Polyglot Persistence Project

SmartRent: Revolutionizing the Rental Market for Property Owners and Renters

Table of Contents

1 Introduction 1

2 Problem Statement 1

3 Application Use case 1

4 Solution Design 2

4.1 Project Assumptions 2

4.2 System Architecture 2

4.3 Justification for Database selection. 3

4.4 ER Diagram 5

4.5 Logical Database Design 5

4.6 The database design language 6

4.7 Data Dictionary 7

4.8 Advance SQL functionality- 8

4.9 NoSQL- MongoDB Structure 8

4.10 Python Libraries and Tools Used. 9

4.11 Ethics 9

4.12 Analytics 10

4.13 Limitations 13

5 Summary 13

6 References 14

1 Introduction

SmartRent offers a platform where property owners can list properties they want to rent out and renters can view the properties available in the area they are interested. This platform allows renters and owners to connect directly with each other without any middlemen. Moreover, the platform also offers a feedback mechanism for tenants giving them the opportunity to express their views on properties listed on the platform.

2 Problem Statement

We live in an era where the market for renting housing properties is dominated by agents and owners of properties. The problem with this is that customers have no say, whatsoever in how the market affects them. Customers are not able to talk about what works best for them because owners and agents are only concerned about how they make their money and not how customers are affected in terms of their needs and what their feedback is. Feedback is important because primarily, customers are the ones who are direct consumers of this venture and having their input will help know what their needs are as well as what has worked well for them.

3 Application Use case

By understanding the needs of the market, we have created an online application called SmartRent, which will allow renters to see all the properties available for rent in the market along with feedback from other renters. Additional to these there are many functions this project delivers. We have listed the features below.

- Search for properties based on needs with filtering.
- View detailed information about properties listed and their attributes.
- Schedule appointments directly through a platform
- Submit applications for properties renters are interested in.
- Closing deals online, giving opportunities for owners to view applications of all renters and finalizing it
- Provide feedback on properties. Customers can give ratings on different factors like location, pricing, amenities in a range of 1 to 5.

4 Solution Design

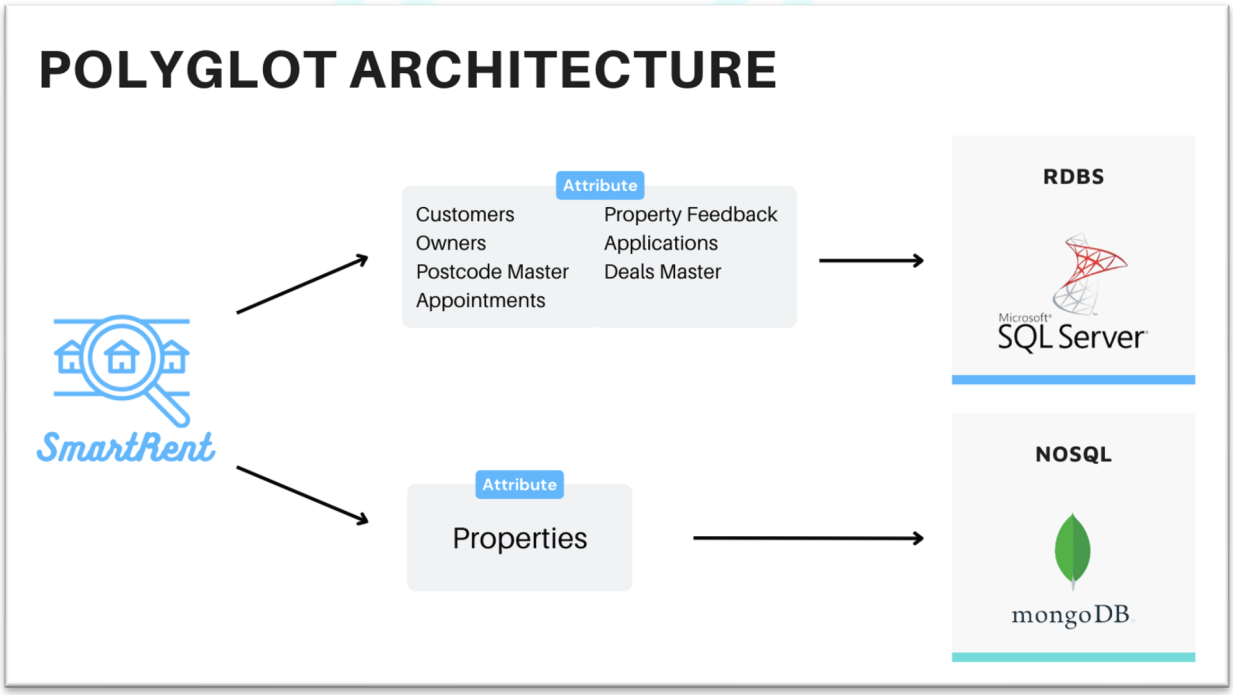
4.1 Project Assumptions

In developing this project, certain assumptions were made. These are,

- One customer can search many properties, make many appointments, and submit multiple applications.
- Owners can list multiple properties for renting, can receive many applications and close deals online.

4.2 System Architecture

Based on the scope of this project, there was a need for polyglot persistence because some data (I.e., customers owners, postcode master, appointments, property feedback, applications, and Deals master) attributes being worked with are strictly structured while other data attributes such as property amenities are not structured. For this application to handle structured and semi-structured data, two types of databases relational and non-relational will need to coexist with each other. This is depicted by the diagram above.



4.3 Justification for Database selection.

In addressing this issue, the use of relational database and non-relational database was incorporated. Relational database because data about applications, customers, property owners, address of properties, appointment booking, and feedback are structured information that are related to one another. According to KRÜGER (2004), relational databases make it possible to ensure character values, thus making it possible to establish relationships with other tables.

A non-relational database was also used because information about the properties is not structured as they contain different types of detail whose structure cannot be held by a relational database for the purpose of what we want to achieve- making enough information available to potential clients to help them decide in a data-driven way, in order to gain value for their money. NoSQL databases are shared, non-relational databases created for multithreaded data processing along a great amount of commodity servers. Using a document database allows more semi-structured information to be stored, allowing for flexibility in the capture of information that will help potential clients make very-informed decisions about the property they rent to have value for money.

Read-only SQL queries are supported through the available connectors for MongoDB. This makes it easy to work with since we also work with a relational database that supports SQL. Neo4J does not support SQL at all and would complicate matters for us. This is why it was not used.

Also, Redis was not used for this project because although its peripherals are extensive, scalability is more comfortable with MongoDB. Also, because of security, it was realized that Redis makes access possible to users with a simple password-based authentication which puts security at risk. MongoDB makes access specific so that users can assign role-based account control which allows for improved security.

The selected relational database was a Microsoft SQL server due to following reasons:

- Microsoft SQL (hereon referred to as MSSQL) allows for easy integration of data into applications.
- MS SQL server allows users to take advantage of a broad set of cognitive services to leverage artificial intelligence at any scale of data (*What Is Microsoft SQL Server and What Is It For?* n.d.).
- It lets you maintain the security, integrity, and consistency of data to ensure customer trust.
- It helps to avoid the need of duplicating data when working from different machines

- It does not require a special toolkit (I.e., cost-efficient), requires a simple installation and can detect automatic updates.
- It has effective data management and mining tools which when combined with disk partitioning, allow for the benefit of the best maintenance available.
- It also uses policy-based management to keep security policies compliant and updated. It thus allows only authorized personnel to access the database (*Why Choose Microsoft SQL Server for Your Website's Database Management?* 2018).

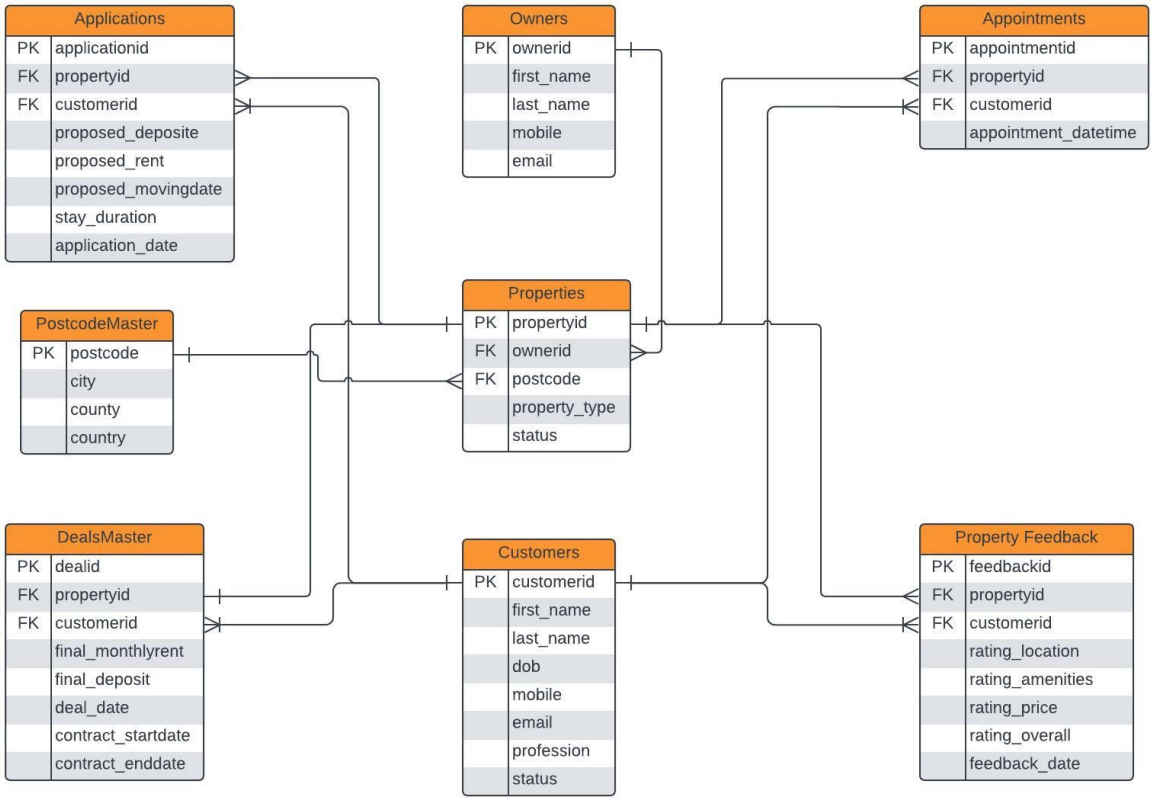
For NoSQL, MongoDB, which is a general-purpose database provides many benefits to our application development processes. It helps build applications that are more future proof with its scaling capabilities and flexible schema. MongoDB was chosen due to the following reasons:

- In relation to our project, MongoDB can accommodate large amounts of semi structured data such as property amenities.
- It is also built to scale up quickly and it supports all the key features of modern databases such as transactions.
- Its flexibility and power can create a single unified view in ways that other databases cannot. It has, as a result, succeeded in bringing such projects to life when approaches using other databases have failed (Why Use MongoDB and When to Use It? n.d.)

In summary, Relational database management system is structured around the concept of relationships, defines data (by forming relationships) via constraints, primary and foreign keys (e.g., a child table references to the master table via its ID). Finally, it offers extreme consistency, critical for some use cases such as daily banking. In contrast, document database system (hereon after referred to as document DB) focuses on data rather than relationships by organizing data into tuples (or rows), have properties without theoretical definitions, instead of rows. Document DB does not use DDL to create relationships through the definition of schemas. Relationships are represented through a nested loop and not foreign keys. Finally, Document Db offers consistency with a period of inconsistency (*Document Database {Definition, Features, Use Cases}*, 2021).

4.4 ER Diagram

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how "entities" such as people, objects or concepts relate to each other within a system. It is a graphical representation of a database that illustrates the connections between its constituent parts.



4.5 Logical Database Design

As part of the database architecture normalization process, we utilized the Third Normal Form(3NF) to create tables, which helped us to remove the redundant data and improve the DBMS's useability by improving its efficiency. We can say that a relation is in its Third Normal Form if it is in first and second normal form and in which no non-primary key attribute is transitively dependent on the primary key. In other words, A relation is said to be in 3NF if at least one of the following condition holds in every nontrivial function dependency X -> Y: -

a. X is a super key
b. Y is a prime attribute

Prime attribute means each element of Y is part of some candidate key.

Relation	Relationship Type
Customers -> Application	One-to-many
Customers -> Appointment	One-to-many
Customer -> DealsMaster	One-to-many
Customer -> Property Feedback	One-to-many
Property -> Application	One-to-many
Property -> Appointment	One-to-many
Property -> DealsMaster	One-to-one
Property -> Property Feedback	One-to-many
Property -> Owner	Many-to-one
Property -> PostcodeMaster	Many-to-one

4.6 The database design language

Bracketing notation of DBDL as follows:

Bracketing Notation	Primary Key	Foreign Key(s)
Customers (Customer ID, First name, Last name, Date of Birth, Mobile, Email, Profession Status)	Customer ID	
Owners (Owner ID, First name, Last name, Mobile, Email)	Owner ID	
Postcode Master (Postcode, City, County, Country)	Postcode	
Appointments (Appointment ID, Property ID, Customer ID, Appointment datetime)	Appointment ID	Property ID, Customer ID
Property Feedback (Feedback ID, Property ID, Customer ID, Rating location, Rating amenities, Rating price, Rating overall, Feedback date)	Feedback ID	Property ID, Customer ID
Applications (Application ID, Property ID, Customer ID, Proposed deposit, Proposed rent, Proposed moving-in-date, Stay duration, Application date)	Application ID	Property ID, Customer ID
Deals Master (Deals ID, Property ID, Customer ID, Final monthly rent, Final deposit, Deal date, Contract start date, Contract end date)	Deals ID	Property ID, Customer ID
Properties (Property ID, Owner ID, Postcode, Property type, Status)	Property ID	Owner ID, Postcode

4.7 Data Dictionary

Collection name	Attribute Name	Attribute Type	Format	PK or FK
Customers	customerid	int		PK
	first_name	varchar(25)		
	last_name	varchar(25)		
	dob	date	YYYY-MM-DD	
	mobile	varchar(15)		
	email	varchar(50)		
	profession	varchar(25)		
	status	int		
Owners	ownerid	int		PK
	firstname	varchar(25)		
	lastname	varchar(25)		
	mobile	varchar(15)		
	email	varchar(50)		
PostcodeMaster	postcode	varchar(10)		PK
	city	varchar(25)		
	county	varchar(25)		
	country	varchar(10)		
Appointments	appointmentid	int		PK
	propertyid	int		FK
	customerid	int		FK
	ppointment_datetim	datetime	YYYY-MM-DD hh:mm:ss	
PropertyFeedback	feedbackid	int		PK
	propertyid	int		FK
	customerid	int		FK
	rating_location	int		
	rating_amenities	int		
	rating_price	int		
	rating_overall	int		
	feedbackdate	date	YYYY-MM-DD	
Applications	applicationid	int		PK
	propertyid	int		FK
	customerid	int		FK
	proposed_deposit	int		
	proposed_rent	int		
	roposed_movingdat	date	YYYY-MM-DD	
	stayduration	int		
	applicationdate	date	YYYY-MM-DD	
DealsMaster	dealsid	int		PK
	propertyid	int		FK
	customerid	int		FK
	final_monthlyrent	int		
	final_deposit	int		
	dealdate	date	YYYY-MM-DD	
	contract_startdate	date	YYYY-MM-DD	
	contract_enddate	date	YYYY-MM-DD	
Properties	propertyid	int		PK
	ownerid	int		FK
	postcode	varchar(10)		FK
	property_type	varchar(25)		
	status	boolean	0(no) or 1(yes)	

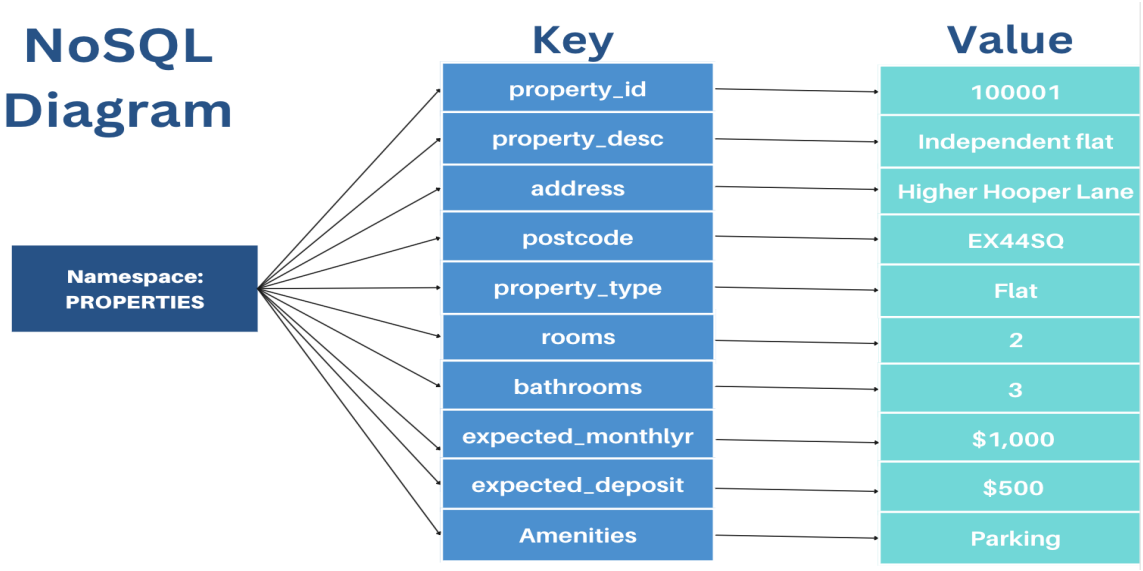
4.8 Advance SQL functionality-

In this application we have used Triggers and Views functionalities of MS SQL. Trigger is used for updating property status from available to rented once deal is finalized, this Trigger is made on the dealmaster table. We have also created a View, which combines data of renter's profile, their feedback, and application data. So that owners can view all the information together.

4.9 NoSQL- MongoDB Structure

In MongoDB, data is allocated to collections, which can be compared to tables in relational databases. Contrarily, collections do not necessitate a fixed schema, unlike tables. Each document within a collection may possess its unique structure and fields, resulting in enhanced flexibility regarding data storage. A document in MongoDB can incorporate diverse data types, such as strings, numbers, Booleans, and other documents. Furthermore, the inclusion of nested data structures is possible, enabling the storage of intricate data.

In the NoSQL diagram below, data storage within MongoDB is illustrated utilizing a key-value paradigm. The diagram features a collection entitled 'Properties', which serves as a container for individual documents, each representing a distinct property through a series of key-value pairs. An exemplary key within the diagram is 'Amenities', accompanied by its corresponding values. Owing to the schema-less characteristic of MongoDB, it offers the flexibility to accommodate a wide range of data types and structures with ease. This attribute is particularly advantageous for handling heterogeneous data sets and evolving data requirements.



Pseudo code:

1. Import necessary libraries (pymongo, pandas)
2. Connect to MongoDB using MongoClient, select the database, and create a "Properties" collection.
3. Insert property details into the "Properties" collection using 'insert_many' and 'insert_one' methods.
4. Loop through inserted data, print success or display error message.
5. Query the "Properties" collection with basic and advanced filters and print the results.
6. Update single and multiple documents in the "Properties" collection using 'update_one' and 'update_many' methods.
7. Delete single and multiple documents in the "Properties" collection using 'delete_many' method.
8. Retrieve property data from the "Properties" collection and create a pandas DataFrame.

4.10 Python Libraries and Tools Used.

PYMONGO, PYODBC, PANDAS, SEABORN, & MATPLOTLIB.PYPLOT are the libraries being used in the code to communicate with the respective databases, manipulating the data stored in the respective databases and performing the necessary analytics to understand the distribution of data, descriptive statistics of the measurable values, and relation between variables.

4.11 Ethics

Ethical considerations were made to ensure that clients are given the chance to give feedback regarding their experience because what already exists is an agent/owner-centric market that does not give clients the opportunity to give feedback for property. And to make sure this information is kept safe, access to the database is given to only authorized personnel. Also, clients' data will only be shared with connected third parties such as owners of properties, so they know the kind of people staying with them. Data collected from clients are used solely to

provide a service to clients. This is about ensuring transparency, accountability, and fairness in our work, which is in line with the UK data ethics framework (Data Ethics Framework, n.d.). Data was generated using mockaroo.com, chat-GPT and Microsoft Bing, in relation to transparency. Also, this work complied with guidelines and rules given in the module (accountability) and finally, in relation to fairness, the project aims to provide equal opportunity for all potential clients.

4.12 Analytics

We have done several essential analytics from data stored in both SQL and NoSQL database such as descriptive tables, data distribution chart, scatter plot with fitted line, and heat map of correlation among the variables-

Identifying patterns and trends: By analyzing the mean, median, and standard deviation of key factors, the application can identify patterns in the rental market. For example, the mean expected monthly rent and deposit can offer insights into the affordability of properties, allowing customers to make informed decisions based on their budget.

	rooms	bathrooms	expected_monthlyrent	expected_deposit
count	33.00	33.00	33.00	33.00
mean	2.15	1.55	2139.39	2306.06
std	1.12	0.75	887.39	945.03
min	1.00	1.00	600.00	800.00
25%	1.00	1.00	1600.00	1700.00
50%	2.00	1.00	2000.00	2000.00
75%	3.00	2.00	2600.00	2600.00
max	5.00	3.00	4000.00	5000.00

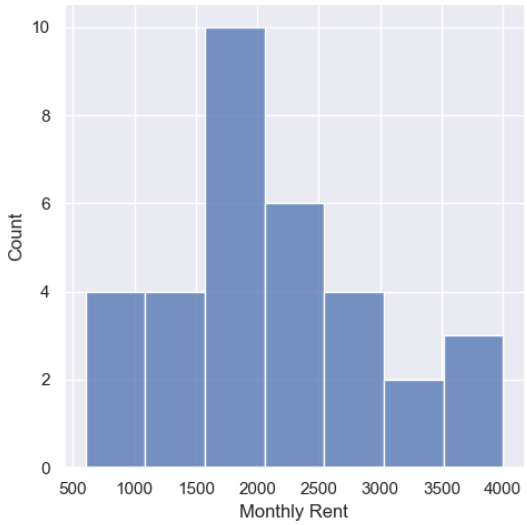


Figure 1.0

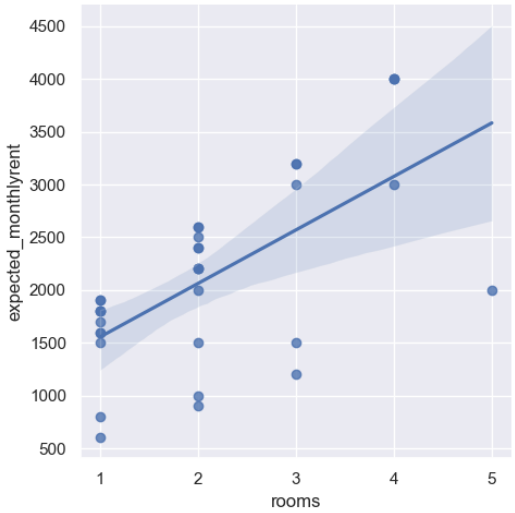
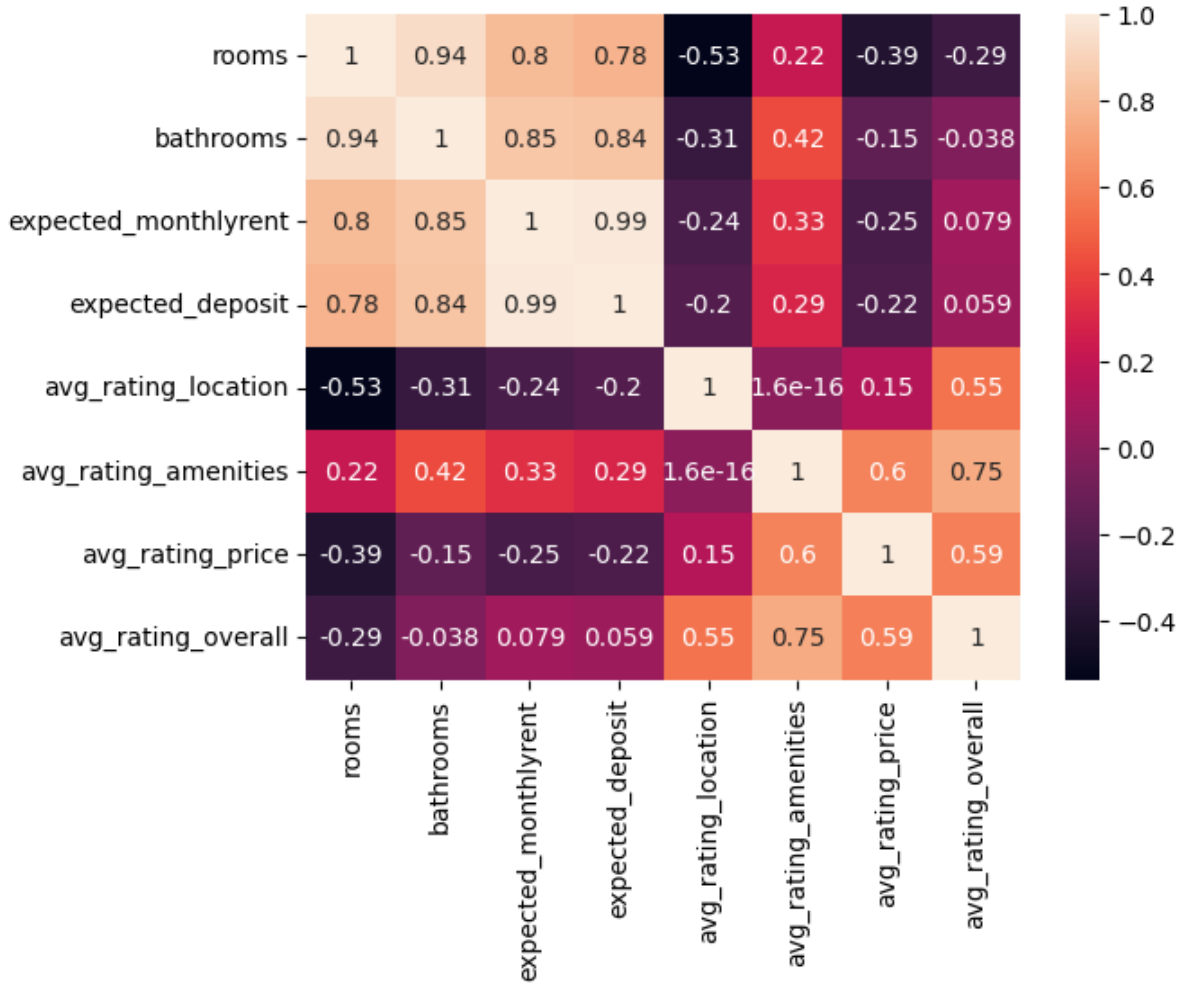


Figure 2.0

- **Understanding Data distribution-** Figure 1.0 (histogram) shows the distribution of monthly rent for the listed properties. This can help customers understand the rent range and most common rents more easily.
- **Understanding impact of number of rooms and rent-** Figure 2.0 (scatter plot), shows a relation between number of rooms and rent for listed properties. With the help of figure customers and owners both can see the relation between rooms and rent and confirm the obvious which is more rent equals more room.
- **Understanding relation between property variables and ratings-** The diagram uses data from the relational and non-relational databases. The heatmap represents the variables (rooms, bathrooms, expected monthly rent, expected deposit, ratings for location, amenities, price, and overall satisfaction) and how these variables are correlated. This correlation can help both owners and renters check the impact the property variables have on ratings and tweak them as per the need and can benefit both parties.



- **Evaluating customer satisfaction**: Average ratings for location, amenities, price, and overall satisfaction provide an understanding of how well properties meet customer needs. By focusing on these metrics, the application can identify areas for improvement and ensure that customer feedback is considered when selecting or recommending properties.

	rooms	Bath-rooms	expected_monthlyrent	expected_deposit	avg_rating_location	avg_rating_amenities	avg_rating_price	avg_rating_overall
count	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00
mean	2.50	1.63	1375.00	2000.00	2.50	2.75	3.13	2.63
std	1.41	0.92	790.57	1392.84	0.76	0.46	0.64	0.52
min	1.00	1.00	600.00	800.00	1.00	2.00	2.00	2.00
25%	1.75	1.00	875.00	1150.00	2.00	2.75	3.00	2.00
50%	2.00	1.00	1100.00	1500.00	3.00	3.00	3.00	3.00
75%	3.25	2.25	1625.00	2250.00	3.00	3.00	3.25	3.00
max	5.00	3.00	3000.00	5000.00	3.00	3.00	4.00	3.00

- **Prioritizing customer needs**: The application's objective is to emphasize customer needs and preferences. By analyzing the descriptive statistics and incorporating customer feedback into the database and application, the application can better tailor its recommendations to individual customers, ensuring that their requirements are met.

4.13 Limitations

This project was to demonstrate key learning concepts in the module and as a result, this did not focus on high level of complexity as data was created and not obtained. Also, analysis was not performed for every variable but only a few just to demonstrate learning. This does not allow this project to be used to its full potential. Finally, only a few CRUD operations were performed to demonstrate that this project works and demonstrates key learning. In summary, the project is not for full operational use but is meant only for the purposes of assessment.

5 Summary

This project aims to showcase renting properties using an online platform, allowing potential clients to make applications to rent properties based on their needs. It uses a relational database to store structured data as well as a document database to capture enough information on property to help potential clients make informed decisions. Data was generated using mockaroo.com, chat-GPT and Microsoft Bing. Ethical considerations were made to ensure that clients are given the chance to give feedback on property, and access to the database is only given to authorized personnel. Limitations of this project are linked to the fact that it is not for full operational use but only serves the purposes of demonstrating the key learning outcomes of this module.

6 References

Intelequia. (n.d.). *What is Microsoft SQL Server and what is it for?* Retrieved March 19, 2023, from https://intelequia.com/en/blog/post/what-is-microsoft-sql-server-and-what-is-it-for#:~:text=Microsoft%20SQL%20Server%20allows%20you

MongoDB. (n.d.). *Why Use MongoDB and When to Use It?* Retrieved from https://www.mongodb.com/why-use-mongodb#:~:text=Using%20MongoDB%20can%20provide%20many

PhoenixNAP. (2021, May 13). *Document Database {Definition, Features, Use Cases}*. Knowledge Base by PhoenixNAP. https://phoenixnap.com/kb/document-database

KRÜGER, B. (2004). WHY SHOULD YOU USE A RELATIONAL DATABASE INSTEAD OF A SPREADSHEET. Cybernetics and Systems, 35(7-8), 683-696. https://doi.org/10.1080/01969720490499461

Upadhyay, M. (2019b, July 31). *Third Normal Form (3NF) - GeeksforGeeks*. GeeksforGeeks. https://www.geeksforgeeks.org/third-normal-form-3nf/

Data Ethics Framework. (n.d.). GOV.UK. https://www.gov.uk/government/publications/data-ethics-framework