MAGS:
The Online Hotel Booking System

Group 7

BEMM459

University of Exeter

Contents

1 PROJECT OVERVIEW

We have decided to create an application for online hotel reservations called MAGS. We addressed the issues that one has while reserving a hotel room since our group is made up of all foreign students who have travelled from all over the globe. We chose this as our project since we all noticed that today's booking systems are riddled with convoluted data structures attempting to squeeze in their adverts and phoney reviews to promote major chain hotels rather than providing true consumer feedback. Because of the uncertainty, individuals have given up and booked their rooms via travel agents, who charge clients exorbitant service fees to complete basic jobs. Looking at the data structure, we saw that other online platforms had made an apparently simple process considerably more difficult. There are websites that excel at this role, like Agoda.com and Booking.com, which served as inspiration for us. Since the re-opening of air travel in most of the pandemic-affected nations, hotels have observed an increase in the number of individuals seeking to travel after experiencing claustrophobia from remaining indoors for so long.

We have designed an online Hotel Booking platform where users can book their reservations, generate invoices, and later provide feedback to the management. The platform relates to databases of various hotels, so that the user gets up-to-date information regarding room availability and other details. It's an application that can be used by both, the hotel management and also the by customers. The data that comes from this platform is sent to a back-end platform, and afterwards, hotels will have access to the booking and customers' information. It helps to improve hotels' efficiency as the room availability is updated automatically in the system. The aim of the project is to simplify the process of booking the hotel for our users and

provide them with unbiased reviews and prices. We have a hotel database which can help to search for the proper hotel and see the availability of the rooms.

2 OUR AIM

The task of creating hotel reservations easier for our customers is one of the primary focuses of our project. We maintain a hotel database that allows users to search for the appropriate hotels and view information about the rooms that are available. The following is a list of our businesses primary functions:

1. Take Reservations
2. Generate Invoices
3. Ability to apply discounts if any
4. Take Feedback

3 APPLICATION USE CASE

Main scenarios:

1. **Search**. Customers are able to do hotel searches based on their individual preferences.
2. **Availability**. Customers are able to view the number of available rooms at a certain hotel.
3. **Meal**. Customers have the option to include dinner reservations in their bookings.
4. **Guest Limit**.The number of visitors is equivalent to the kind of accommodation (for example 3 people go to triple room).
5. **Duration**. The programme will do the calculation for the duration for you automatically.
6. **Discounts**. Customers who make reservations at the hotel are eligible for discounts on such reservations.
7. **Invoice**. The invoices are created automatically, either with or without the deductions for discounts.
8. **Feedback**. Following the completion of a booking, the customers are sent feedback questionnaires to complete.
9. **Ranking**. Using customer rating we are able to rank hotels accordingly.

4 DATA REQUIREMENT RELATED TO THE USE-CASE

When a customer makes a reservation at a hotel, they are required to provide their personal information in order to do so. The customer then searches for the hotel and makes the reservation, which generates a unique booking ID and other booking information, including the foreign key of the hotel and customer table. In order to use our hotel management system, we require six tables in the relational database. The purpose of the room table is to ensure that the availability information shown by the system is always valid by providing generic room parameters as well as foreign keys to the hotel and booking tables. After a customer's stay, they are provided with a feedback form that includes references to the hotel and the booking so that they can rate their experience and provide feedback regarding the hotel's performance. The process is finished off with the creation of an invoice that includes references to the booking.

5 DATABASE SELECTION

In order to create polyglot persistence, we made use of both relational and non-relational databases. It is generally recommended to store data using a variety of data storage methods, and this should be done according to the function that the data serves. The ability to communicate in a number of languages offers the possibility to simplify operations and boost overall efficiency. We showed how the relational database management system (RDBMS) connects six different titles in our demonstration. We made use of NoSQL in order to improve the efficiency of our process for the non-relational database.

5.1 RELATIONAL DBMS

Relational databases are utilised in order to store information that is connected and to provide access to that information. They do this by presenting the data in a visual tabular format using the relational paradigm. Each row in the table represents a record, and each record has a key, which serves as a one-of-a-kind identifier. It is typically the case that each item in a table column has a value for each data attribute. This makes it straightforward to construct relationships between the many data points.

In order to interface with the database, we have decided to use SQLite, which is a relational data management system. SQLite offers a number of advantages, including the following:

1. Safety. It is safe and secure since it only allows a restricted number of people, access to the data.

2. Better performance. It is possible for the programme to load only the data that it needs rather than reading the entire file and maintaining a full parser in memory.

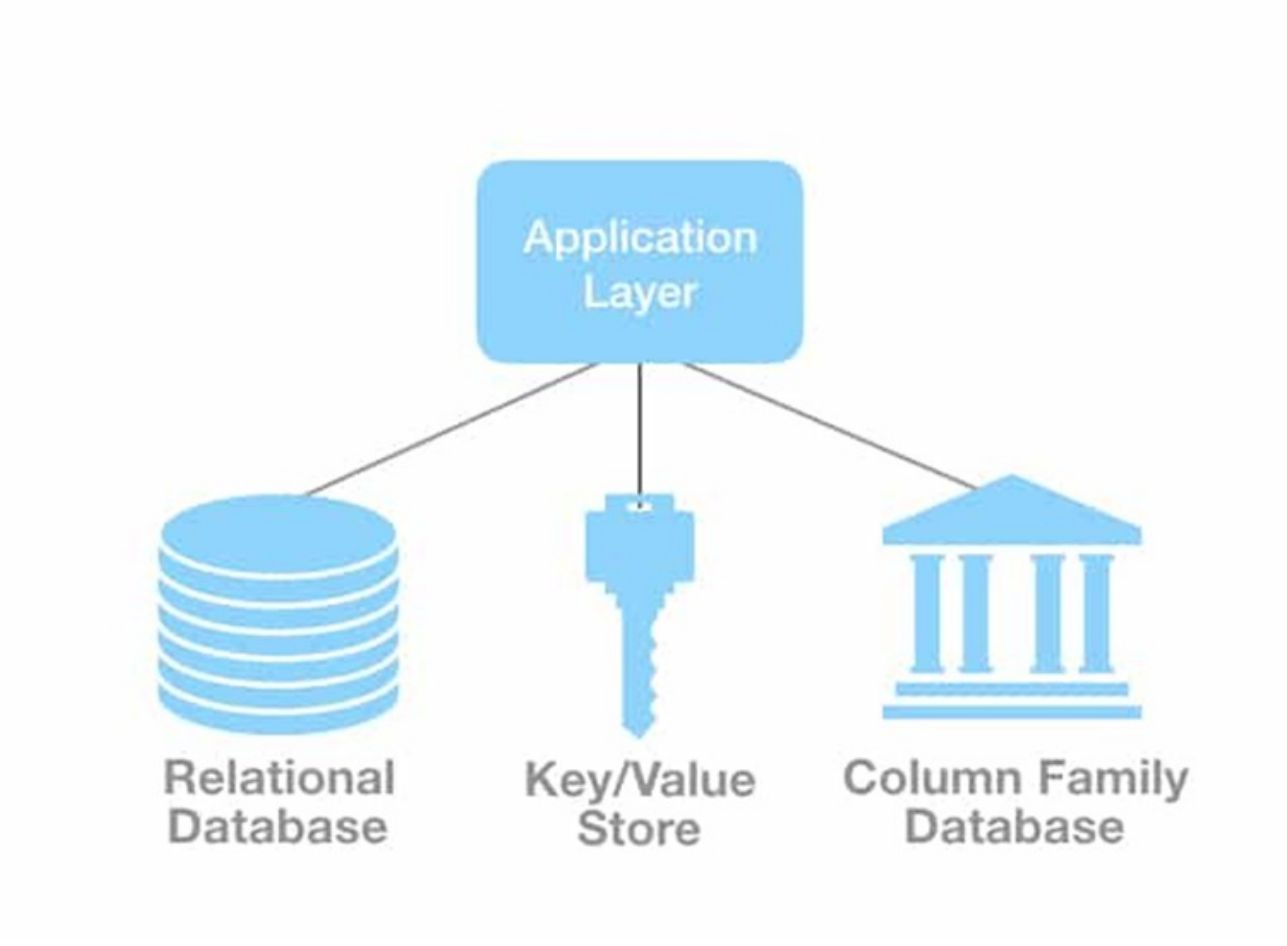

FIGURE 1: APPLICATION LAYERS

3. Speed. It offers a method that is both quick and effective for storing and manipulating the data. The DBMS components are integrated into the application, and the method in which they are called remains unchanged. As a result, access to them is a lot quicker than when various processes are communicating with one another.
4. Integration. Since it supports both Python and R without any issues.
5. The value of simplicity. Learning the language and downloading it are both very simple processes.
6. Reliability. Every part of the programme has been put through its paces throughout the testing process. Because of this, many people believe that SQLite is a trustworthy DBMS that has a low risk of unexpected behaviour.

5.2 NON RELATIONAL DBMS

Our application consists of 1 non relational table. NoSQL databases are perfect for applications that need to process a large amount of data that is organised in a variety of ways quickly and with as little lag as possible. As a direct consequence of this, non-relational storages have focused their efforts to Big Data. The key benefits of using NoSQL are as follows:

1. There is linear scalability. NoSQL is able to manage enormous amounts of data in a simple manner.
2. Adaptability. The schemas in NoSQL databases are flexible, which enables you to work with the databases more rapidly and to develop applications in stages. Compatibility exists with both semi-structured and unstructured data types.
3. Big Data Capability. The ability to modify different representations of different types of information.
4. Increased speed at performance by optimising for specific data model types.
5. A wide range of functionalities
6. Open source. It is possible to install NoSQL databases at a low cost since their licensing fees are low, and they may operate on hardware that is not very expensive.

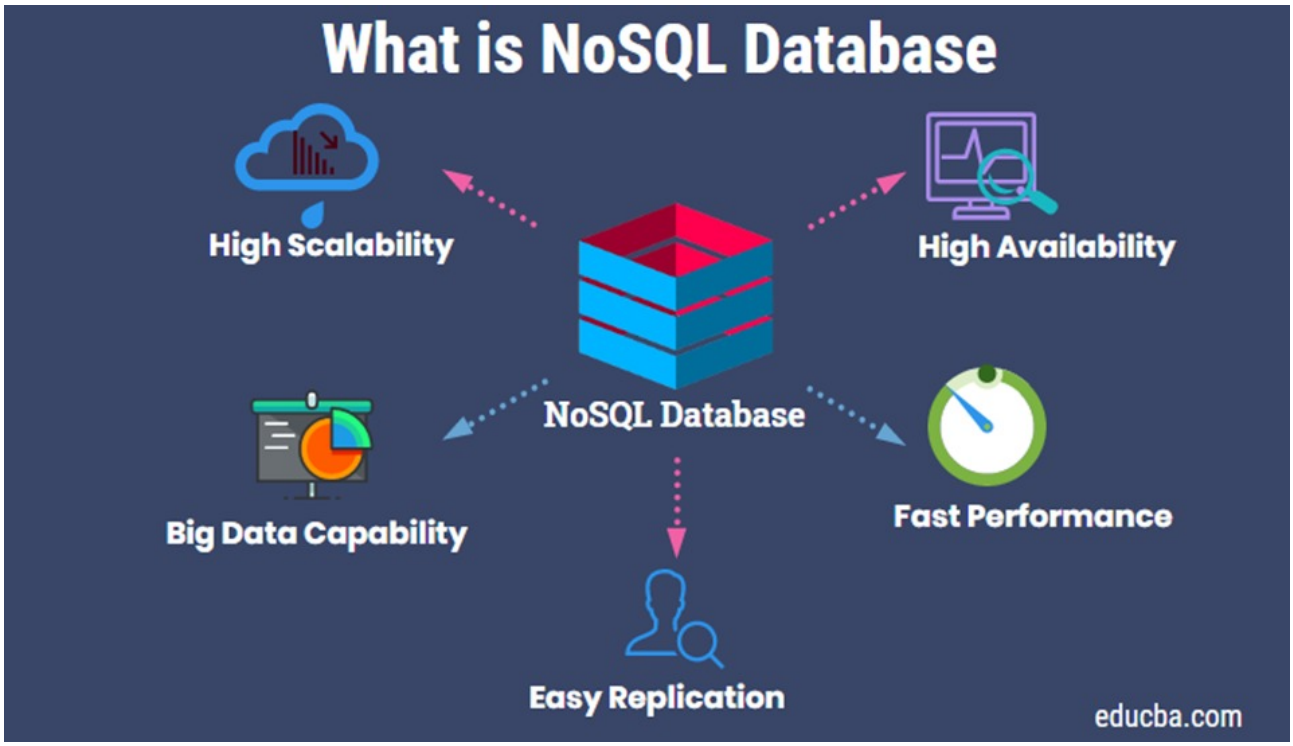

FIGURE 2: WHAT IS NOSQL?

6 MODELING

6.1 ER DIAGRAM

An Entity Relationship Diagram, often known as an ERD, is a graphical representation of a database that illustrates the connections between its constituent parts. Cardinalities are unique line ends that are used to indicate how two different database items interact with one another. These line endings are used to illustrate the relationships that exist between the entities that make up the database. An ER diagram represents a database table for each entity it depicts. There are six entities in our system, each with its own primary key and associated foreign key. The entities are related with each other as follows:

1. A single customer can make many bookings, and each booking will have its own unique ID.
2. A single booking may only be under one customer ID.
3. A booking can create an invoice and one invoice can have one only booking at a time.
4. Each Booking will get a feedback and each feedback can be only for one booking.
5. A booking may have many rooms while one rooms can only have one booking.
6. A room can only be there in one hotel while a hotel can have many rooms.
7. A hotel can have many feedbacks while one feedback can only be given for one hotel.
8. A hotel can have multiple bookings while one booking can only. have one hotel.

Based on the preceding relationships, we may characterise the nature of each relationship as follows:

6.2 LOGICAL DATABASE DESIGN

We used the Third Normal Form (3NF) to construct tables as part of the normalisation process for the database architecture. This helped us cut down on unnecessary data and made the DBMS easier to use.

It is a requirement of 3NF that each non-key column can only rely on the key column; in other words, there cannot be any transitive functional relationships between the columns. Database normalisation is often performed with the goals of removing duplicate data and ensuring that data is preserved in an appropriate manner.

To normalise a whole data model needs a few different processes to complete. There are many other kinds of normalisation forms, but the first three are the most crucial ones to understand. In order to accomplish the second normal form (2NF), it is necessary to first get the first

Relation	Relationship Type
Customers -> Bookings	one-to-many
Bookings -> Customers	one-to-one
Bookings -> Invoice	one-to-one
Invoice -> Bookings	one-to-one
Booking -> Feedback	one-to-one
Feedback-> Booking	one-to-one
Bookings -> Rooms	one-to-many
Room -> Booking	one-to-one
Room -> Hotel	one-to-one
Hotel -> Room	one-to-many
Hotel -> Feedback	one-to-many
Feedback-> Hotel	one-to-one
Hotel -> Booking	one-to-many
Booking -> Hotel	one-to-one

TABLE 1: RELATIONS

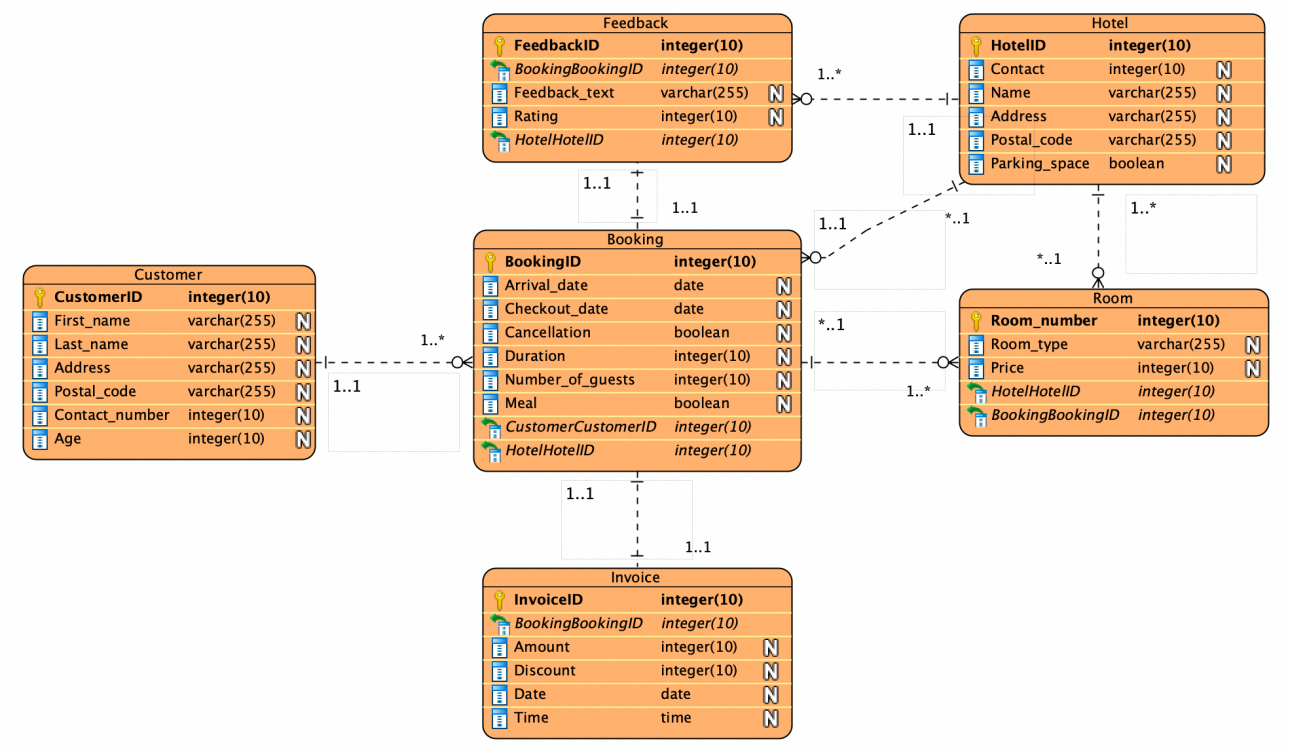

FIGURE 3: ENTITY RELATIONSHIP MODEL

normal form (1NF). Obtaining success with the second normal form (2NF) is a prerequisite for moving on to the third normal form (3NF).

Table/Collection name	Attribute name	Type	Format	PK or FK
CUSTOMER	Customer ID	Integer		PK
	First name	Varchar		
	Last name	varchar		
	Address	Varchar		
	Postal Code	Varchar		
	Contact number	Integer		
	Age	Integer		
BOOKING	BookingID	Integer		PK
	Arrival Date	Date	YYYY-MM-DD	
	Checkout Date	Date	YYYY-MM-DD	
	HotelID	Interger		FK
	Cancellation	Boolean	0(no), 1(yes)	
	Duration	Integer		
	Number of guests	Integer		
	Meal	Boolean	0(no), 1(yes)	
	Customer ID	Integer		FK
ROOM	room number	Integer		PK
	Room type	Varchar		
	Price	Varchar		
	HotelID	Integer		FK
	BookingID	Integer		FK
ROOM	HotelID	Integer		PK
	Contact	Integer		
	Name	Varchar		
	Address	Varchar		
	Postal Code	Varchar		
	Parking Space	Boolean	0 (no), 1(yes)	
FEEDBACK	FeedbackID	Integer		PK
	Feedback Text	Varchar		
	Rating	Integer		
	Booking ID	Integer		FK
	HotelID	Integer		FK
INVOICE	InvoiceID	Integer		PK
	Amount	Integer		
	Discount	Integer		
	Date	Date	YYYY-MM-DD	
	Time	Time	YYYY-MM-DD	
	Booking ID	Integer		FK

TABLE 2: DATA DICTIONARY

6.3 THE DATABASE DESIGN LANGUAGE

The DBDL of our tables is as follows:

Customers (CustomerId, First name, Last name,
Address, Postal code, Contact number, Age
PK: CustomerID

Booking (BookingID, Arrival date, Checkout date,
Room type, Cancellation, Duration, Number of guests, Meal)
PK: BookingID
FK: CustomerID, HotelID

Rooms (Room number, Room type, Price)
PK: Room number
FK: HotelID, BookingID

Hotel (HotelID, Contact, Name, Address, Postal Code, Parking space)
PK: HotelID

Invoice (InvoiceID, Amount, Discount, Date, Time)
PK: InvoiceID
FK: BookingID

Feedback (FeedbackID, Feedback text, Rating)
PK: FeedbackID
FK: BookingID, HotelID

6.4 NOSQL

In contrast to the tabular structure of rows and columns that is used by the relational database, the data is stored in the key-value format as well as JSON documents. We have built our NoSQL database by using MongoDB, and by utilising widgets, we are able to easily enter enormous amounts of data. The ability to quickly retrieve information associated to the list of hotels is a major benefit provided by NoSQL. This feature may play an important role in the overall user experience provided by search engines. When a user searches for a list of hotels in a certain region, country, or with a specified number of ratings, we are able to deliver the results instantaneously. This is a significant benefit of using NoSQL, and it is one of the main reasons why we decided to use it. Because of the nature of our circumstance, we decided to keep the database for our hotel in MongoDB so that we could more easily analyse it. Every single key-value pair is saved in what MongoDB refers to as a ”bucket,” which is essentially a collection. As a consequence of this, the same key could appear many times with different values each time.

Pseudo code: *Import Mongodb, widget and pandas library -> Establish connection with Mongo client -> Assign database to a variable -> Define function for saving the hotel information -> Create different key and value function for all variables -> Create a list with attributes of hotel list -> Use Insert once function for data insert operations. -> Create a loop to print if data is entered successfully else give an error message. -> Use widget function for text to mention that input will be taken in form of text box. -> Use widget button to save the entered information. -> Once the user enters the information and clicks on the save record button the hotel data will be saved and we can see the new entry by running the find operation.*

We follow the same structure for delete operation as well but use delete one command instead of the insert command to remove the record.

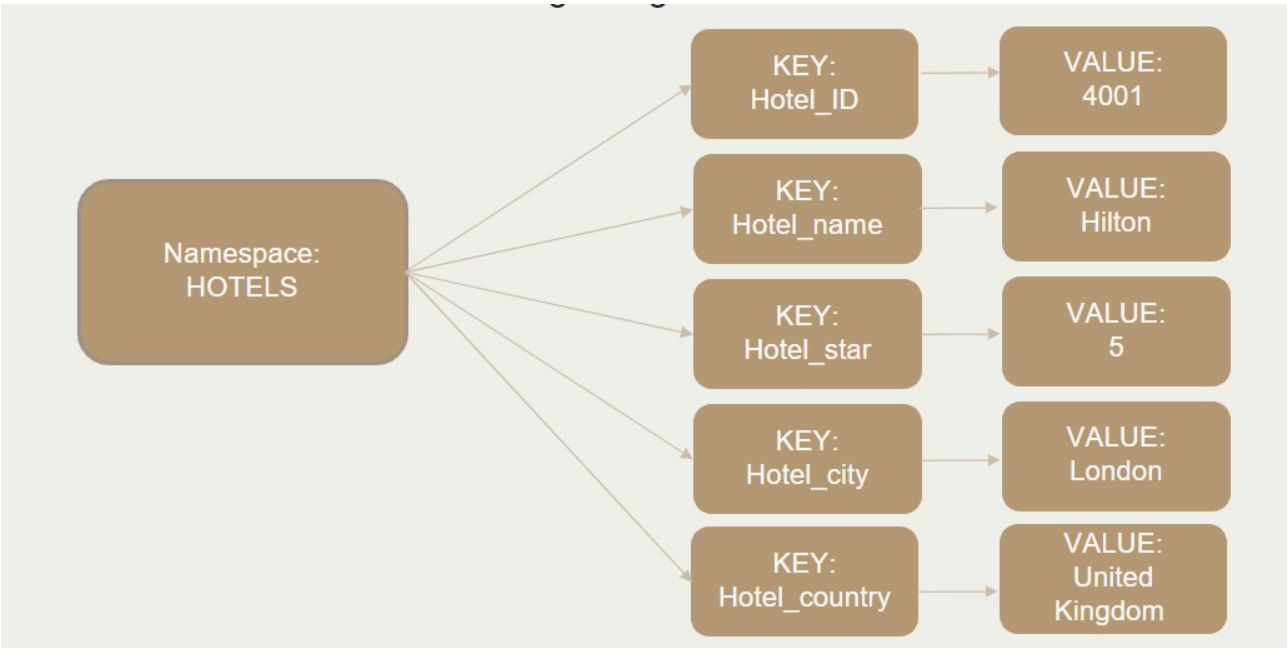

FIGURE 4: NOSQL KEY-VALUE

7 ANALYSIS

In order to provide a more comprehensive demonstration of our user case, we have finished a number of essential analytics activities, such as a rank table and summary statistics. The summary statistics may provide us with information on the booking frequency, the average amount spent per reservation, the highest and lowest discounts that are given, and other relevant information. These statistics are dynamic, which means that they are subject to change whenever the underlying database is altered.

Over here in Fig.5 we can see that the mean price of hotel per night in our database is 168 GBP per night. In fig.6 Using this ranking system you can see that hotel ID ”4002” and ”4006” have received 5 star ratings and are top of the ranking system. Over here in Fig.7 you can see the ratings distribution of the hotels in our database.

	Amount	Discount
count	10.000000	10.000000
mean	168.400000	27.400000
std	25.447986	13.226237
min	133.000000	13.000000
25%	154.000000	19.250000
50%	162.500000	23.500000
75%	185.250000	32.000000
max	220.000000	60.000000

FIGURE 5: MEAN OF AMOUNT

HotelHotelID	Rating
4002	5
4006	5
4004	4
4007	4
4009	4
4008	3
4001	2
4005	2
4003	1
4010	1

FIGURE 6: RANKING

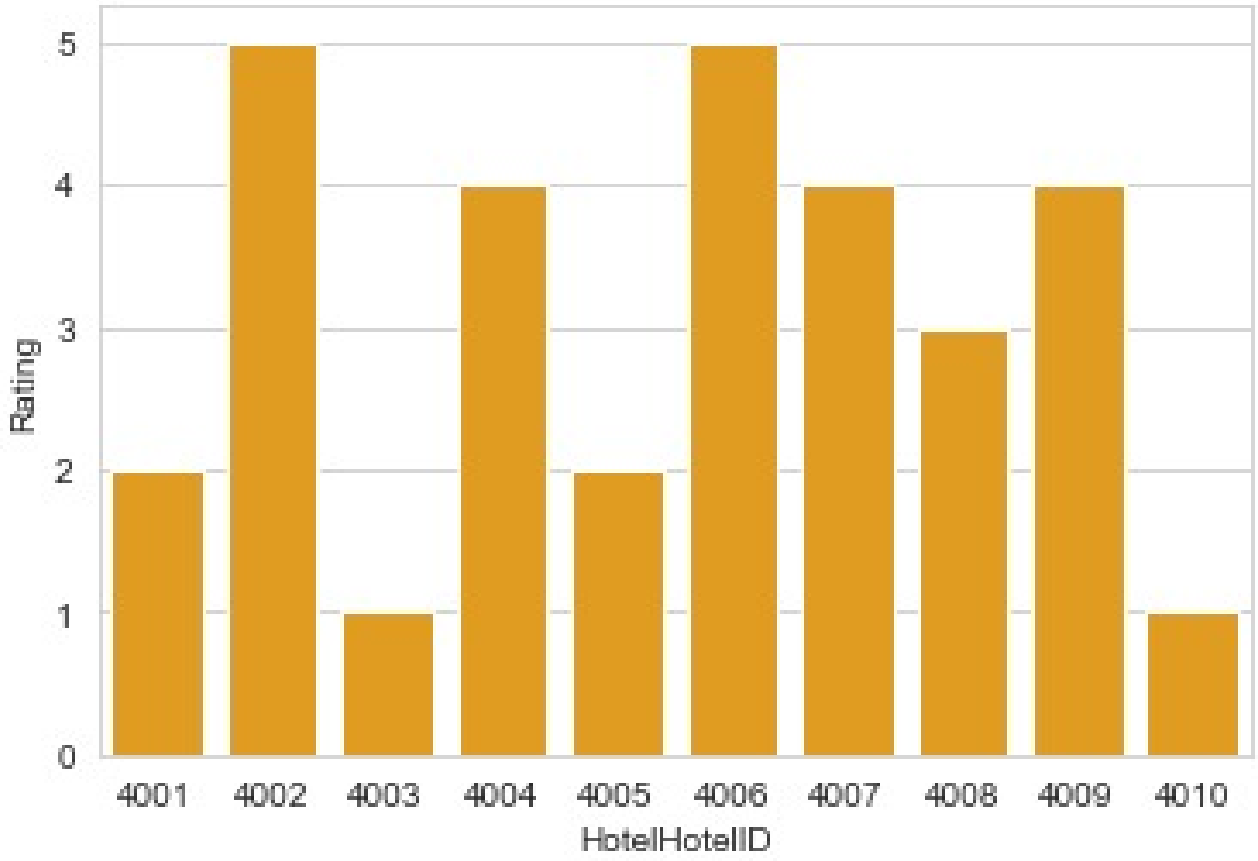

FIGURE 7: HOTEL RATINGS

8 CONCLUSION

In this report, we provide a documentation of the process by which we developed our online hotel booking application. The purpose of this project was to streamline the room reservation process and enhance the database of the hotel. We designed a use case for the application that depicts the whole of the customer experience, beginning with the search for an appropriate hotel room and ending with the customer rating and commenting on the institution. Throughout the whole of the project, we made use of Polyglot persistence to store and analyse data in a number of different data storage systems. For the relational database, we decided to go with SQLite, and for the non-relational database, we went with MongoDB. In addition, we developed an entity relationships diagram that presents a visual representation of the links that exist between the various entities included inside the system. The tables make use of the Third Normal form, which helps to eliminate data duplication and prevent data anomalies (3NF). We demonstrated core CRUD procedures for both relational and non-relational databases, as well as data analytics, in our part on the code.