

Room Booking System

Contents

1. Introduction.....	3
2. Data collection, preprocessing and analysis	3
2.1. Data Description and data collection.....	3
2.2. Data preprocessing	5
2.3. Data analysis.....	7
3. Database Design.....	10
3.1. ER diagram.....	10
3.2. Assumption.....	10
3.3. Functional dependencies	11
4. SQL codes and explanation.....	12
4.1. Insertion.....	12
4.2. Search	12
4.3. Deletion	14
5. Website design and feature implementation	14
6. Conclusion	17

1. Introduction

Our project is called UBooking system, which is a platform that can provide different kinds of rooms with people. When people travel or business work, they will encounter accommodation booking problems. Providing such a platform can solve the life problems everyone will encounter, provide travelers with more diversified travel options, help many people have a more comfortable experience, also promote the development of tourism.

In our project, we mainly focus on different kinds of user type like booker and host. People in our webpage can provide accommodation such as apartments or hotels, others can book those accommodation such as tourists or businessmen. Visitors can browse all kinds of rooms of hotels in different places on this website and search and filter them. When they log in their personal account, they can book hotel rooms according to the date, price, location and other information, and can directly talk with the host to learn about the information. The host can put his own house resources on this website, and can adjust the state of the house at any time. Finally, the administrator of the website can examine the house and transaction information and update.

So our project mainly divide into three parts: data preprocessing and analysis by Python, design a database, web design in Django and connect data with web to give function implement through SQL.

2. Data collection, preprocessing and analysis

2.1. Data Description and data collection

Our datasets contain detailed listings data, review data, room data and host data and all of our data are Airbnb listings. We mainly focus on five city which are Berlin, Sydney, Amsterdam, Melbourne, and Madrid. Since those data have a lot of the same attributes, so it is much easier for us to handle them in the following procedure. The dataset has

been analyzed, cleansed and aggregated where appropriate to facilitate public discussion. Also in our dataset, some of data we collect come from Airbnb site through Python.

Besides, most of our data come from Kaggle website. The data is sourced from publicly available information from the Airbnb site. We collected some data from Airbnb using python as well. The following is a code example that we collect room information of Sydney from Airbnb site. Firstly we found that there are 20 room information each city's page, and there are totally 15 pages of each city. Then we use for loop to acquire all information of a city at once. We use same process to get room information of other city as well. After we data collection, we converted them to the form as same as our schemas.

```
from bs4 import BeautifulSoup
import requests
import pandas as pd

urls1 = list(range(0, 281, 20))
urls1 = [str(i) for i in urls1]
urls2 = ['https://www.airbnb.com/s/Sydney/homes?tab_id=home_tab&refinement_paths%5B%5D=%2Fhomes&federated_search_session_id=fdeb87d0-b061-
+x+&search_type=pagination' for x in urls1]

Name = []
Stars = []
Facilities = []
Type = []
Capacity = []
Price = []
URL = []

for i in urls2:
    r = requests.get(i)
    soup = BeautifulSoup(r.text, "html.parser")
    table = soup.find('div', class_='fhph4u')
    for row in table.find_all('div', class_='8ssblpx'):
        name_el = row.find(class_='1jbo9b6h')
        name = name_el.text.strip()

        c_el = row.find(class_='1ulsev2')
        c = c_el.text.rstrip("\n")

        types_el = row.find(class_='13qbpeg')
        if types_el == None:
            types_el = row.find(class_='190542tw')
        types = types_el.text.strip()

        facility_el = row.find('div', class_='1ulsev2', style='margin-top:4px')
        facility = facility_el.text.strip()

        price_el = row.find('span', class_='1p7iugi')
        price = price_el.text.strip()[7:]

        stars_el = row.find('span', class_='3zgr580')
        if stars_el == None:
            stars = '5.0'
        else:
            stars = stars_el.text.strip()

        urls = row.find('img').get('data-original-uri')

        Name.append(name)
        Capacity.append(c)
        Type.append(types)
        Facilities.append(facility)
        Price.append(price)
        Stars.append(stars)
        URL.append(urls)

df_Sydney = pd.DataFrame({'Name':Name, "Capacity":Capacity, "Stars":Stars, "Price":Price, "Type":Type, "Facilities":Facilities, "URL":URL})
df_Sydney
```

There are a lot of garbled code in the CSV file we downloaded. We guess the most likely reason is that the data set provider transcoded the data in order to store the data in the CSV file. Therefore, most of the data in the dataset are mixed with Chinese characters. It is impossible to convert these garbled codes back to English letters, and to convert them into UTF-8 codes that are easy for Python to read and edit, the existence of garbled codes is inevitable. So we decided to delete the data containing the Chinese characters and select the perfect data for us to use.

(Downloaded data set that contain a lot of Chinese characters)

```
def is_Chinese(word):
    word1=str(word)
    for ch in word1:
        if '\u4e00' <= ch <= '\u9fff':
            return True
    return False
```

Then we synthesize the classified table, for example, the room information of each city, into one table through python, then traverse each grid of CSV file, and check whether it contains Chinese through the functions we wrote before. Once we detect that the data contains Chinese characters, we will turn the whole row into a null value. There are two reasons not to delete this line of data directly here. One is that the reduction of data

volume will cause the traversal to exceed the scope and result in an error. The second is that when the null value is deleted at the last time, "dropna" can be used to delete all empty roll make by garbled code and dataset provider. Finally, export to a new CSV file.

```
import pandas as pd
import numpy as np

for i in range(0, len(data)):
    for k in range(0, data.shape[1]):
        if is_Chinese(data.iloc[i, k]) == True:
            data.iloc[i, :] = np.nan
data = data.dropna(how='all')
data.to_csv('room.csv')
```

(A function that cleans up garbled code and removes empty lines)

After cleaning the files, the information between them does not match each other. For example, each room must have a host, and some rooms' hosts were deleted, so we need to remove redundant data. So we have two methods. For the data with less data, my team members choose to use Python to operate on the two tables.

```
data = pd.read_csv('host.csv')
room = pd.read_csv('room.csv')
HID = room['HID']
for i in range(0, len(room)):
    if HID.iloc[i] not in data['HID']:
        room.iloc[i, :] = np.nan
room = room.dropna(how='all')
room.to_csv('roomdan.csv')
```

(clean out all the room has no host in host.csv)

If there is a large amount of data, you can join directly through the SQL statement of the database, so that you can clear the unpaired data in multiple tables at the same time.

```
SELECT
    *
FROM
    search_room,
    detail_host
WHERE
    search_room.RID = detail_host
```

(simple and fast)

What we need to do next is that the wrong row of data caused by dataset provider, for

example, the ID column of host is the self-description of host. At this time, we use the filtering and sorting function of Excel, which can simply classify and sort the data types, and we can simply delete the data that does not belong to this column of data types. After completing the above steps, our data has been basically sorted out and can be uploaded to the database.

	id	name	summar	space	rhhood	notes	transi	access	interact	use ru
50010	20910897		0 5 minutes walk to Sydney of university	7 minutes walk to Redfern stati						
57650	22422314	1	"Vinya" iVinya is Explore tVinya offVinya is Open accePaddy is happy to							
81360	30533215	5	hhhh							
64181	24460520	52 4	lager tWORKING PERSON							
51759	21361303	150	Die Wohnung ist im potsdamer Platz	sony center im mitte von berlin ..g						
48840	20591521	452	A really lovely place to stay ...	200 metres from the beach. Huge priva						
5415	2943509	1119	Nice and stone floors... decorated in greys and beige.	I travel a lot gi						
77556	29724887	2703	very new property							
8904	4745015	1	cosy rAt 1 min Single rcAt the intersectic	The city wi-fi, fulThe ownerSmoking						
5902	3254077	ApartmerRuim en eComfortatGelegen iGrote gezTussen Station RAI	Voor vrageNo pets,							
8210	4486178	BEST 4 COur moderModern aqWe are situated in	Trams areGuests c&As I am rwe only							
9754	5072599	BushlandOur home My home iThey are The pool	Yes bus eThe guestAs much eWith a r							

(After sorting "name", it simply filters out the wrong data with "ID")

2.3. Data analysis

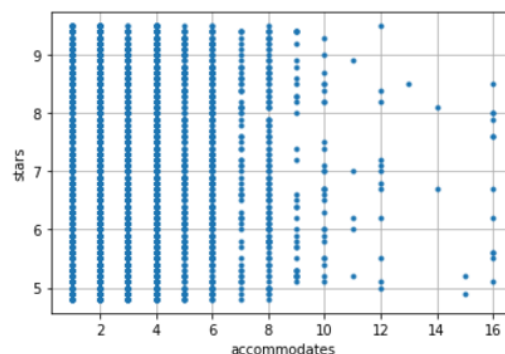
In this section, we will mainly analyze our data and through Python to illustrate some relationship among our datasets.

```
In [2]: import matplotlib
from pandas import read_csv
import matplotlib.pyplot as plt

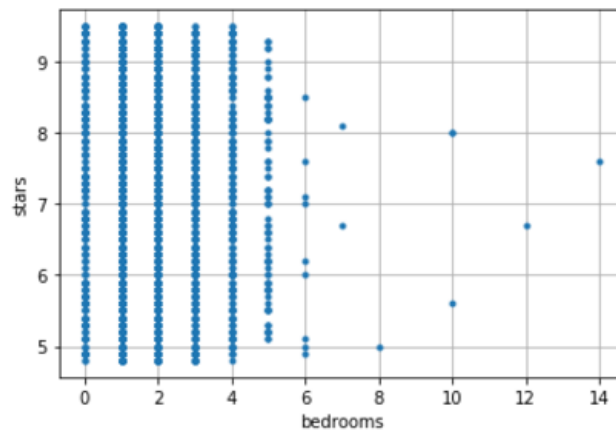
data = read_csv("F:\大二下 Python\project\detail_room_info.csv")

In [3]: plt.plot(data['accommodates'], data['stars'], '.')
```

#Assign the data to (x,y)
#Set the name of the X-axis
#Set the name of the Y-axis
#Render grid lines
#Visual presentation



```
plt.plot(data['bedrooms'], data['stars'], '.') #Assign the data to (x,y)
plt.xlabel('bedrooms') #Set the name of the X-axis
plt.ylabel('stars') #Set the name of the Y-axis
plt.grid(True) #Render grid lines
plt.show() #Visual presentation
```



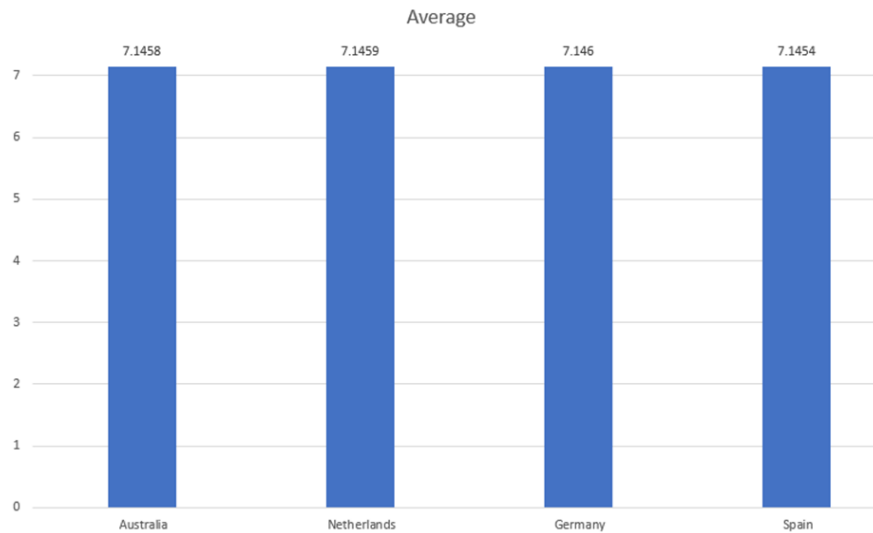
These two scatter plots focus on accommodates, bedrooms and stars. We wonder if there is any connection between stars and them. From the chart we can see the stars and accommodates distribution is relatively uniform, so it does not mean that luxury large room can obtain high stars. In addition, in the second picture, there are not enough samples to increase the persuasion when the number of bedrooms increase. Therefore, according to these two tables, the distribution of room should be even, so as to meet the needs of different customers.

And the following two pictures are analysis between the country distribution of rooms and stars. At first, we wanted to get the country information of all the rooms, so we wrote a function in excel to get the country distribution of the room from the street attribute in the listing table.

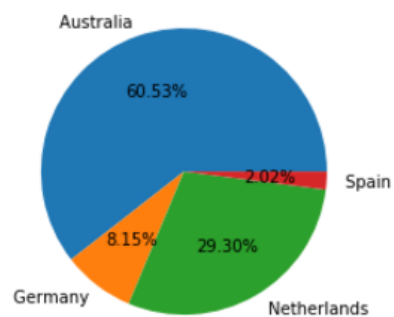
B1		=RIGHT(A1,LEN(A1)-FIND("@",SUBSTITUTE(A1,"","@",LEN(A1)-LEN(SUBSTITUTE(A1,"",""))),1))			
	A	B	C	D	E
1	Amsterdam, Noord-Holland, Netherlands	Netherlands	3209		You will love our spacious (9
2	Bulleen, VIC, Australia	Australia	9835		
3	Brunswick East, VIC, Australia	Australia	10803		A large air conditioned room
4	Pyrmont, NSW, Australia	Australia	12351		Come stay with Vinh & Stuar
5	St Kilda East, VIC, Australia	Australia	16760		
6	Balmain, NSW, Australia	Australia	20865		Hi! We are a married profess
7	Amsterdam, North Holland, Netherlands	Netherlands	25428		
8	Bellevue Hill, NSW, Australia	Australia	26174		
9	Amsterdam, North Holland, Netherlands	Netherlands	27886		Stylish and romantic houseb
10	Amsterdam, North Holland, Netherlands	Netherlands	28871		

Then we join it in room_info table and made a pie chart in python to show their proportions. In the second picture, we present the average stars of all the rooms in the same country in a bar chart. Finally, we found that although the distribution

proportion of the sample in the four countries was quite different, the average stars of each country did not differ significantly, which means that stars were not necessarily related to countries.

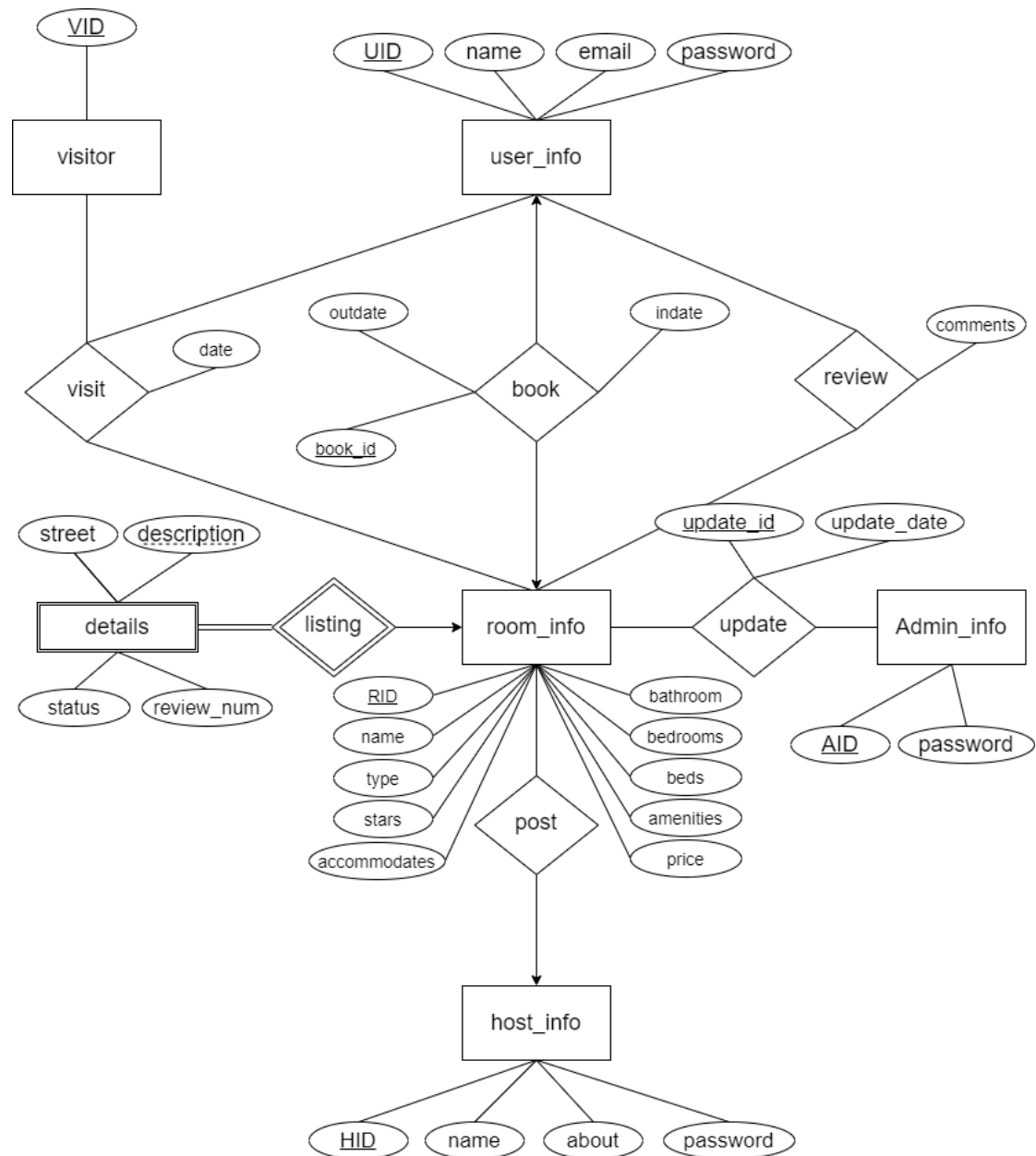


```
plt.pie(gb['count'], labels=gb['country'], autopct='%.2f%%')  
## 2f means keep two decimal places and %% means %
```



3. Database Design

3.1. ER diagram



3.2. Assumption

Based on our project requirement, we made some assumption to give constraints for our database.

1. Every user of our website has their own unique id. For example, VID is a primary key of visitor, UID is a primary key of user, HID is a primary key of host and AID is a primary key of administer.

2. User and room has a one to one relationship, which means one room can only be booked by one specific user in particular time and is identified by book_id.
3. Listing relationship connect week entity details to strong entity romm_info, all details must have one corresponding RID
4. Host_info and room_info has a one to many relationship, each room can only be posted by one host.
5. Visitor just have one function, which is visit the search_listing.
6. The number of each room are set to be one, which means once a booker choose the room A, then other bookers cannot book this room anymore and status in listing will be changed to zero.

3.3. Functional dependencies

Here are all functional dependencies that we made based on the knowledge that we learn before.

1. visit = (VID, RID, date), $F=\{VID,RID \rightarrow date\}$, Key = {VID,RID}
2. user_info=(UID, name, email, password), $F=\{UID \rightarrow name, email, password\}$, Key = {UID}
3. book = {UID, RID, book_id, indate }, $F=\{ book_id \rightarrow UID,RID,indate \}$, Key = {book_id}
4. review = {UID, RID, comments}, Key = {UID, RID, comments}
5. room_info = {RID, name, type, stars, accommodates, bathrooms, bedrooms, beds, amenities, price}

 $F=\{RID \rightarrow name,type,stars,accommodates,bathrooms,bedrooms,beds,amenities,price\}$, Key = {RID}
6. listing = {RID, street, description, status}, $F=\{RID \rightarrow street,description,status\}$, Key = {RID, description}
7. Adimin_info = {AID, password}, $F=\{AID \rightarrow password\}$, Key = {AID}
8. update = {RID, AID, update_id, update_date}, $F=\{update_id \rightarrow update_date,RID,AID\}$, Key = {update_id}
9. host_info = {HID, name, about, password}, $F=\{HID \rightarrow uname,about,password\}$, Key = {HID}
10. post = {RID, HID}, $F=\{RID \rightarrow HID\}$, Key = {RID}

4. SQL codes and explanation

4.1. Insertion

In our project, whenever we register a new user, submit a reservation form, or host uploads a room information, these will insert data into the corresponding table in our database.

```
✓ 插入了 1 行。  
插入的行 id: 6845 (查询花费 0.0186 秒。)  
  
INSERT INTO `detail_room_info` (`id`, `RID`, `name`, `type`, `accommodates`, `stars`, `bathrooms`, `bedrooms`, `amenities`, `price`,  
`image_url`) VALUES (NULL, '12345', 'Comfortable room', 'Entire home/apt', '5', '0', '1', '2', 'Internet,Wifi,Kitchen', '160',  
'https://a0.muscache.com/im/pictures/88955424/496ae7bb_original.jpg?aki_policy=large')
```

This code means insert a new room that has a “room ID”, “room name”, “room type”, “number of people allowed”, “rating stars”, and number of bed and bedroom and other equipment in this room.

```
✓ 插入了 1 行。  
插入的行 id: 41416 (查询花费 0.0267 秒。)  
  
INSERT INTO `login_users` (`id`, `username`, `password`, `c_time`, `email`, `usertype`) VALUES (NULL, 'Joyce', '1234', '2020-05-26  
17:11:33.991128', '15824304@qq.com', '1')
```

This row code means insert a new user (through create account function) with name as “Joyce” password as “1234”, and login date is 2020-05-26 ,also with a email and user type is booker(In our system, we set “1” is booker and “2” is host)

```
✓ 插入了 1 行。  
插入的行 id: 811 (查询花费 0.0035 秒。)  
  
INSERT INTO `center_book` (`UID`, `RID`, `book_id`, `indate`) VALUES ('264427', '3209', '1', '2020-3-18')
```

This line of code inserts a new order from booker ID 264427 for room ID 3209, and record the check date.

4.2. Search

In our webpages, we often need to get information according to keywords or room and user ID and display it on the page. In order to complete this operation, it is often necessary to join two tables to search for information. Since our project only uses one database from beginning to end, we can safely use extra to implement join operation.

```

→RID_record = room_info.objects.filter(RID=RID).extra(
→→→select=['hostname': 'search_host_info.name',
          'about': 'search_host_info.about',
          'email': 'search_host_info.email',
          'location': 'search_listing.street',
          'HID': 'search_host_info.HID'],
→→→tables=['search_listing', 'search_host_info'],
→→→where=['detail_room_info.RID=search_listing.RID',
          'search_listing.HID=search_host_info.HID'])

```

The purpose of this function is to get the room ID from the room_ Info and host_ Info table to get room and host information, and display on the web page. It takes more time but is more accurate. So we translate it into SQL statements and test the run time.

```

SELECT
    detail_room_info.name,
    detail_room_info.image_url,
    search_host_info.name,
    about,
    email,
    street,
    search_host_info.HID
FROM
    search_listing,
    search_host_info,
    detail_room_info
WHERE
    detail_room_info.RID = 3209 AND
    search_listing.RID = detail_room_info.RID AND
    search_listing.HID = search_host_info.HID

```

✓ 正在显示第 0 - 0 行 (共 1 行, 查询花费 0.0018 秒。)

```

SELECT detail_room_info.name, detail_room_info.image_url, search_host_info.name, about, email,
street, search_host_info.HID FROM search_listing, search_host_info, detail_room_info WHERE
detail_room_info.RID = 3209 AND search_listing.RID = detail_room_info.RID AND
search_listing.HID = search_host_info.HID

```

This is the most complex query operation in our project. It takes only 0.0018 seconds (far less than the required two seconds) to extract information from three tables. Another case where similar SQL codes are used is that we have made a function for customers to sort the displayed rooms by price, number of review, or rating. We take sorting by price as an example.

```

— listings = listing.objects.filter(street__contains=country, statement=1).extra(
select={'name': 'detail_room_info.name',
        'pic': 'detail_room_info.image_url',
        'rating': 'detail_room_info.stars',
        'price': 'detail_room_info.price'},
tables=['detail_room_info'],
where=['search_listing.RID=detail_room_info.RID']
).order_by('price')

```

(after joining the two tables, use order_by to sort)

✓ 正在显示第 0 - 24 行 (共 4137 行, 查询花费 0.0724 秒。)

```

SELECT NAME , street, image_url, stars, price FROM search_listing, detail_room_info WHERE
search_listing.RID = detail_room_info.RID AND street LIKE "%Australia%" ORDER BY price

```

This is the converted SQL code. It also takes only 0.0724 seconds, far less than the required 2 seconds.

4.3. Deletion

The only once our projects that uses the delete function is that Booker cancels his order. When the delete button is clicked, the function will search the database for data with the same booking ID and delete it.

```

def delete(request, type, id):
    — if type=="booker":
    —     UID=request.session.get('bookid')
    —     dingdan=book.objects.get(UID=UID)
    —     listing.objects.filter(RID=dingdan.RID).update(statement='1')
    —     book.objects.filter(id=id).delete()
    —

```

✓ 影响了 1 行。 (查询花费 0.0036 秒。)

```

DELETE FROM center_book WHERE center_book.id = 811

```

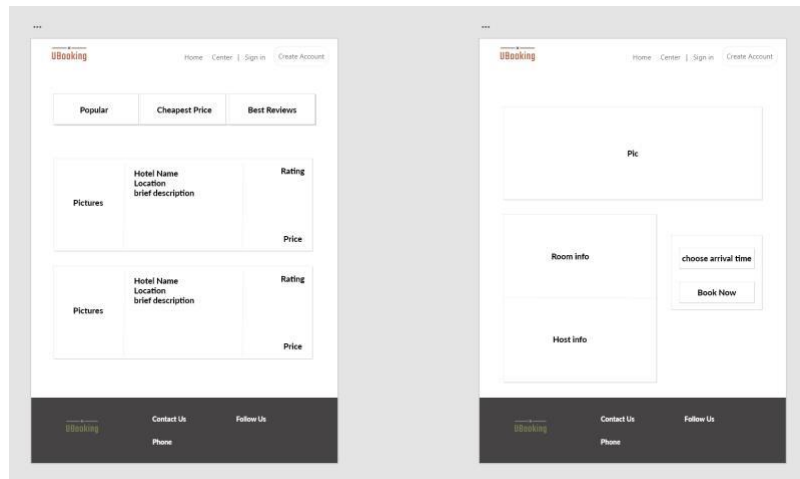
[\[编辑内嵌\]](#) [\[编辑\]](#) [\[创建 PHP 代码\]](#)

The above SQL statement deleted the order with ID 811, which took only 0.0036 seconds. So the code of our delete operation is qualified.

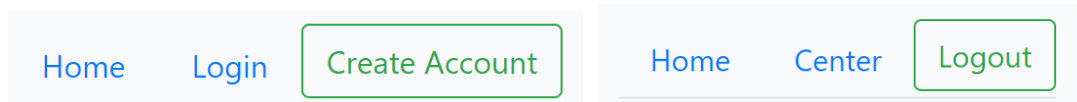
5. Website design and feature implementation

To give a perfect design of our websites, we consider all the data we collect and other websites like

Airbnb and booking to give a outline of our webpages through Adobe XD.



So we take about one week to accomplish our web design and also implement our function. There are some of function in our website. First, we have login/logout, once they use their own account to enter website, then they can find their own center and book a room or provide rooms.



Then the second function is to search room they want. Basically, we divide rooms into their corresponding country, and in each search listing, user can sort all room by cheapest price, popular and best review.

- Germany -

Popular Cheapest Price Best Review

The next function is to book a room as a booker. Once they find the room they desire, they can choose an arrival time, then click book button. After that, our webpage will tell you that you have reserved successfully. Once a person book, then this room is not available to others, which means other user cannot book this room any more.

Nice Room in Prenzlauerberg,

Location: Berlin, Berlin, Germany

\$44

Room ID: 31440



Arrival Date

2020-06-03

Choose Date!

Book Right Now!

127.0.0.1:8000 显示

Reserve Successfully!

确定

After booking, if user want to check out their book history, they have right to view all book history in their own center page.

Hi, BOOKER(hooa)

Say something

Booker ID: 9110

All Book History



Nice Room in Prenzlauerberg,

Learn more!

Cancel

Now if this booker have no to enjoy his room and he want to cancel his booking, then just click cancel button and booking will be disappeared.

All Book History

You don't have any Booking!

Besides we can post new room in the website as host. Once they fill the form and provide room information in their center page, the website will show them a message said submit successfully, then they can also check from their center if they submit or not.

Provided Room
Post New Room

You haven't provide any Room!

Provided Room
Post New Room

Please Fill this Form

Room Name

Location

Country

Room Type

Accommodates


Bathrooms


Bedrooms

127.0.0.1:8000 显示
Submit Successfully!

确定

Provided Room
Post New Room



Joyce Room
Xiaogang Road Binjiang
Street,Germany
 Super comfortable. Great for family!

Room ID: 7960674

\$23

6. Conclusion

In this project, we find a lot of difficulties that we encounter. In the previous preprocessing part, we get and scrape many data from websites but we have no idea to consider which attributes are the one that we want. So before the preprocessing, we gather all the data and discuss together to confirm the data that are not necessary like discount and available day. Then when we start to design the websites, we have final confirm which attribute should be removed.

Another difficult is that it is hard to negotiate something in the online discussion. Normally, we can show our outcome with each other face to face. However, we have to describe something orally. So we decided open an online discussion from afternoon to midnight day by day. Once we have something that we need give a hand from others, then just speak up in the meeting and give a

screenshot in the WeChat group.

The last difficult is that we need to look for a lot of other concept by ourselves. For example, when we want to decide pagination in our website, we have to look for some related guideline from Internet. So we browse different kinds of websites then continue to revise our websites. Another example is that how to connect Django with our database. Actually, we are all confused about it and have no idea to start. But fortunately, we find some guideline from others and they even teach us how to join two database relations to illustrate our information. Since we got limited time, we think it is good enough for us to finish our project and overcome these difficulties.