PES UNIVERSITY, BANGALORE

Department of Computer Science and Engineering

B. Tech (CSE) – 5th Semester – Aug-Dec 2023

UE21CS351A – Database Management Systems

PROJECTREPORT

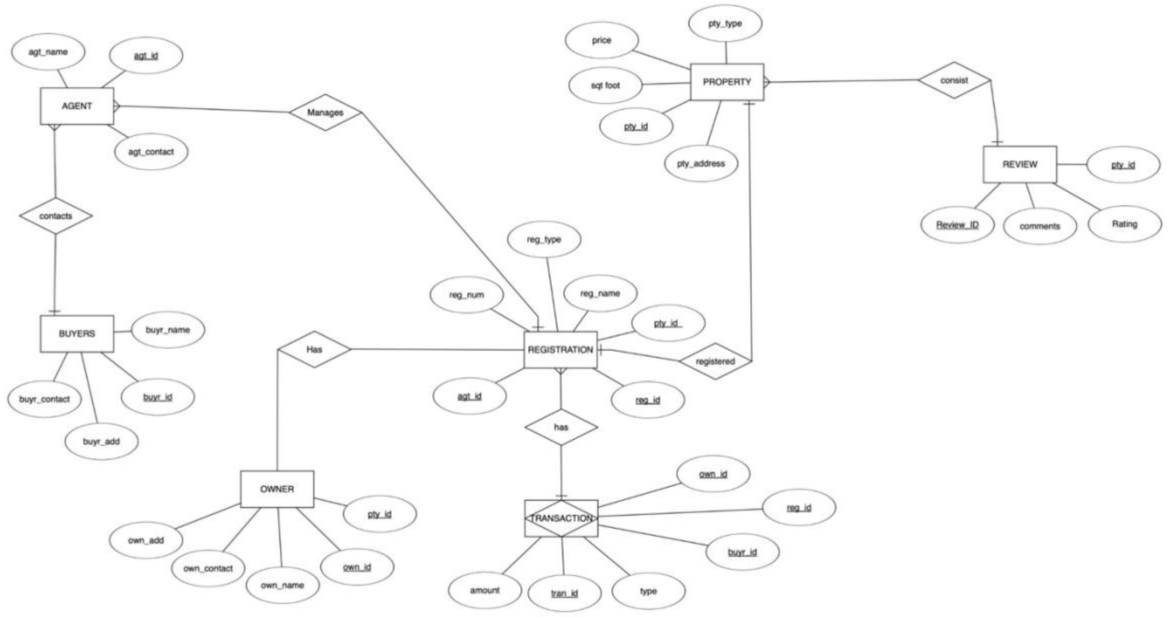
Real-Estate Database Management System

Pranav TP	PES1UG21CS433
Pranjal TS	PES1UG21CS438

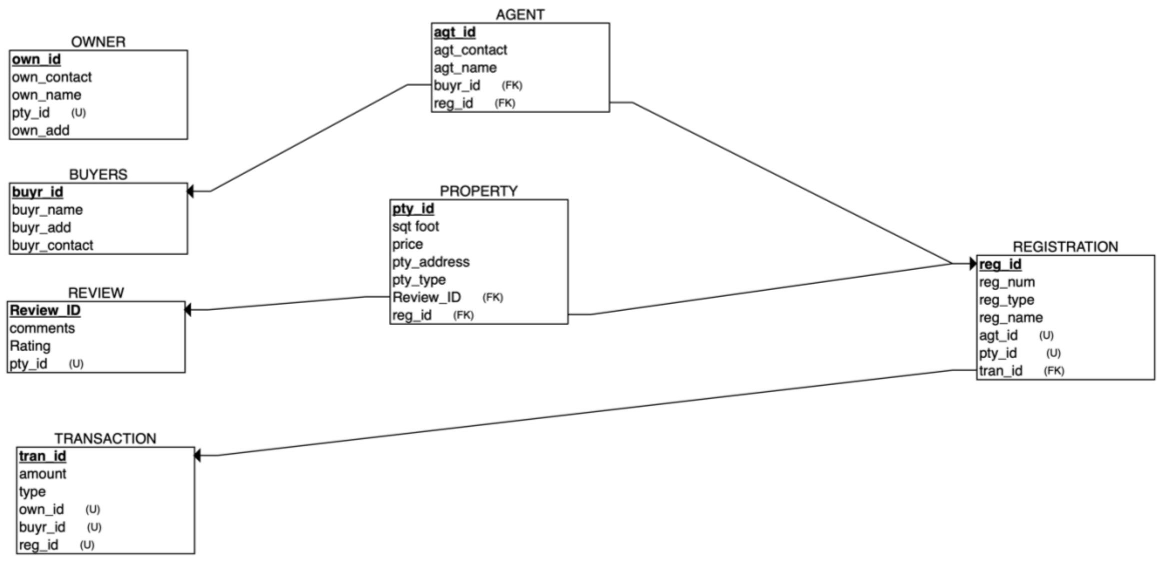
Short Abstract

The Real Estate Management System, implemented using Streamlit and MySQL, is a sophisticated application designed to handle a broad spectrum of real estate management tasks. The code encompasses a range of functionalities, including property registration, comprehensive owner and buyer management, dynamic property searches, SQL query execution, agent registrations, transaction handling, and a user-friendly review submission system.The system places a strong emphasis on delivering a visually enhanced experience through Streamlit's interactive components. Key features, such as owner-property associations and cascading deletes, contribute to the efficiency of real estate resource management. The application's architecture ensures security, particularly in user authentication processes, prioritizing data integrity and protection..

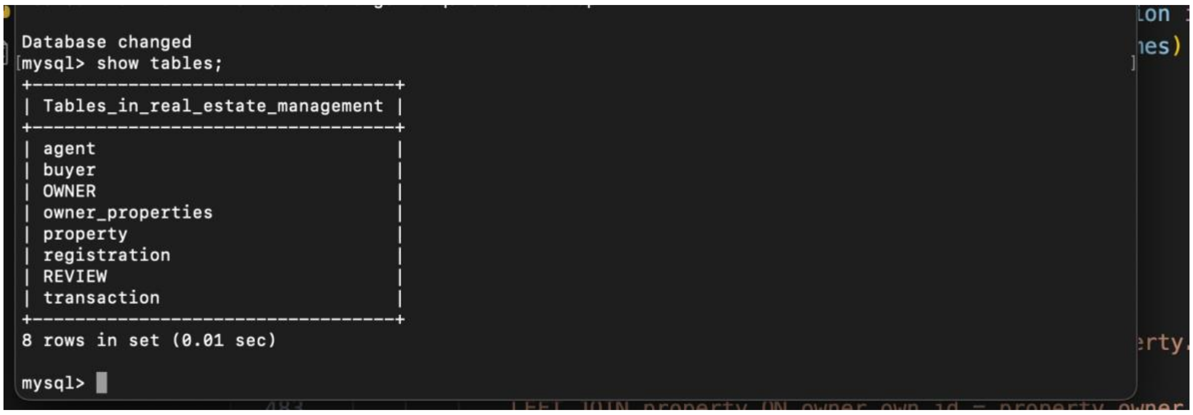
Entity Relationship Diagram



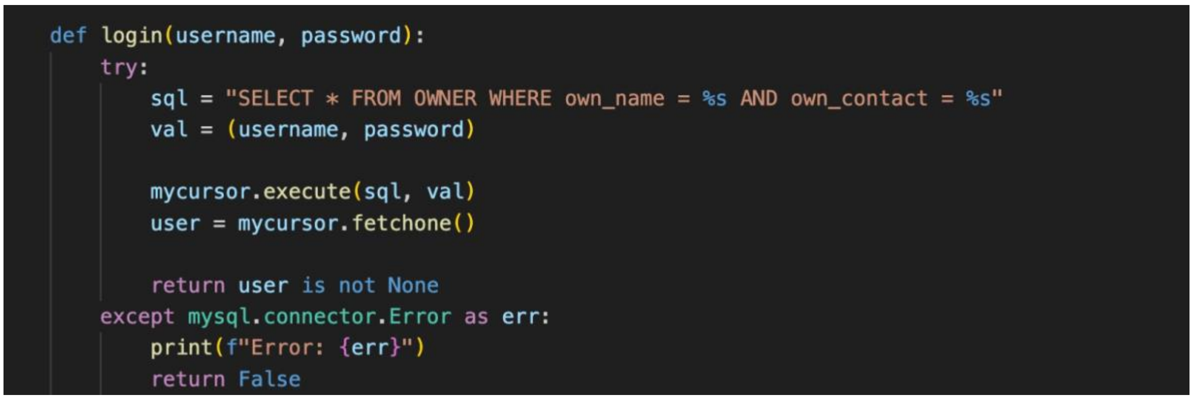
Relational Schema



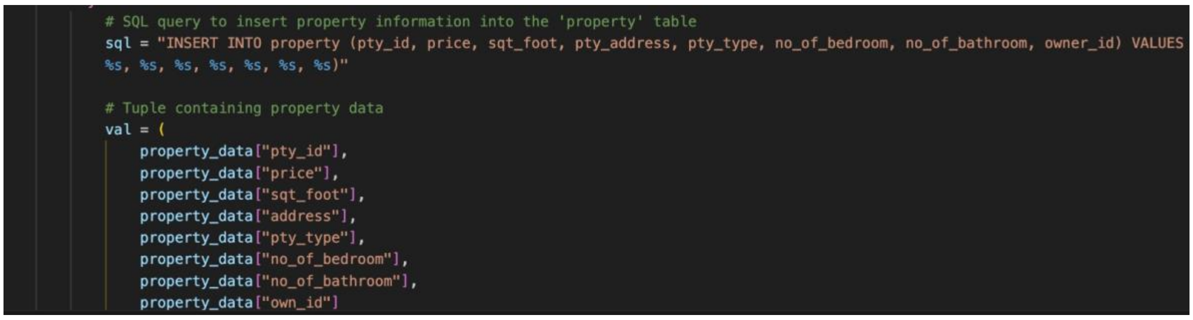
Data Definition Language – SQL Commands and CRUD Operations



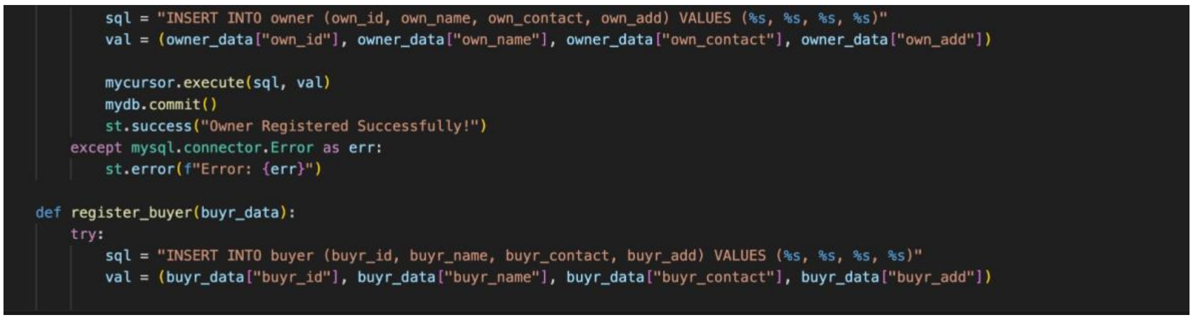
```
Database changed
mysql> show tables;
+-----------------------------------+
| Tables_in_real_estate_management |
+-----------------------------------+
| agent                             |
| buyer                             |
| OWNER                             |
| owner_properties                  |
| property                          |
| registration                      |
| REVIEW                            |
| transaction                       |
+-----------------------------------+
8 rows in set (0.01 sec)

mysql>
```



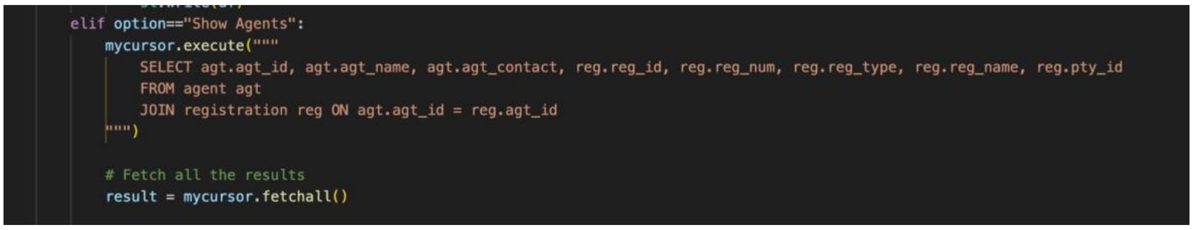
```python
def login(username, password):
    try:
        sql = "SELECT * FROM OWNER WHERE own_name = %s AND own_contact = %s"
        val = (username, password)

        mycursor.execute(sql, val)
        user = mycursor.fetchone()

        return user is not None
    except mysql.connector.Error as err:
        print(f"Error: {err}")
        return False
```



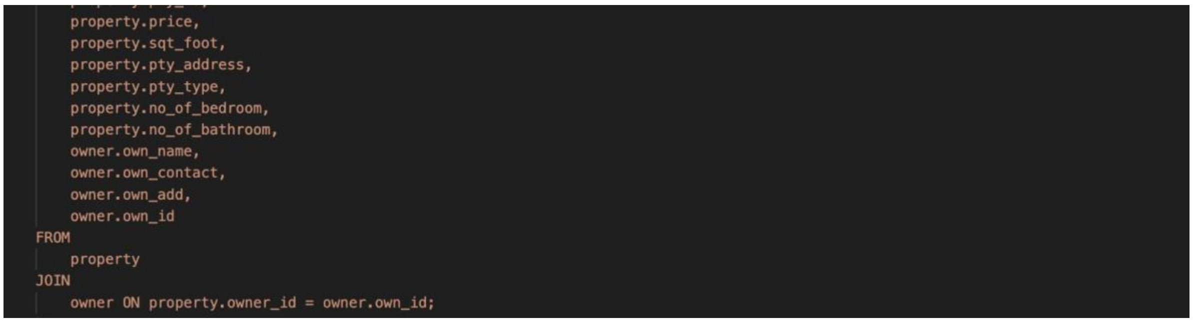
```python
        # SQL query to insert property information into the 'property' table
        sql = "INSERT INTO property (pty_id, price, sqt_foot, pty_address, pty_type, no_of_bedroom, no_of_bathroom, owner_id) VALUES
        %s, %s, %s, %s, %s, %s, %s)"

        # Tuple containing property data
        val = (
            property_data["pty_id"],
            property_data["price"],
            property_data["sqt_foot"],
            property_data["address"],
            property_data["pty_type"],
            property_data["no_of_bedroom"],
            property_data["no_of_bathroom"],
            property_data["own_id"]
```



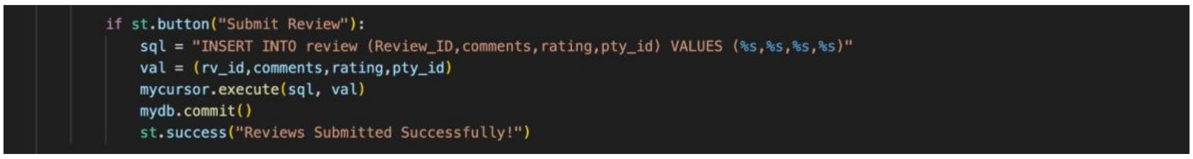
```python
        sql = "INSERT INTO owner (own_id, own_name, own_contact, own_add) VALUES (%s, %s, %s, %s)"
        val = (owner_data["own_id"], owner_data["own_name"], owner_data["own_contact"], owner_data["own_add"])

        mycursor.execute(sql, val)
        mydb.commit()
        st.success("Owner Registered Successfully!")
    except mysql.connector.Error as err:
        st.error(f"Error: {err}")

def register_buyer(buyr_data):
    try:
        sql = "INSERT INTO buyer (buyr_id, buyr_name, buyr_contact, buyr_add) VALUES (%s, %s, %s, %s)"
        val = (buyr_data["buyr_id"], buyr_data["buyr_name"], buyr_data["buyr_contact"], buyr_data["buyr_add"])
```



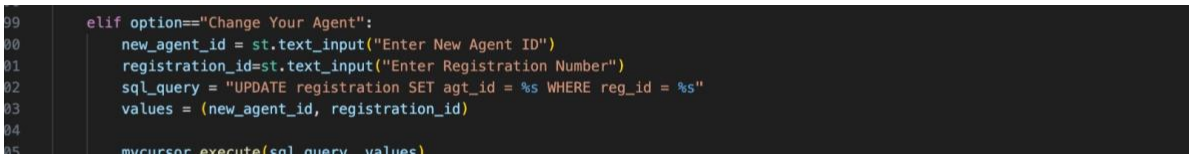
```
elif option=="Show Agents":
    mycursor.execute("""
        SELECT agt.agt_id, agt.agt_name, agt.agt_contact, reg.reg_id, reg.reg_num, reg.reg_type, reg.reg_name, reg.pty_id
        FROM agent agt
        JOIN registration reg ON agt.agt_id = reg.agt_id
    """)

    # Fetch all the results
    result = mycursor.fetchall()
```



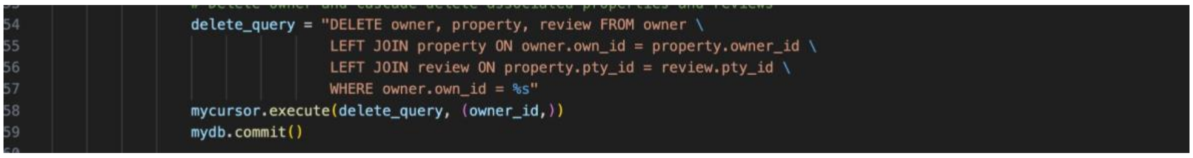
```
    property.price,
    property.sqt_foot,
    property.pty_address,
    property.pty_type,
    property.no_of_bedroom,
    property.no_of_bathroom,
    owner.own_name,
    owner.own_contact,
    owner.own_add,
    owner.own_id
FROM
    property
JOIN
    owner ON property.owner_id = owner.own_id;
```



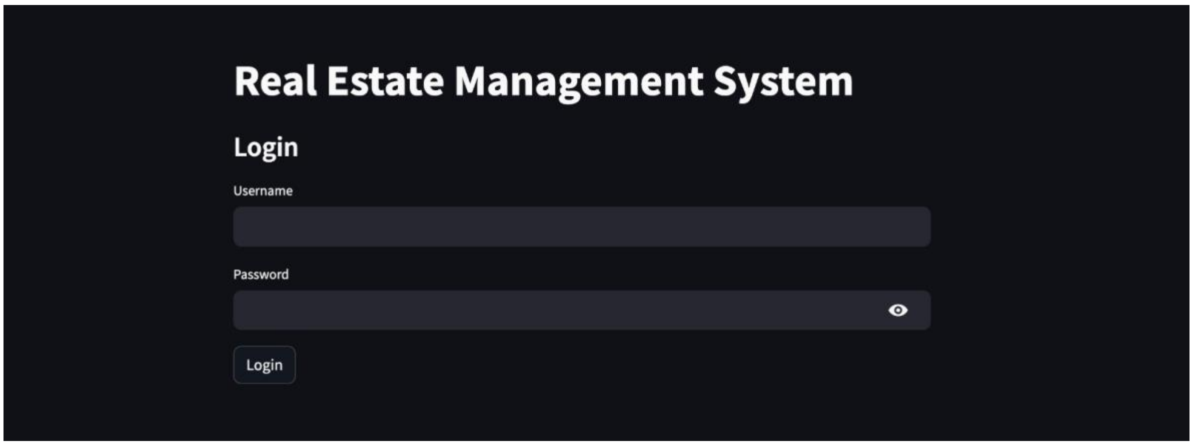
```
if st.button("Submit Review"):
    sql = "INSERT INTO review (Review_ID,comments,rating,pty_id) VALUES (%s,%s,%s,%s)"
    val = (rv_id,comments,rating,pty_id)
    mycursor.execute(sql, val)
    mydb.commit()
    st.success("Reviews Submitted Successfully!")
```

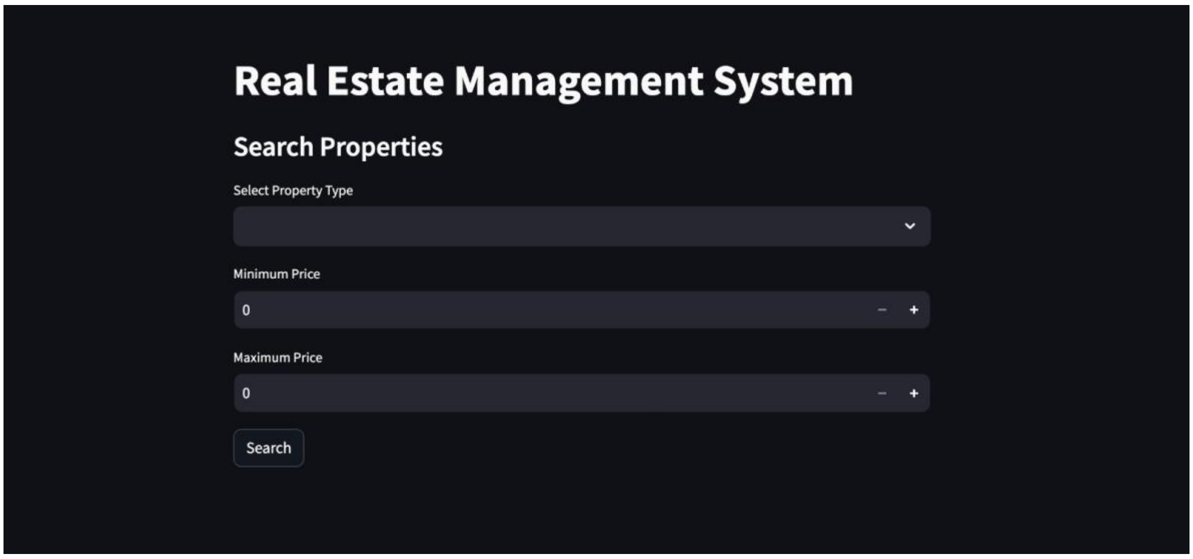


```
elif option=="Change Your Agent":
    new_agent_id = st.text_input("Enter New Agent ID")
    registration_id=st.text_input("Enter Registration Number")
    sql_query = "UPDATE registration SET agt_id = %s WHERE reg_id = %s"
    values = (new_agent_id, registration_id)

    mycursor.execute(sql_query, values)
```



```
delete_query = "DELETE owner, property, review FROM owner \
                LEFT JOIN property ON owner.own_id = property.owner_id \
                LEFT JOIN review ON property.pty_id = review.pty_id \
                WHERE owner.own_id = %s"
mycursor.execute(delete_query, (owner_id,))
mydb.commit()
```





Real Estate Management System

Search Properties

Select Property Type

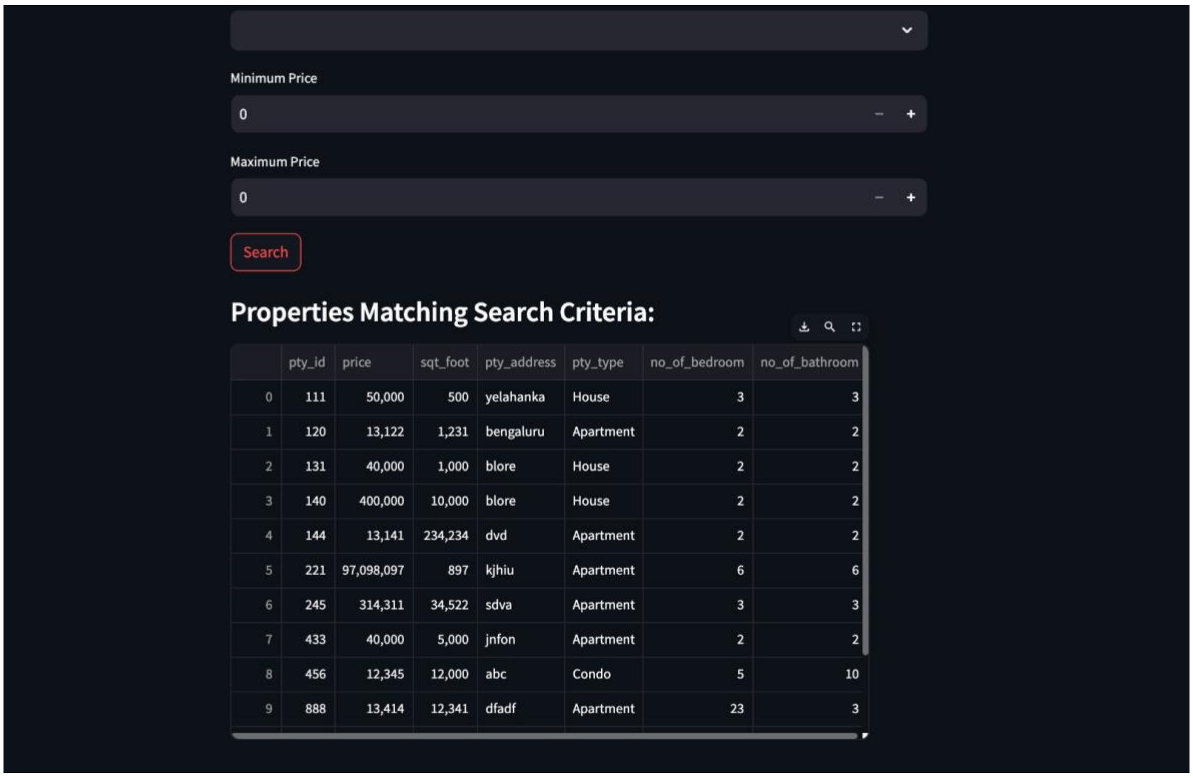
Minimum Price

0

Maximum Price

0

Search



Minimum Price

0

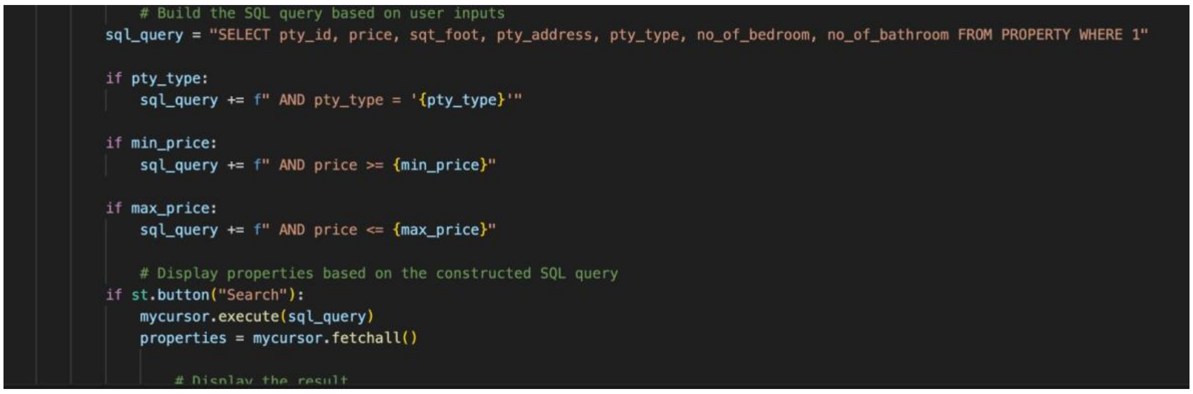
Maximum Price

0

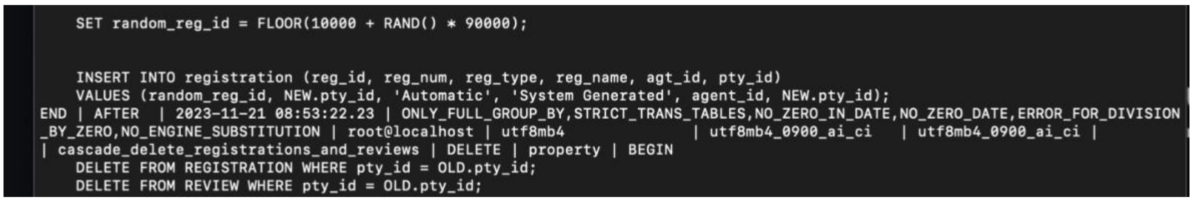
Search

Properties Matching Search Criteria:

	pty_id	price	sqt_foot	pty_address	pty_type	no_of_bedroom	no_of_bathroom
0	111	50,000	500	yelahanka	House	3	3
1	120	13,122	1,231	bengaluru	Apartment	2	2
2	131	40,000	1,000	blore	House	2	2
3	140	400,000	10,000	blore	House	2	2
4	144	13,141	234,234	dvd	Apartment	2	2
5	221	97,098,097	897	kjhiu	Apartment	6	6
6	245	314,311	34,522	sdva	Apartment	3	3
7	433	40,000	5,000	jnfon	Apartment	2	2
8	456	12,345	12,000	abc	Condo	5	10
9	888	13,414	12,341	dfadf	Apartment	23	3



```python
    # Build the SQL query based on user inputs
sql_query = "SELECT pty_id, price, sqt_foot, pty_address, pty_type, no_of_bedroom, no_of_bathroom FROM PROPERTY WHERE 1"

if pty_type:
    sql_query += f" AND pty_type = '{pty_type}'"

if min_price:
    sql_query += f" AND price >= {min_price}"

if max_price:
    sql_query += f" AND price <= {max_price}"

    # Display properties based on the constructed SQL query
if st.button("Search"):
    mycursor.execute(sql_query)
    properties = mycursor.fetchall()

        # Display the result
```



```
    SET random_reg_id = FLOOR(10000 + RAND() * 90000);


    INSERT INTO registration (reg_id, reg_num, reg_type, reg_name, agt_id, pty_id)
    VALUES (random_reg_id, NEW.pty_id, 'Automatic', 'System Generated', agent_id, NEW.pty_id);
END | AFTER  | 2023-11-21 08:53:22.23 | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION
_BY_ZERO,NO_ENGINE_SUBSTITUTION | root@localhost | utf8mb4              | utf8mb4_0900_ai_ci   | utf8mb4_0900_ai_ci |
| cascade_delete_registrations_and_reviews | DELETE | property | BEGIN
    DELETE FROM REGISTRATION WHERE pty_id = OLD.pty_id;
    DELETE FROM REVIEW WHERE pty_id = OLD.pty_id;
```

List of Functionalities :

Login: Users can log in with their credentials (owner's name and contact).

Register Owner: Owners can register in the system by providing their details (ID, name, contact, and address).

Login Buyer: Buyers can log in with their credentials (buyer's name and contact).

Register Buyer: Buyers can register in the system by providing their details (ID, name, contact, and address).

Register Your Property: Property owners can register their properties, providing details such as ID, price, area, address, type, number of bedrooms and bathrooms, and owner ID.

Search Properties: Users can search for properties based on various criteria, including property type, minimum and maximum price.

Write SQL Query: Users can write custom SQL queries and execute them against the database.

Show Owners: Displays a table of property details along with owner information.

Write a Review: Users can submit reviews for specific properties, including comments and ratings.

Show Reviews: Displays reviews for a specific property, optionally filtered by rating.

Count Owner Properties: Shows a count of properties owned by each owner.

Search Property by Country Code: Searches for properties based on the country code of the owner's contact.

Show Properties: Displays details of properties based on various criteria.

Make a Transaction: Buyers can make transactions for purchasing properties, providing details such as transaction amount, type, buyer ID, and owner ID.

Register Agent: Agents can be registered in the system with their ID, name, and contact.

Show Agents: Displays a table of agents along with registration details.

Update Property: Allows users to update the price of a property.

Delete: Deletes an owner along with associated properties and reviews using cascade delete.

Change Your Agent: Allows users to change the agent associated with a registration.