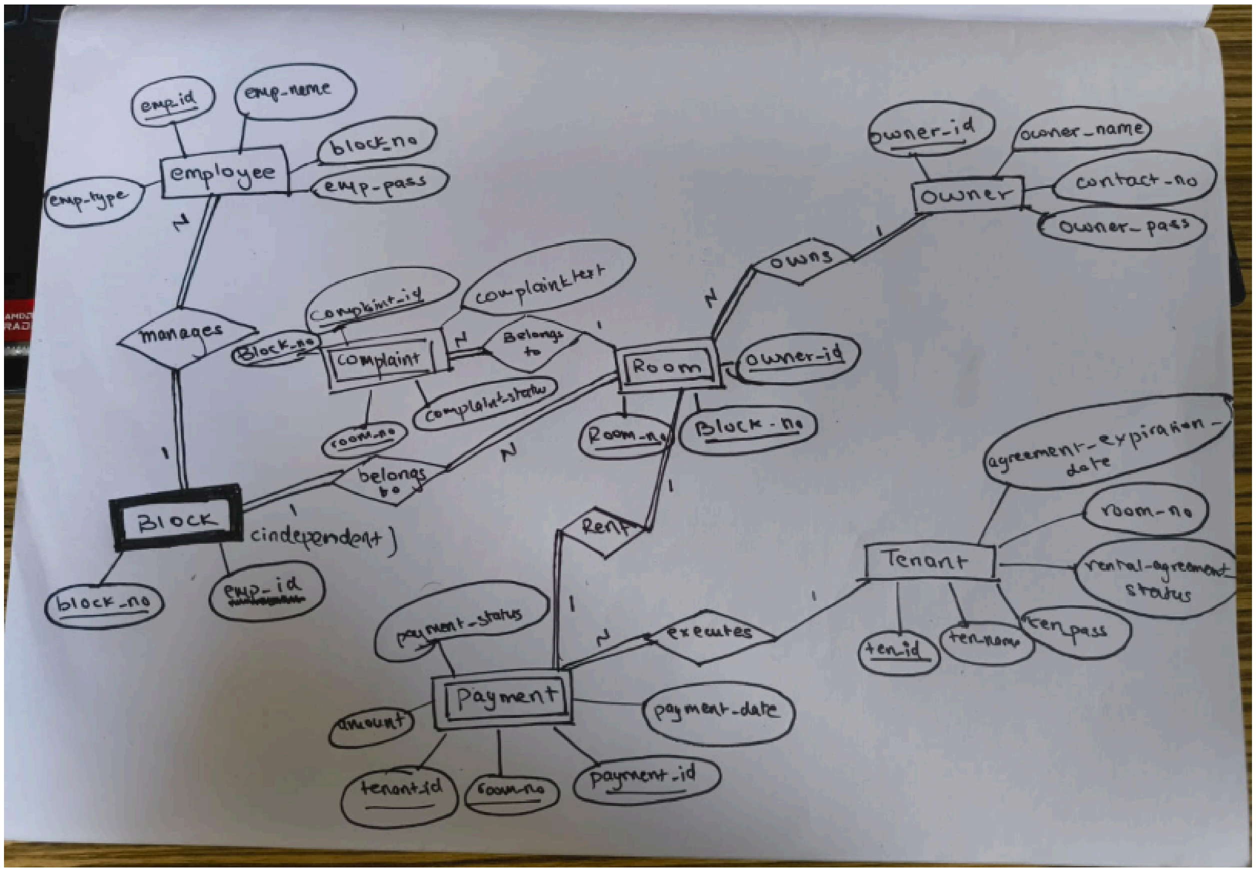
- **Templates**: HTML templates show the web pages users see, filled with data from the backend.
- **Dynamic Display**: Jinja2 lets templates show real-time data, such as tenant or payment information.
- **User Dashboards**: There are separate login pages and dashboards for owners, tenants, and employees.
- **Feedback Messages**: Flash messages notify users about their actions (e.g., login success, errors).

How It Works:

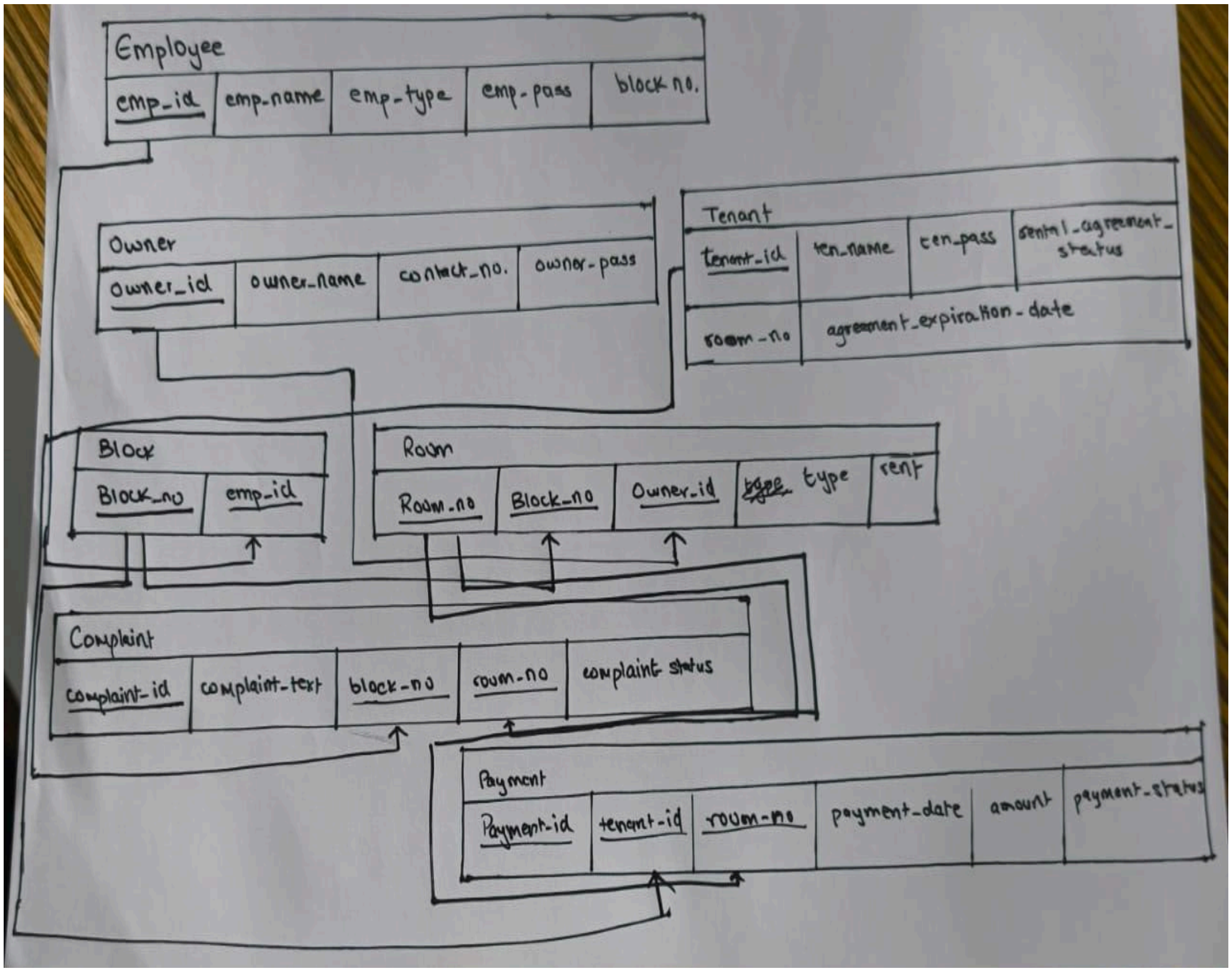
1. **Starting Point**: Users visit the home page and choose to log in as an owner, tenant, or employee.

2. **Login & Routing**: The hello.py file checks user credentials and routes them to their dashboard if successful.
3. **Data Handling**: Flask fetches or updates data in MySQL using PyMySQL when users interact (e.g., make payments or file complaints).
4. **Page Display**: Data is passed to HTML templates and shown on the page using Jinja2.
5. **User Actions**: Users can do things like renew agreements, pay rent, or handle complaints, with WTForms ensuring inputs are valid.

ENTITY RELATIONSHIP DIAGRAM

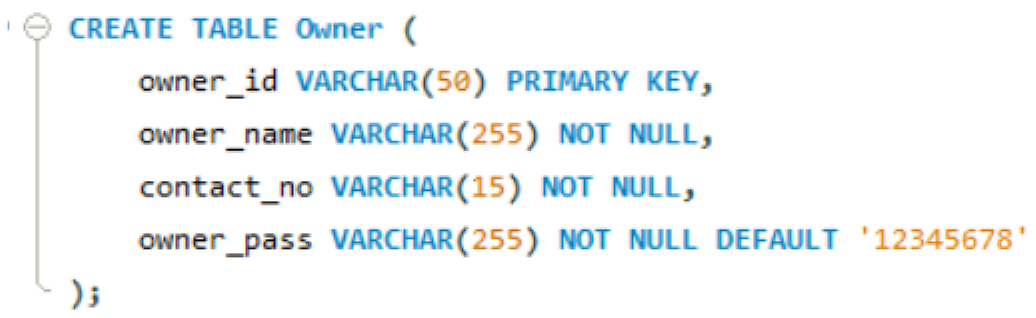


RELATIONAL SCHEMA

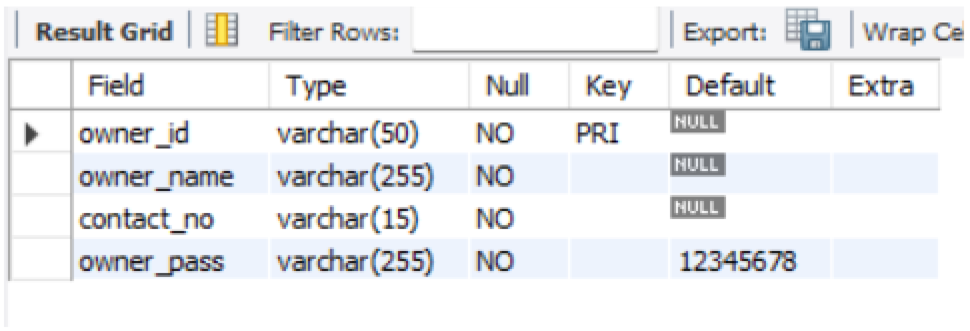


TABLES

1. Owner table

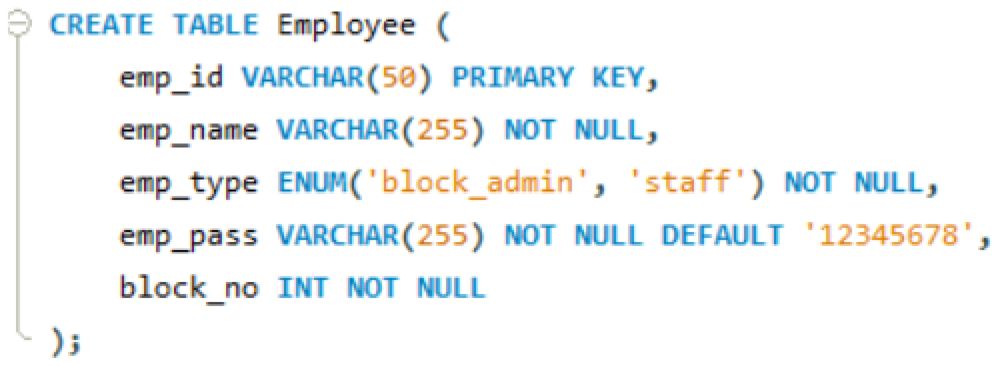


```
CREATE TABLE Owner (
    owner_id VARCHAR(50) PRIMARY KEY,
    owner_name VARCHAR(255) NOT NULL,
    contact_no VARCHAR(15) NOT NULL,
    owner_pass VARCHAR(255) NOT NULL DEFAULT '12345678'
);
```

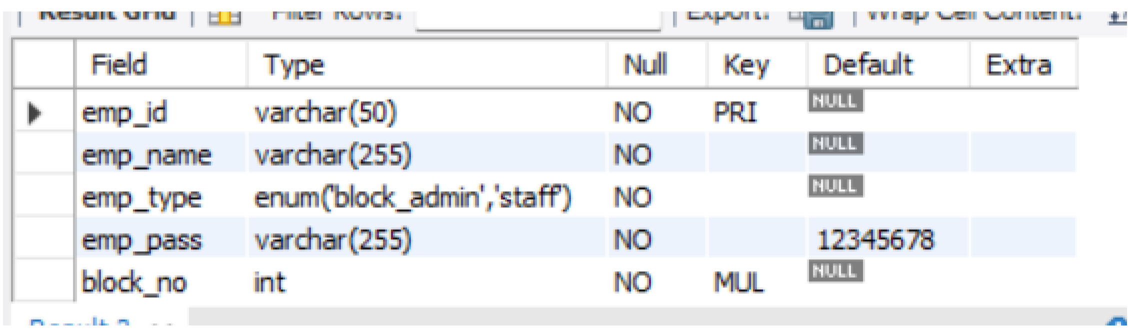


Field	Type	Null	Key	Default	Extra
owner_id	varchar(50)	NO	PRI	NULL	
owner_name	varchar(255)	NO		NULL	
contact_no	varchar(15)	NO		NULL	
owner_pass	varchar(255)	NO		12345678	

2. Employee table

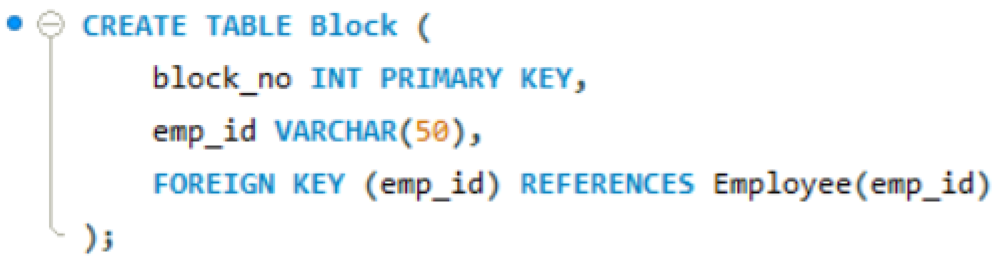


```
CREATE TABLE Employee (
    emp_id VARCHAR(50) PRIMARY KEY,
    emp_name VARCHAR(255) NOT NULL,
    emp_type ENUM('block_admin', 'staff') NOT NULL,
    emp_pass VARCHAR(255) NOT NULL DEFAULT '12345678',
    block_no INT NOT NULL
);
```

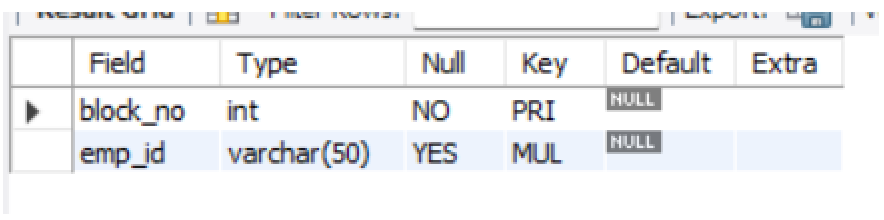


Field	Type	Null	Key	Default	Extra
emp_id	varchar(50)	NO	PRI	NULL	
emp_name	varchar(255)	NO		NULL	
emp_type	enum('block_admin','staff')	NO		NULL	
emp_pass	varchar(255)	NO		12345678	
block_no	int	NO	MUL	NULL	

3. Block Table

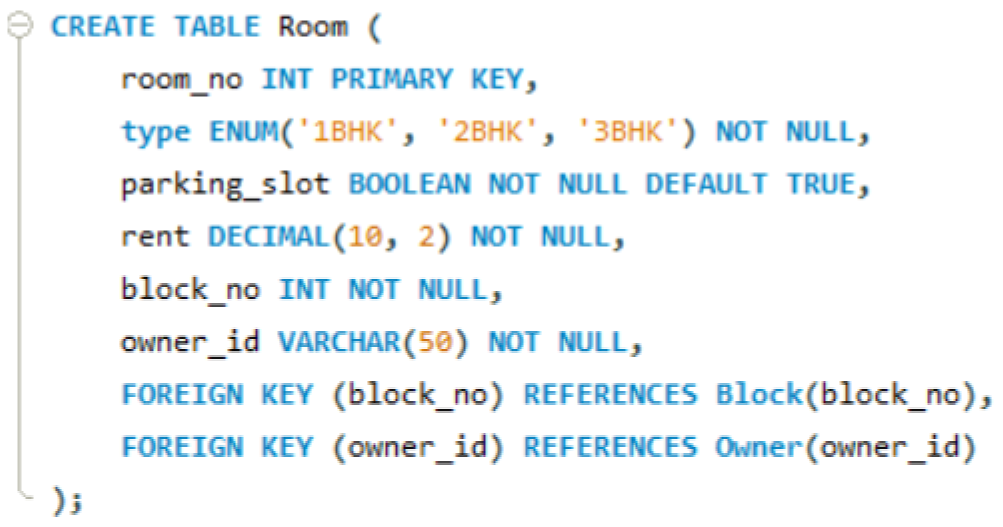


```
CREATE TABLE Block (
    block_no INT PRIMARY KEY,
    emp_id VARCHAR(50),
    FOREIGN KEY (emp_id) REFERENCES Employee(emp_id)
);
```

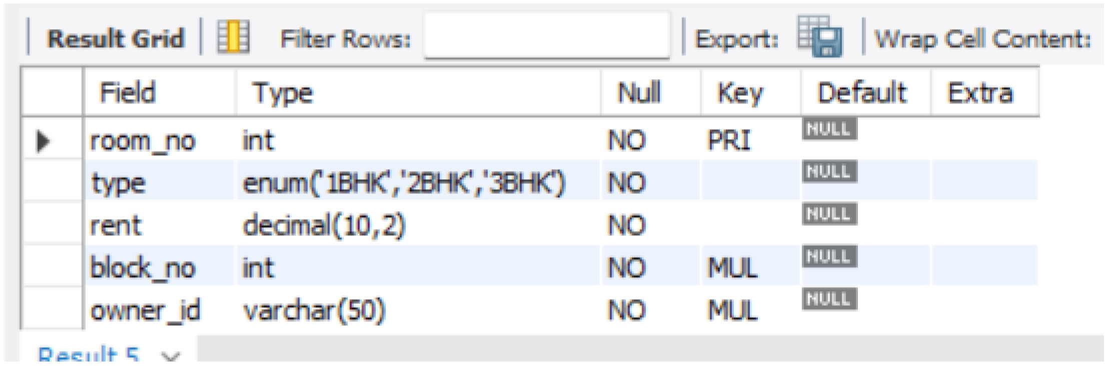


Field	Type	Null	Key	Default	Extra
block_no	int	NO	PRI	NULL	
emp_id	varchar(50)	YES	MUL	NULL	

4. Rooms Table



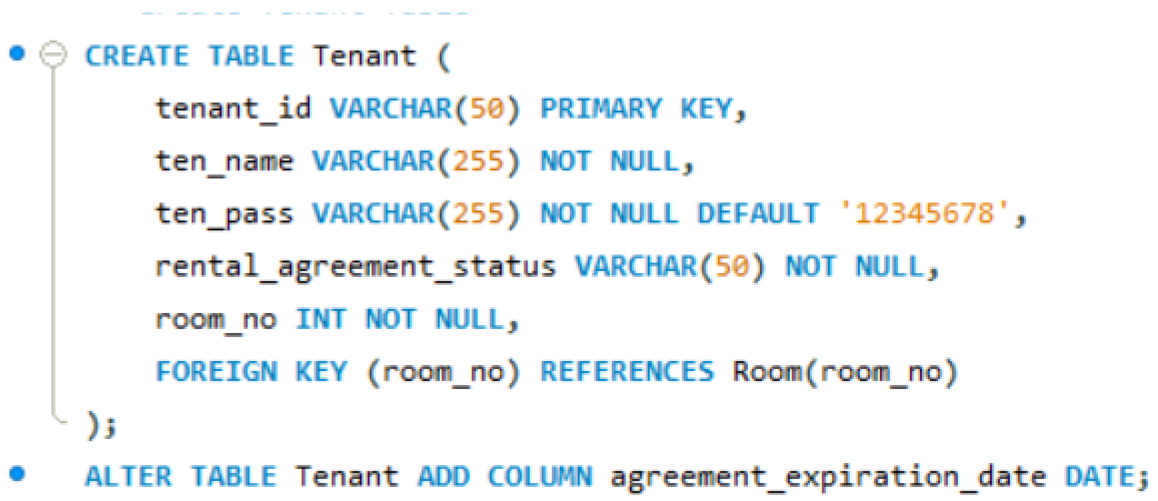
```
CREATE TABLE Room (
    room_no INT PRIMARY KEY,
    type ENUM('1BHK', '2BHK', '3BHK') NOT NULL,
    parking_slot BOOLEAN NOT NULL DEFAULT TRUE,
    rent DECIMAL(10, 2) NOT NULL,
    block_no INT NOT NULL,
    owner_id VARCHAR(50) NOT NULL,
    FOREIGN KEY (block_no) REFERENCES Block(block_no),
    FOREIGN KEY (owner_id) REFERENCES Owner(owner_id)
);
```

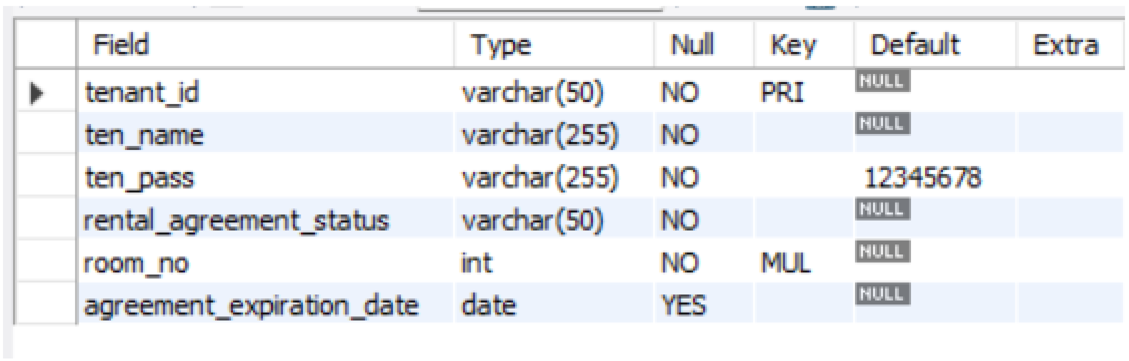

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	Field	Type	Null	Key	Default	Extra
▶	room_no	int	NO	PRI	NULL	
	type	enum('1BHK','2BHK','3BHK')	NO		NULL	
	rent	decimal(10,2)	NO		NULL	
	block_no	int	NO	MUL	NULL	
	owner_id	varchar(50)	NO	MUL	NULL	

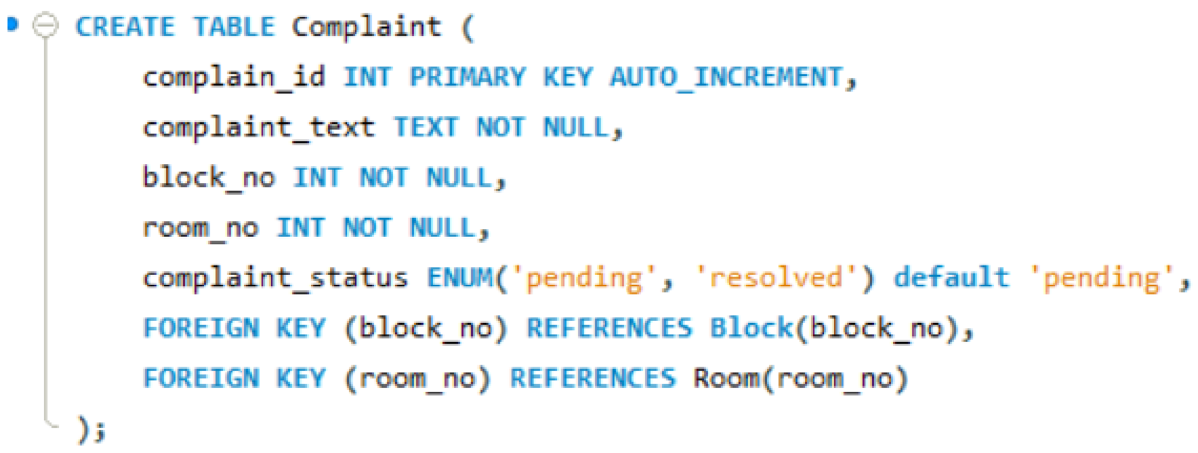
Result 5

5. Tenant Table

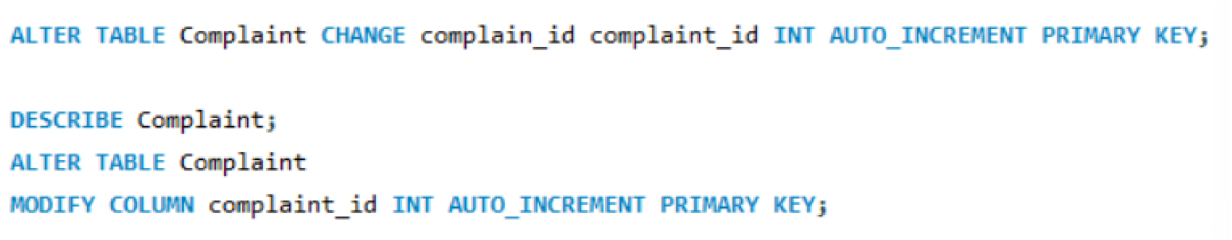

```
CREATE TABLE Tenant (
    tenant_id VARCHAR(50) PRIMARY KEY,
    ten_name VARCHAR(255) NOT NULL,
    ten_pass VARCHAR(255) NOT NULL DEFAULT '12345678',
    rental_agreement_status VARCHAR(50) NOT NULL,
    room_no INT NOT NULL,
    FOREIGN KEY (room_no) REFERENCES Room(room_no)
);
ALTER TABLE Tenant ADD COLUMN agreement_expiration_date DATE;
```


	Field	Type	Null	Key	Default	Extra
▶	tenant_id	varchar(50)	NO	PRI	NULL	
	ten_name	varchar(255)	NO		NULL	
	ten_pass	varchar(255)	NO		12345678	
	rental_agreement_status	varchar(50)	NO		NULL	
	room_no	int	NO	MUL	NULL	
	agreement_expiration_date	date	YES		NULL	

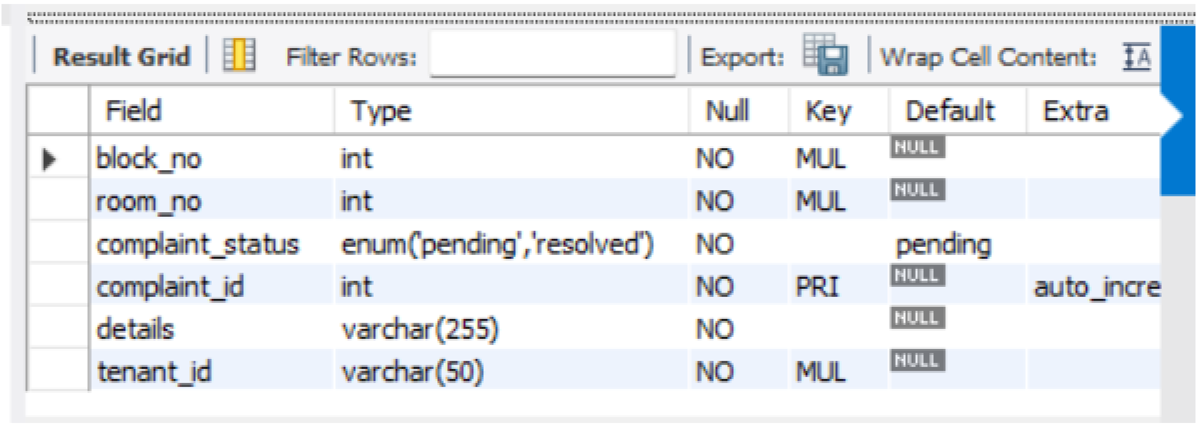
6. Complaint Table



```
CREATE TABLE Complaint (
    complain_id INT PRIMARY KEY AUTO_INCREMENT,
    complaint_text TEXT NOT NULL,
    block_no INT NOT NULL,
    room_no INT NOT NULL,
    complaint_status ENUM('pending', 'resolved') default 'pending',
    FOREIGN KEY (block_no) REFERENCES Block(block_no),
    FOREIGN KEY (room_no) REFERENCES Room(room_no)
);
```

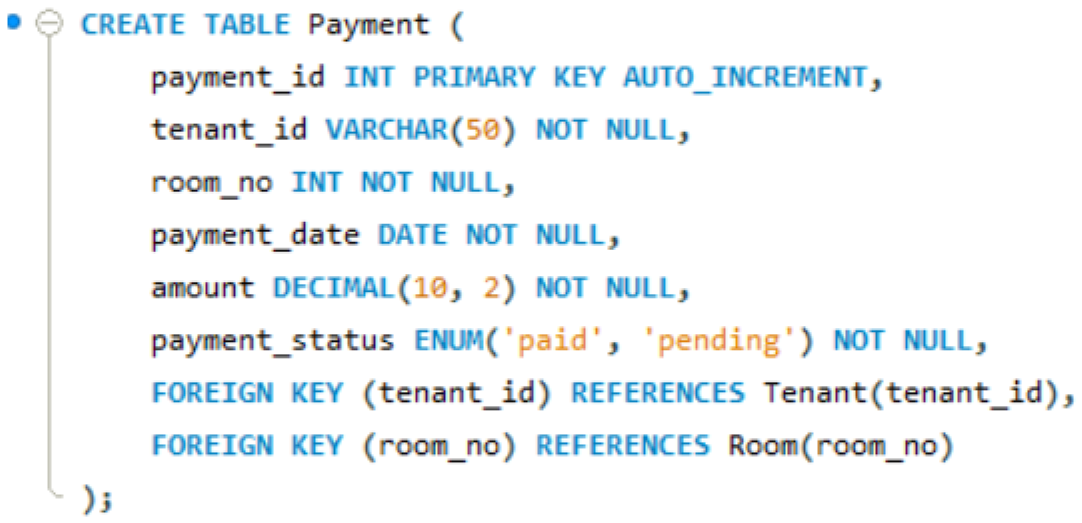


```
ALTER TABLE Complaint CHANGE complain_id complaint_id INT AUTO_INCREMENT PRIMARY KEY;

DESCRIBE Complaint;
ALTER TABLE Complaint
MODIFY COLUMN complaint_id INT AUTO_INCREMENT PRIMARY KEY;
```

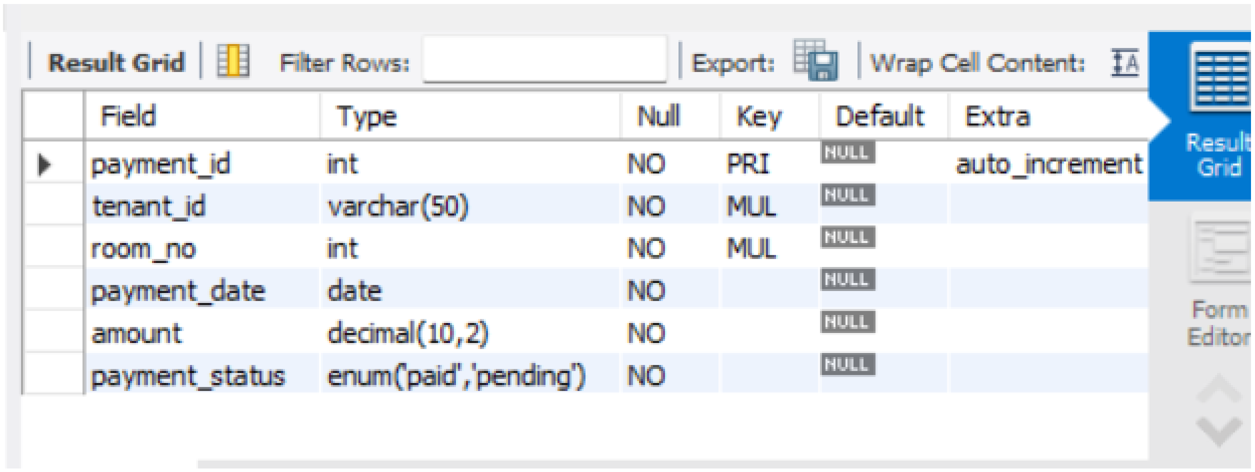


Field	Type	Null	Key	Default	Extra
block_no	int	NO	MUL	NULL	
room_no	int	NO	MUL	NULL	
complaint_status	enum('pending','resolved')	NO		pending	
complaint_id	int	NO	PRI	NULL	auto_incre
details	varchar(255)	NO		NULL	
tenant_id	varchar(50)	NO	MUL	NULL	

7. Payment Table

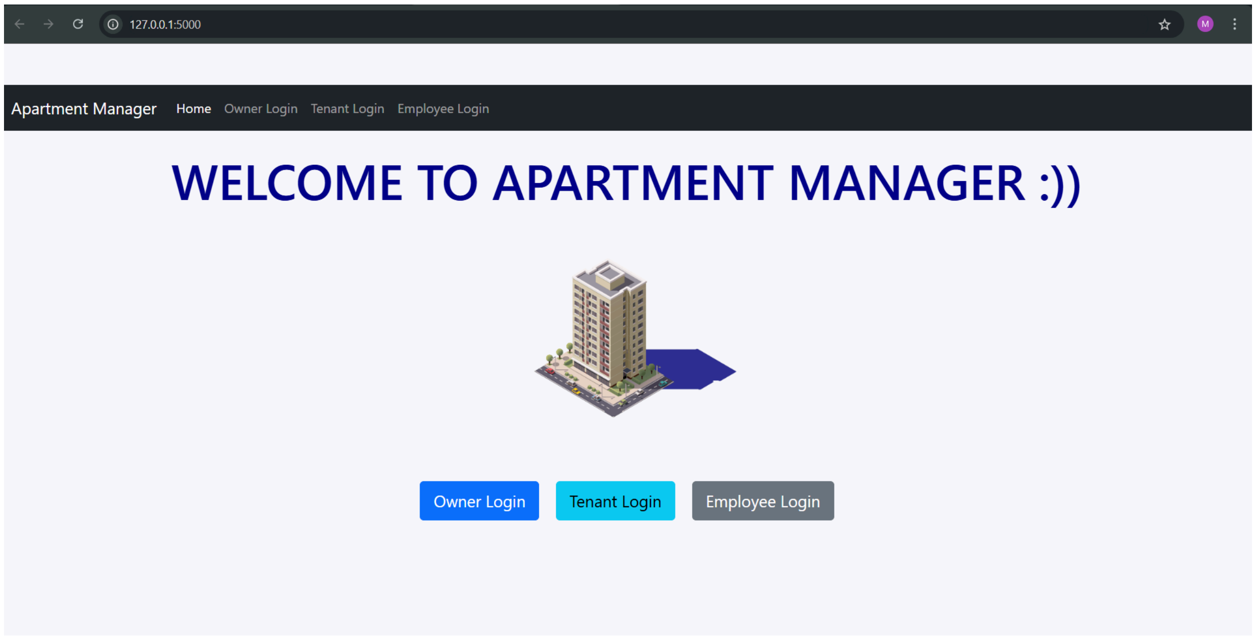


```
CREATE TABLE Payment (
    payment_id INT PRIMARY KEY AUTO_INCREMENT,
    tenant_id VARCHAR(50) NOT NULL,
    room_no INT NOT NULL,
    payment_date DATE NOT NULL,
    amount DECIMAL(10, 2) NOT NULL,
    payment_status ENUM('paid', 'pending') NOT NULL,
    FOREIGN KEY (tenant_id) REFERENCES Tenant(tenant_id),
    FOREIGN KEY (room_no) REFERENCES Room(room_no)
);
```

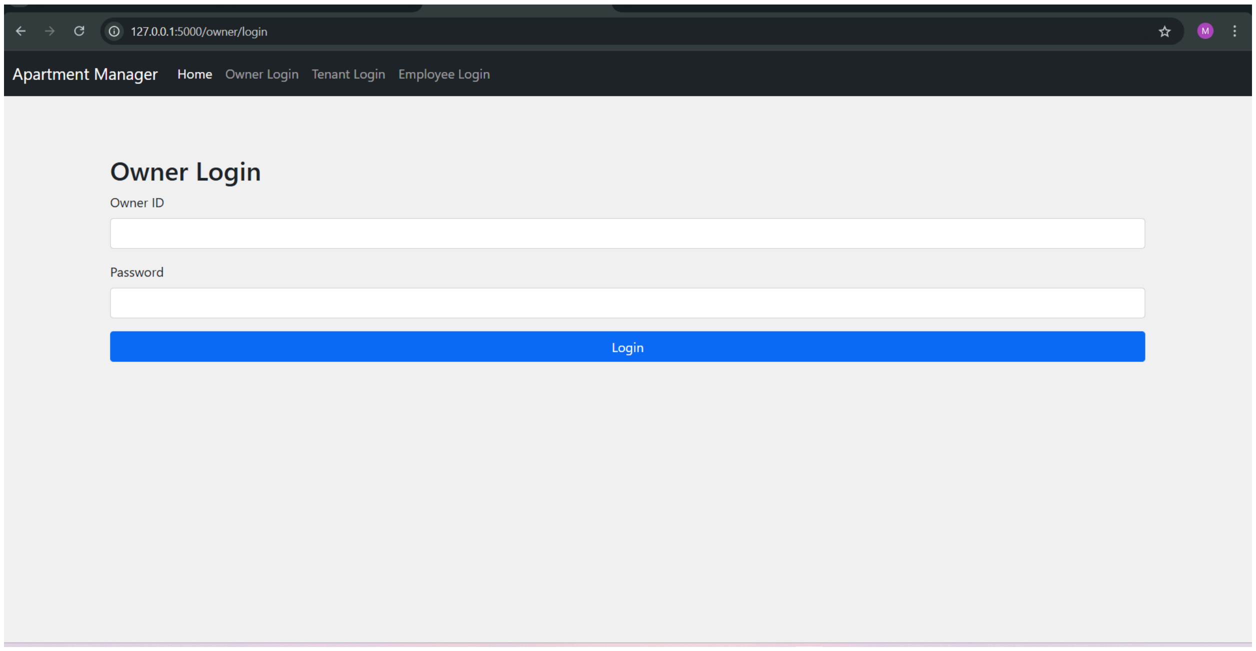


Field	Type	Null	Key	Default	Extra
payment_id	int	NO	PRI	NULL	auto_increment
tenant_id	varchar(50)	NO	MUL	NULL	
room_no	int	NO	MUL	NULL	
payment_date	date	NO		NULL	
amount	decimal(10,2)	NO		NULL	
payment_status	enum('paid','pending')	NO		NULL	

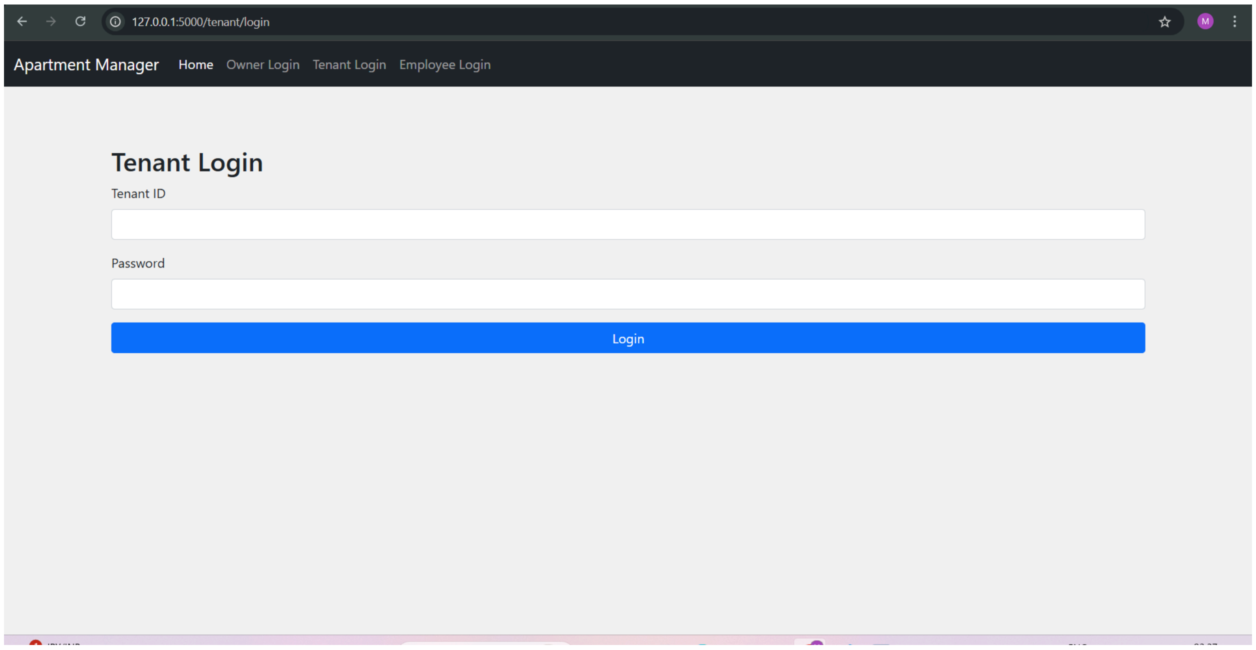
HOME PAGE



OWNER LOGIN



TENANT LOGIN


127.0.0.1:5000/tenant/login

Apartment Manager Home Owner Login Tenant Login Employee Login

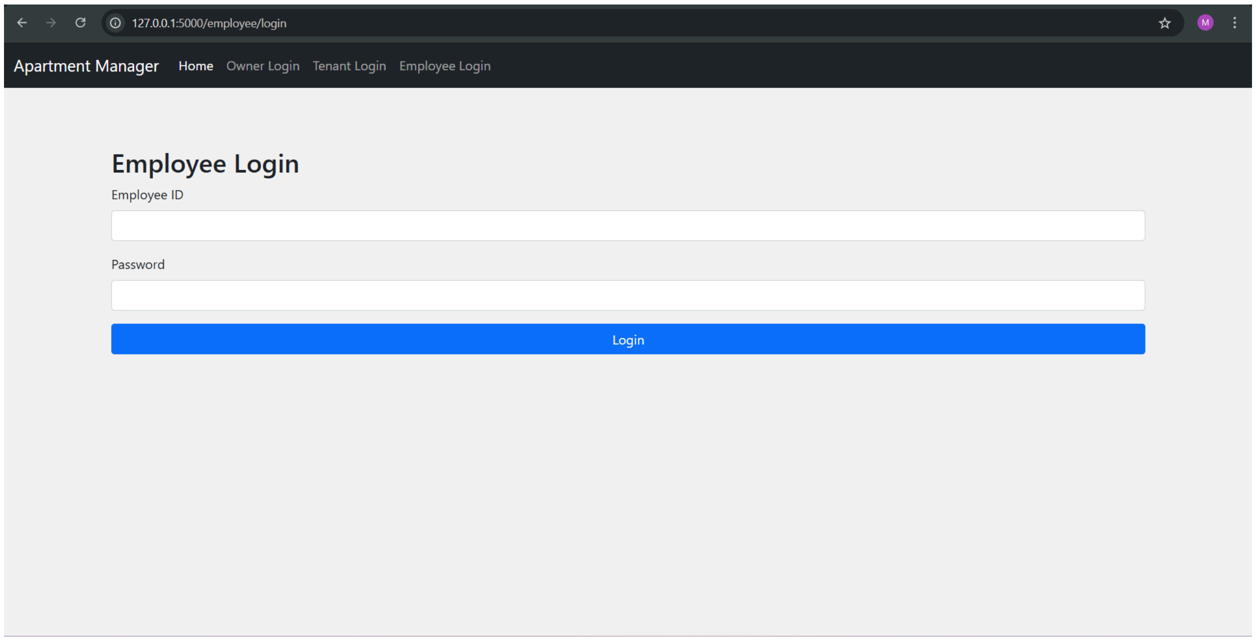
Tenant Login

Tenant ID

Password

Login

EMPLOYEE LOGIN


127.0.0.1:5000/employee/login

Apartment Manager Home Owner Login Tenant Login Employee Login

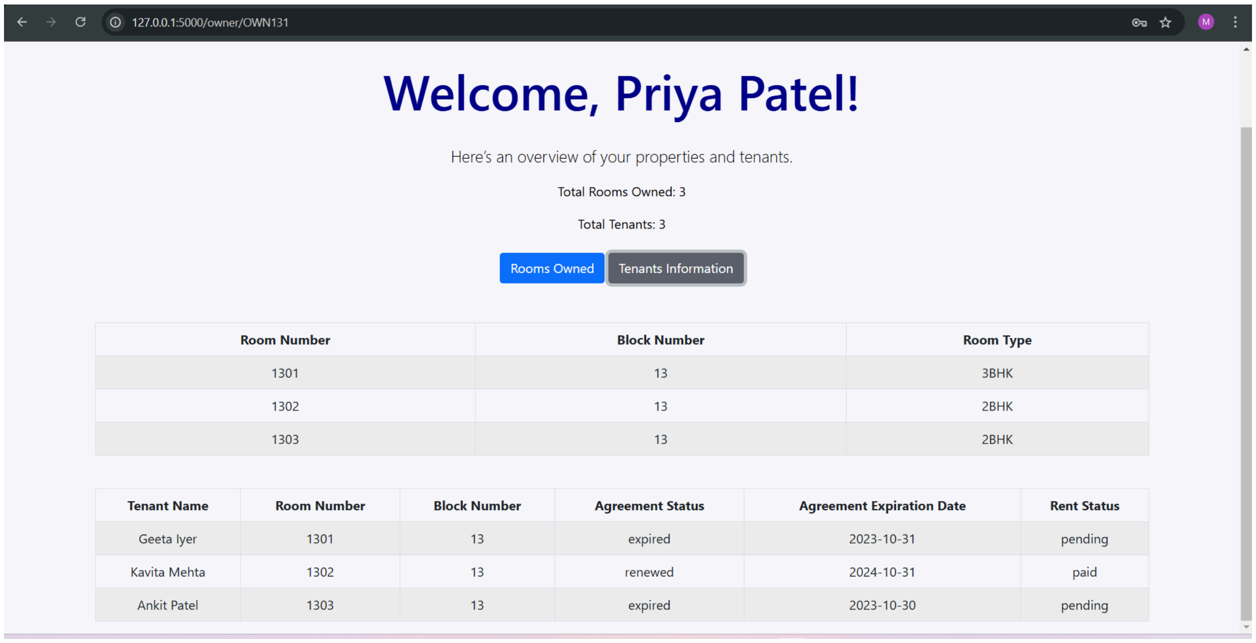
Employee Login

Employee ID

Password

Login

OWNER DASHBOARD



127.0.0.1:5000/owner/OWN131

Welcome, Priya Patel!

Here's an overview of your properties and tenants.

Total Rooms Owned: 3

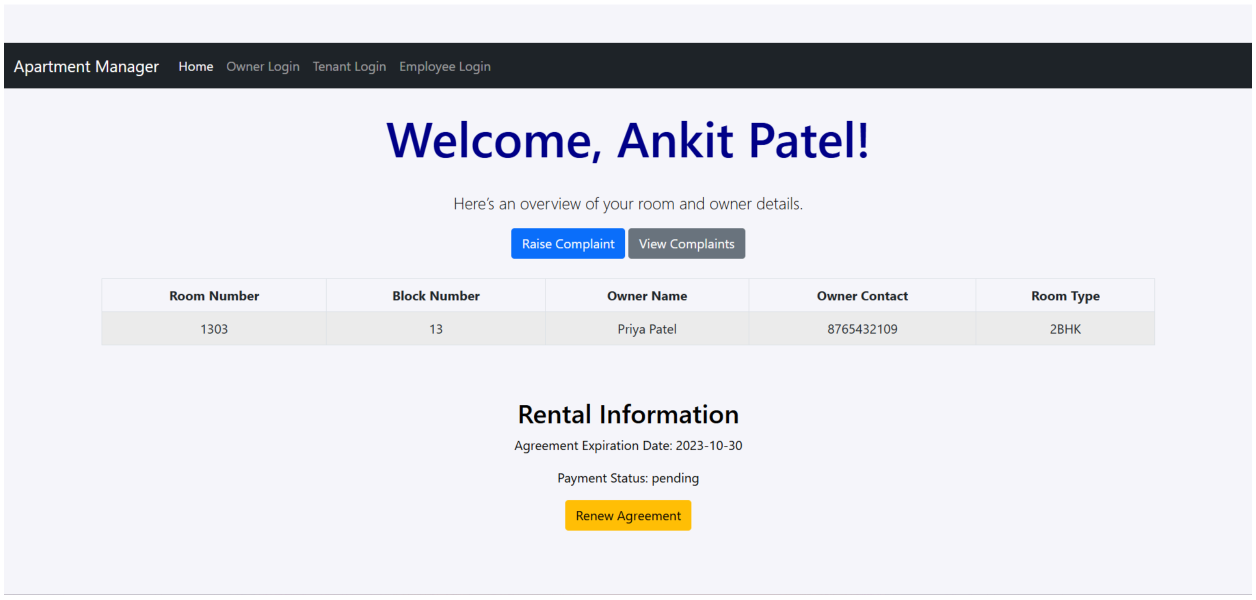
Total Tenants: 3

Room Number	Block Number	Room Type
1301	13	3BHK
1302	13	2BHK
1303	13	2BHK

Tenant Name	Room Number	Block Number	Agreement Status	Agreement Expiration Date	Rent Status
Geeta Iyer	1301	13	expired	2023-10-31	pending
Kavita Mehta	1302	13	renewed	2024-10-31	paid
Ankit Patel	1303	13	expired	2023-10-30	pending

TENANT DASHBOARD

rental agreement status = expired & payment status = pending



Welcome, Ankit Patel!

Here's an overview of your room and owner details.

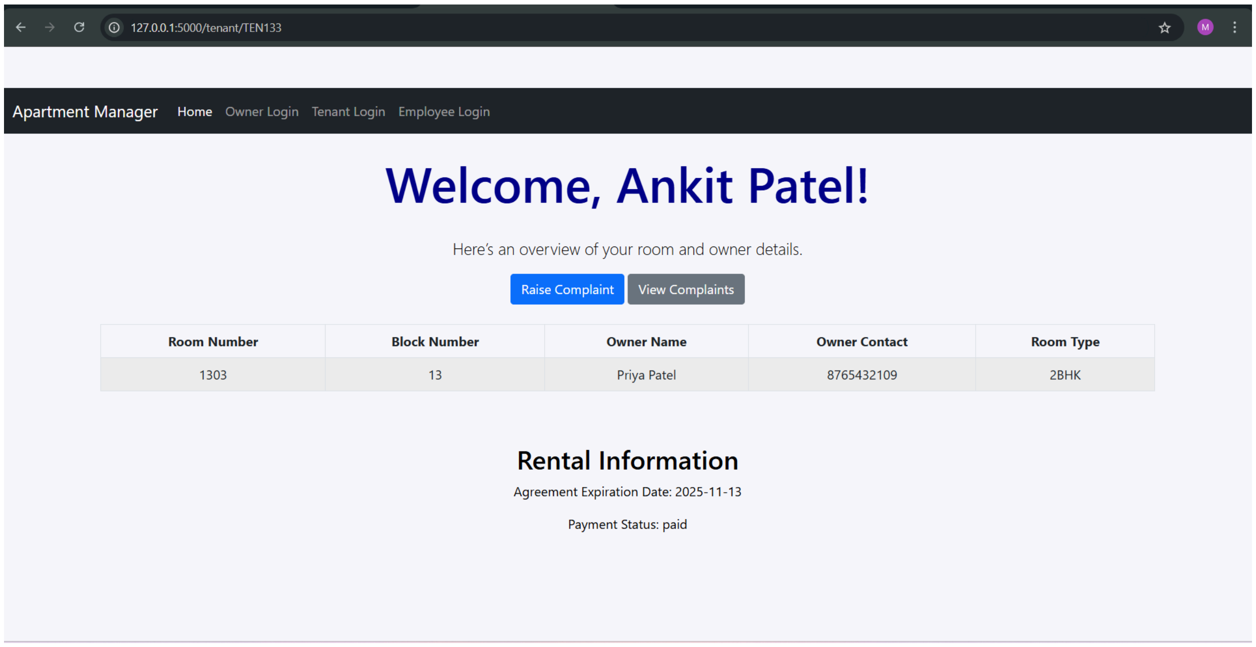
Room Number	Block Number	Owner Name	Owner Contact	Room Type
1303	13	Priya Patel	8765432109	2BHK

Rental Information

Agreement Expiration Date: 2023-10-30

Payment Status: pending

rental agreement status = renewed & payment status = paid


127.0.0.1:5000/tenant/TEN133

Welcome, Ankit Patel!

Here's an overview of your room and owner details.

Room Number	Block Number	Owner Name	Owner Contact	Room Type
1303	13	Priya Patel	8765432109	2BHK

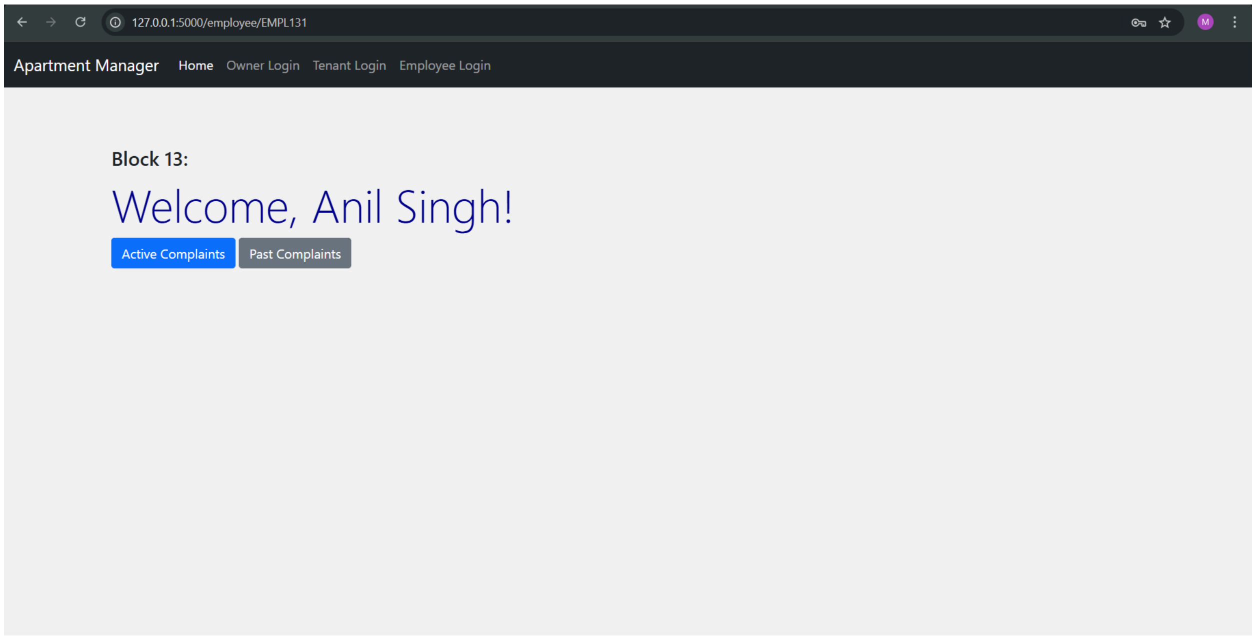
Rental Information

Agreement Expiration Date: 2025-11-13

Payment Status: paid

<u>EMPLOYEE DASHBOARD</u>

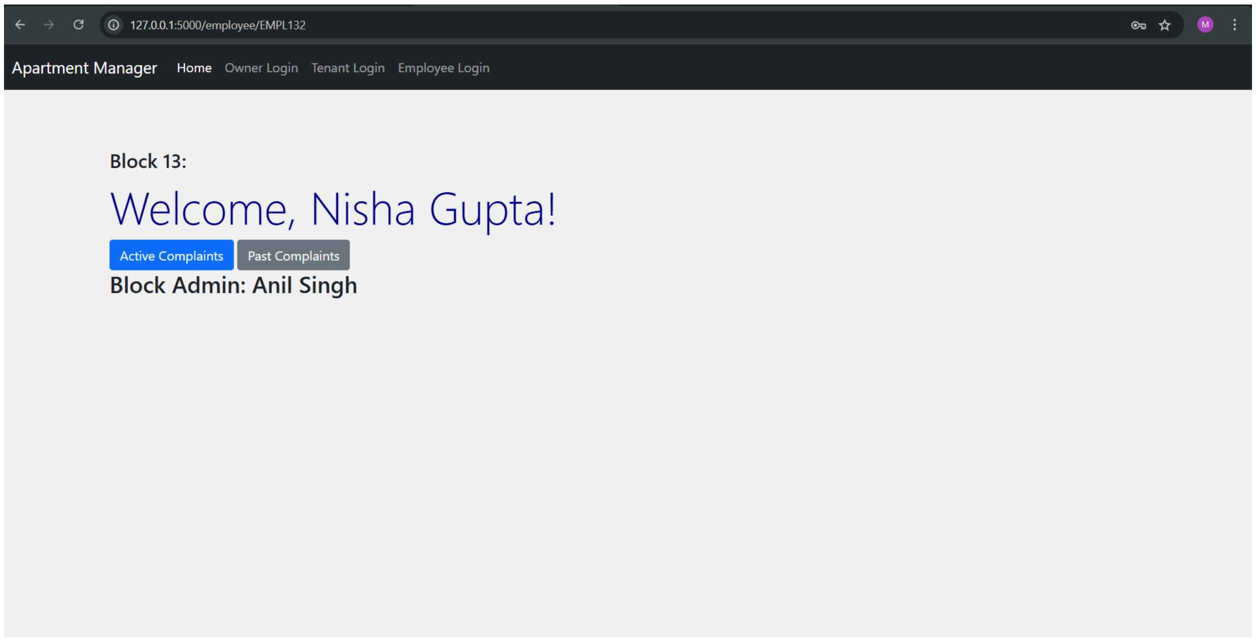
For Block Admin


127.0.0.1:5000/employee/EMPL131

Block 13:

Welcome, Anil Singh!

For staff working in that block


127.0.0.1:5000/employee/EMPL132

Apartment Manager Home Owner Login Tenant Login Employee Login

Block 13:

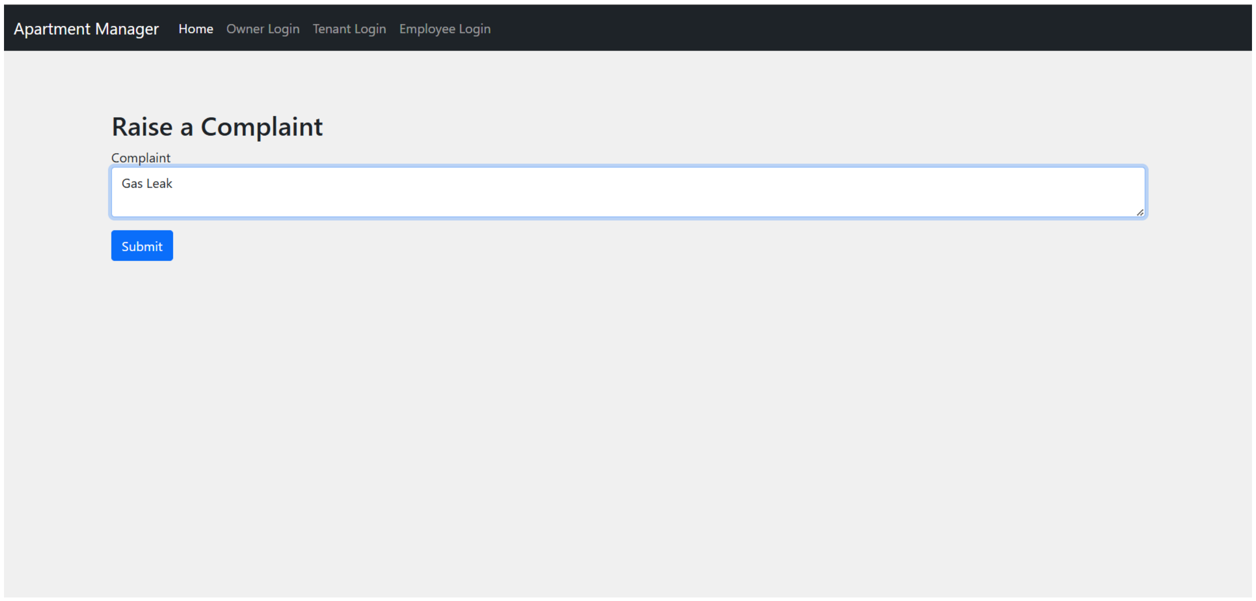
Welcome, Nisha Gupta!

Active Complaints Past Complaints

Block Admin: Anil Singh

COMPLAINTS

Raise complaints (Tenant)


Apartment Manager Home Owner Login Tenant Login Employee Login

Raise a Complaint

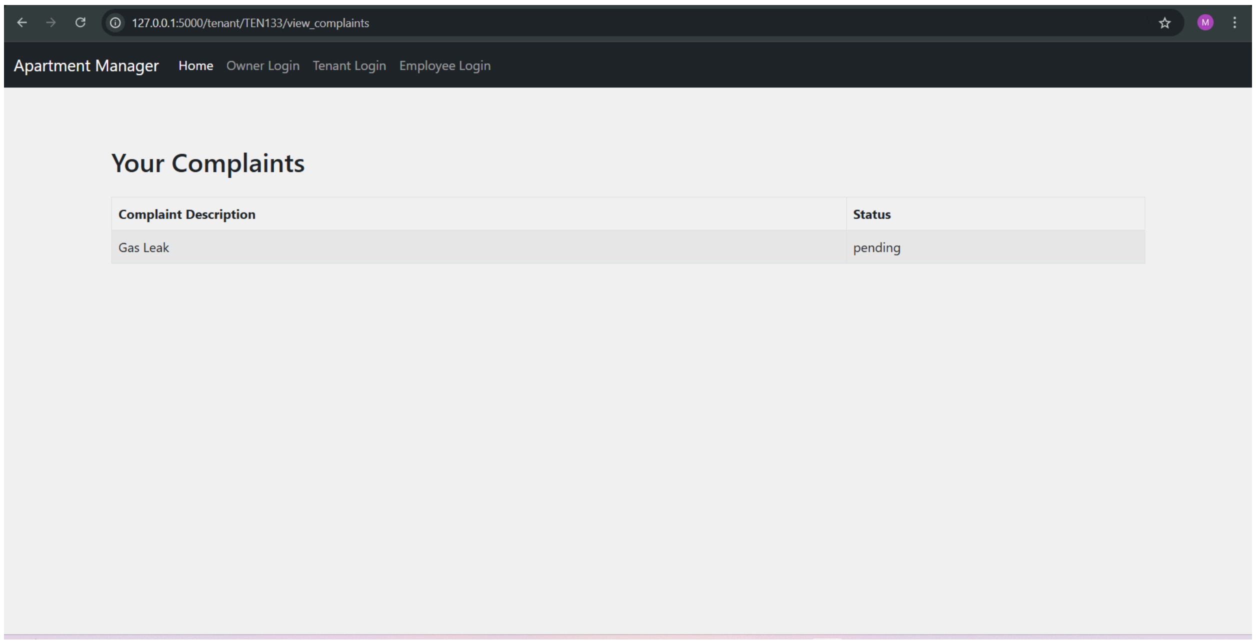
Complaint

Gas Leak

Submit

View complaint (Tenant)

Screenshot 1:



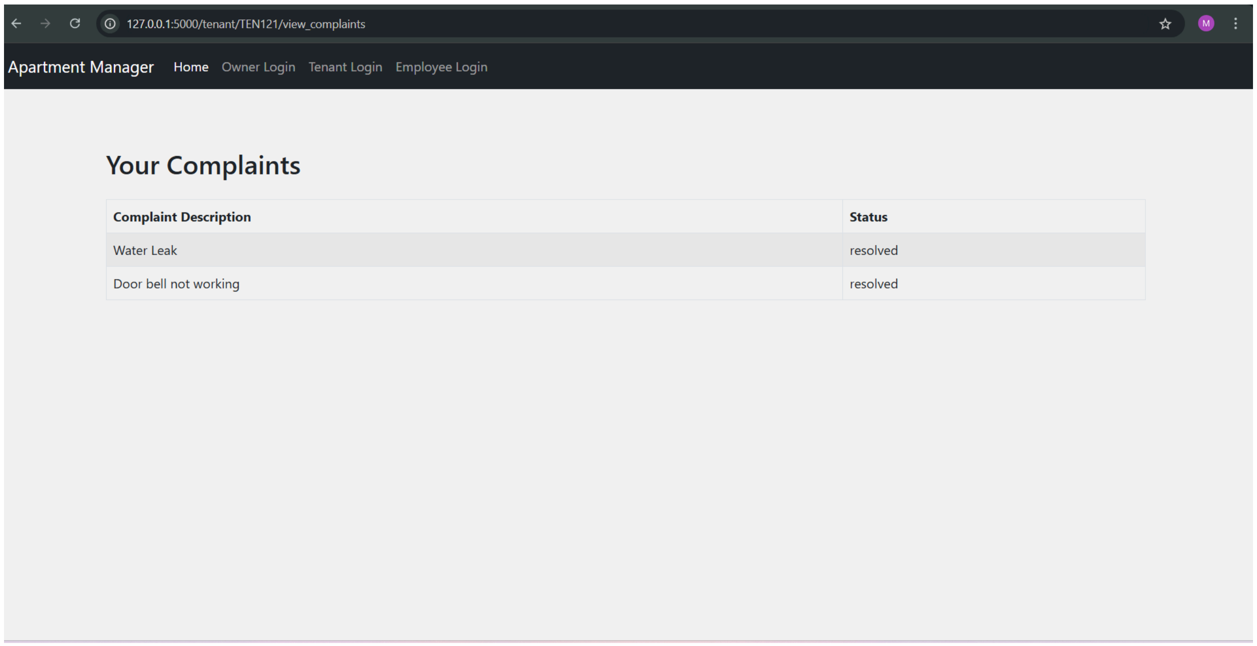
127.0.0.1:5000/tenant/TEN133/view_complaints

Apartment Manager Home Owner Login Tenant Login Employee Login

Your Complaints

Complaint Description	Status
Gas Leak	pending

Screenshot 2:



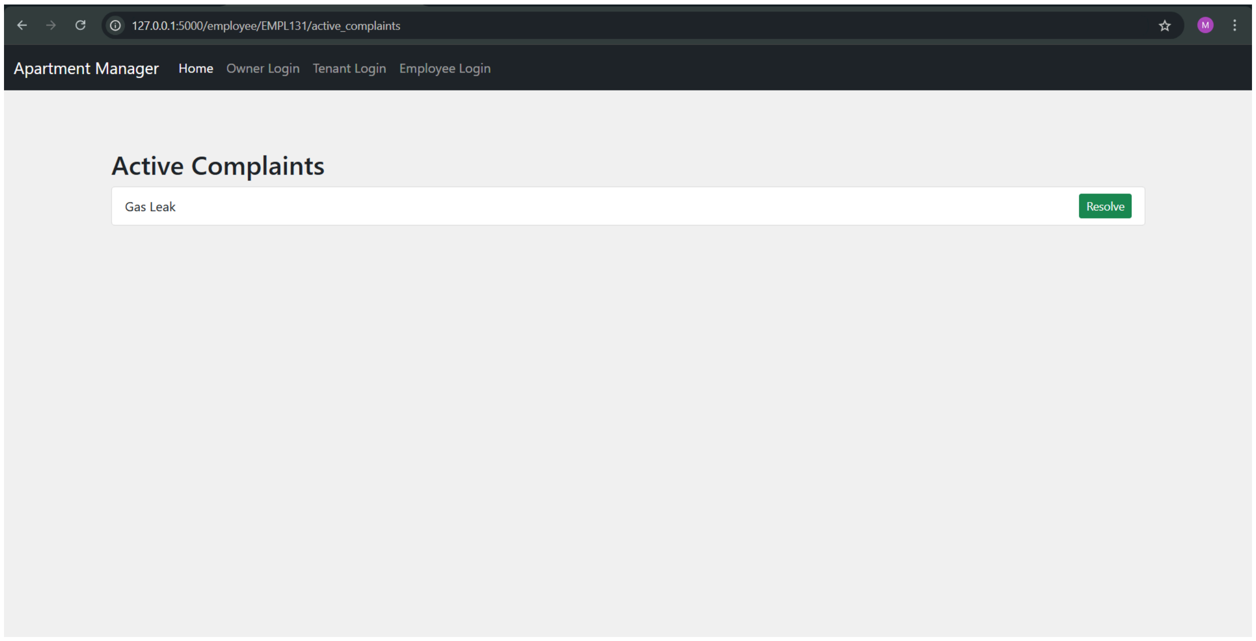
127.0.0.1:5000/tenant/TEN121/view_complaints

Apartment Manager Home Owner Login Tenant Login Employee Login

Your Complaints

Complaint Description	Status
Water Leak	resolved
Door bell not working	resolved

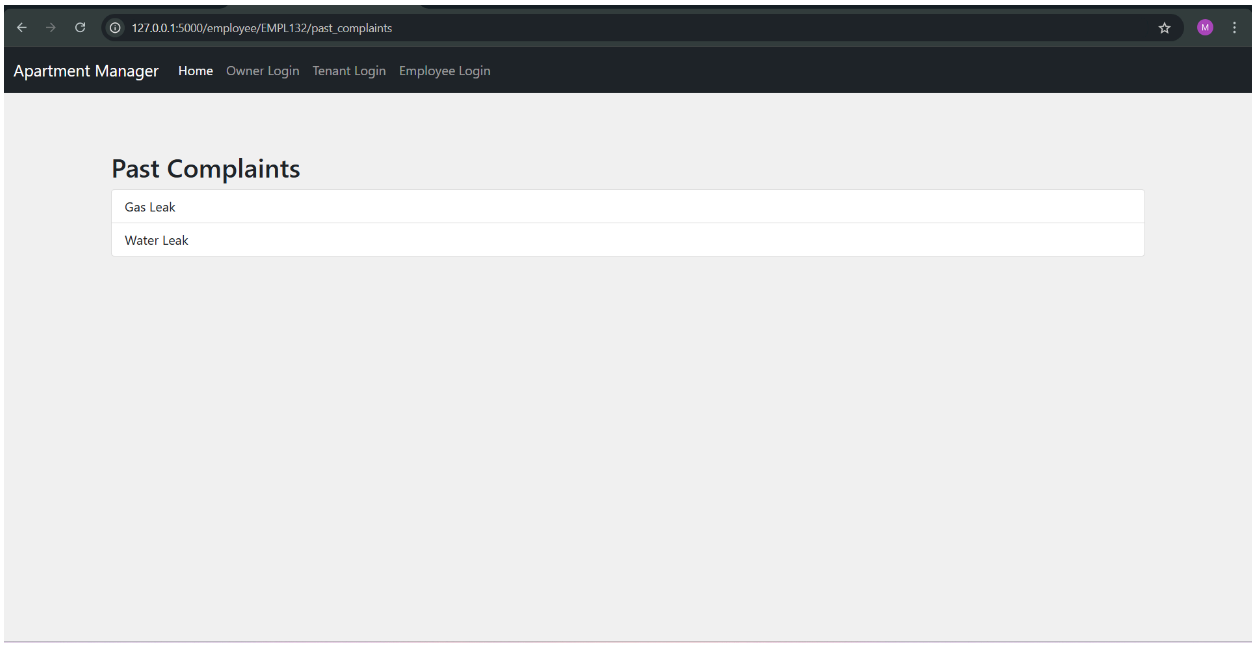
Active complaints (Employee)


127.0.0.1:5000/employee/EMPL131/active_complaints

Active Complaints

Gas Leak

Past complaints (Employee)

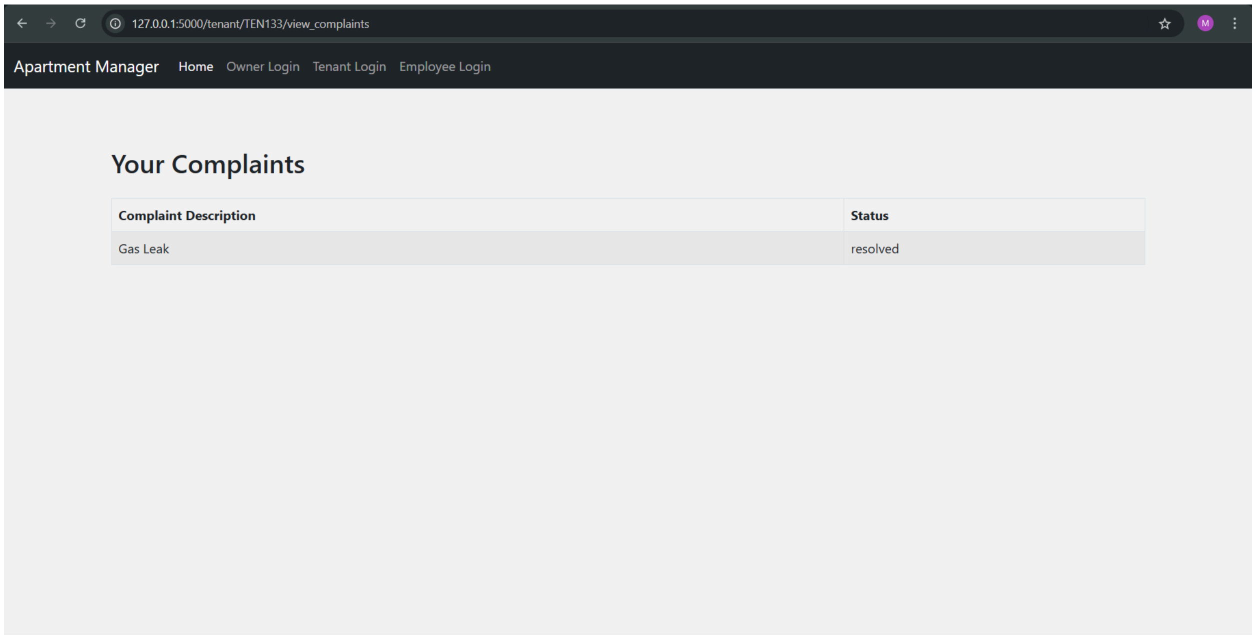

127.0.0.1:5000/employee/EMPL132/past_complaints

Past Complaints

Gas Leak

Water Leak

Complaint Resolved (Tenant)

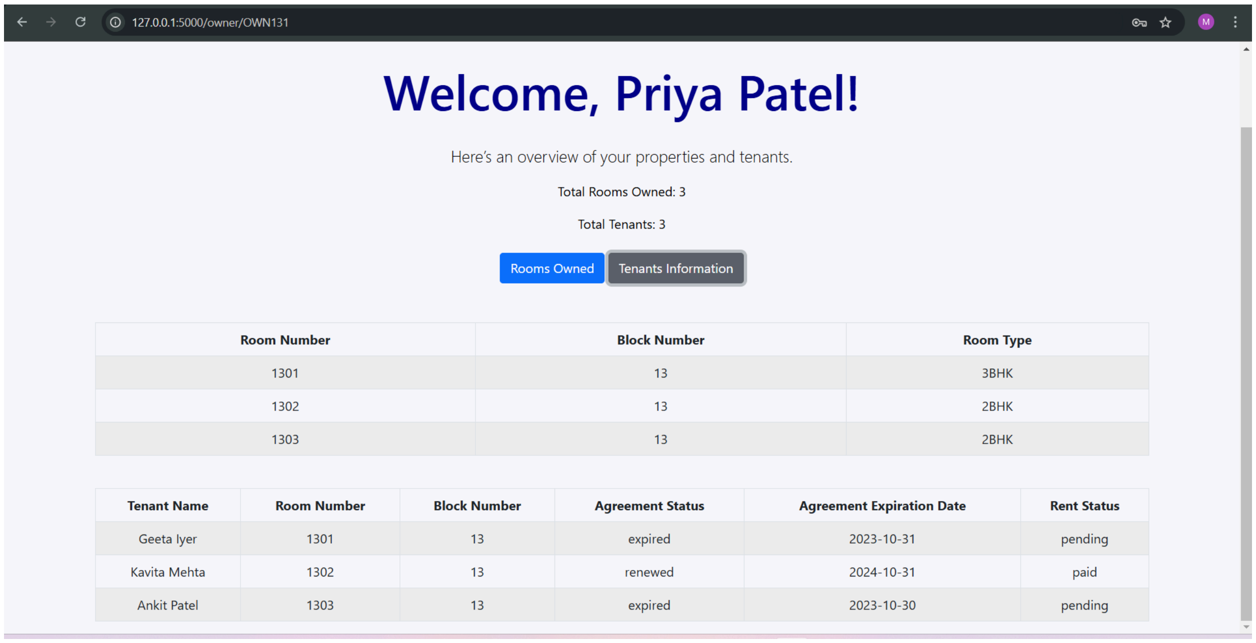


Your Complaints

Complaint Description	Status
Gas Leak	resolved

PAYMENTS

Ankit Patel: Agreement status expired and rent status pending



Welcome, Priya Patel!

Here's an overview of your properties and tenants.

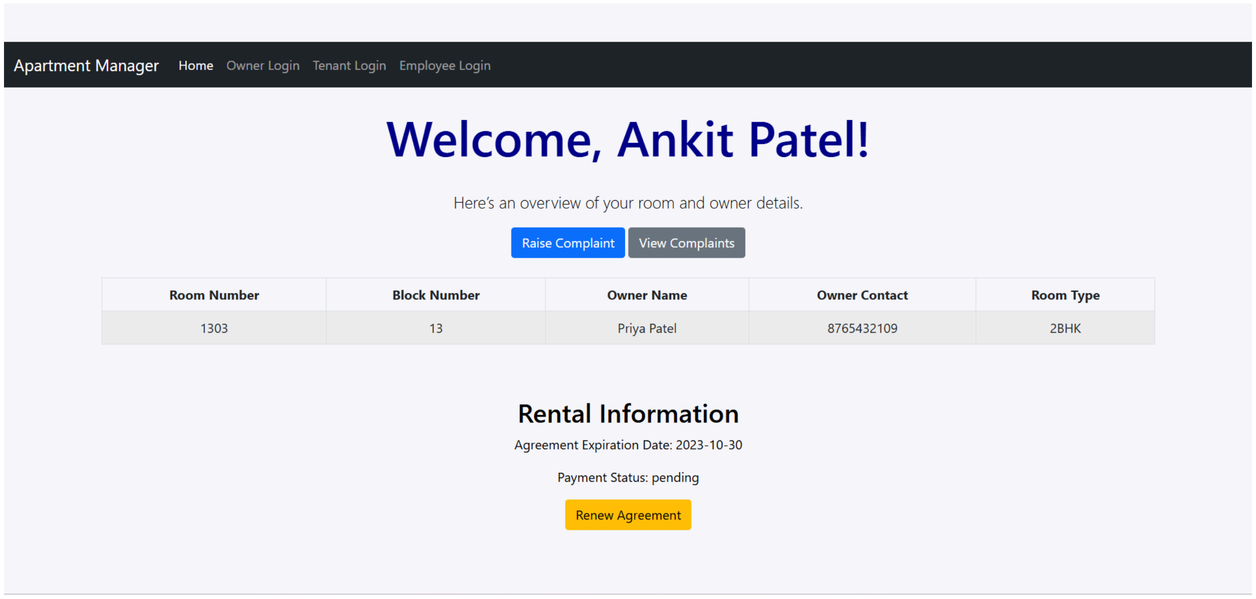
Total Rooms Owned: 3

Total Tenants: 3

Room Number	Block Number	Room Type
1301	13	3BHK
1302	13	2BHK
1303	13	2BHK

Tenant Name	Room Number	Block Number	Agreement Status	Agreement Expiration Date	Rent Status
Geeta Iyer	1301	13	expired	2023-10-31	pending
Kavita Mehta	1302	13	renewed	2024-10-31	paid
Ankit Patel	1303	13	expired	2023-10-30	pending

Renew agreement and make payment as tenant



Welcome, Ankit Patel!

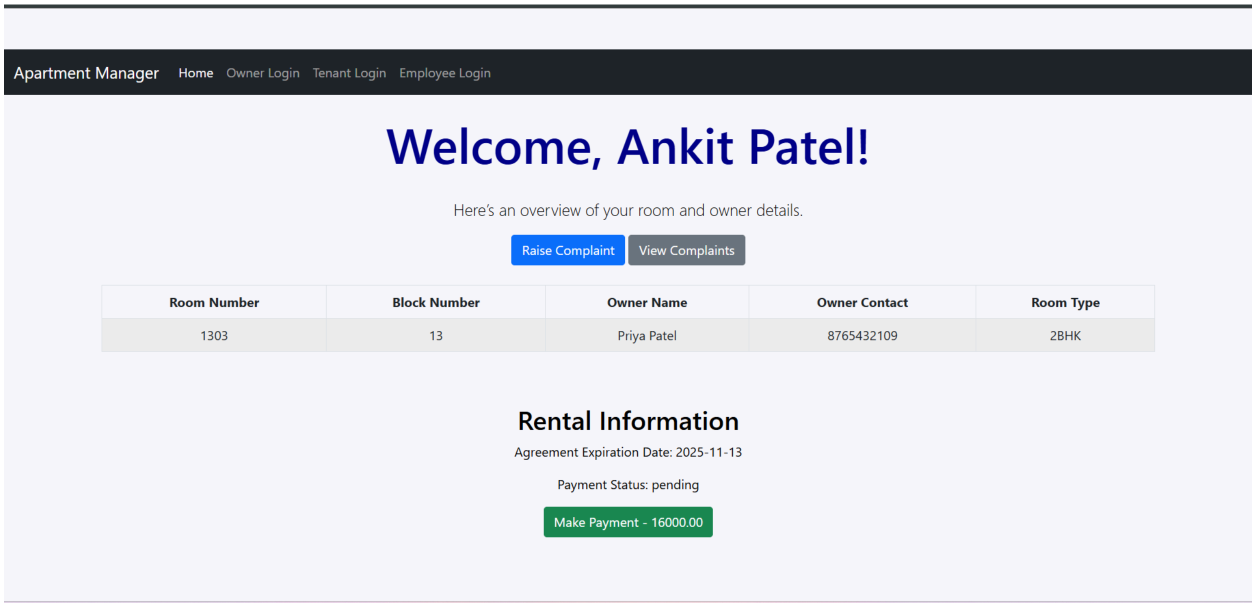
Here's an overview of your room and owner details.

Room Number	Block Number	Owner Name	Owner Contact	Room Type
1303	13	Priya Patel	8765432109	2BHK

Rental Information

Agreement Expiration Date: 2023-10-30

Payment Status: pending



Welcome, Ankit Patel!

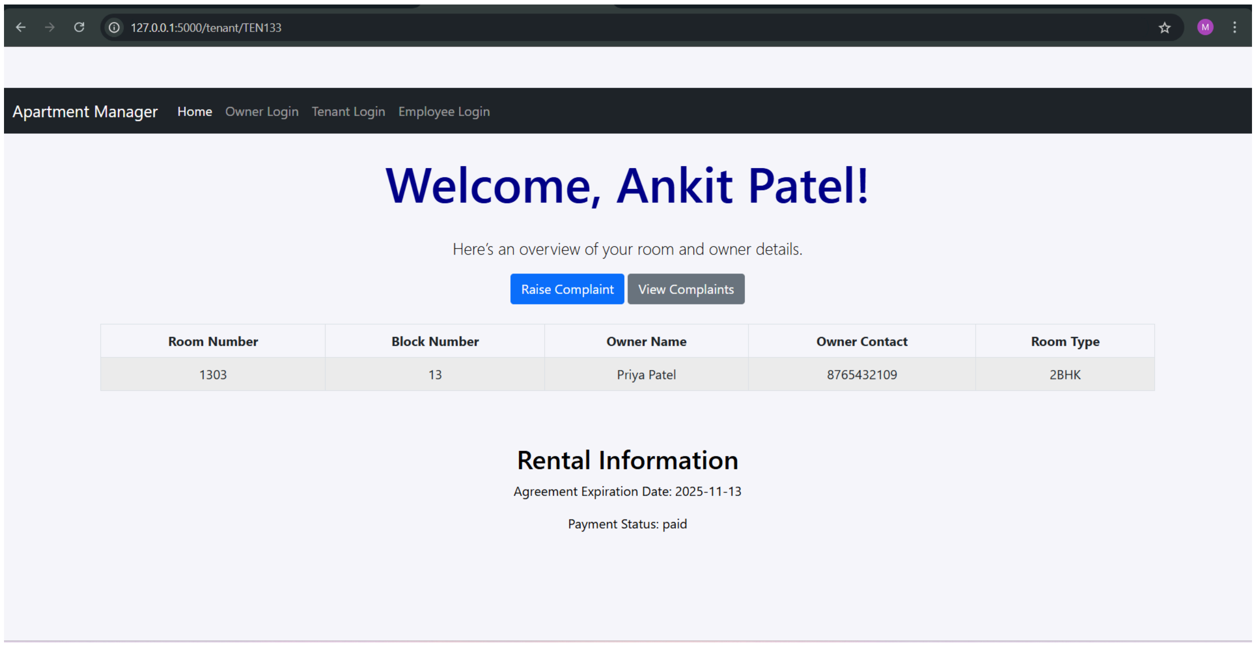
Here's an overview of your room and owner details.

Room Number	Block Number	Owner Name	Owner Contact	Room Type
1303	13	Priya Patel	8765432109	2BHK

Rental Information

Agreement Expiration Date: 2025-11-13

Payment Status: pending


127.0.0.1:5000/tenant/TEN133

Apartment Manager Home Owner Login Tenant Login Employee Login

Welcome, Ankit Patel!

Here's an overview of your room and owner details.

Raise Complaint View Complaints

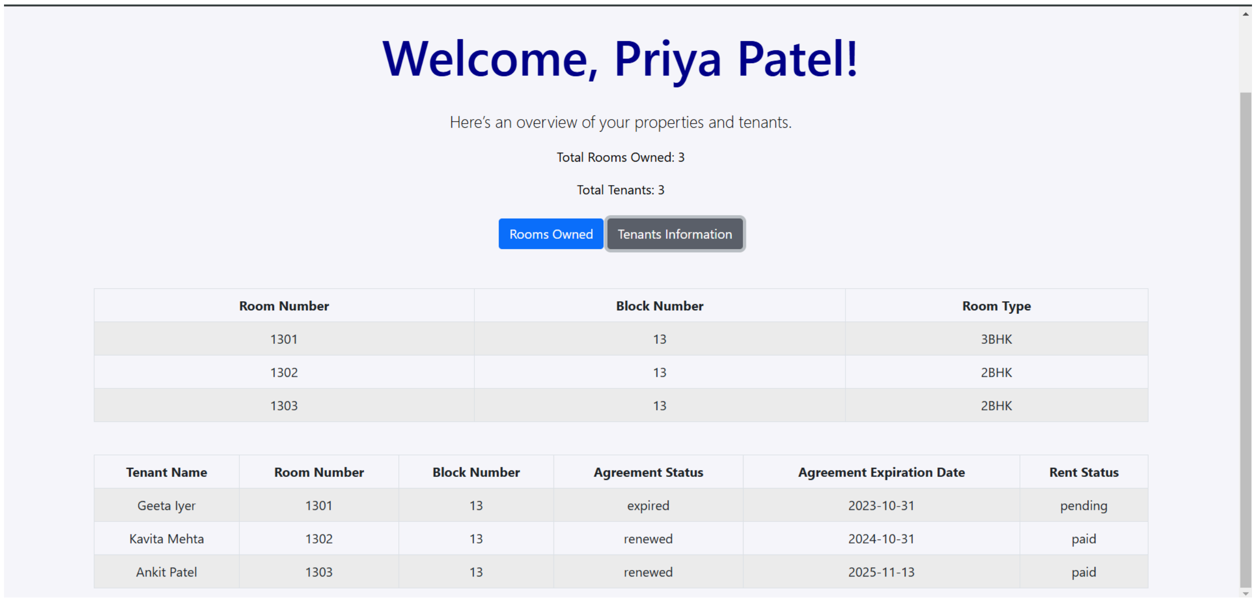
Room Number	Block Number	Owner Name	Owner Contact	Room Type
1303	13	Priya Patel	8765432109	2BHK

Rental Information

Agreement Expiration Date: 2025-11-13

Payment Status: paid

Status updated for Ankit Patel (tenant) on owner dashboard


Welcome, Priya Patel!

Here's an overview of your properties and tenants.

Total Rooms Owned: 3

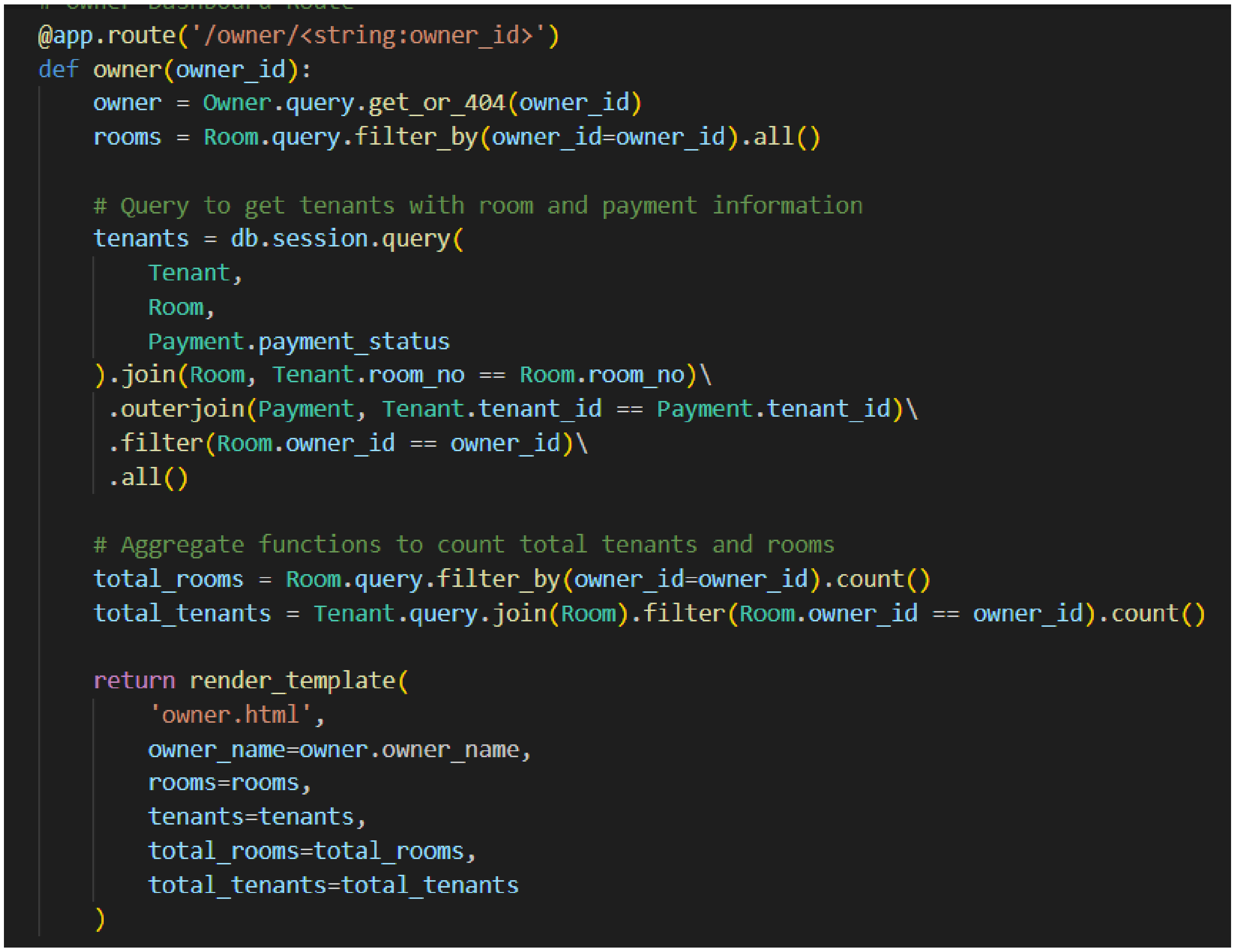
Total Tenants: 3

Rooms Owned Tenants Information

Room Number	Block Number	Room Type
1301	13	3BHK
1302	13	2BHK
1303	13	2BHK

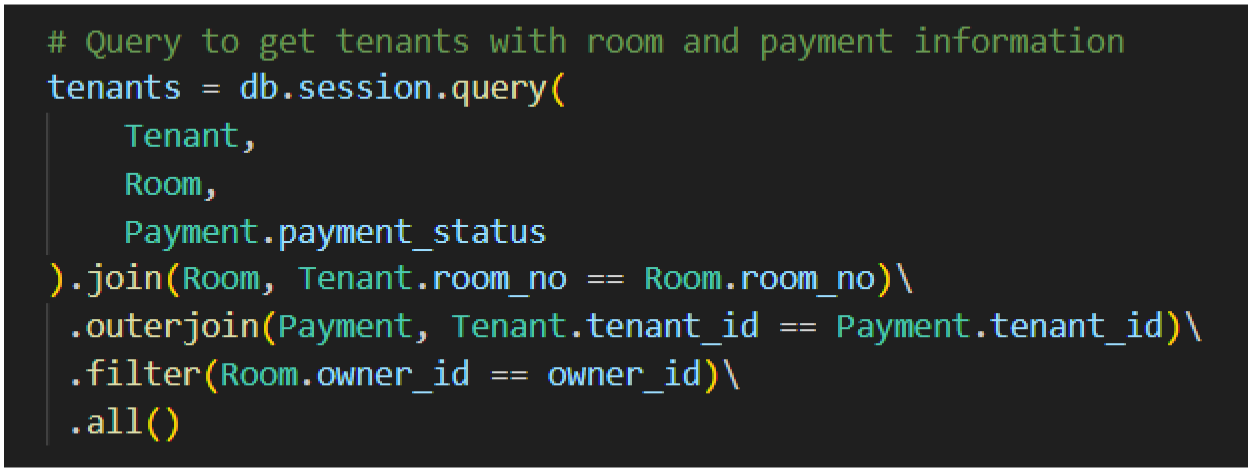
Tenant Name	Room Number	Block Number	Agreement Status	Agreement Expiration Date	Rent Status
Geeta Iyer	1301	13	expired	2023-10-31	pending
Kavita Mehta	1302	13	renewed	2024-10-31	paid
Ankit Patel	1303	13	renewed	2025-11-13	paid

Owner dashboard route(hello.py file)



```python
@app.route('/owner/<string:owner_id>')
def owner(owner_id):
    owner = Owner.query.get_or_404(owner_id)
    rooms = Room.query.filter_by(owner_id=owner_id).all()

    # Query to get tenants with room and payment information
    tenants = db.session.query(
        Tenant,
        Room,
        Payment.payment_status
    ).join(Room, Tenant.room_no == Room.room_no)\
     .outerjoin(Payment, Tenant.tenant_id == Payment.tenant_id)\
     .filter(Room.owner_id == owner_id)\
     .all()

    # Aggregate functions to count total tenants and rooms
    total_rooms = Room.query.filter_by(owner_id=owner_id).count()
    total_tenants = Tenant.query.join(Room).filter(Room.owner_id == owner_id).count()

    return render_template(
        'owner.html',
        owner_name=owner.owner_name,
        rooms=rooms,
        tenants=tenants,
        total_rooms=total_rooms,
        total_tenants=total_tenants
    )
```

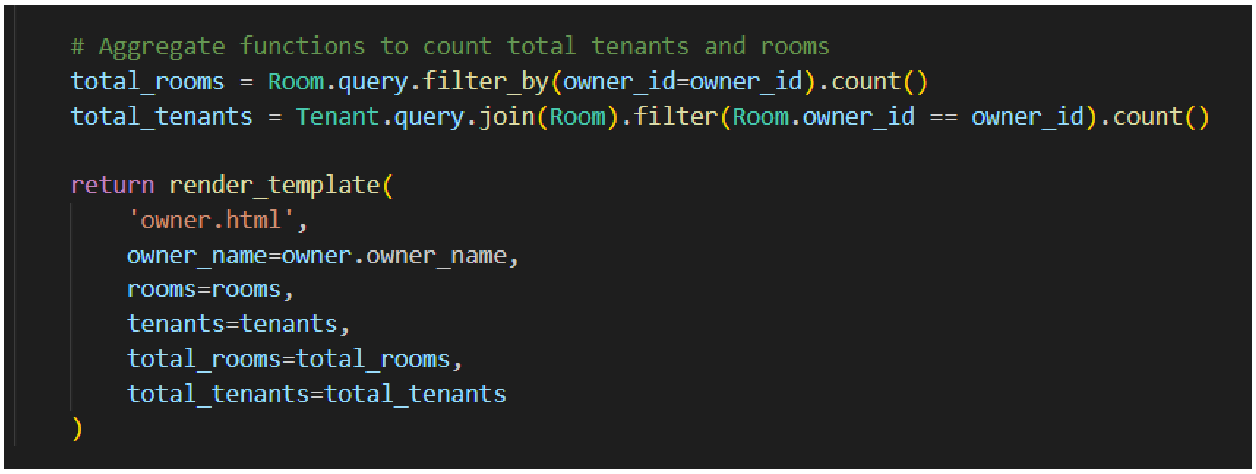
JOIN



```python
# Query to get tenants with room and payment information
tenants = db.session.query(
    Tenant,
    Room,
    Payment.payment_status
).join(Room, Tenant.room_no == Room.room_no)\
 .outerjoin(Payment, Tenant.tenant_id == Payment.tenant_id)\
 .filter(Room.owner_id == owner_id)\
 .all()
```

AGGREGATE

To display total rooms owned by an owner and total tenants under an owner

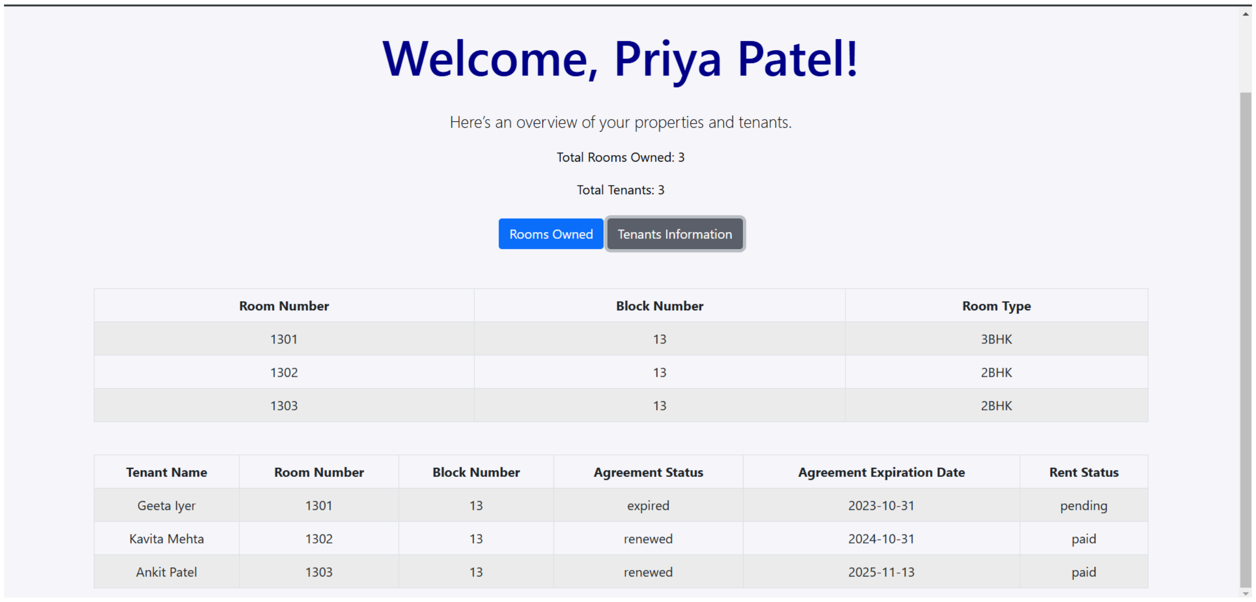


```
# Aggregate functions to count total tenants and rooms
total_rooms = Room.query.filter_by(owner_id=owner_id).count()
total_tenants = Tenant.query.join(Room).filter(Room.owner_id == owner_id).count()

return render_template(
    'owner.html',
    owner_name=owner.owner_name,
    rooms=rooms,
    tenants=tenants,
    total_rooms=total_rooms,
    total_tenants=total_tenants
)
```

OUTPUT

AGGREGATE: Total rooms owned and Total Tenants

JOIN: Displaying tenants under a particular owner with room and payment information



Welcome, Priya Patel!

Here's an overview of your properties and tenants.

Total Rooms Owned: 3

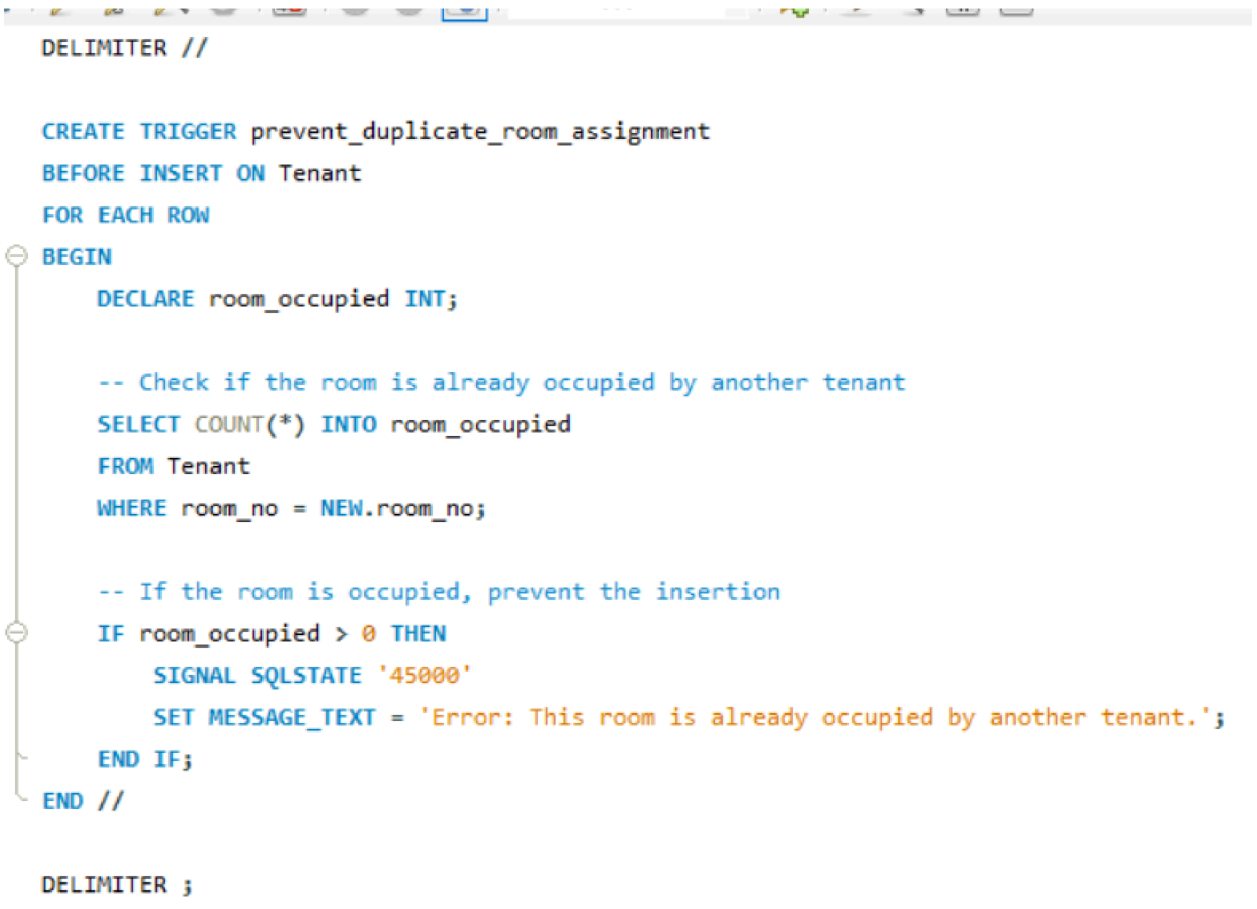
Total Tenants: 3

Room Number	Block Number	Room Type
1301	13	3BHK
1302	13	2BHK
1303	13	2BHK

Tenant Name	Room Number	Block Number	Agreement Status	Agreement Expiration Date	Rent Status
Geeta Iyer	1301	13	expired	2023-10-31	pending
Kavita Mehta	1302	13	renewed	2024-10-31	paid
Ankit Patel	1303	13	renewed	2025-11-13	paid

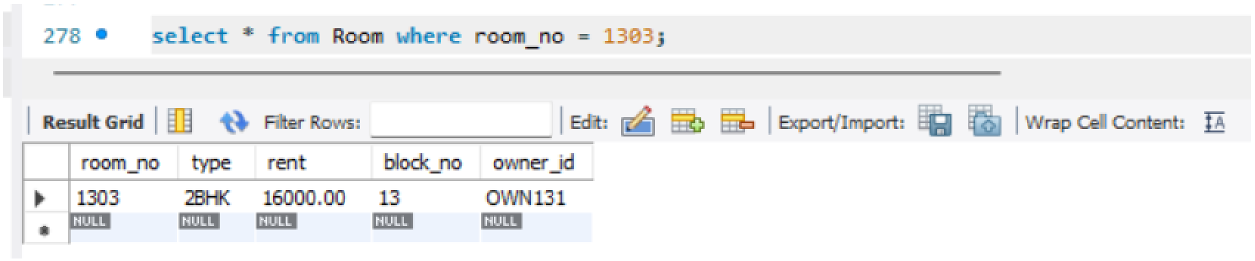
TRIGGERS

1. Trigger to prevent duplicate room assignment


```
DELIMITER //

CREATE TRIGGER prevent_duplicate_room_assignment
BEFORE INSERT ON Tenant
FOR EACH ROW
BEGIN
    DECLARE room_occupied INT;

    -- Check if the room is already occupied by another tenant
    SELECT COUNT(*) INTO room_occupied
    FROM Tenant
    WHERE room_no = NEW.room_no;

    -- If the room is occupied, prevent the insertion
    IF room_occupied > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: This room is already occupied by another tenant.';
    END IF;
END //

DELIMITER ;
```

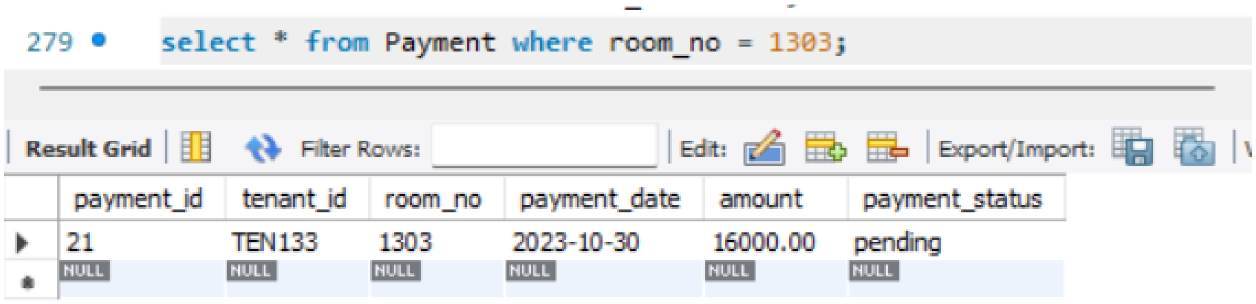
2. To set payment amount in payment table to that of rent in room table

```
CREATE TRIGGER set_payment_amount
BEFORE INSERT ON Payment
FOR EACH ROW
BEGIN
    DECLARE room_rent DECIMAL(10, 2);

    -- Get the rent for the room from the Room table
    SELECT rent INTO room_rent
    FROM Room
    WHERE room_no = NEW.room_no;

    -- Set the payment amount
    SET NEW.amount = room_rent;
END; //

DELIMITER ;
```



```
select * from Room where room_no = 1303;
```

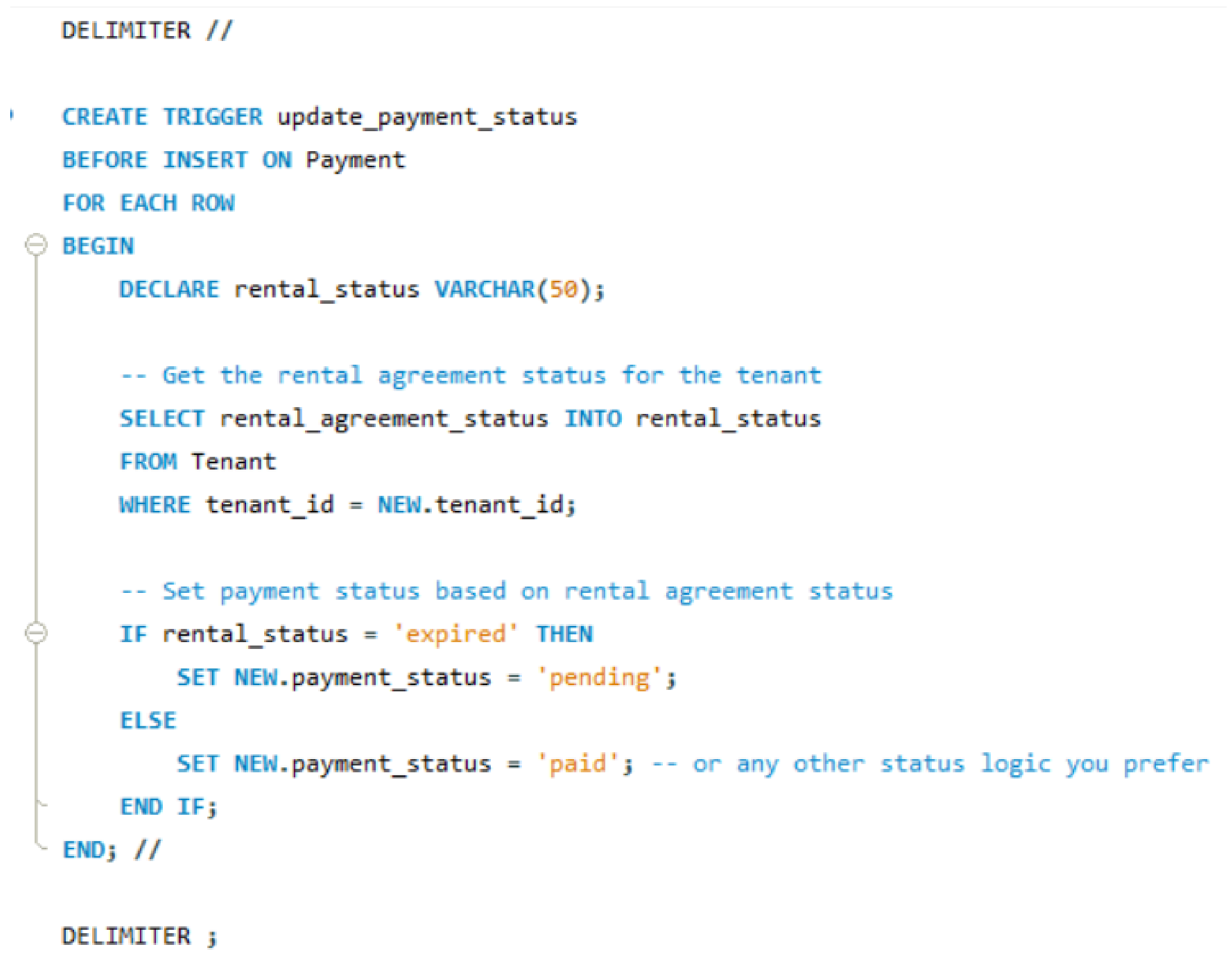
room_no	type	rent	block_no	owner_id
1303	2BHK	16000.00	13	OWN131
NULL	NULL	NULL	NULL	NULL



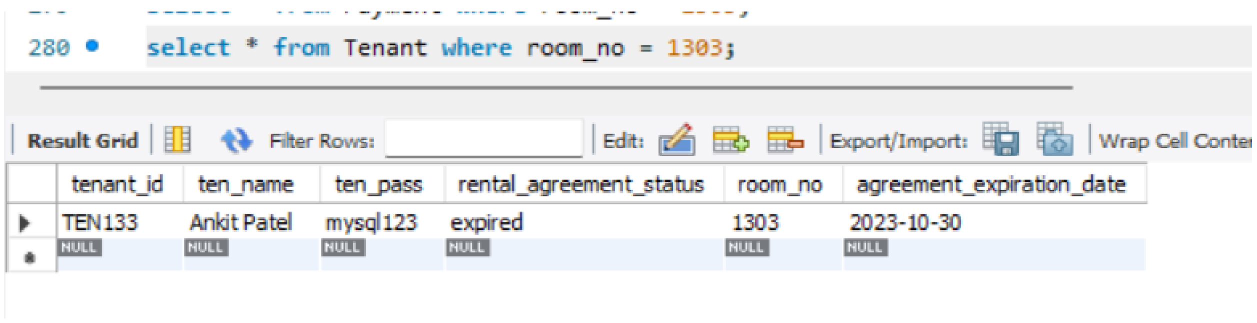
```
select * from Payment where room_no = 1303;
```

payment_id	tenant_id	room_no	payment_date	amount	payment_status
21	TEN133	1303	2023-10-30	16000.00	pending
NULL	NULL	NULL	NULL	NULL	NULL

3. To set payment status to pending if rental agreement status is expired

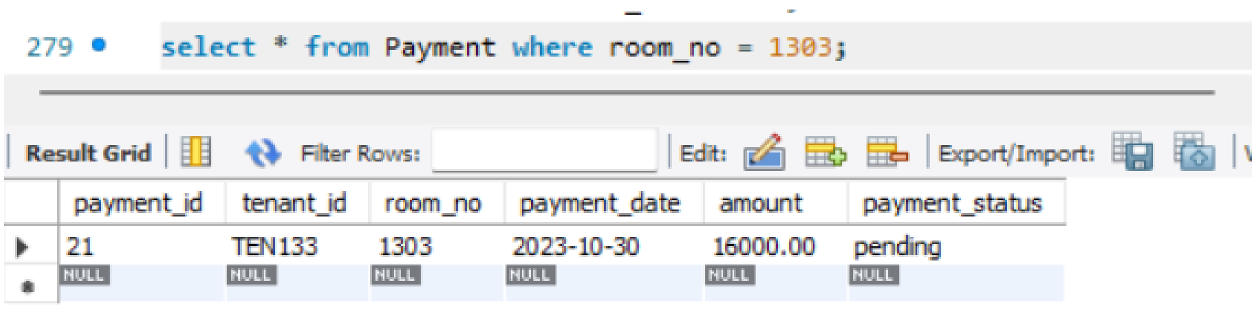


```
DELIMITER //

CREATE TRIGGER update_payment_status
BEFORE INSERT ON Payment
FOR EACH ROW
BEGIN
    DECLARE rental_status VARCHAR(50);

    -- Get the rental agreement status for the tenant
    SELECT rental_agreement_status INTO rental_status
    FROM Tenant
    WHERE tenant_id = NEW.tenant_id;

    -- Set payment status based on rental agreement status
    IF rental_status = 'expired' THEN
        SET NEW.payment_status = 'pending';
    ELSE
        SET NEW.payment_status = 'paid'; -- or any other status logic you prefer
    END IF;
END; //

DELIMITER ;
```



```
select * from Tenant where room_no = 1303;
```

tenant_id	ten_name	ten_pass	rental_agreement_status	room_no	agreement_expiration_date
TEN133	Ankit Patel	mysql123	expired	1303	2023-10-30
NULL	NULL	NULL	NULL	NULL	NULL

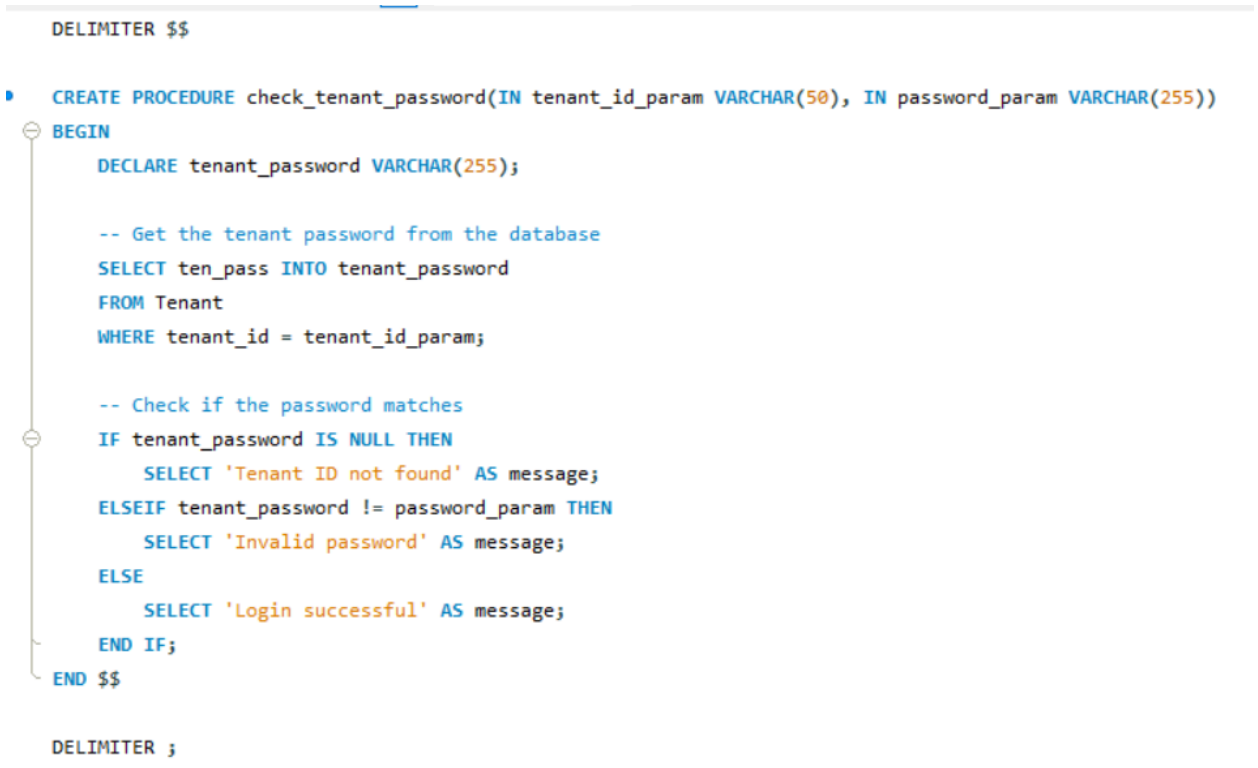


```
select * from Payment where room_no = 1303;
```

	payment_id	tenant_id	room_no	payment_date	amount	payment_status
▶	21	TEN133	1303	2023-10-30	16000.00	pending
*	NULL	NULL	NULL	NULL	NULL	NULL

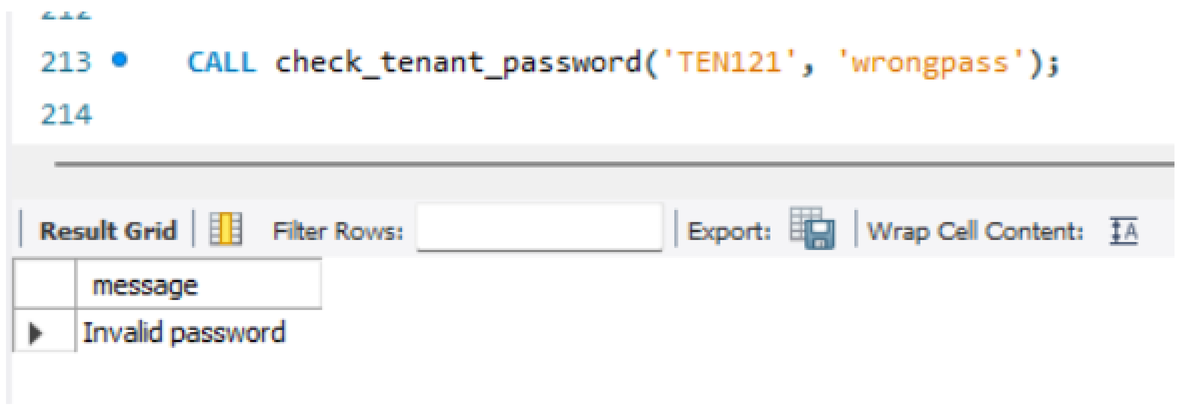
FUNCTIONS/PROCEDURES

1. To check if tenant login credentials is correct



```
DELIMITER $$

CREATE PROCEDURE check_tenant_password(IN tenant_id_param VARCHAR(50), IN password_param VARCHAR(255))
BEGIN
    DECLARE tenant_password VARCHAR(255);

    -- Get the tenant password from the database
    SELECT ten_pass INTO tenant_password
    FROM Tenant
    WHERE tenant_id = tenant_id_param;

    -- Check if the password matches
    IF tenant_password IS NULL THEN
        SELECT 'Tenant ID not found' AS message;
    ELSEIF tenant_password != password_param THEN
        SELECT 'Invalid password' AS message;
    ELSE
        SELECT 'Login successful' AS message;
    END IF;
END $$

DELIMITER ;
```

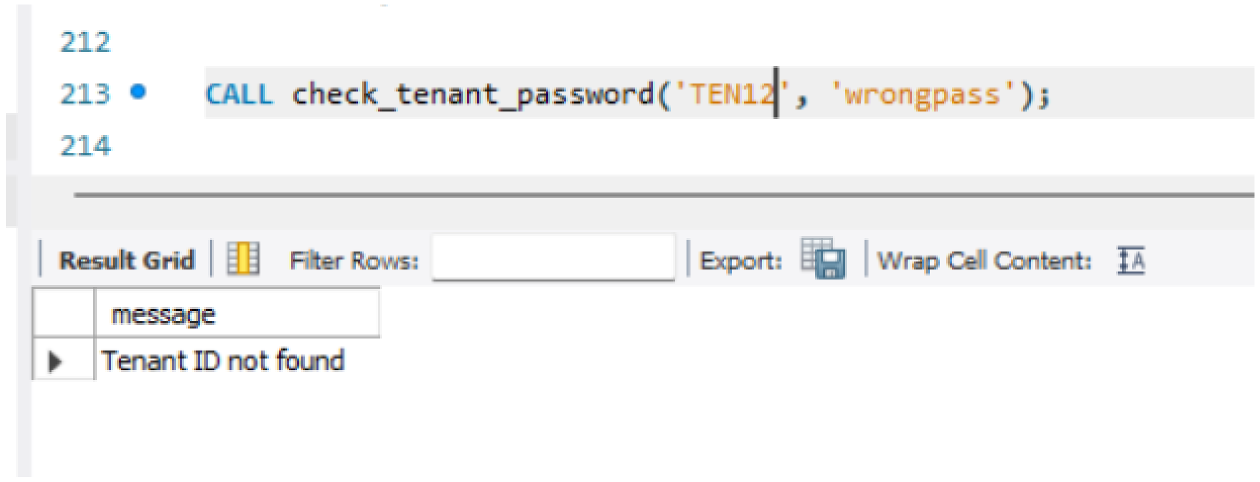
Wrong password



```
CALL check_tenant_password('TEN121', 'wrongpass');
```

message
Invalid password

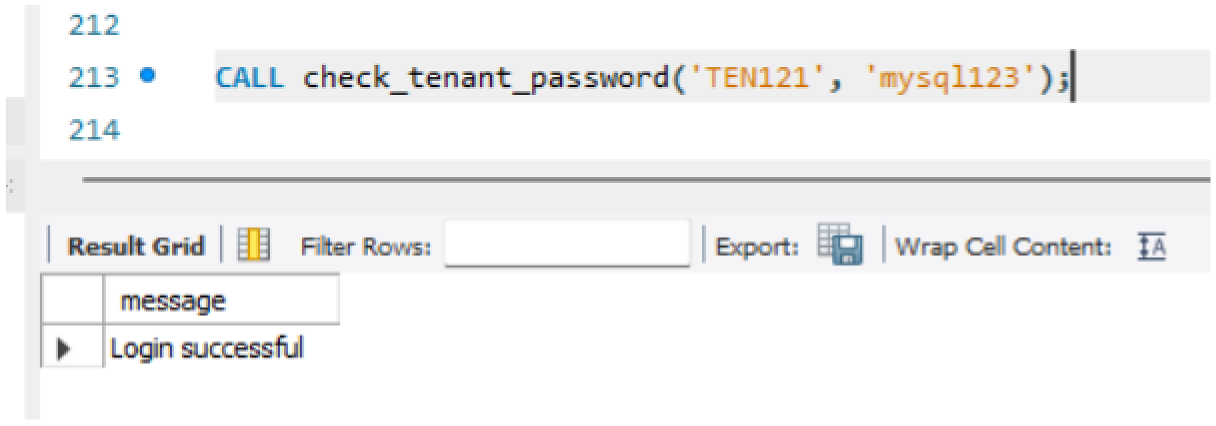
TEN12 doesn’t exist in the database:



```
CALL check_tenant_password('TEN12', 'wrongpass');
```

message
Tenant ID not found

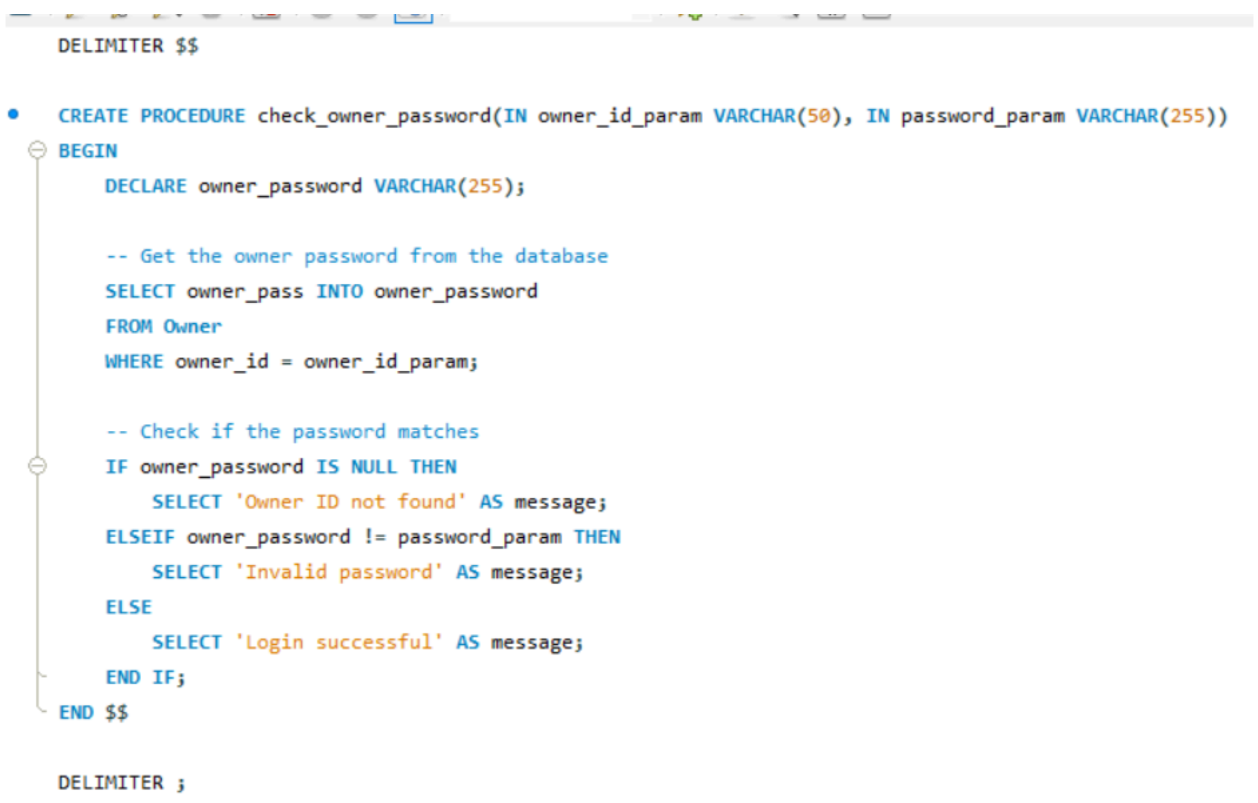
Valid Credentials



```
CALL check_tenant_password('TEN121', 'mysql123');
```

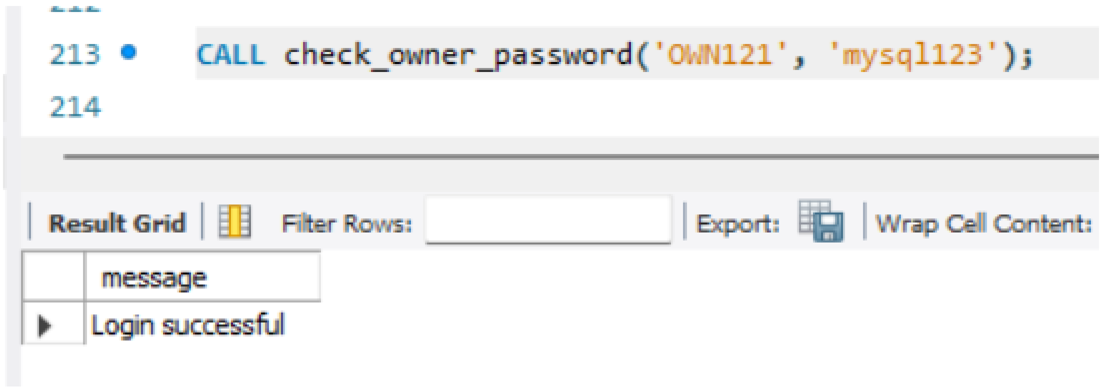
message
Login successful

2. To check if owner login credentials is correct



```
DELIMITER $$

CREATE PROCEDURE check_owner_password(IN owner_id_param VARCHAR(50), IN password_param VARCHAR(255))
BEGIN
    DECLARE owner_password VARCHAR(255);

    -- Get the owner password from the database
    SELECT owner_pass INTO owner_password
    FROM Owner
    WHERE owner_id = owner_id_param;

    -- Check if the password matches
    IF owner_password IS NULL THEN
        SELECT 'Owner ID not found' AS message;
    ELSEIF owner_password != password_param THEN
        SELECT 'Invalid password' AS message;
    ELSE
        SELECT 'Login successful' AS message;
    END IF;
END $$

DELIMITER ;
```

Valid credentials

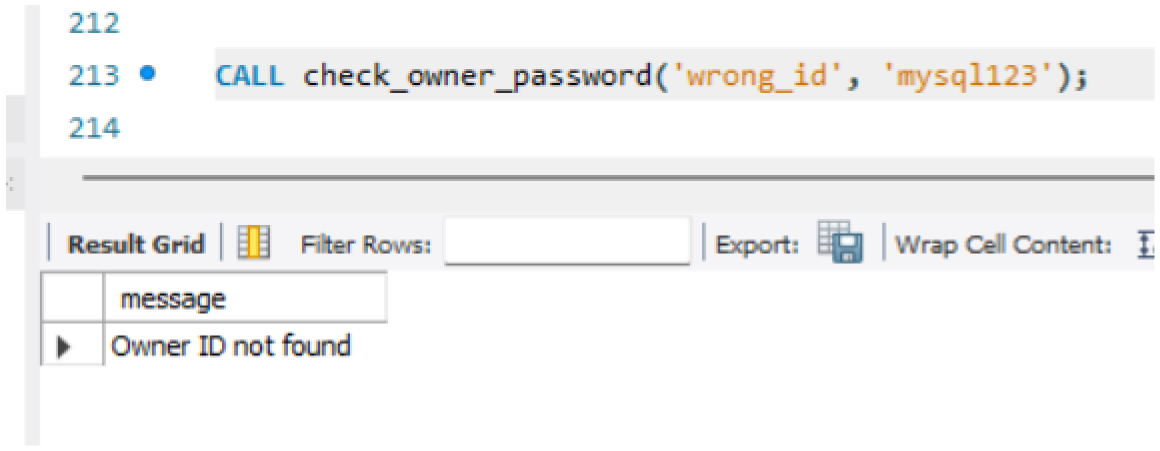


```
CALL check_owner_password('OWN121', 'mysql123');
```

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

```
message
Login successful
```

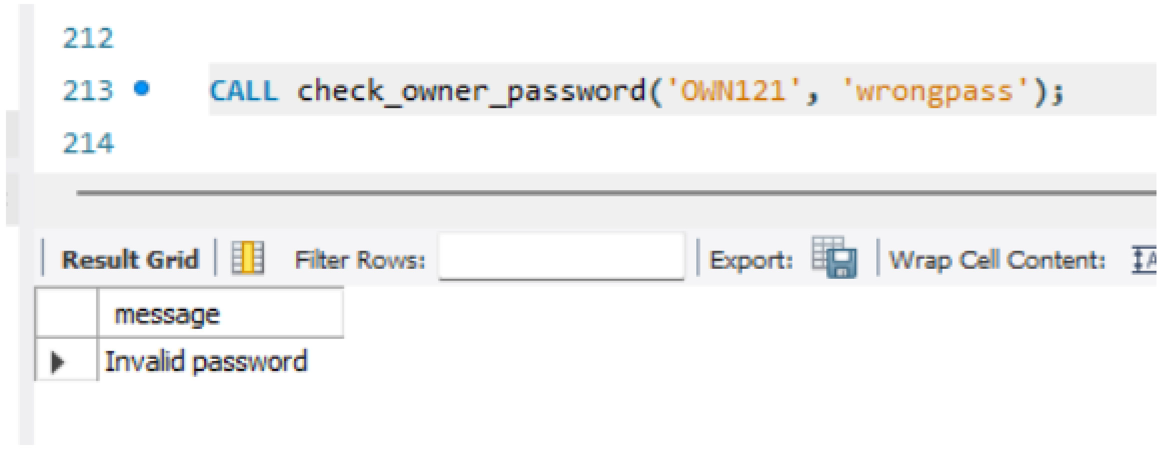
Invalid owner_id



```
CALL check_owner_password('wrong_id', 'mysql123');
```

```
message
Owner ID not found
```

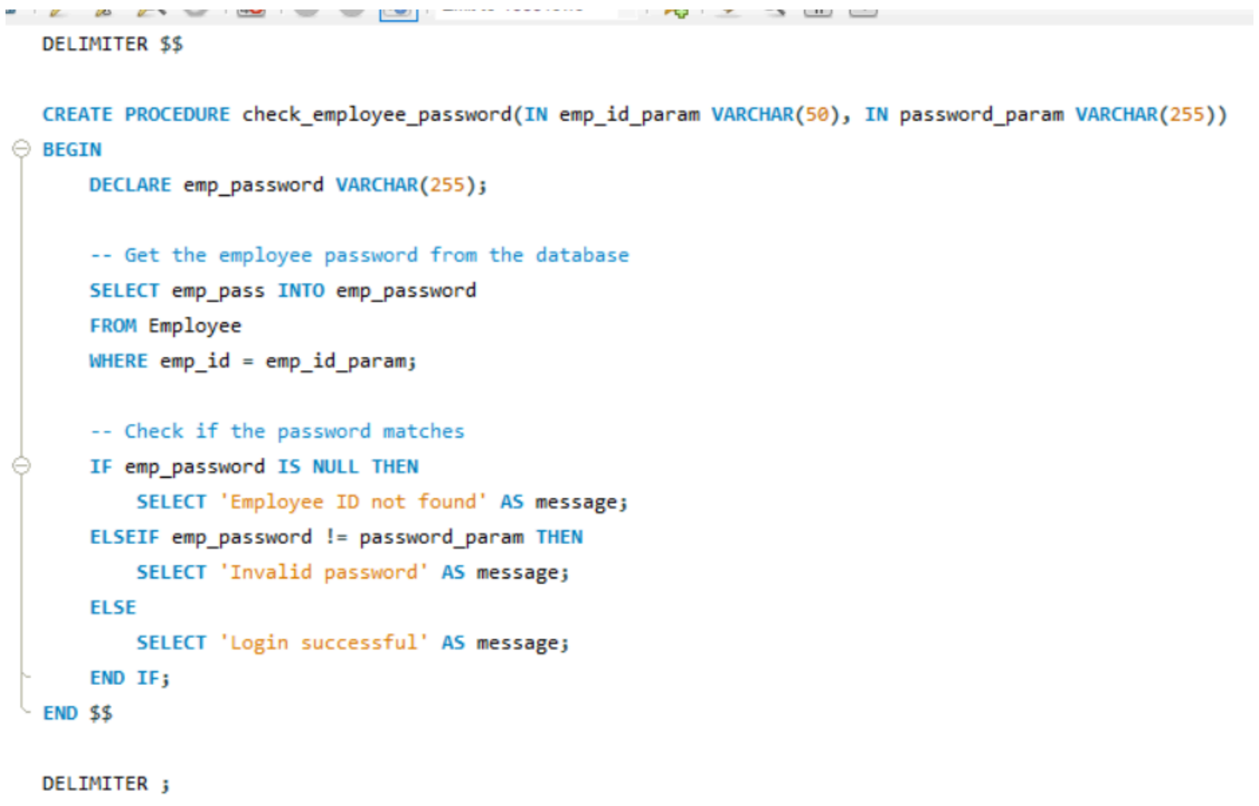
Invalid password



```
CALL check_owner_password('OWN121', 'wrongpass');
```

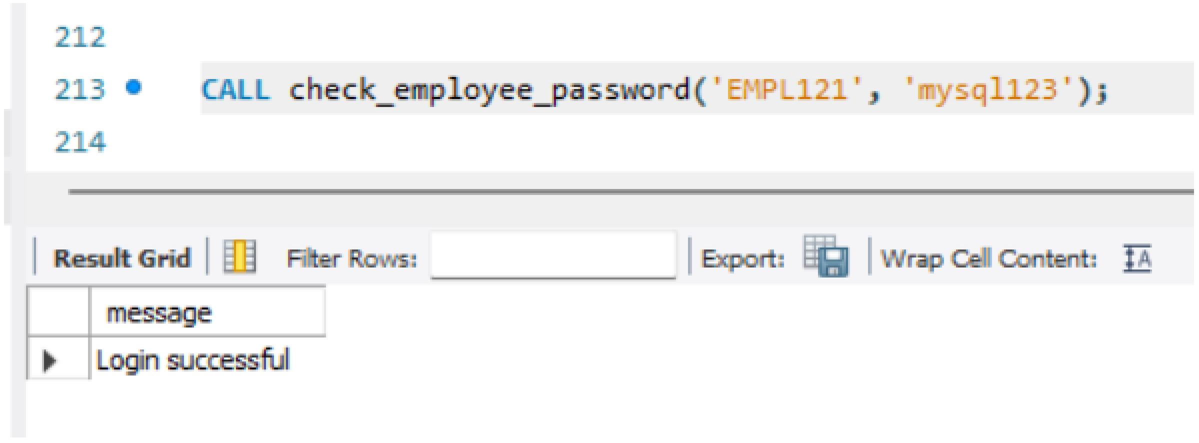
```
message
Invalid password
```

3. To check employee login credentials



```
DELIMITER $$

CREATE PROCEDURE check_employee_password(IN emp_id_param VARCHAR(50), IN password_param VARCHAR(255))
BEGIN
    DECLARE emp_password VARCHAR(255);

    -- Get the employee password from the database
    SELECT emp_pass INTO emp_password
    FROM Employee
    WHERE emp_id = emp_id_param;

    -- Check if the password matches
    IF emp_password IS NULL THEN
        SELECT 'Employee ID not found' AS message;
    ELSEIF emp_password != password_param THEN
        SELECT 'Invalid password' AS message;
    ELSE
        SELECT 'Login successful' AS message;
    END IF;
END $$

DELIMITER ;
```

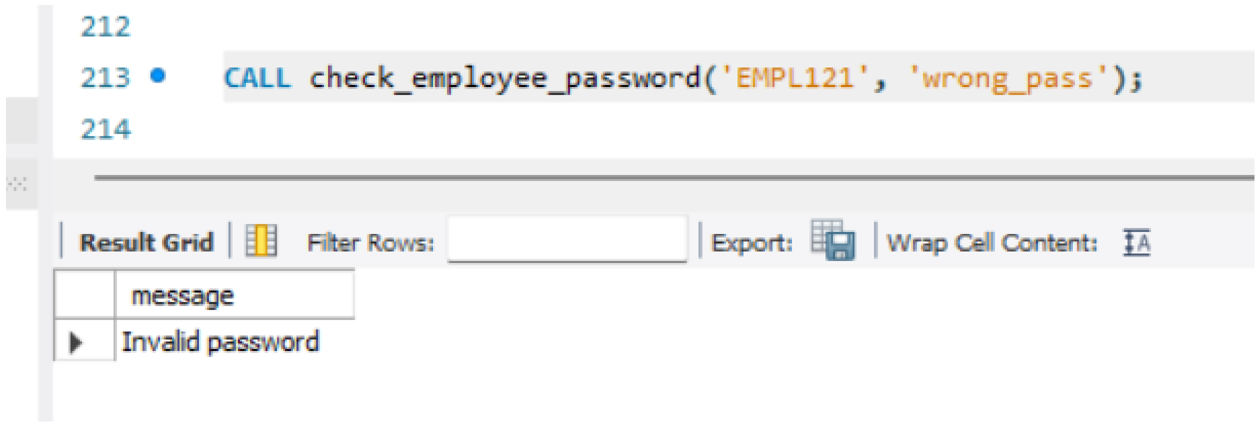
Valid credentials



```
CALL check_employee_password('EMPL121', 'mysql123');
```

```
message
Login successful
```

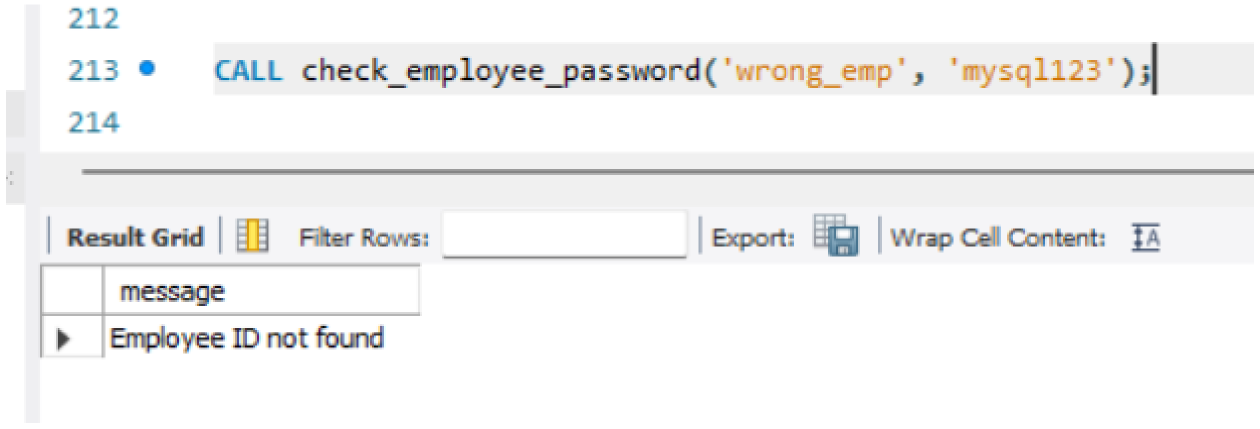
Invalid password



```
CALL check_employee_password('EMPL121', 'wrong_pass');
```

```
message
Invalid password
```

Invalid employeeID



```
CALL check_employee_password('wrong_emp', 'mysql123');
```

```
message
Employee ID not found
```