USE CASE STUDY REPORT

Group No.: Group 03

Student Names: Sai Anish Sreeramagiri and Sai Prashanth Vaditha

Executive Summary

The main goal of the ParkBnB project was to create and implement a relational database system that addresses parking problems in urban areas. Like Airbnb, ParkBnB is based on the idea of the sharing economy and aims to improve how parking spaces are accessed in crowded cities. The platform connects drivers with private parking spot owners, helping reduce the time spent looking for parking and making better use of available spaces, ultimately improving city mobility.

We started developing the ParkBnB database by researching business requirements. After brainstorming, we outlined the platform's needs and created conceptual diagrams to represent the key entities and their relationships. Once the conceptual model was ready, we designed a relational model, using primary and foreign keys to structure the data and ensure minimal violations of normal forms. This model was then implemented in MySQL to manage and track important data like user accounts, parking spot listings, reservations, and transactions.

Next, we developed a Python application to integrate with the MySQL database. This application allowed us to query data and create visualizations, helping us understand user behavior, parking trends, and platform usage. We also explored MongoDB to build a NoSQL version of the database, considering its potential for scalability in the future.

The project achieved its goals by improving operational efficiency and providing useful insights through data analysis. In the future, the platform could be enhanced by adding real-time parking availability updates, scaling to handle more users, and including public parking spots, which would make it even more effective and valuable to the community.

I. Introduction

During a trip to New York City, we experienced how frustrating and time-consuming it can be to find parking in a busy city. After driving around for over an hour we finally found one, only to discover that the cost was nearly as much as our hotel stay! This experience made us realize just how broken the parking system is in major urban cities. Drivers like us waste precious time and money searching for spots, while property owners with unused parking spaces miss out on the chance to generate income. The

existing parking systems are inefficient with inconsistent availability and high pricing, making it clear that something needs to change.

That's when the idea for ParkBnB was born. We realized that there was an opportunity to create a platform that could make parking easier, more affordable, and more accessible. By connecting drivers with private property owners who have unutilized parking spaces, ParkBnB aims to simplify the parking process and optimize space usage. The platform would allow drivers to search for, reserve, and pay for parking on-demand, while also giving property owners a way to earn extra income from their unused spaces.

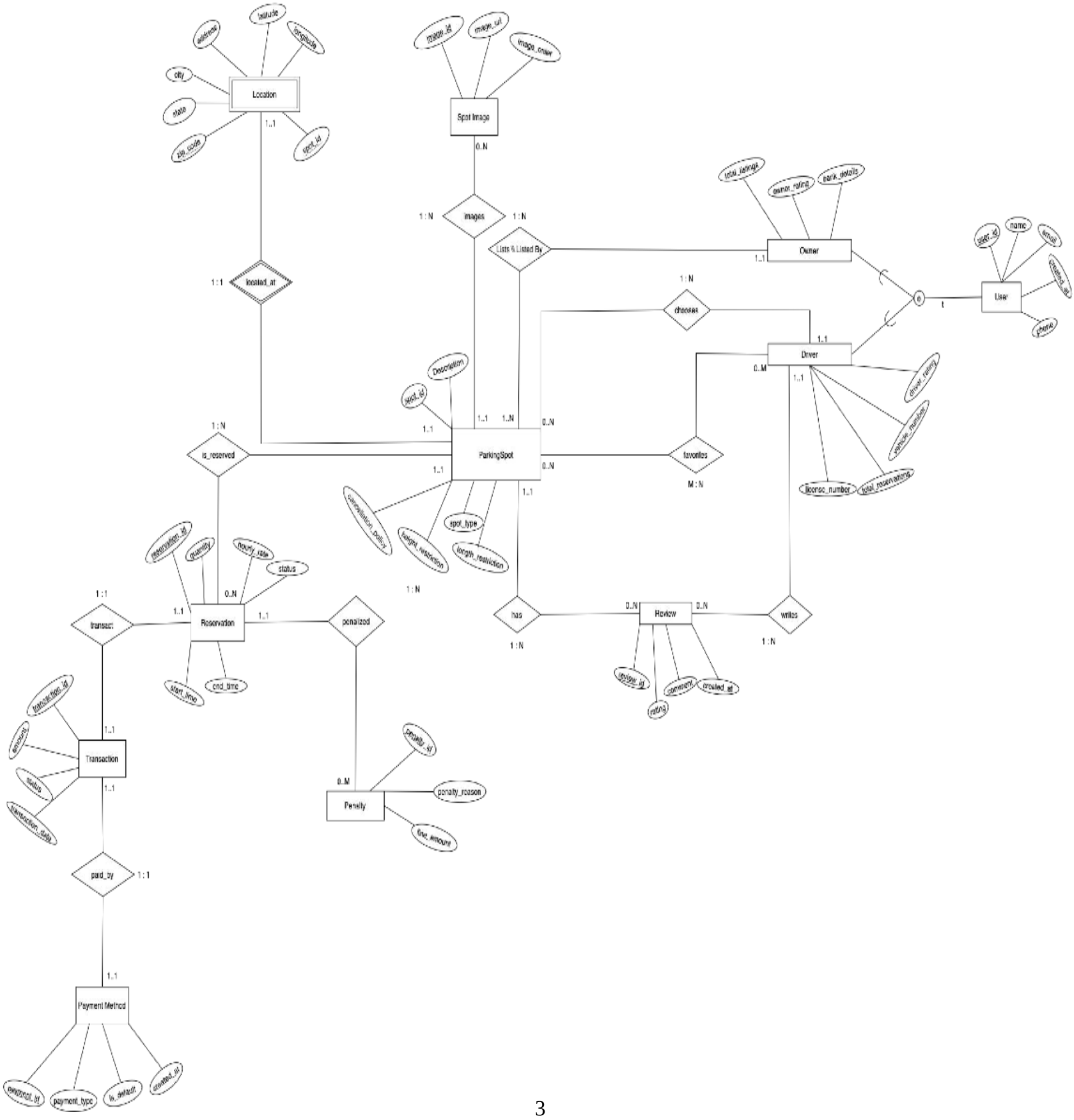
To bring this vision to life, we identified several key requirements:

- User Account Management to securely store and manage user information, ensuring a personalized experience.
- Search and Book Parking functionality, allowing users to filter by location, price, and duration for better customization.
- Payment Integration supporting secure methods, including credit/debit cards, PayPal, and mobile wallets, for seamless transactions.
- Reservation and Confirmation system, providing real-time availability updates and instant booking confirmation for peace of mind.
- Rating and Review System to maintain trust and quality, where users and property owners can rate their experiences.
- Data Analytics and Insights to track parking trends, optimize availability, and provide valuable insights for future improvements.

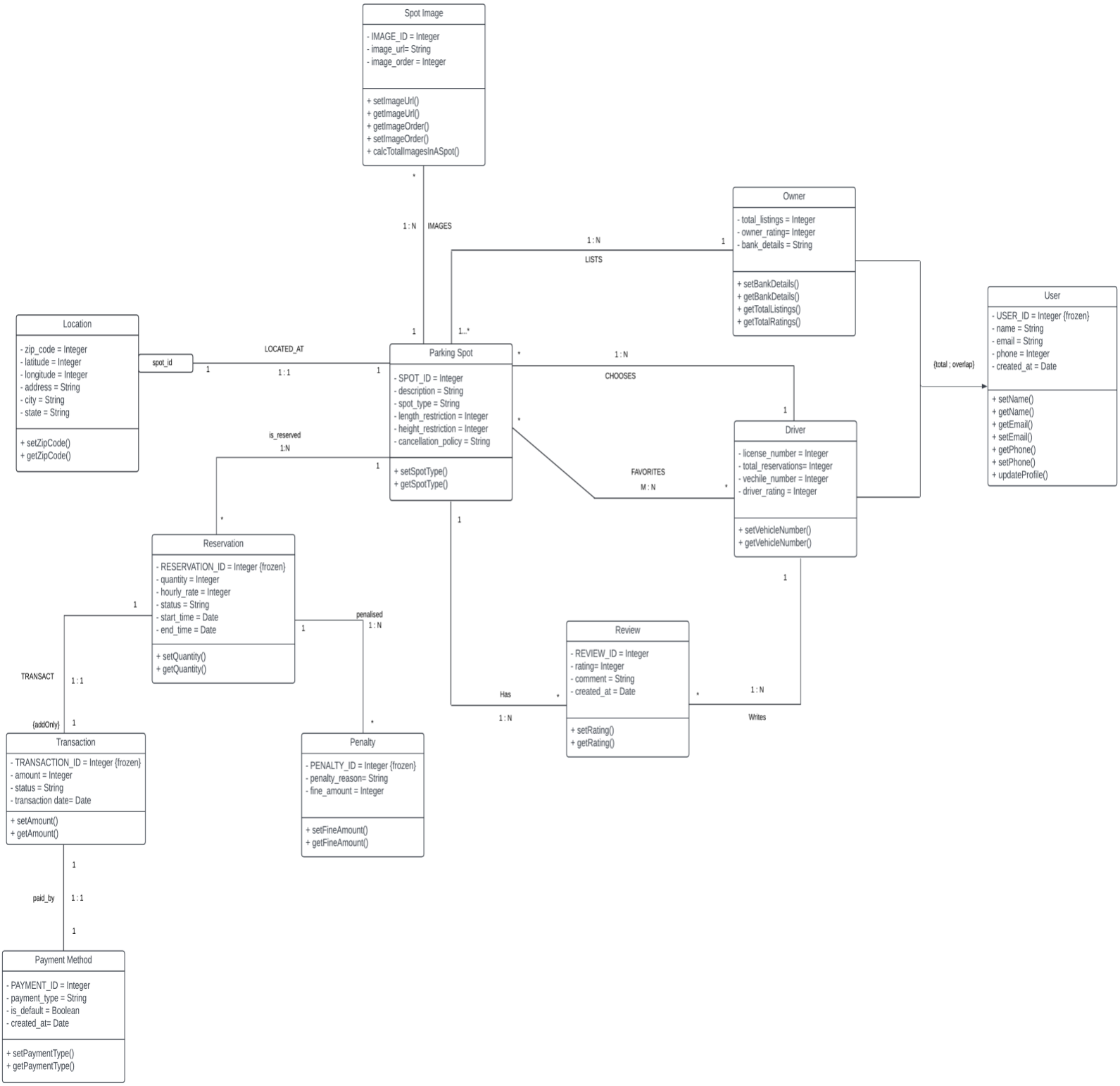
Through ParkBnB, we aim to create a more efficient, affordable, and user-friendly parking experience, benefiting both drivers and property owners in urban centers.

II. Conceptual Data Modeling:

1. EER Diagram:



2. UML Diagram:



III. Mapping Conceptual Model to Relational Model

Primary Key: **Bold and Underlined** Foreign Key: *Italicized and red*

1. User (**user_id**, name, email, created_at, phone)

2. Owner (***owner_id***, total_listings, owner_rating, bank_details)

3. Driver (***driver_id***, license_number, total_reservations, driver_rating)

4. Favorites (***driver_id***, ***spot_id***)
   driver_id and spot_id are primary and foreign keys from Driver and ParkingSpot Tables – NOT NULL for both

5. Review (**review_id**, rating, comment, created_at, *driver_id*, *spot_id*)
   driver_id and spot_id are foreign keys from Driver and ParkingSpot tables; driver_id and spot_id are both NOT NULL

6. ParkingSpot (**spot_id**, description, cancellation_policy, height_restriction, spot_type, length_restriction, *owner_id*, *location_id*)
   owner_id and location_id are foreign keys from Owner and Location tables; both are NOT NULL
7. SpotImage (**image_id**, image_url, image_order, *spot_id*)
   spot_id is the foreign key from ParkingSpot table where it is NOT NULL

8. Location(**location_id**, latitude, longitude, address, *zipcode*)
   Zipcode is foreign key from the Zipcode table

9. Zipcode(**zipcode**, city, state)

10. Reservation (**reservation_id**, quantity, hourly_rate, status, start_time, end_time, *spot_id*, *driver_id*)
    spot_id and driver_id are foreign keys from ParkingSpot and Driver tables and both are NOT NULL

11. Transaction (**transaction_id**, amount, status, transaction_date, *reservation_id*, *payment_id*)
    reservation_id and payment_id are foreign keys; both are NOT NULL

12. PaymentMethod (**payment_id**, payment_type, is_default, created_at)

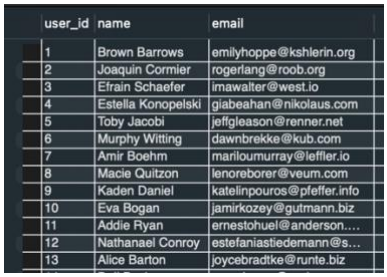
13. Penalty (**penalty_id**, penalty_reason, fine_amount, *reservation_id*)
    reservation_id is foreign key which is NOT NULL

IV. Implementation of Relation Model via MySQL and NoSQL

MySQL Implementation:

Query 1: List of Users

```
SELECT user_id, name, email
FROM User;
```



user_id	name	email
1	Brown Barrows	emilyhoppe@kshlerin.org
2	Joaquin Cormier	rogerlang@roob.org
3	Efrain Schaefer	imawalter@west.io
4	Estella Konopelski	giabeahan@nikolaus.com
5	Toby Jacobi	jeffgleason@renner.net
6	Murphy Witting	dawnbrekke@kub.com
7	Amir Boehm	mariloumurray@leffler.io
8	Macie Quitzon	lenoreborer@veum.com
9	Kaden Daniel	katelinpouros@pfeffer.info
10	Eva Bogan	jamirkozey@gutmann.biz
11	Addie Ryan	ernestohuel@anderson....
12	Nathanael Conroy	estefaniastiedemann@s...
13	Alice Barton	joycebradtke@runte.biz

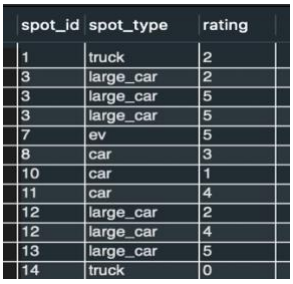
Query 2: Average rating per spot

```
SELECT spot_id, AVG(rating) AS avg_rating
FROM Review
GROUP BY spot_id;
```

spot_id	avg_rating
1	2.0000
3	4.0000
7	5.0000
8	3.0000
10	1.0000
11	4.0000
12	3.0000
13	5.0000
14	0.0000
15	3.0000

Query 3: Parking spots that have been reviewed, along with their type and ratings.

```
SELECT p.spot_id,p.spot_type, r.rating
FROM ParkingSpot p
INNER JOIN Review r ON p.spot_id = r.spot_id
order by p.spot_id;
```



spot_id	spot_type	rating
1	truck	2
3	large_car	2
3	large_car	5
3	large_car	5
7	ev	5
8	car	3
10	car	1
11	car	4
12	large_car	2
12	large_car	4
13	large_car	5
14	truck	0

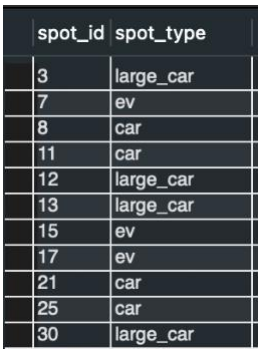
Query 4: Counts of many reservations have been made for each parking spot. If a spot has no reservations, the count will be 0.

```
SELECT p.spot_id, COUNT(r.reservation_id) AS num_reservations
FROM ParkingSpot p
LEFT JOIN Reservation r ON p.spot_id = r.spot_id
GROUP BY p.spot_id;
```

spot_id	num_reservatio...
1	2
2	2
3	1
4	2
5	2
6	3
7	1
8	0
9	2

Query 5: Average rating higher than the overall system's average

```
SELECT spot_id,spot_type
FROM ParkingSpot
WHERE spot_id IN (
   SELECT spot_id
   FROM Review
   GROUP BY spot_id
   HAVING AVG(rating) > (
      SELECT AVG(rating)
      FROM Review
   ));
```



spot_id	spot_type
3	large_car
7	ev
8	car
11	car
12	large_car
13	large_car
15	ev
17	ev
21	car
25	car
30	large_car

Query 6: Find Owners_id and total_listings of owner who Have Reservations with an Average Amount Above $5

```
SELECT o.owner_id,o.total_listings
FROM Owner o
WHERE o.owner_id = (
   SELECT p.owner_id
   FROM ParkingSpot p
   JOIN Reservation r ON p.spot_id = r.spot_id
   WHERE p.owner_id = o.owner_id
   GROUP BY p.owner_id
   HAVING AVG(r.hourly_rate) > 5
);
```

owner_id	total_listings
2	3
3	1
7	7
8	5
11	10
12	3
18	10
23	9
25	7
26	1

Query 7: Drivers who have made reservations where the start time is after 6PM - EXISTS

```
SELECT driver_id, license_number
FROM Driver d
WHERE EXISTS (
   SELECT 1
   FROM Reservation r
   WHERE r.driver_id = d.driver_id
   AND r.start_time > '18:00:00'
);
```

driver_id	license_numb...
820	102774045
100	110380666
727	112258015
603	134988431
523	139999817
297	140103957
287	142187320
492	157107801
824	168839548
695	184268528

Query 8: Parking spot in Seattle and portland -- UNION

```
SELECT p.spot_id FROM parkingspot p , location l, zipcode z
WHERE p.location_id=l.location_id AND l.zipcode=z.zipcode AND z.city='Seattle'
UNION
SELECT p.spot_id FROM parkingspot p , location l, zipcode z
WHERE p.location_id=l.location_id AND l.zipcode=z.zipcode AND z.city='Portland';
```

spot_id
175
432
449
451
57
58
99
198
339
345

Query 9: Find the Total Revenue for Each Owner from Reservations

```
SELECT o.owner_id, o.total_listings,
(SELECT SUM(r.quantity * r.hourly_rate)
FROM Reservation r
JOIN ParkingSpot ps ON r.spot_id = ps.spot_id
WHERE ps.owner_id = o.owner_id) AS total_revenue
FROM Owner o;
```

owner_id	total_listings	total_revenue
2	3	25.50
3	1	19.00
5	10	NULL
7	7	19.00
8	5	13.00
11	10	67.50
12	3	24.50
13	1	NULL

Implementation in NoSQL:

We implemented our database in MongoDB and executed a few queries mentioned below:

Query 1: Find all the reviews

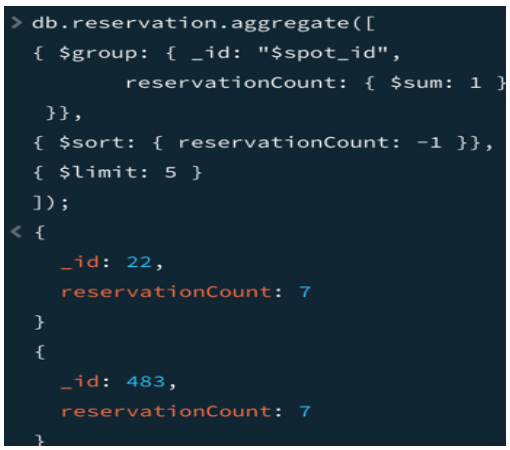
```
db.review.find();
```

Query 2: Find the 5 Most Recent Transactions that were successful

```
db.transaction.find() .
sort({ transaction_date: -1
}).limit(5);
```

Query 3: Find the Most Popular Parking Spots

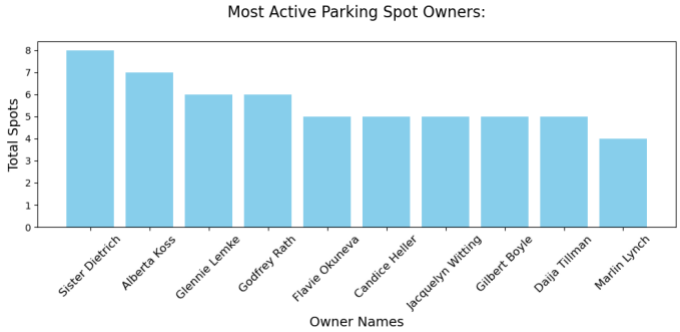
```
db.Reservation.aggregate([
        { $group: { _id: "$spot_id",
                    reservationCount: { $sum: 1 }
         }},
        { $sort: { reservationCount: -1 }},
        { $limit: 5 }
]);
```



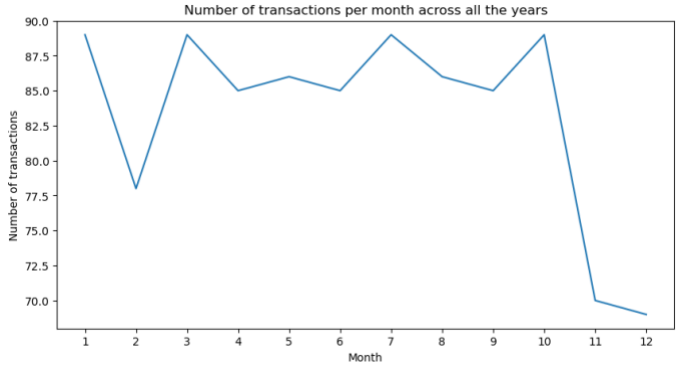
V. Database Access via Python:

Accessed our database in MySQL from Python using pymysql library. Utilized libraries like matplotlib and seaborn to get visualizations from the queried data.

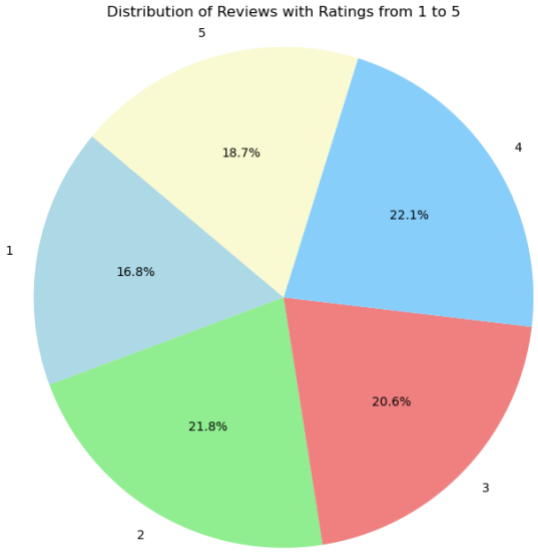
Graph 1: Most Active Parking Spot Owners



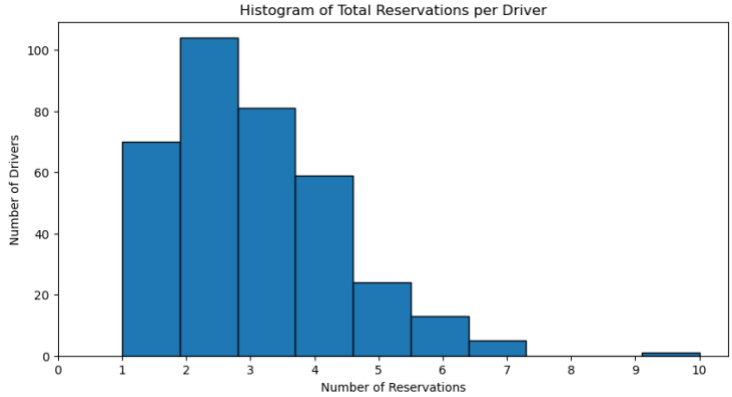
Graph 2: Number of Transactions per month across all years



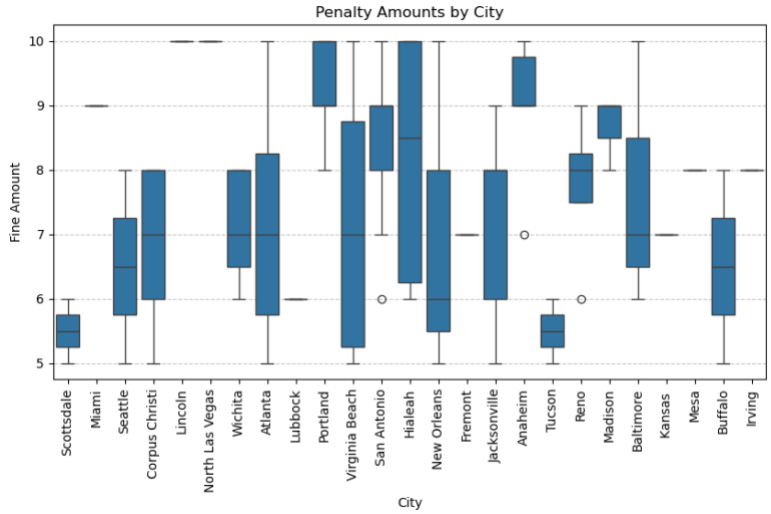
Graph 3: Distribution of Ratings



Graph 4: Histogram of Total Reservations per Driver



Graph 5: Boxplot showing the penalty amounts paid by drivers per City



VII. Summary and recommendation

The ParkBnB project was a great learning experience, and it helped us tackle the issue of urban parking challenges through the design and implementation of a relational database system. Using Python, we were able to integrate the database and extract valuable insights about user behavior and parking trends. Additionally, we experimented with a NoSQL database for future scalability. Overall, the system improved the efficiency of operations and provided solutions for better urban mobility. In the future, we think adding real time parking availability updates, parking spot location tracking and including public parking spots could make the platform even more effective and user friendly.