House Rental System Group 5 Final Report

Course/Term:CS5200 Fall 2024

Members:Chunzhang Liu, Haomiao Shi,Ran Cao,Tianmeng Xia

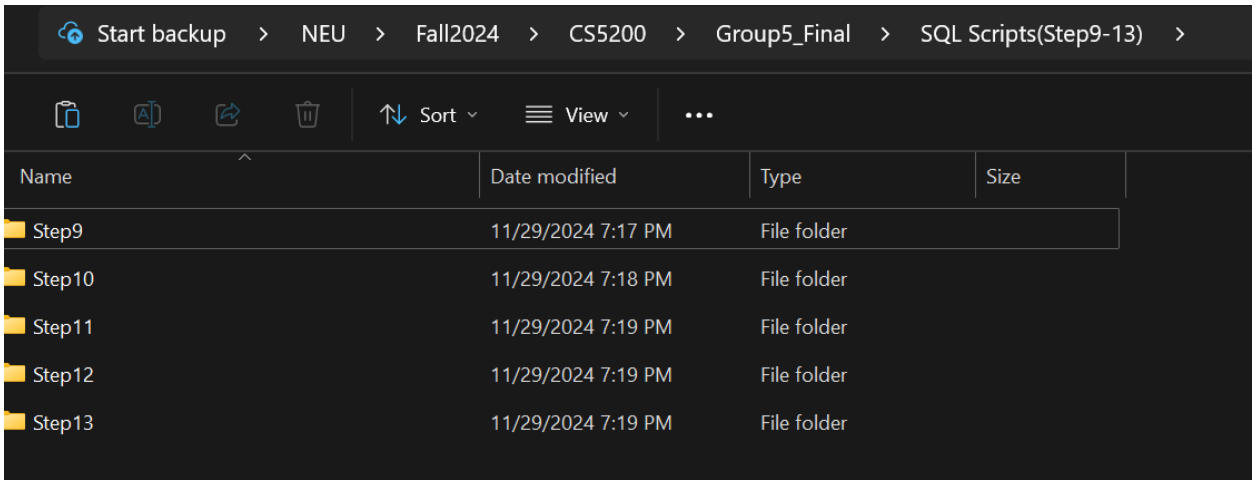
Instructor's Name:Tehmina Amjad

Institution Name: Northeastern University

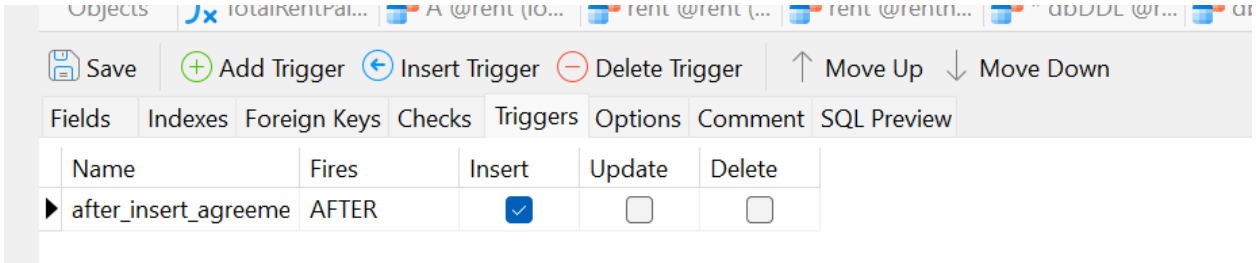
2. Updated Documentation: Revise Phase I & II deliverables if needed.

2.1 Updated ER Diagram

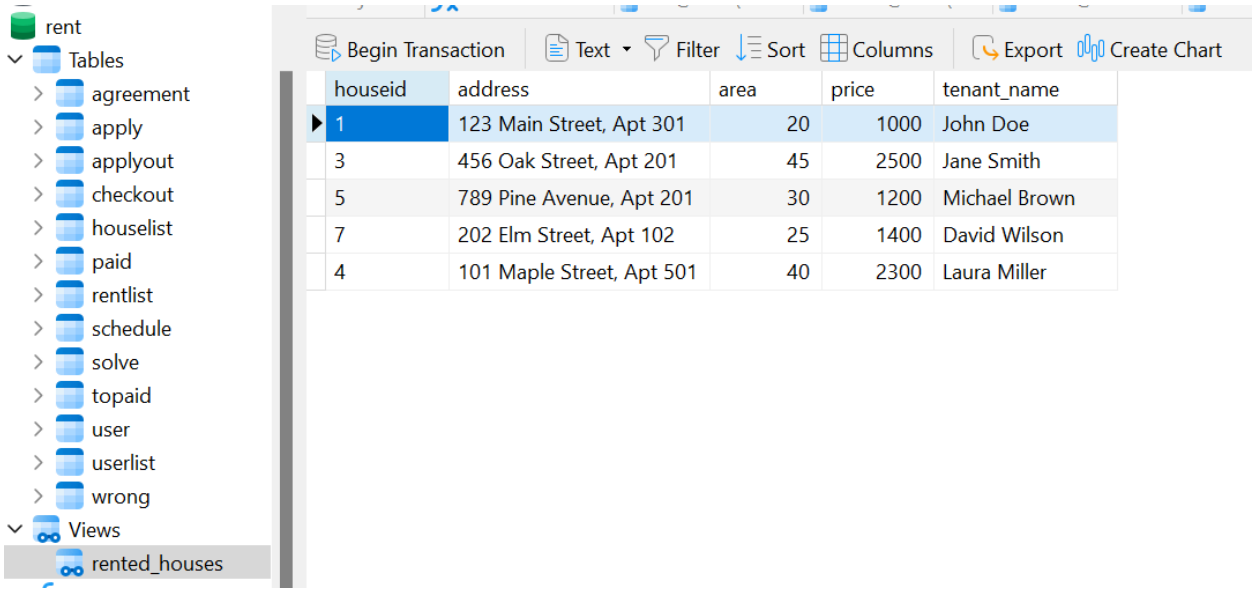
3. SQL Scripts: All scripts from Steps 9-13. (Zip File)



Step9:
dao from DAO folder in the backend code
Step10:
Some updates on the table naming, but the framework is not changed.
We created triggers, view, and functions
trigger:

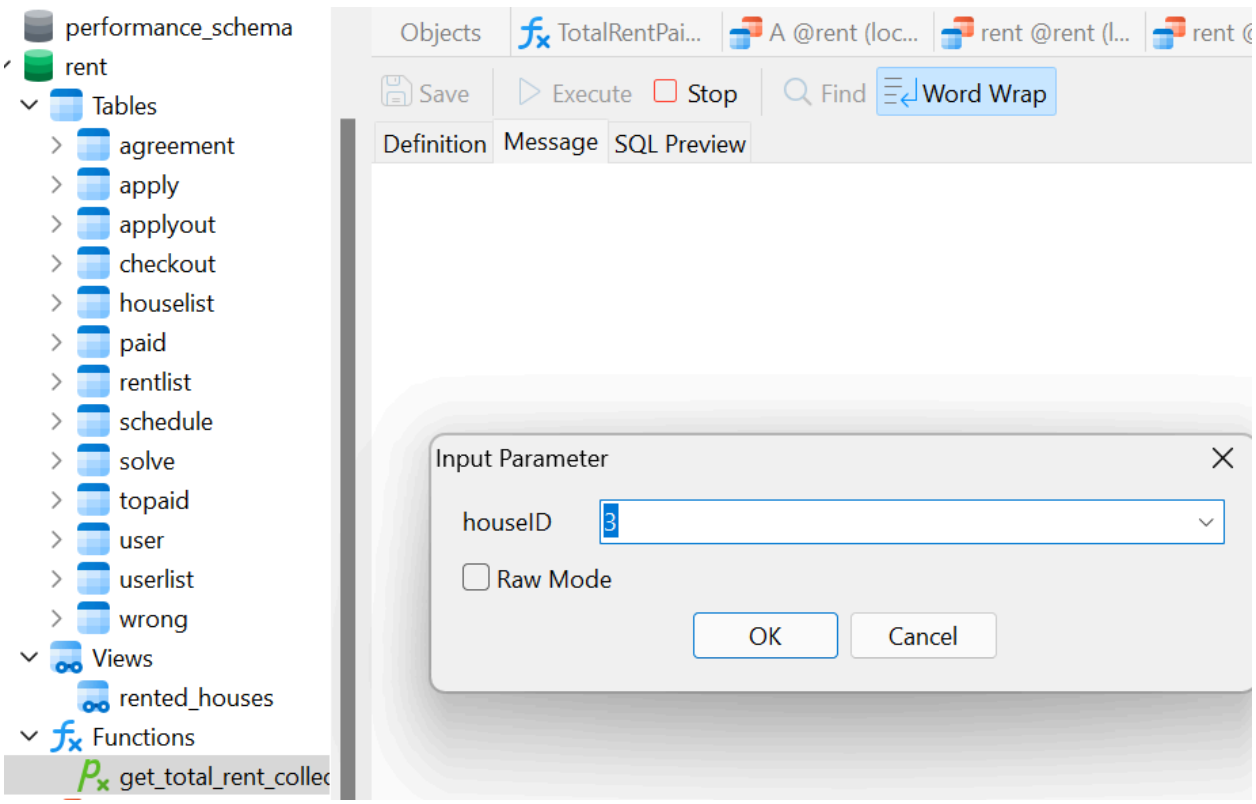


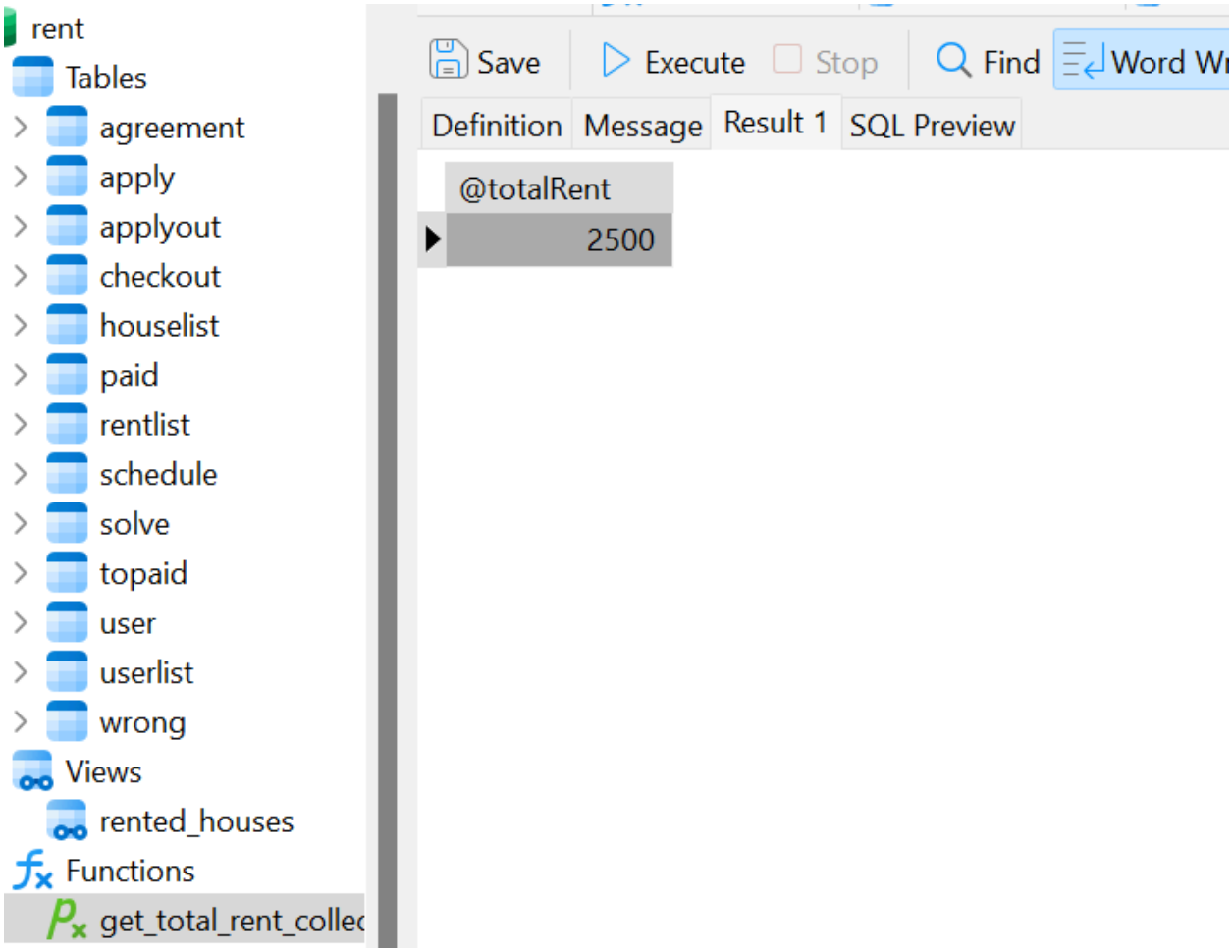
view:



houseid	address	area	price	tenant_name
1	123 Main Street, Apt 301	20	1000	John Doe
3	456 Oak Street, Apt 201	45	2500	Jane Smith
5	789 Pine Avenue, Apt 201	30	1200	Michael Brown
7	202 Elm Street, Apt 102	25	1400	David Wilson
4	101 Maple Street, Apt 501	40	2300	Laura Miller

function:





Step11:
Some updates on the table naming, but the framework is not changed.
Step12:
Some updates on the table naming, but the framework is not changed.
Step13:We created new DQLs to replace origins.
Query1:

```sql
-- Query 1: Join query
-- 1. Join query: List all rented houses with tenant details
SELECT
    h.houseid AS House_ID,
    h.address AS Address,
    h.price AS Rent,
    u.name AS Tenant_Name,
    u.phone AS Tenant_Contact
FROM houselist h
JOIN rentlist r ON h.houseid = r.house_id
JOIN userlist u ON r.userlist_id = u.id
WHERE h.status = 'Rented';



-- Query 2: Aggregate query with GROUP BY and HAVING
-- Find the total rent paid by each user, grouped by user, where t
SELECT
    p.userlist_id,
    u.name AS user_name,
    SUM(p.price) AS total_rent
FROM
    paid p
JOIN
    userlist u
```

Message | Summary | Result 1 | Result 2 | Result 3 | Result 4 | Result 5 | Profile | Status

House_ID	Address	Rent	Tenant_Name	Tenant_Contact
1	123 Main Street, Apt 301	1000	John Doe	12165434090
3	456 Oak Street, Apt 201	2500	Jane Smith	12165434091
5	789 Pine Avenue, Apt 201	1200	Michael Brown	12165434093
7	202 Elm Street, Apt 102	1400	David Wilson	12165434095
4	101 Maple Street, Apt 501	2300	Laura Miller	12165434096

Query2:

Google Cloud db-gro | rent | Run Selected | Stop | Explain Selected

```sql
-- Query 2: Aggregate query with GROUP BY and HAVING
-- Find the total rent paid by each user, grouped by user, where the total exceeds $1000, sorted by total rent in descending order.
SELECT
    p.userlist_id,
    u.name AS user_name,
    SUM(p.price) AS total_rent
FROM
    paid p
JOIN
    userlist u
ON
    p.userlist_id = u.id
GROUP BY
    p.userlist_id, u.name
HAVING
    total_rent > 1000
ORDER BY
    total_rent DESC;

-- Query 3: Subquery
-- Find all houses that have been applied for but not yet added to the `applyout` table.
SELECT
    houseid, address
FROM
    houselist
```

Message | Summary | Result 1 | Result 2 | Result 3 | Result 4 | Result 5 | Profile | Status

userlist_id	user_name	total_rent
3	Jane Smith	2500
5	Michael Brown	2400
2	John Doe	2300
7	David Wilson	1400
4	Emily Johnson	1100

Query3:

```
    total_rent DESC;

-- Query 3: Subquery
-- Find all houses that have been applied for but not yet added to the `applyout` table.
SELECT
    houseid, address
FROM
    houselist
WHERE
    houseid NOT IN (
        SELECT house_id FROM applyout
    );
```

Message | Summary | Result 1 | Result 2 | Result 3 | Result 4 | Result 5 | Profile | Status

houseid	address
7	202 Elm Street, Apt 102

Query4:

```
-- Query 4: Join with multiple tables
-- Find all tenants and the corresponding contracts with their rental house details.
SELECT
    a.tenant AS tenant_name,
    a.address AS agreement_address,
    h.area AS house_area,
    h.price AS house_price,
    a.fromdate,
    a.todate
FROM
    agreement a
JOIN
    houselist h
ON
    a.house_id = h.houseid;

-- Query 5: Aggregate query with GROUP BY, HAVING, and inline subquery
-- Find the average rental price for each user and filter users who rented houses larger than the average house area.
SELECT
    p.userlist_id,
    u.name AS user_name,
    AVG(p.price) AS avg_rent
FROM
    paid p
```

Message | Summary | Result 1 | Result 2 | Result 3 | Result 4 | Result 5 | Profile | Status

tenant_name	agreement_address	house_area	house_price	fromdate	todate
John Doe	123 Main Street, Apt 301	20	1000	2024-01-04	2025-01-01
Emily Johnson	456 Oak Street, Apt 402	22	1100	2024-02-01	2025-02-01
Jane Smith	456 Oak Street, Apt 201	45	2500	2024-01-04	2024-10-03
Laura Miller	101 Maple Street, Apt 501	40	2300	2024-06-01	2025-06-01
Michael Brown	789 Pine Avenue, Apt 201	30	1200	2024-03-01	2025-03-01
Sarah Davis	101 Maple Street, Apt 501	35	1300	2024-04-01	2025-04-01
David Wilson	202 Elm Street, Apt 102	25	1400	2024-05-01	2025-05-01

Query5:

```sql
    a.house_id = h.houseid;

-- Query 5: Aggregate query with GROUP BY, HAVING, and inline subquery
-- Find the average rental price for each user and filter users who rented houses larger than the average house area.
SELECT
    p.userlist_id,
    u.name AS user_name,
    AVG(p.price) AS avg_rent
FROM
    paid p
JOIN
    userlist u
ON
    p.userlist_id = u.id
WHERE
    p.house_id IN (
        SELECT houseid FROM houselist WHERE area > (SELECT AVG(area) FROM houselist)
    )
GROUP BY
    p.userlist_id, u.name
HAVING
    avg_rent > 1200
ORDER BY
    avg_rent DESC;
```

Message | Summary | Result 1 | Result 2 | Result 3 | Result 4 | Result 5 | Profile | Status

userlist_id	user_name	avg_rent
3	Jane Smith	2500

Attention: Below is for GCP, Above is for Local

```
#jdbc.driver=com.mysql.cj.jdbc.Driver
#jdbc.url=jdbc:mysql://localhost:3306/rent?characterEncoding=utf-8&serverTimezone=America/Los_Angeles
#jdbc.username=root
#jdbc.password=20120337Aa


jdbc.driver=com.mysql.cj.jdbc.Driver
jdbc.url=jdbc:mysql://34.168.219.226:3306/rent?characterEncoding=utf-8&serverTimezone=GMT%2B8
jdbc.username=root
jdbc.password=stone
```

4. Application Source Code: Include frontend and backend code with setup instructions. (GitHub Links)
https://github.com/stone-coding/CS5200Group5

5. Conclusion: ○ Reflect on challenges, learnings, and improvements.

challenges:

- Database Design Complexity: Ensuring proper normalization while accommodating all functional requirements was a significant challenge. Balancing between avoiding redundancy and maintaining performance was a key consideration.
- Integration Issues: Configuring the backend to work seamlessly with the MySQL database on GCP required troubleshooting connectivity issues and optimizing queries for remote access.
- Session Management: Handling session conflicts between admin and tenant users on the same browser exposed flaws in our session management logic, requiring additional debugging and testing.
- Technology Learning Curve: For some team members, working with SpringMVC and MyBatis was a new experience, which initially slowed down the development process.

Learnings

Despite the challenges, the project provided invaluable learning opportunities:

- Database Optimization: We gained hands-on experience in designing scalable databases, indexing for performance, and using SQL scripts for efficient data population.
- Framework Proficiency: Working with Spring, SpringMVC, and MyBatis deepened our understanding of backend frameworks and best practices for building maintainable codebases.
- Cloud Integration: Setting up and connecting to a MySQL instance on GCP improved our skills in cloud database management and taught us the importance of secure configurations.
- Collaborative Development: The project emphasized the importance of clear communication, version control, and task delegation in team environments.

Improvements

Looking forward, we identified areas for improvement that could elevate the quality of similar projects in the future:

- Error Handling: Implementing more robust error-handling mechanisms, especially for session conflicts and database connectivity issues, would improve the reliability of the application.
- Testing Strategy: Adopting a more structured testing approach, such as automated unit and integration tests, could ensure higher code quality and catch issues earlier in development.
- Frontend Optimization: The user experience on the frontend can be enhanced by introducing responsive design techniques and improving the interface for better usability.
- Scalability Considerations: Refactoring parts of the application to support horizontal scaling and larger datasets would prepare the system for real-world deployment scenarios.