Introduction

Renting a property can be a very good option for people but only if they get a good landlord. According to a national survey of housing tenants, renters in Australia feel that they lack the power to demand standard property maintenance.[1] Tenants feel that even for basic property management or damage they have to make multiple calls to Landlords. Landlords act as middlemen between renters and tradesman, they manage when the renters would be home so that the traders can come and complete their repair jobs. This system makes the entire process full for hassle which results in delays, improper maintenance, and bad repair jobs. Looking at this from a landlord's point of view, one person can have multiple properties to look after, and it can get difficult to manage all property problems together. The system that I have developed here allows the tenants to file a complaint which the tradesmen can see and act upon. The system will show to the tradesmen - the complaint description and property ID. Traders will be able to edit or delete a complaint. The landlords will be able to see a combined result of property with their respective complaints filed. The current development of the idea is on a very small scale, but I believe this idea can be converted into a full-scale project and can be implemented in real life.

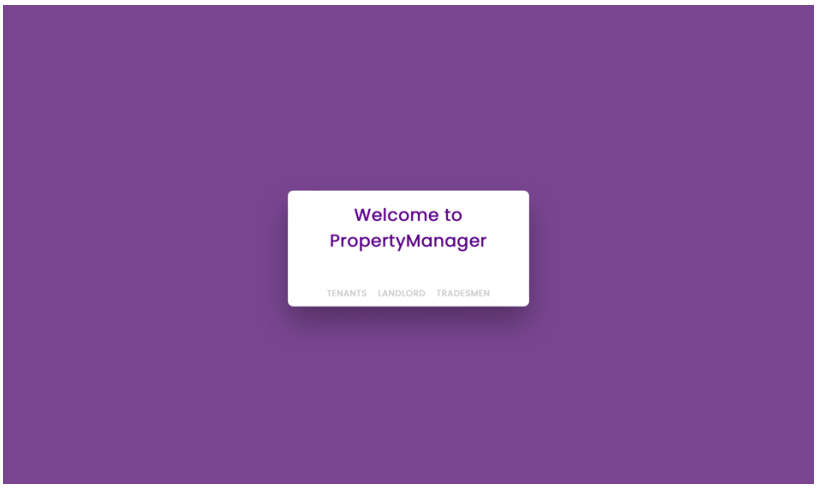


Fig 1: Welcome page of PropertyManager

Requirements:

Programming language:

Front end: HTML, CSS, Javascript, Bootstrap

Back end: MySQL, Python, Flask

Software: VS Code, XAMPP (For localhost and SQL database)

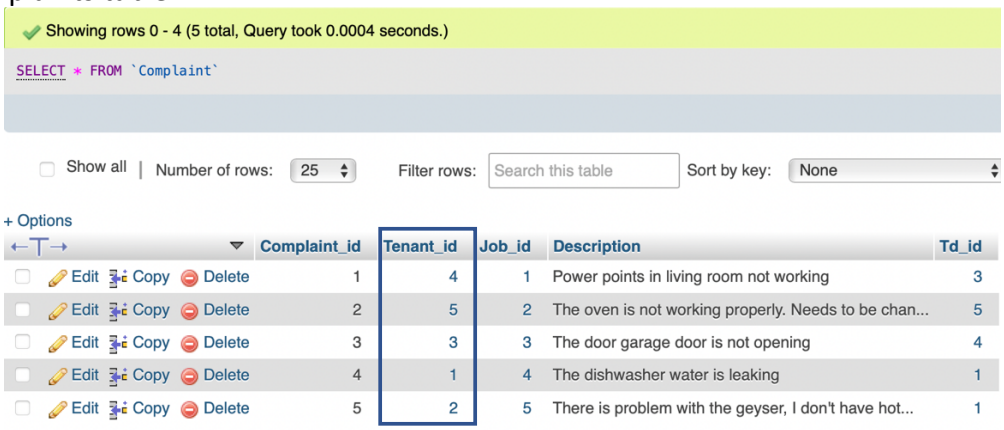
[1] https://www.abc.net.au/news/2017-02-21/australians-share-their-rental-horror-stories/8277394

Query Demonstration

1. Join query:

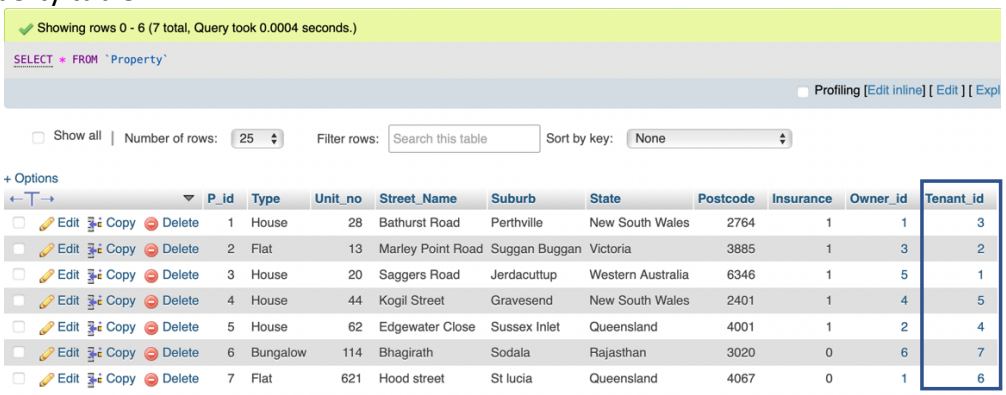
I have implemented join query between 'Complaints' table and 'Property' table to show results of all complaints filled corresponding to the property.
Both the tables had the attribute T_id (Tenant ID). I used this attribute to implement inner join.
This join query is useful for the landlord to keep an eye on all the complaints with the corresponding property details.

Complaints table:



	Complaint_id	Tenant_id	Job_id	Description	Td_id
Edit Copy Delete	1	4	1	Power points in living room not working	3
Edit Copy Delete	2	5	2	The oven is not working properly. Needs to be chan...	5
Edit Copy Delete	3	3	3	The door garage door is not opening	4
Edit Copy Delete	4	1	4	The dishwasher water is leaking	1
Edit Copy Delete	5	2	5	There is problem with the geyser, I don't have hot...	1

Property table:



	P_id	Type	Unit_no	Street_Name	Suburb	State	Postcode	Insurance	Owner_id	Tenant_id
Edit Copy Delete	1	House	28	Bathurst Road	Perthville	New South Wales	2764	1	1	3
Edit Copy Delete	2	Flat	13	Marley Point Road	Suggan Buggan	Victoria	3885	1	3	2
Edit Copy Delete	3	House	20	Saggers Road	Jerdacuttup	Western Australia	6346	1	5	1
Edit Copy Delete	4	House	44	Kogil Street	Gravesend	New South Wales	2401	1	4	5
Edit Copy Delete	5	House	62	Edgewater Close	Sussex Inlet	Queensland	4001	1	2	4
Edit Copy Delete	6	Bungalow	114	Bhagirath	Sodala	Rajasthan	3020	0	6	7
Edit Copy Delete	7	Flat	621	Hood street	St lucia	Queensland	4067	0	1	6

SQL CODE:

```
SELECT Property.Unit_no, Property.Street_Name, Property.Suburb, Property.State, Property.Postcode, Property.Insurance, Property.Owner_id, Property.Tenant_id, Complaint.Complaint_id, Complaint.Description, Complaint.Td_id FROM Property INNER JOIN Complaint ON Property.Tenant_id=Complaint.Tenant_id ORDER BY Complaint.Complaint_id ASC;
```

Pictorial UI:

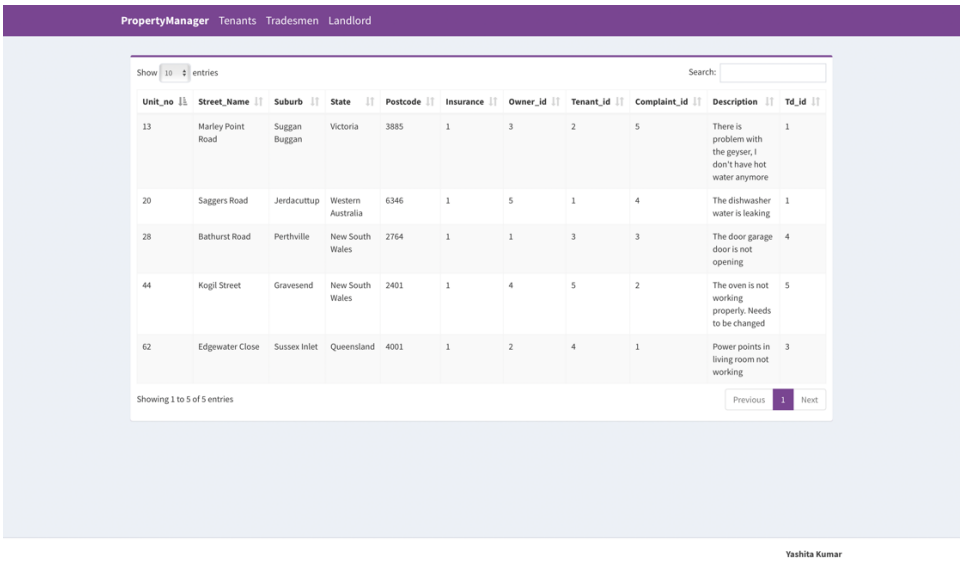
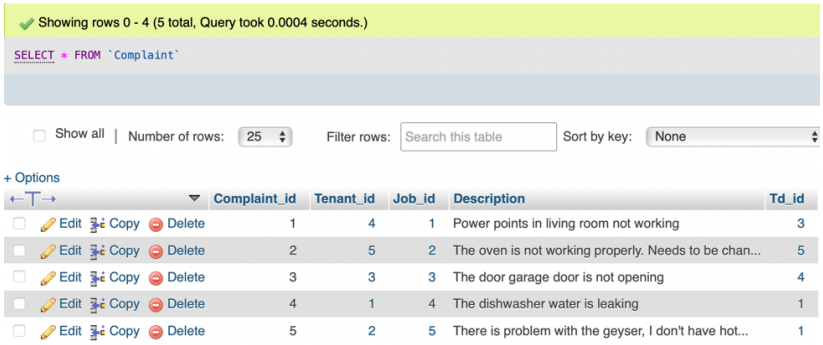


Fig 2: Results from join query visible on Landlord's page

2. Update operation:

I implemented the update query on the tradesmen page the action button  allows the tradesmen to edit anything in the complaint. Changes when submitted will be updated in the database and on the tradesmen's main page.
The update operation can be used by the tradesmen to update the description of the complaint to show the status, which is then visible to landlords and tenants.

Complaint table:



SQL CODE:

UPDATE Complaint set Complaint_id = %s, Tenant_id = %s, Job_id = %s, Description = %s, Td_id = %s where Complaint_id = %s",
(data['Complaint_id'],data['Tenant_id'],data['Job_id'],data['Description'],data['Td_id'],id,)

Pictorial UI:

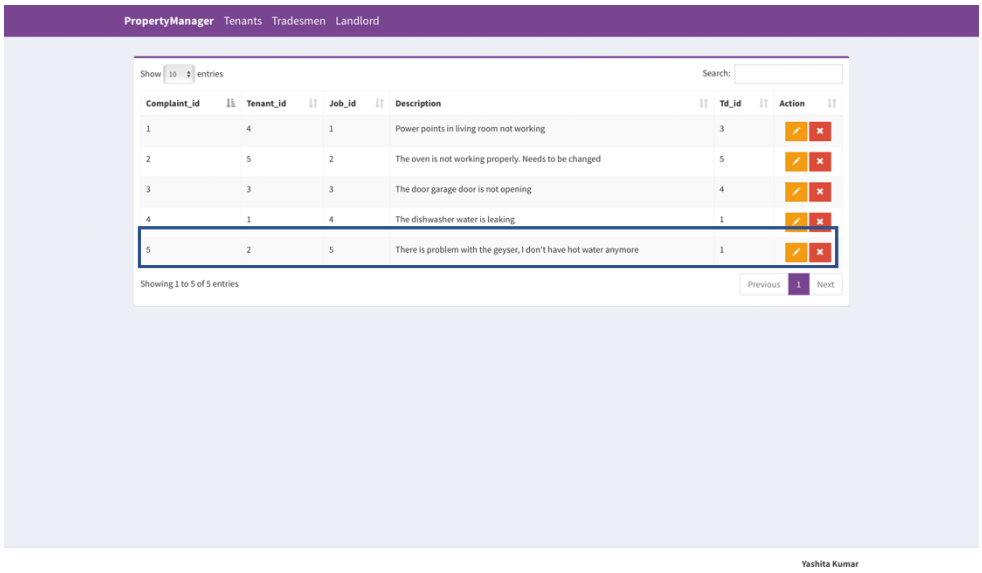


Fig 3: Tradesmen's home page shows the result of Complaint table

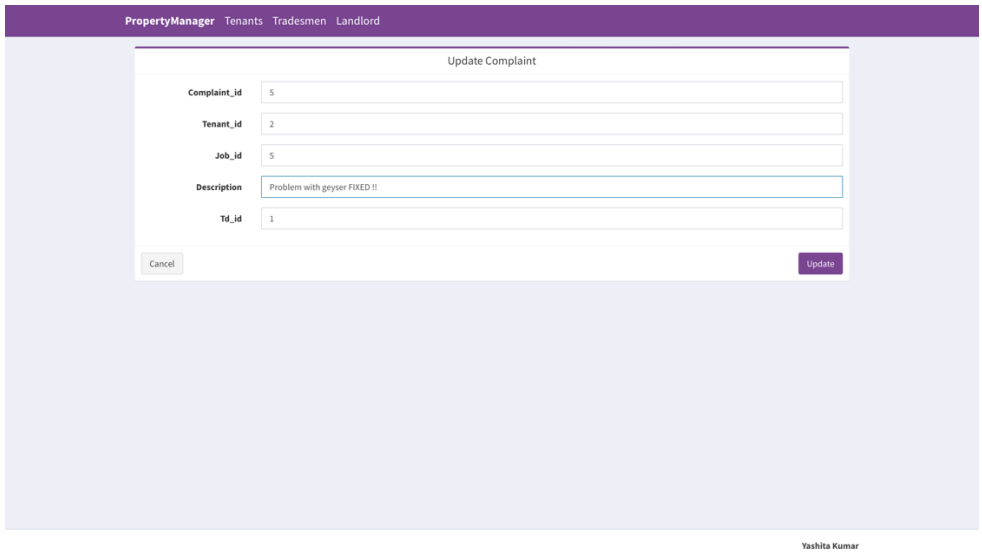


Fig 4: Update complaint page on Tradesmen portal

In the figure I changed the description of Complaint_id = 5 from "There is problem with geyser, I don't get hot water anymore" to "Problem with geyser FIXED !!"

The GUI send a pop-up message to show that the complaint has been updated.

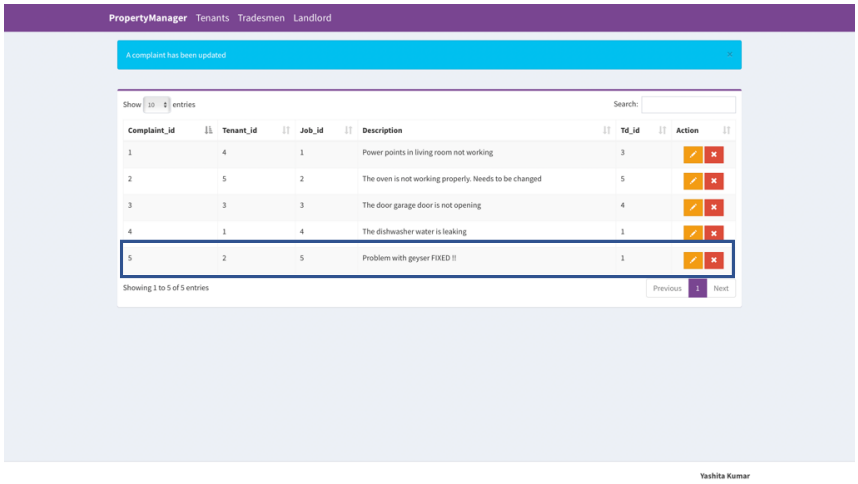


Fig 5: UI's response to update query

When the user clicks on update, they get a message informing them that the query has been performed. The update on the complaint description is visible in the tradesmen and landlord portals.

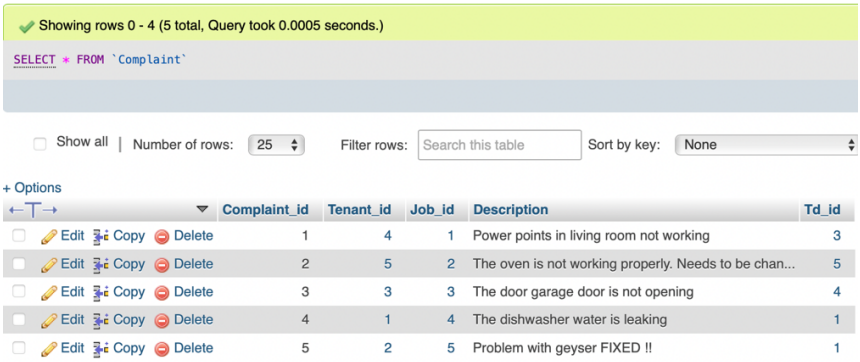


Fig 6: 'Complaint' table after update operation

3. Delete operation with cascade:

I implemented the delete query on the tradesmen page where the action button allows the tradesmen to delete any complaint. When deleted the complaint gets deleted from the 'Complaints' Table and therefore will no longer be visible on the landlord's page.
Tradesmen can use this functionality to delete a complaint which has been fixed and completed or to delete a complaint which the tenant might have filed by mistake.
The cascade constraints have been set in PhpMyAdmin and therefore when a 'complaint' is deleted the corresponding 'Job' details are also deleted.

Complaint table:

Showing rows 0 - 4 (5 total, Query took 0.0004 seconds.)

```
SELECT * FROM `Complaint`
```

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

	Complaint_id	Tenant_id	Job_id	Description	Td_id
Edit Copy Delete	1	4	1	Power points in living room not working	3
Edit Copy Delete	2	5	2	The oven is not working properly. Needs to be chan...	5
Edit Copy Delete	3	3	3	The door garage door is not opening	4
Edit Copy Delete	4	1	4	The dishwasher water is leaking	1
Edit Copy Delete	5	2	5	There is problem with the geyser, I don't have hot...	1

SQL CODE:

```
DELETE FROM Complaint where Complaint_id = 5;
```

Pictorial UI:

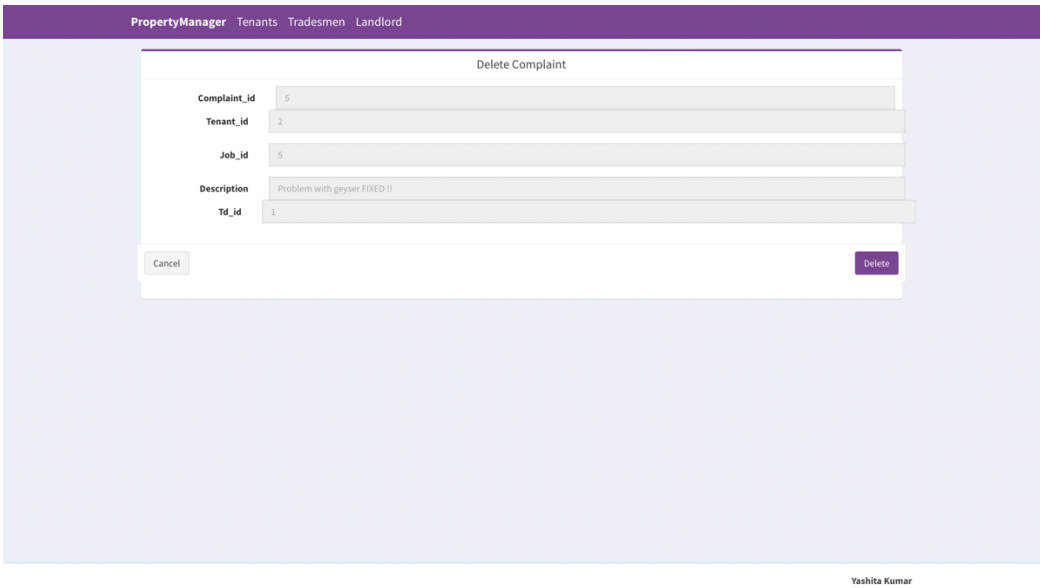


Fig 7: Delete complaint page on the tradesmen portal

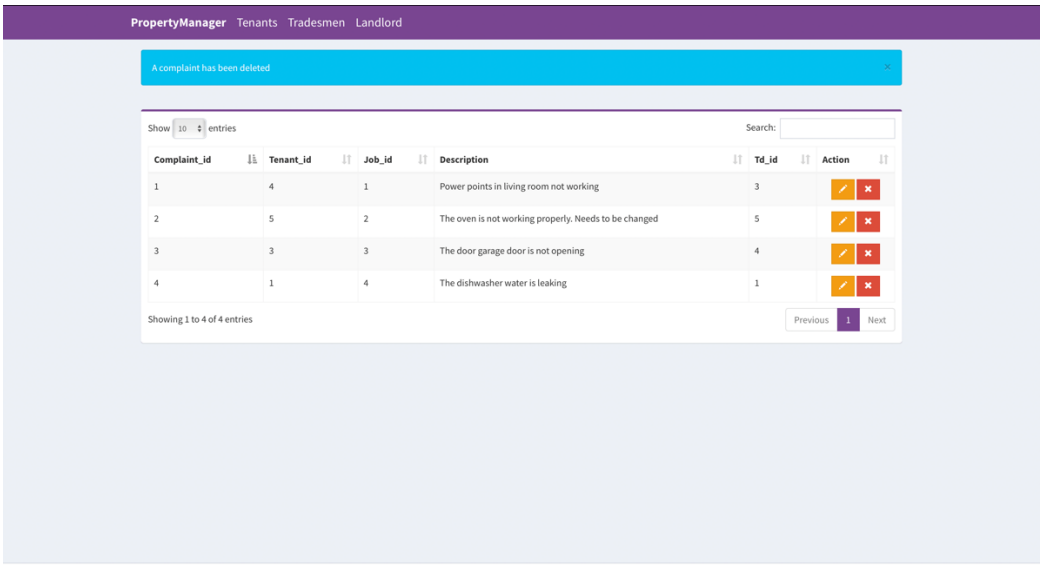


Fig 8: UI's response to delete operation

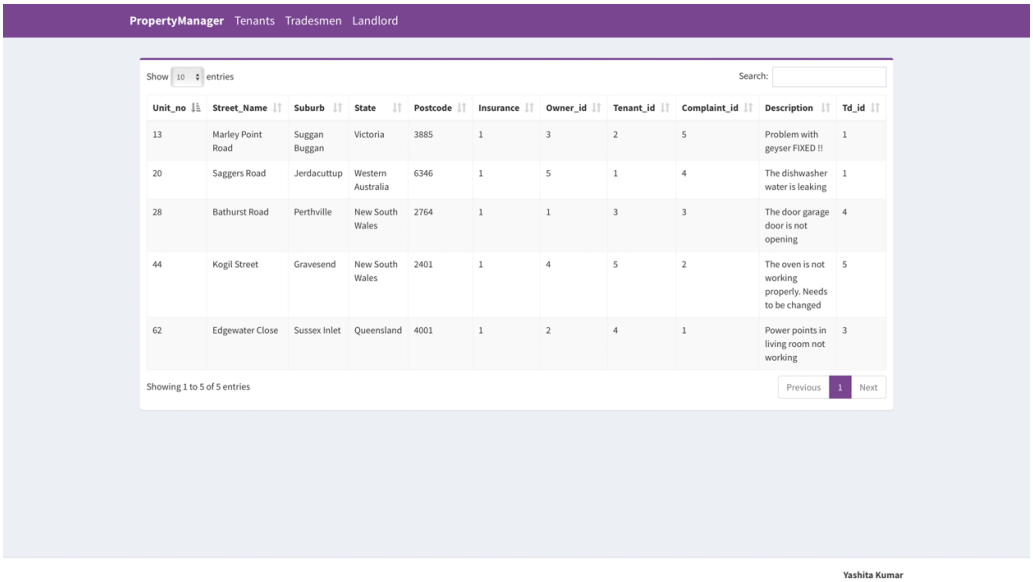


Fig 9: Landlord's page before the delete operation

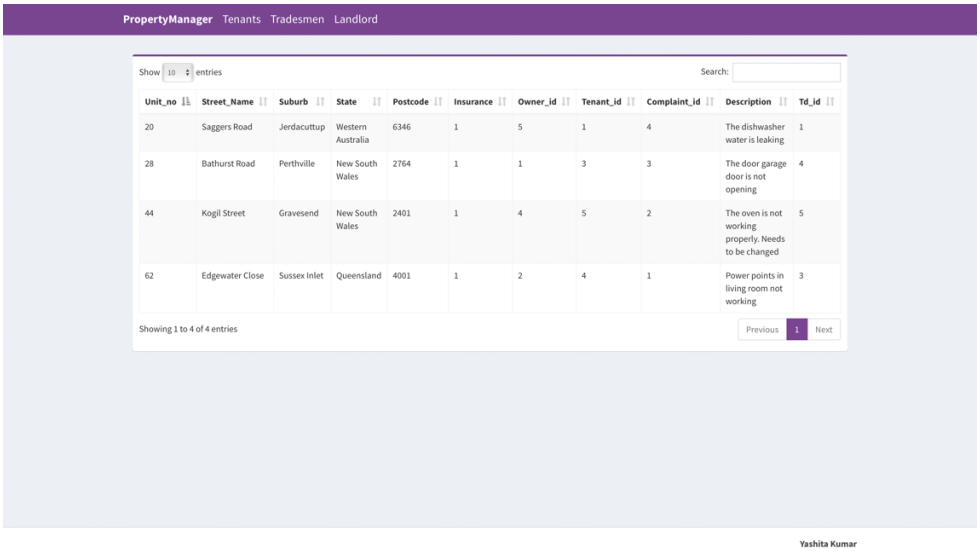
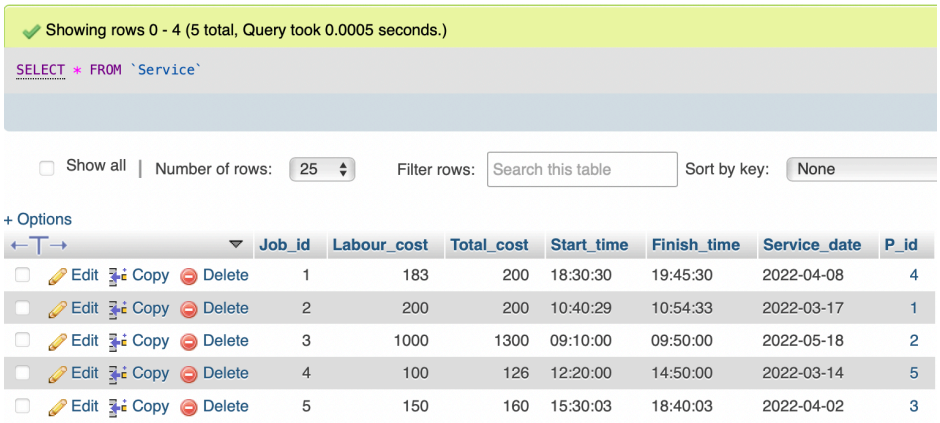


Fig 10: Landlord's page after the delete operation

4. Aggregation query (functions such as min, max, average or count)

I have implemented the aggregation query to find the average cost of service from the 'Service' table. This will be helpful for the tenants as well as landlords to have an idea about what the average cost of service would be.

Service table:



SQL CODE:

```
SELECT AVG(Total_cost) as average_cost FROM `Service`
```

Pictorial UI:

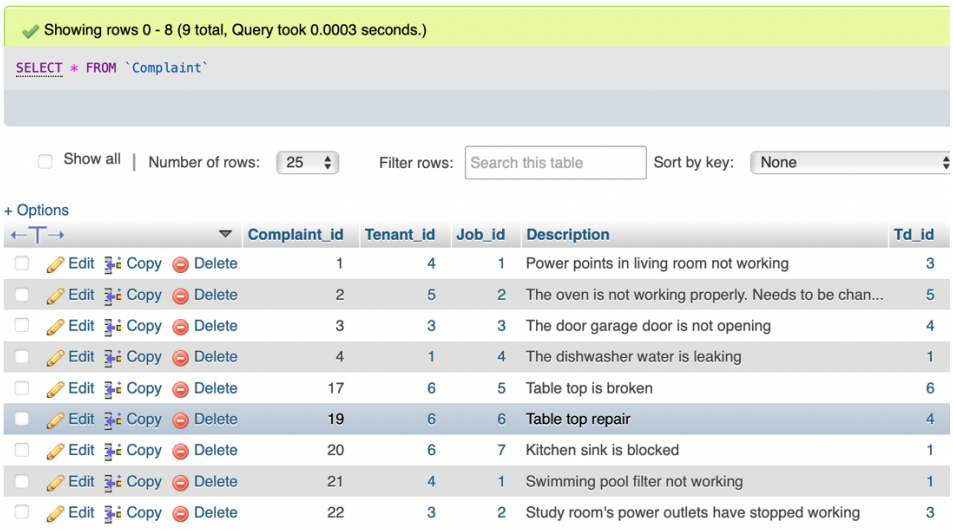
Fig 11: Result from the aggregation query

After performing the above query to find the average cost of service, it can be seen that the average_cost = $669.4286

5. Aggregation with group-by (aggregated value for each group)

The query I have implemented here would allow the Landlord to see the number of complaints filled by each tenant. I have also applied order by to sort the number of complaints from highest to lowest (descending).

Complaint table:



	Complaint_id	Tenant_id	Job_id	Description	Td_id
Edit Copy Delete	1	4	1	Power points in living room not working	3
Edit Copy Delete	2	5	2	The oven is not working properly. Needs to be chan...	5
Edit Copy Delete	3	3	3	The door garage door is not opening	4
Edit Copy Delete	4	1	4	The dishwasher water is leaking	1
Edit Copy Delete	17	6	5	Table top is broken	6
Edit Copy Delete	19	6	6	Table top repair	4
Edit Copy Delete	20	6	7	Kitchen sink is blocked	1
Edit Copy Delete	21	4	1	Swimming pool filter not working	1
Edit Copy Delete	22	3	2	Study room's power outlets have stopped working	3

SQL CODE:

```
SELECT Tenant_id, COUNT(Complaint_id) as Number_of_complaints FROM
`Complaint` GROUP BY Tenant_id ORDER BY COUNT(Complaint_id) DESC
```

Pictorial UI:

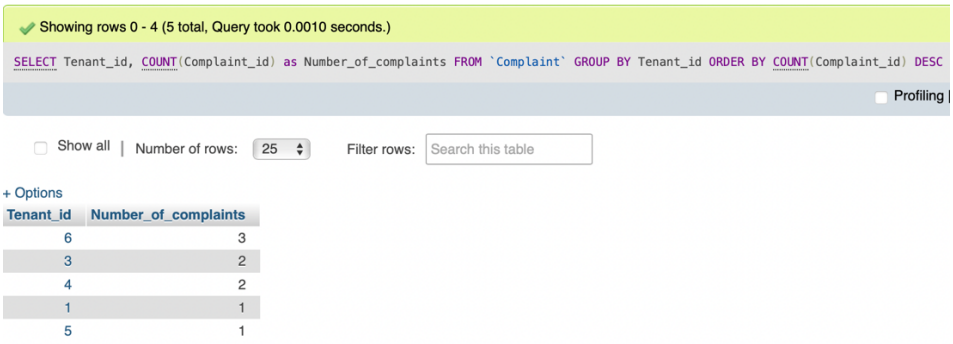


Fig 12: The result from the aggregation with group by query

After performing the above query, it can be seen that tenant_id = 6 has filed the maximum number of complaints which is equal to 3.

CONCLUSION

The project is still in its initial stages and has a lot of scope to improve, I plan to keep working on it and implement more functionalities for the system. I have a background in programming and have developed various websites therefore I would say that implementing the SQL queries was the easiest part for me, but I have never worked with python and flask to build projects so getting familiar with a new language was fun!
I would say I had spent the maximum amount of time coming up with an idea which would be useful in real life and then designing the database.
Time spent on parts of project:

1. Idea: Almost a month, because I had a lot of ideas, and it was difficult to decide which idea I should work on.
2. Database design: Almost 1.5 months because after deciding what I wanted to do I had to thinking about how to implement my idea using an actual system.
3. Implementation: Around 2 weeks because I'm familiar with building websites using SQL database. For the GUI, I used the skeleton structure provided by our tutor.[2]
4. Report: Personally, it was the least time-consuming task for me. It took me 2 days to complete it.

For me the most interesting part was to learn how to use a new language (flask) to create webpages. The practicals helped me a lot, but I also used various YouTube videos to build my project. There is nothing in particular that I didn't like about this project, it was a fun assignment that we had to work on the entire duration of our semester.

The best way to learn something new is practising it, just studying theory would not help you practice your skills for industry. If someone really wants to learn how to design a database, I would suggest them to come up with ideas and try to implement them and work on live projects.

[2] https://github.com/xurong-liang/crud_flask_mysql