BOSTON HOUSE RENTAL MANAGEMENT SYSTEM

Group No: 41

Student Names: Gargi Gokhale AND Tanay Malviya

Executive Summary:

The Boston House Rental Management System is a comprehensive and efficient solution designed to streamline the management and administration of rental properties within the vibrant city of Boston. This system leverages the power of Structured Query Language (SQL) to provide a robust and scalable platform for property owners, tenants, and administrators to effectively handle the diverse aspects of house rentals.
The Boston House Rental Management System is built on a robust SQL database, enabling efficient data storage, retrieval, and management. SQL queries are strategically employed to ensure data integrity, relational connections between tables, and optimal performance in handling complex queries.
The Boston House Rental Management System leverages SQL as its backbone to offer a powerful, secure, and scalable solution for managing the intricate processes associated with house rentals in Boston. By embracing technology, the system aims to enhance the overall rental experience for property owners and tenants alike, promoting efficiency, transparency, and satisfaction in the dynamic real estate landscape of Boston.

I. INTRODUCTION:

Amidst the ever-evolving landscape of technology, there is a growing need to transcend complexities and embrace solutions that simply work. The current paradigm shift in technology demands an appreciation of its power, prompting the housing sector to adopt innovative strategies for efficient Boston house rental management. Recognizing this urgency, there arises a necessity to develop a Rental House Management System tailored to streamline the workflows of rental managers, ensuring efficiency and effectiveness in their daily operations.

The Rental Management System specifically caters to the diverse needs of individuals seeking accommodations such as apartments, paying guest spaces, offices, and houses in metropolitan cities. Anchored on the interaction between property owners and customers, this system provides a comprehensive platform for managing property details, facilitating seamless communication, and enhancing the overall rental experience.

For property owners, the system offers real-time updates on apartment, office, house, and paying guest details. Property managers can efficiently manage their listings, ensuring accurate and up-to-date information for potential customers. The system acts as a centralized hub for property details, eliminating the need for repetitive explanations and making communication with eligible individuals more straightforward.

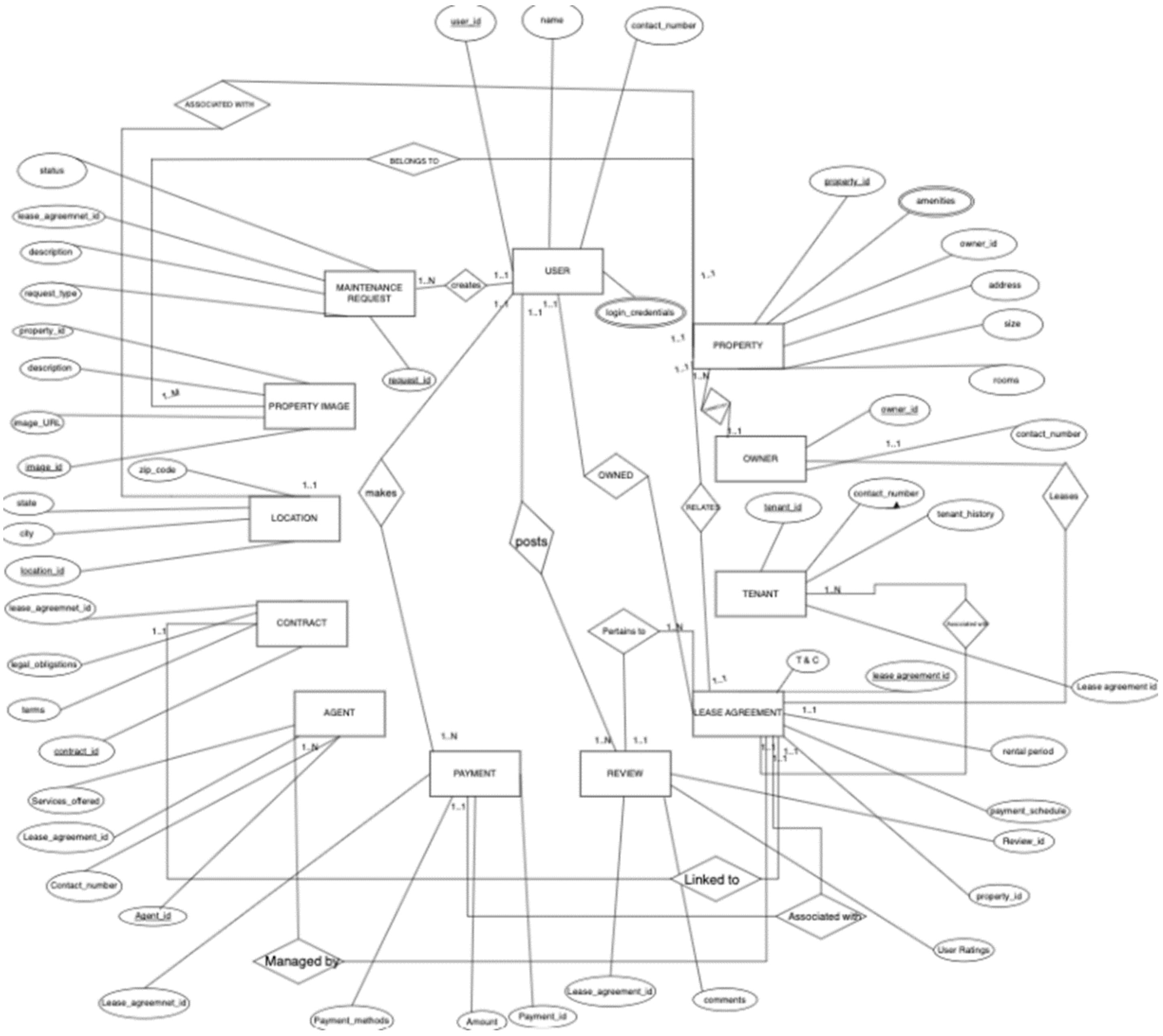
Customers, on the other hand, benefit from a user-friendly interface that provides information about available room spaces, rental costs, and address details. The Rental Management System becomes a valuable tool for individuals looking to find suitable rental spaces in metropolitan cities. The search process is streamlined based on preferences, such as budget constraints and occupancy limits, ensuring that customers quickly identify the most suitable housing options.

This system is particularly advantageous for property owners as it not only saves time but also ensures direct and efficient communication with eligible individuals. The ease of contact and the elimination of the need to repeatedly explain property details contribute to a seamless rental management experience. The Boston Rental Management System proves to be a highly effective application in urban settings, simplifying the customer's search process for apartments, offices, paying guest accommodations, and houses based on budget constraints and occupancy preferences.

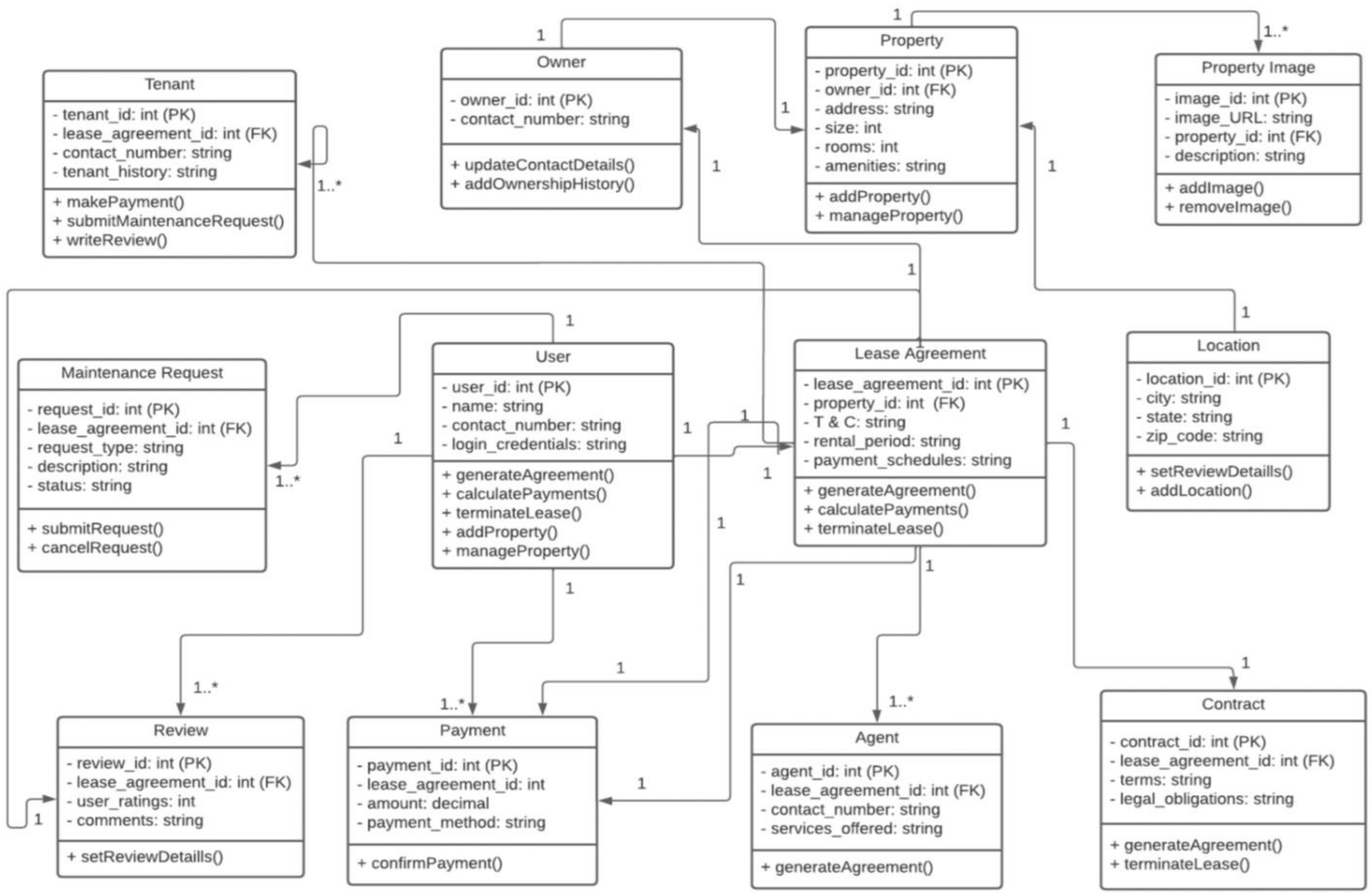
In conclusion, the Boston House Rental Management System is a cutting-edge solution designed to simplify and enhance the house rental process. By leveraging technology, this system provides property owners and customers with an efficient, user-friendly platform that streamlines property management and rental searches, ultimately making the entire process more accessible and time-effective.

II. CONCEPTUAL DATA MODELING:

1. EER Diagram



2. UML Diagram



III. MAPPING CONCEPTUAL MODEL TO RELATIONAL MODEL

Primary Key: Underlined **Foreign Key: *Italicized***

User(User_id,name,contact_number,login_credentials)

Maintanence_request(status, lease_agreement_id, description, request_type, request_id, *user_id*)

User_id: foreign keys refers to user_id in relation user: NULL not allowed

Property_image(property_id, description, image_url, image_id, *lease_agreement_id*)

lease_agreement_id: foreign keys refers to *lease_agreement_id* in relation Lease_agreement: NULL not allowed

Location(zip_code, state, city, location_id)

Contract(lease_agreement_id, legal_obligation, terms, contract_id)

Agent(agent_id, contact_number, services_offered, *lease_agreement_id*)

lease_agreement_id: foreign keys refers to *lease_agreement_id* in relation Lease_agreement: NULL not allowed

Payment(payment_id, amount, payment_methods, *lease_agreement_id, user_id*)

lease_agreement_id: foreign keys refers to *lease_agreement_id* in relation Lease_agreement: NULL not allowed

User_id: foreign keys refers to user_id in relation user: NULL not allowed

Review(review_id, user_rating, comments, *lease_agreement_id, user_id*)

lease_agreement_id: foreign keys refers to *lease_agreement_id* in relation Lease_agreement: NULL not allowed

User_id: foreign keys refers to user_id in relation user: NULL not allowed

Lease_agreement(lease_agreement_id, T&C, rental_period, payment_schedule, property_id)

Tenant(tenant_id, contact_number, tenant_history, *lease_agreement_id*)

lease_agreement_id: foreign keys refers to *lease_agreement_id* in relation Lease_agreement: NULL not allowed

Owner(owner_id, contact_number)

Property(property_id, ammenties, *owner_id,* address, size, rooms)

owner_id: foreign keys refers to *owner_id* in relation Owner: NULL not allowed

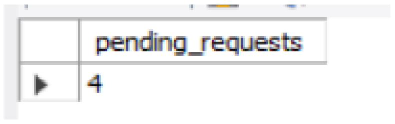
IV. IMPLEMENTATION OF RELATIONAL MODEL VIA MYSQL AND NOSQL

MySQL Implementation:

The database was created in MySQL and the following queries were performed:

Query 1: Finding the total number of maintenance requests in "Pending" status

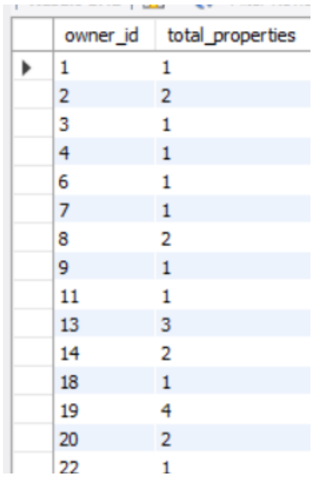
```
select count(*) as pending_requests
from maintenance_request
where status = 'Pending';
```



pending_requests
4

Query 2: Finding the total number of properties owned by each owner

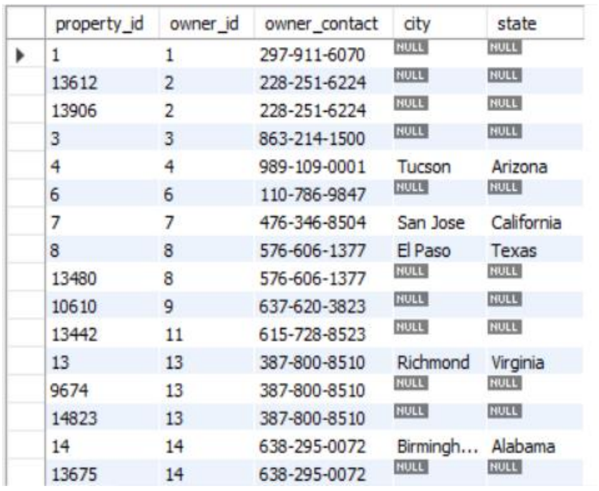
```
SELECT o.owner_id, COUNT(p.property_id) AS total_properties
FROM Owner o
JOIN Property p ON o.owner_id = p.owner_id
GROUP BY o.owner_id;
```



owner_id	total_properties
1	1
2	2
3	1
4	1
6	1
7	1
8	2
9	1
11	1
13	3
14	2
18	1
19	4
20	2
22	1

Query 3: Retrieving all properties with their owners and locations

```
SELECT p.property_id, o.owner_id, o.contact_number AS owner_contact, l.city, l.state
FROM Property p
JOIN Owner o ON p.owner_id = o.owner_id
LEFT JOIN Location l ON p.property_id = l.location_id;
```



property_id	owner_id	owner_contact	city	state
1	1	297-911-6070	NULL	NULL
13612	2	228-251-6224	NULL	NULL
13906	2	228-251-6224	NULL	NULL
3	3	863-214-1500	NULL	NULL
4	4	989-109-0001	Tucson	Arizona
6	6	110-786-9847	NULL	NULL
7	7	476-346-8504	San Jose	California
8	8	576-606-1377	El Paso	Texas
13480	8	576-606-1377	NULL	NULL
10610	9	637-620-3823	NULL	NULL
13442	11	615-728-8523	NULL	NULL
13	13	387-800-8510	Richmond	Virginia
9674	13	387-800-8510	NULL	NULL
14823	13	387-800-8510	NULL	NULL
14	14	638-295-0072	Birmingh...	Alabama
13675	14	638-295-0072	NULL	NULL

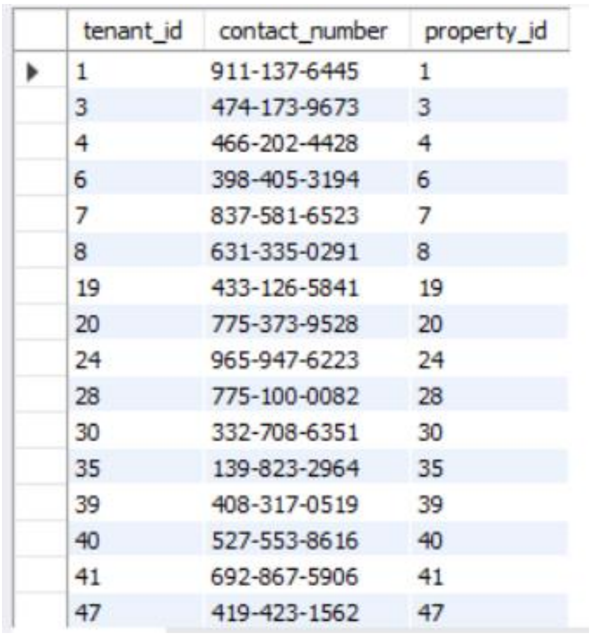
Query 4: Find tenants with a lease agreement for properties in a specific city

SELECT t.tenant_id, t.contact_number, la.property_id

FROM Tenant t

JOIN Lease_Agreement la ON t.lease_agreement_id = la.lease_agreement_id

WHERE la.property_id IN (SELECT property_id FROM Location WHERE city = 'Boston');



tenant_id	contact_number	property_id
1	911-137-6445	1
3	474-173-9673	3
4	466-202-4428	4
6	398-405-3194	6
7	837-581-6523	7
8	631-335-0291	8
19	433-126-5841	19
20	775-373-9528	20
24	965-947-6223	24
28	775-100-0082	28
30	332-708-6351	30
35	139-823-2964	35
39	408-317-0519	39
40	527-553-8616	40
41	692-867-5906	41
47	419-423-1562	47

Quey 5: Finding tenants with the highest payment amount

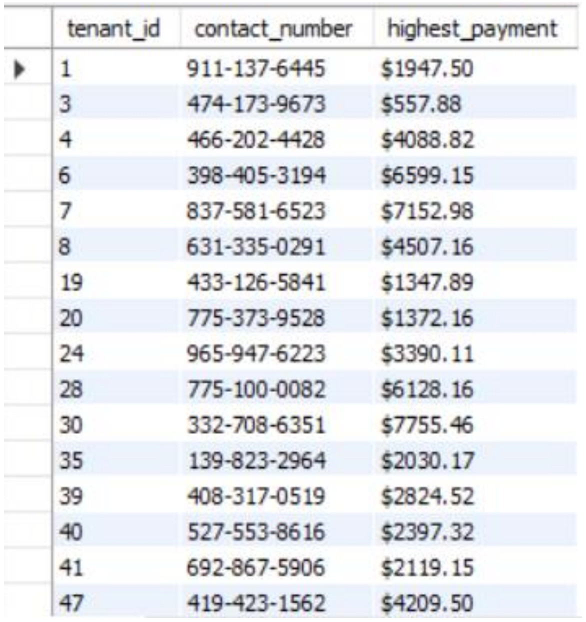
SELECT t.tenant_id, t.contact_number, MAX(p.amount) AS highest_payment

FROM Tenant t

JOIN Lease_Agreement la ON t.lease_agreement_id = la.lease_agreement_id

JOIN Payment p ON la.lease_agreement_id = p.lease_agreement_id

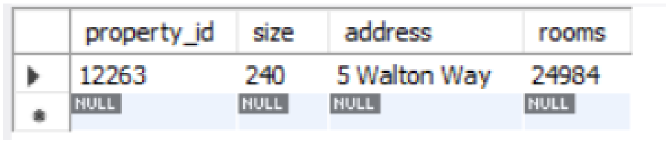
```
WHERE p.amount = (
        SELECT MAX(amount)
        FROM Payment
        WHERE lease_agreement_id = la.lease_agreement_id
    )
GROUP BY t.tenant_id, t.contact_number;
```



tenant_id	contact_number	highest_payment
1	911-137-6445	$1947.50
3	474-173-9673	$557.88
4	466-202-4428	$4088.82
6	398-405-3194	$6599.15
7	837-581-6523	$7152.98
8	631-335-0291	$4507.16
19	433-126-5841	$1347.89
20	775-373-9528	$1372.16
24	965-947-6223	$3390.11
28	775-100-0082	$6128.16
30	332-708-6351	$7755.46
35	139-823-2964	$2030.17
39	408-317-0519	$2824.52
40	527-553-8616	$2397.32
41	692-867-5906	$2119.15
47	419-423-1562	$4209.50

Query 6: Find properties where the size is greater than all other properties in terms of rooms

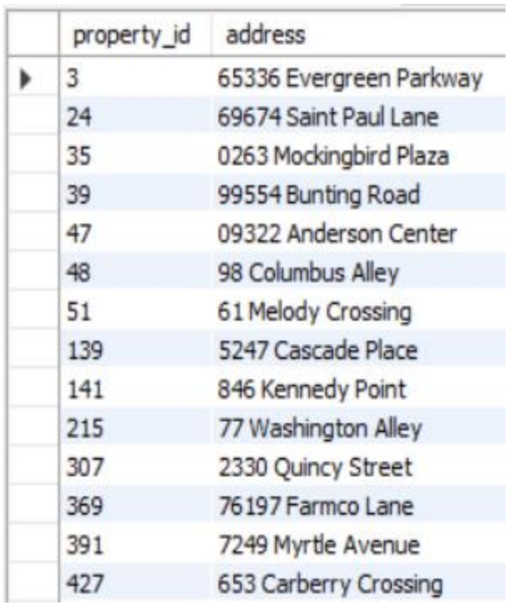
```
SELECT p.property_id, p.size, p.address, p.rooms
FROM Property p
WHERE p.rooms > ALL (
    SELECT rooms
    FROM Property
    WHERE property_id <> p.property_id
);
```



property_id	size	address	rooms
12263	240	5 Walton Way	24984
NULL	NULL	NULL	NULL

Query 7: Find properties with no maintenance requests

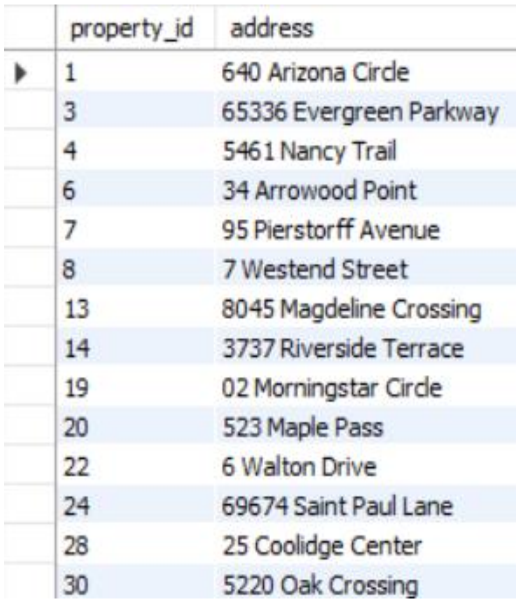
```
SELECT p.property_id, p.address
FROM Property p
WHERE NOT EXISTS (
    SELECT 1
    FROM Maintenance_Request mr
    WHERE mr.lease_agreement_id IN (
        SELECT la.lease_agreement_id
        FROM Lease_Agreement la
        WHERE la.property_id = p.property_id
    )
);
```



property_id	address
3	65336 Evergreen Parkway
24	69674 Saint Paul Lane
35	0263 Mockingbird Plaza
39	99554 Bunting Road
47	09322 Anderson Center
48	98 Columbus Alley
51	61 Melody Crossing
139	5247 Cascade Place
141	846 Kennedy Point
215	77 Washington Alley
307	2330 Quincy Street
369	76197 Farmco Lane
391	7249 Myrtle Avenue
427	653 Carberry Crossing

Query 8: Find properties with maintenance requests or reviews

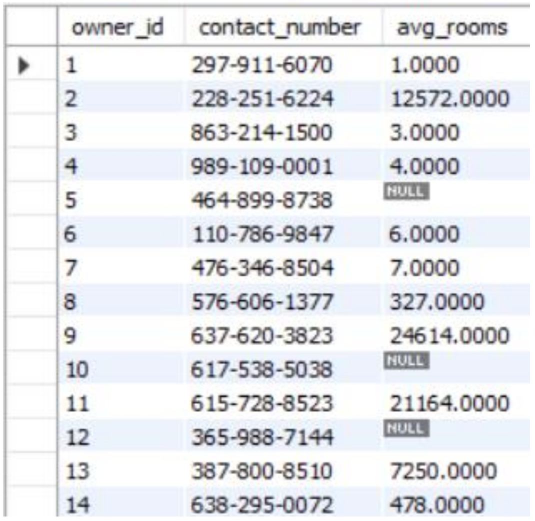
```
SELECT DISTINCT p.property_id, p.address
FROM Property p
WHERE p.property_id IN (
    SELECT DISTINCT la.property_id
    FROM Maintenance_Request mr
    JOIN Lease_Agreement la ON mr.lease_agreement_id = la.lease_agreement_id
    UNION
    SELECT DISTINCT la.property_id
    FROM Review r
    JOIN Lease_Agreement la ON r.lease_agreement_id = la.lease_agreement_id
);
```



property_id	address
1	640 Arizona Circle
3	65336 Evergreen Parkway
4	5461 Nancy Trail
6	34 Arrowood Point
7	95 Pierstorff Avenue
8	7 Westend Street
13	8045 Magdeline Crossing
14	3737 Riverside Terrace
19	02 Morningstar Circle
20	523 Maple Pass
22	6 Walton Drive
24	69674 Saint Paul Lane
28	25 Coolidge Center
30	5220 Oak Crossing

Query 9: Calculate the average number of rooms for properties owned by each owner

```
SELECT o.owner_id, o.contact_number,
    (SELECT AVG(rooms) FROM Property WHERE owner_id = o.owner_id) AS avg_rooms
FROM Owner o;
```



owner_id	contact_number	avg_rooms
1	297-911-6070	1.0000
2	228-251-6224	12572.0000
3	863-214-1500	3.0000
4	989-109-0001	4.0000
5	464-899-8738	NULL
6	110-786-9847	6.0000
7	476-346-8504	7.0000
8	576-606-1377	327.0000
9	637-620-3823	24614.0000
10	617-538-5038	NULL
11	615-728-8523	21164.0000
12	365-988-7144	NULL
13	387-800-8510	7250.0000
14	638-295-0072	478.0000

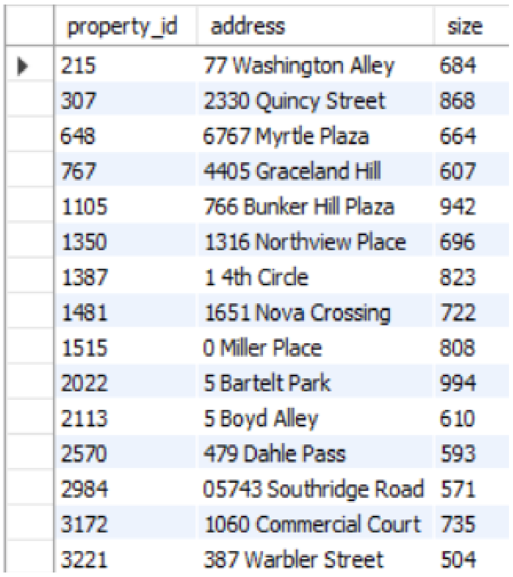
Query 10: Retrieve properties with a size greater than the average size of properties

SELECT p.property_id, p.address, p.size

FROM Property p

JOIN (SELECT AVG(size) AS avg_size FROM Property) AS avg_prop

ON p.size > avg_prop.avg_size;



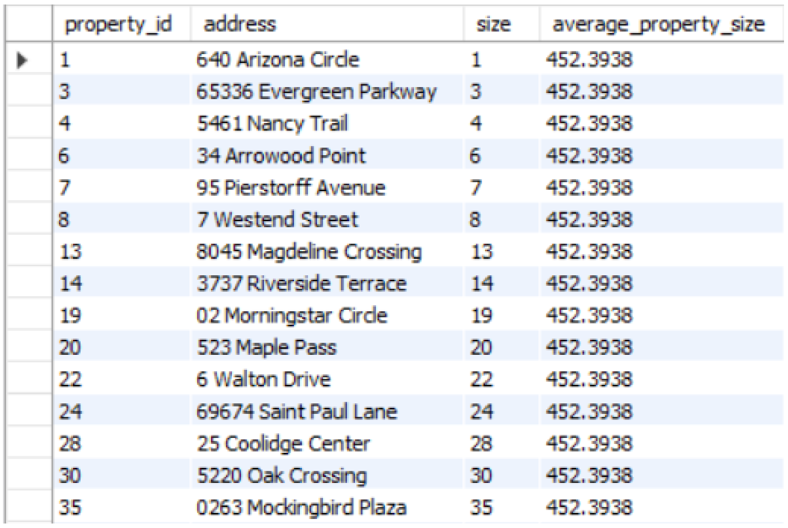
property_id	address	size
215	77 Washington Alley	684
307	2330 Quincy Street	868
648	6767 Myrtle Plaza	664
767	4405 Graceland Hill	607
1105	766 Bunker Hill Plaza	942
1350	1316 Northview Place	696
1387	1 4th Circle	823
1481	1651 Nova Crossing	722
1515	0 Miller Place	808
2022	5 Bartelt Park	994
2113	5 Boyd Alley	610
2570	479 Dahle Pass	593
2984	05743 Southridge Road	571
3172	1060 Commercial Court	735
3221	387 Warbler Street	504

Query 11: Retrieve the average size of properties and display it for each property

SELECT property_id, address, size,

(SELECT AVG(size) FROM Property) AS average_property_size

FROM Property;

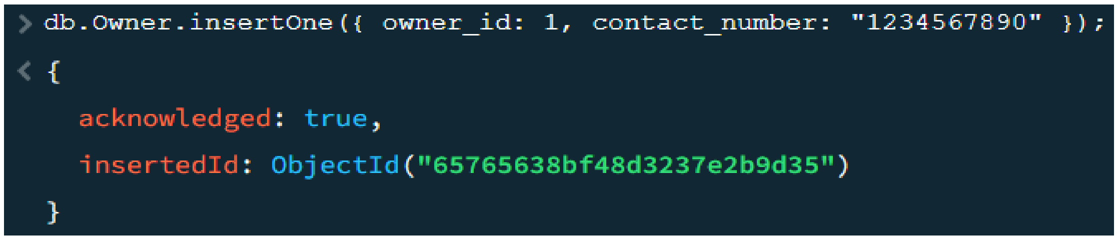


property_id	address	size	average_property_size
1	640 Arizona Circle	1	452.3938
3	65336 Evergreen Parkway	3	452.3938
4	5461 Nancy Trail	4	452.3938
6	34 Arrowood Point	6	452.3938
7	95 Pierstorff Avenue	7	452.3938
8	7 Westend Street	8	452.3938
13	8045 Magdeline Crossing	13	452.3938
14	3737 Riverside Terrace	14	452.3938
19	02 Morningstar Circle	19	452.3938
20	523 Maple Pass	20	452.3938
22	6 Walton Drive	22	452.3938
24	69674 Saint Paul Lane	24	452.3938
28	25 Coolidge Center	28	452.3938
30	5220 Oak Crossing	30	452.3938
35	0263 Mockingbird Plaza	35	452.3938

NoSQL Implementation:

Query 1: Insert document

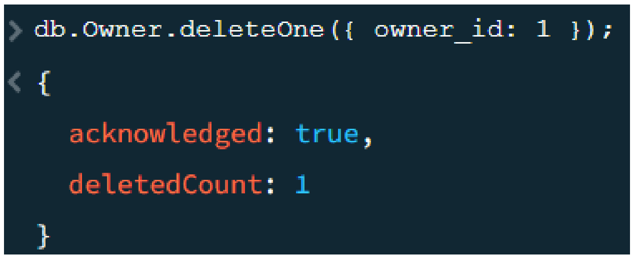
db.Owner.insertOne({ owner_id: 1, contact_number: "1234567890" });



```
> db.Owner.insertOne({ owner_id: 1, contact_number: "1234567890" });
< {
    acknowledged: true,
    insertedId: ObjectId("65765638bf48d3237e2b9d35")
  }
```

Query 2: Delete document

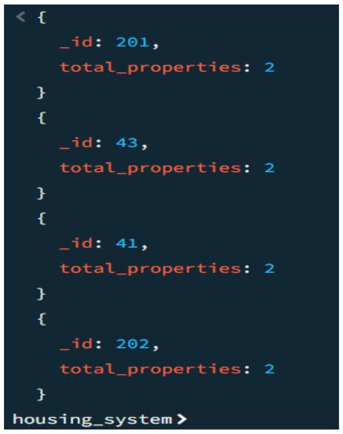
db.Owner.deleteOne({ owner_id: 1 });



```
> db.Owner.deleteOne({ owner_id: 1 });
< {
    acknowledged: true,
    deletedCount: 1
  }
```

Query 3: Count the number of properties owned by each owner

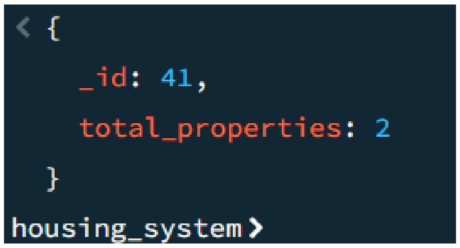
```
db.Property.aggregate([
  {
    $group: {
      _id: "$owner_id",
      total_properties: { $sum: 1 }
    }
  }
]);
```



```
< {
    _id: 201,
    total_properties: 2
  }
  {
    _id: 43,
    total_properties: 2
  }
  {
    _id: 41,
    total_properties: 2
  }
  {
    _id: 202,
    total_properties: 2
  }
housing_system>
```

Query 4: Find the owner with the most properties

```
db.Property.aggregate([
  {
    $group: {
      _id: "$owner_id",
      total_properties: { $sum: 1 }
    }
  },
  {
    $sort: { total_properties: -1 }
  },
  {
    $limit: 1
  }
]);
```

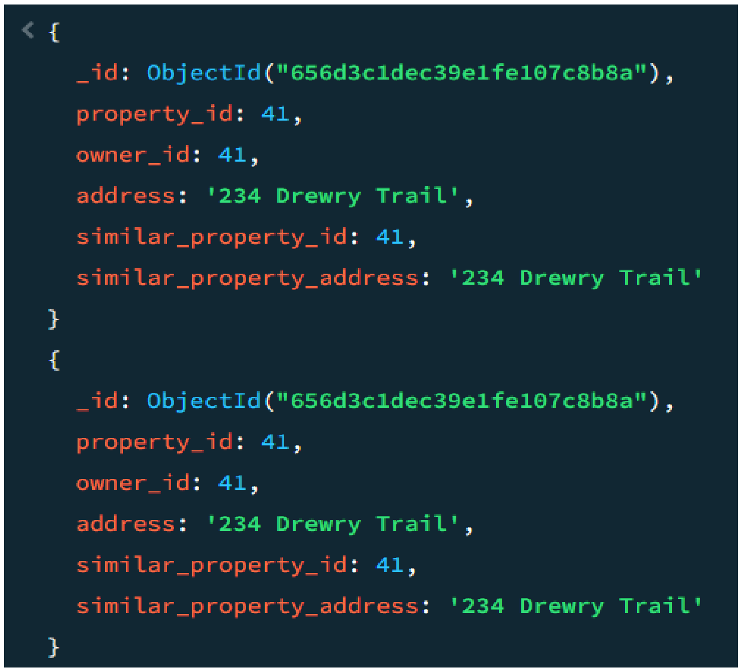


```
< {
    _id: 41,
    total_properties: 2
  }
housing_system>
```

Query 5: Find Properties with Similar Amenities

```
db.Property.aggregate([
  {
    $lookup: {
      from: "Property",
      localField: "amenities",
      foreignField: "amenities",
      as: "similar_properties"
    }
  },
  {
```

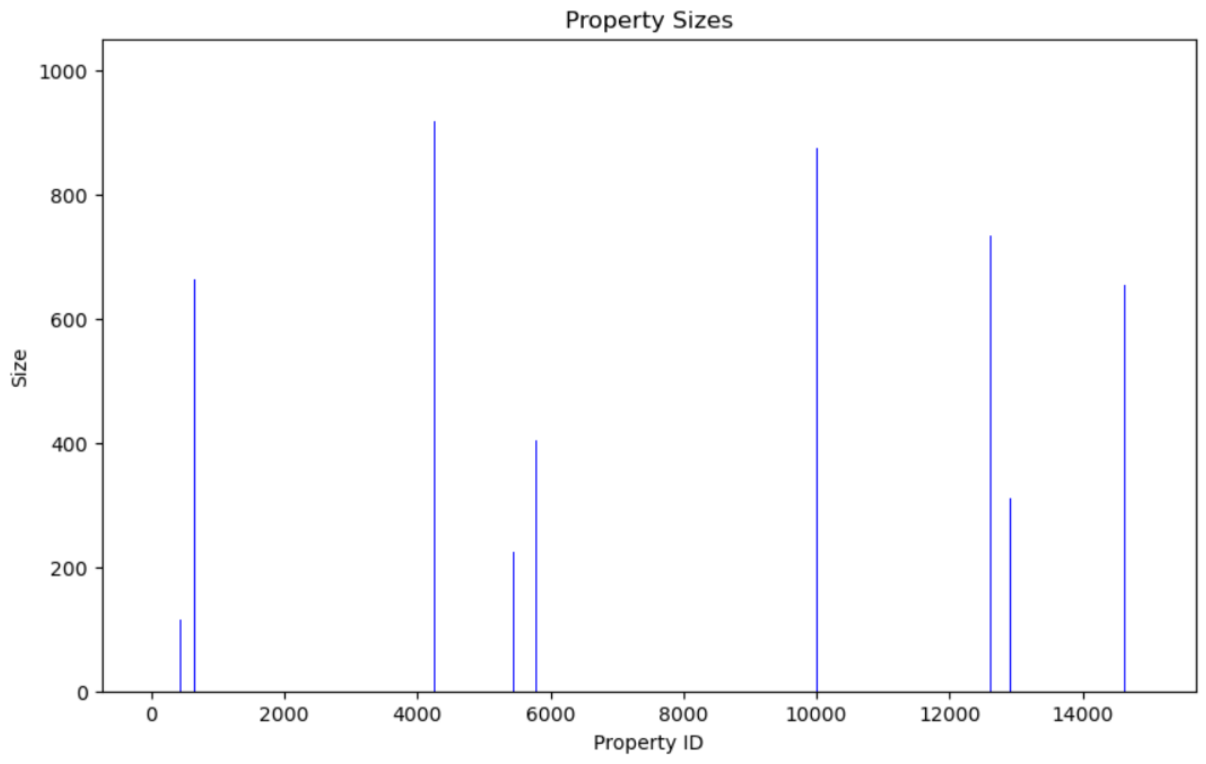
```
  $unwind: "$similar_properties"
 },
 {
  $match: {
    "similar_properties.property_id": { $ne: "$property_id" }
  }
 },
 {
  $project: {
    property_id: 1,
    address: 1,
    owner_id: 1,
    owner_contact: 1,
    similar_property_id: "$similar_properties.property_id",
    similar_property_address: "$similar_properties.address"
  }
 }
]);
```



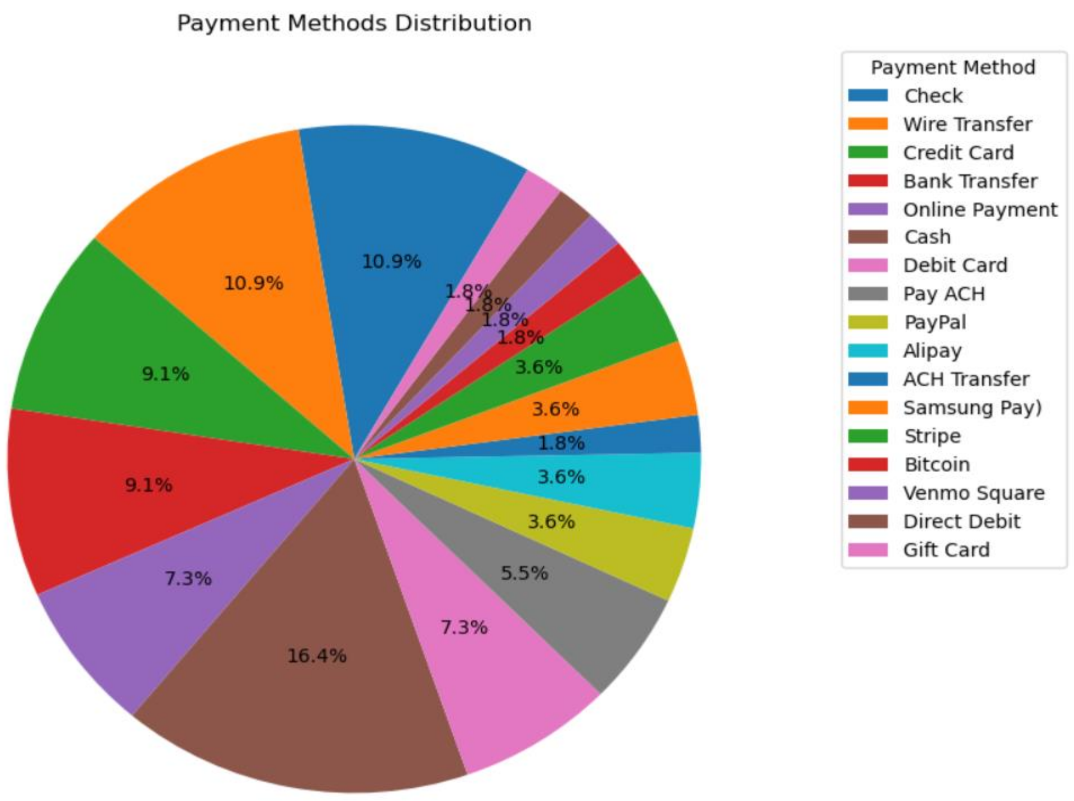
```
< {
    _id: ObjectId("656d3c1dec39e1fe107c8b8a"),
    property_id: 41,
    owner_id: 41,
    address: '234 Drewry Trail',
    similar_property_id: 41,
    similar_property_address: '234 Drewry Trail'
  }
  {
    _id: ObjectId("656d3c1dec39e1fe107c8b8a"),
    property_id: 41,
    owner_id: 41,
    address: '234 Drewry Trail',
    similar_property_id: 41,
    similar_property_address: '234 Drewry Trail'
  }
```

V. DATABASE ACCESS VIA PYTHON

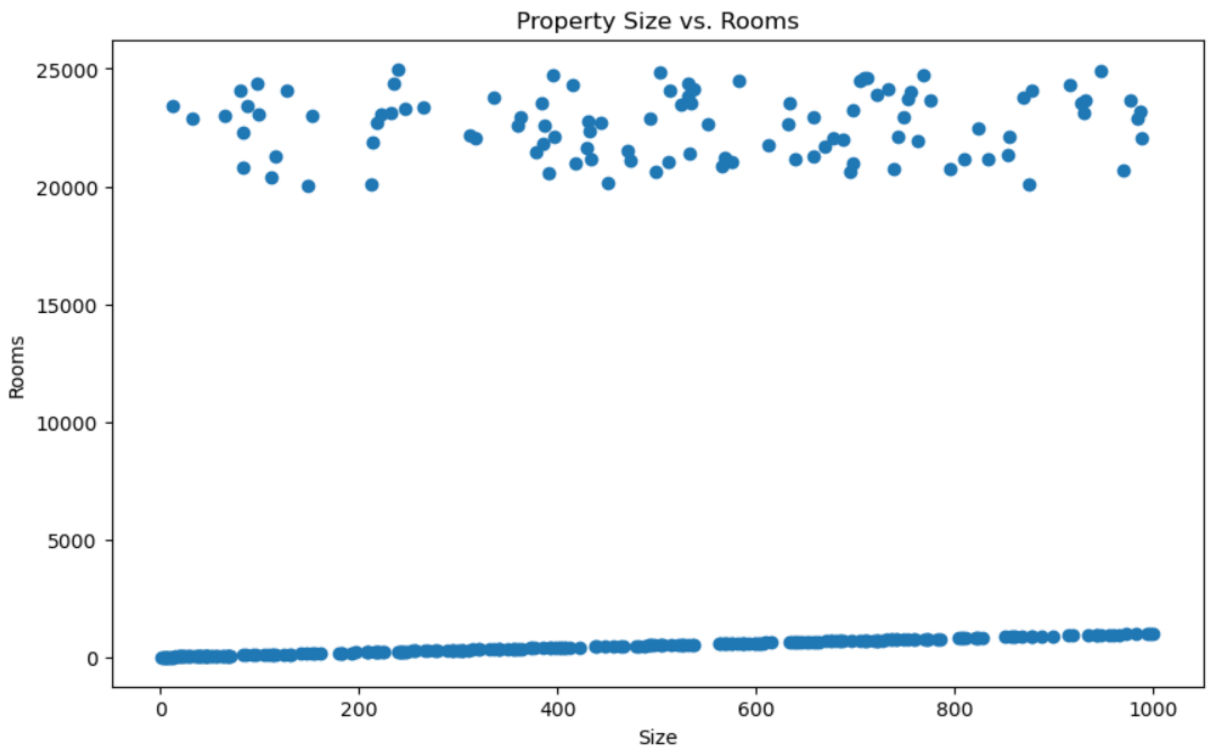
The database is accessed using Python and visualization of analyzed data is shown below. The connection of MySQL to Python is done using mysql.connector, followed by cursor.execute to run and fetch from query, followed by converting the list into a dataframe using pandas library and using matplotlib to plot the graphs for the analytics.



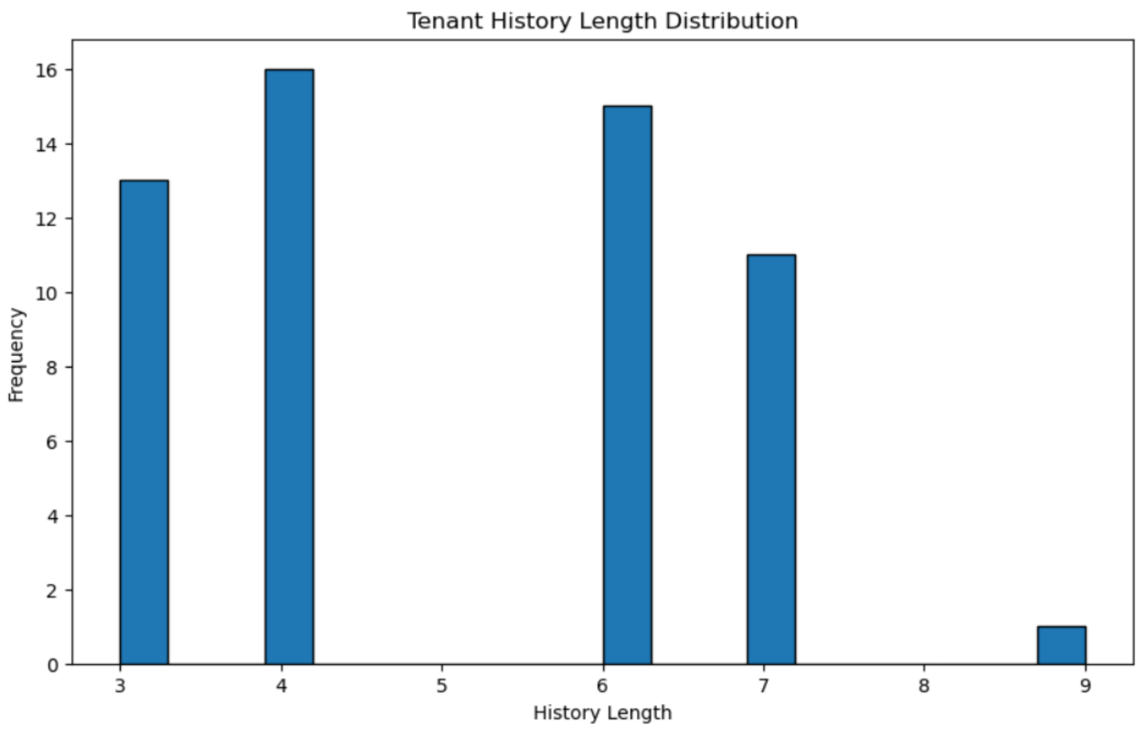
Graph1: Property Sizes



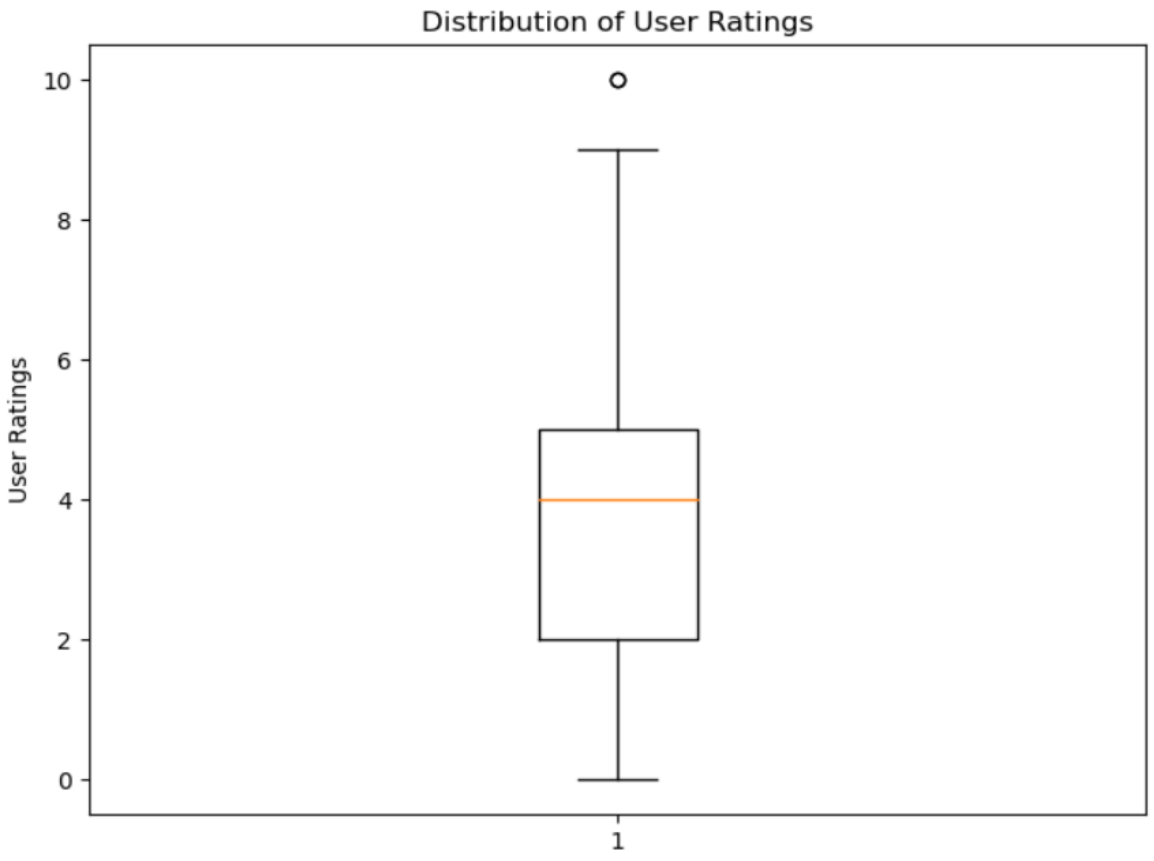
Graph2: Payment Methods Distribution



Graph3: Property Size VS Rooms



Graph4: Tenant History Length Distribution



Graph5: Distribution of User Ratings

VII. SUMMARY:

The Boston House Rental Management System presents an innovative and efficient solution to the challenges faced in the rental property management landscape. Rooted in the power of technology and a robust relational database, the system streamlines the entire house rental process for property owners, tenants, and administrators in the dynamic city of Boston.

The system addresses the complexities of lease management, financial transactions, maintenance requests, and background checks, providing a user-friendly interface and a centralized hub for property details. Leveraging SQL queries ensures optimal data integrity and performance, offering a seamless experience for all stakeholders.

By offering real-time updates for property owners and simplified search processes for tenants, the Rental Management System becomes a pivotal tool in organizing and managing the Boston housing market efficiently. The introduction of an employee entity enhances transparency and security, allowing for background checks on property management personnel.

VIII. RECOMMENDATION:

Based on the comprehensive features and advantages offered by the Boston House Rental Management System, it is recommended for implementation across the rental property landscape in Boston. The system's ability to significantly reduce data duplication, streamline processes, and enhance overall efficiency aligns with the current technological shift in the real estate industry.

The implementation of this system promises substantial benefits for property owners, tenants, and administrators by providing a centralized platform for efficient communication, transparent property management, and simplified search processes. The relational database structure, driven by SQL, ensures a secure and organized framework, paving the way for a more accessible and streamlined house rental experience in Boston.

To maximize the impact of the Boston Rental Management System, a phased implementation approach is recommended. Commencing with a pilot rollout to a subset of properties will allow for comprehensive testing and fine-tuning of the system. Subsequent phases can then gradually expand the system's coverage to encompass a wider range of properties and stakeholders.

In conclusion, the Boston House Rental Management System stands as a transformative solution poised to elevate the efficiency and transparency of house rental processes. Its implementation is recommended for property management entities and administrators seeking to navigate the complexities of the Boston housing market with agility and technological finesse.