

PROJECT REPORT

IB9HPO - DATA MANAGEMENT

MODULE CONVENOR:
DR NIKOLAOS KORFIATIS



GROUP 7

Tables of Contents

TABLES OF CONTENTS	1
1. PART A – STRUCTURED DATABASE DESIGN	3
1.1. CONCEPTUAL DESIGN	3
1.1.1. INTRODUCTION OF ATTRIBUTES AND ENTITIES.....	4
1.1.2. RELATIONSHIP TYPES	8
1.1.3. ASSUMPTIONS.....	8
1.1.4. ASSUMPTIONS FOR CARDINALITY	9
1.2. LOGICAL DESIGN	10
1.3. DDL QUERIES.....	12
2. PART B -SEMI STRUCTURED DATA(FILES).....	33
3. PART C – SEMI STRUCTURED DATA (WEB).....	36
4. PART D – DASHBOARD WITH R/SHINY	42
4.1. SHINY DASHBOARD.....	44
REFERENCES.....	47

Index of figures

Figure 1: E-R Diagram.....	3
Figure 2: Logical E-R Diagram	11
Figure 3: Presence in UK	44
Figure 4: Food Hygine Dataset	45
Figure 5: Location Wise Ratings	45
Figure 6: LocationWise Data Coventry	46

GROUP 7

Index of tables

Table 1: Relationships and cardinalities	9
--	---

1. Part A – Structured Database Design

1.1. Conceptual Design

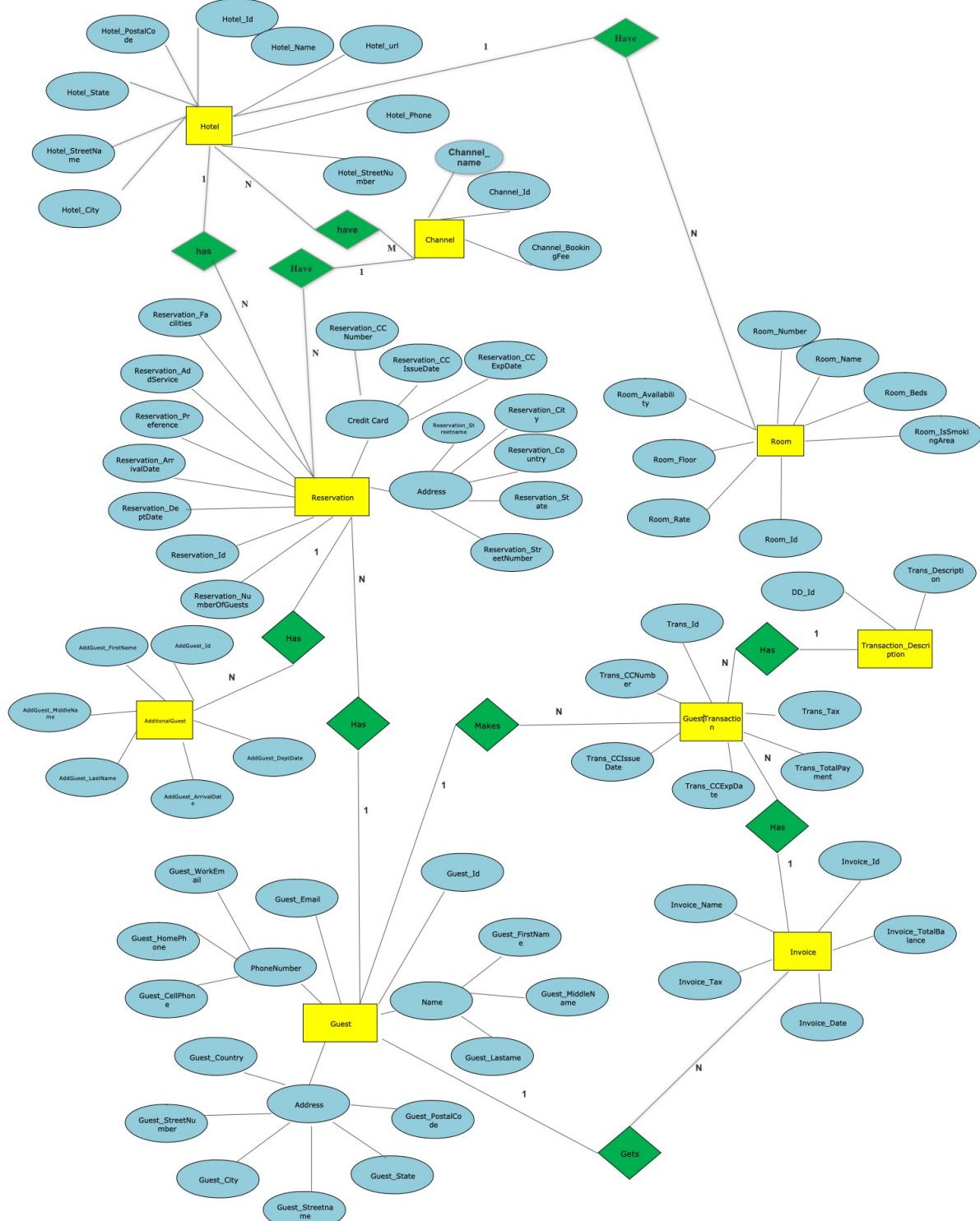


Figure 1: E-R Diagram

GROUP 7

The conceptual design shown in Diagram is an entity-relationship model which is a visual representation of the entities as well as their relationship with each other. The rectangular boxes represent the entities connected to the attributes shown as the oval shape through the diamond shape which creates a relationship between the two of them. In addition, the relationships between entities can be represented by different cardinalities such as whether it is optional, mandatory or the limit of the relationship.

There are 8 Entities: Hotel, Guest, AdditonalGuest, Reservation, Channel, Transaction, Invoice, and Transaction_description

1.1.1. Introduction of Attributes and Entities

Hotel:

- Hotel_id is the primary key of the hotel entity
- Hotel_Name is the name of the hotel
- Hotel_WebsiteUrl is the electronic address of the hotel
- Hotel_Phone is the phone number of the hotel
- Hotel_StreetNumber is the street number of the address of the hotel
- Hotel_StreetName is the street name of the address of the hotel
- Hotel_City is the city where the hotel is located in
- Hotel_State is the state the hotel is located in
- Hotel_PostalCode is the postal code of the address of the hotel

Guest:

- Guest_Id is the primary key of the CUSTOMER entity
- Guest_FirstName is the first name of the customer
- Guest_MiddleName is the middle name of the customer
- Guest_Lastame is the last name of the customer
- Guest_Email is the electronic mail address of the customer
- Guest_WorkEmail is the customer's work electronic mail address
- Guest_HomePhone is the customer's phone number at home

GROUP 7

- Guest_CellPhone is the customer's cell phone number
- Guest_StreetNumber is the street number of the customer's home address
- Guest_Streetname is the street name of the customer's home address
- Guest_City is the city of the customer's home address
- Guest_State is the state of the customer's home address
- Guest_Country is the country of the customer's home address
- Guest_PostalCode is the postal code of the customer's home address

AdditionalGuest:

- AddGuest_Id is the primary key of the GUEST entity
- Reservation_Id is the reservation id on which the guest stay is registered on
- Room_Id is the room id which is assigned to the guest
- AddGuest_FirstName is the guest's first name
- AddGuest_MiddleName is the guest's middle name
- AddGuest_LastName is the guest's last name
- AddGuest_ArrivalDate is the time and date of planned arrival of the guest to the hotel
- AddGuest_DeptDate is the time and date of the actual departure of the guest from the hotel

Reservation:

- Reservation_Id is the primary key of the RESERVATION entity
- Hotel_Id is the foreign key for the hotel id
- Guest_Id is the foreign key for the customer id
- Channel_Id is the foreign key for the channel id
- Trans_Id is the foreign key for the transaction id
- Reservation_DeptDate is the planned reservation departure date and time
- Reservation_ArrivalDate is the planned reservation arrival date and time
- Reservation_Preference is the reservation preference
- Reservation_AddService is the additional services booked with the reservation
- Reservation_Facilities is the additional facilities booked with the reservation
- Reservation_CCNumber is the number of the credit card used for booking the reservation

GROUP 7

- Reservation_CCIssueDate is the issue date of the credit card used for booking the reservation
- Reservation_CCExpDate is the expiry date of the credit card used for booking the reservation
- Reservation_NumberOfGuests is the number of additional guests under the same reservation apart from the customer booking the reservation
- Reservation_Status is the status of the reservation either pending, completed or cancelled
- Reservation_Type is the type of reservation either group or family/no group, and the data type is Boolean
- Reservation_StreetNumber is the customer's street number referenced to the reservation
- Reservation_Streetname is the customer's street name referenced to the reservation
- Reservation_City is the customer's city name referenced to the reservation
- Reservation_State is the customer's state name referenced to the reservation
- Reservation_Country is the customer's country referenced to the reservation

Channel

- Channel_Id is the primary key of the CHANNEL entity
- Hotel_Id is the foreign key for the hotel id
- Channel_Name is the name of the channel the booking is made from
- Channel_BookingFee is the charge of the respective channel for the booking

Transaction

- Trans_Id is the primary key of the TRANSACTION entity
- Guest_Id is the foreign key for the customer id
- TD_Id is the foreign key for the transaction description id
- Trans_CCNumber is the number of the credit card used
- Trans_CCIssueDate is the issued date of the credit card used
- Trans_CCExpDate is the expiration date of the credit card used
- Trans_TotalPayment is the total amount paid
- Trans_Tax is the respective tax amount for each transaction

GROUP 7

Invoice

- Invoice_Id is the primary key of the INVOICE entity
- Trans_Id is the foreign key for the transaction id
- Guest_ID is the foreign key for the customer id
- Invoice_Name is the name under which the invoice is printed
- Invoice_TotalBalance is the total balance of the invoice
- Invoice_Tax is the total tax for the total balance of the invoice
- Invoice_Date is the date and time the invoice is printed

Room

- Room_Id is the primary key of the ROOM entity
- Reservation_Id is the foreign key for the reservation id
- Hotel_Id is the foreign key for the hotel id
- Room_Number is the number of the room
- Room_Name is the name of the room
- Room_Beds is the number of beds available in the room. Also represents the number of people allowed to sleep in the room
- Room_IsSmokingArea is the permission of smocking in the room
- Room_Floor is the floor the room is located on
- Room_Rate is the rate per night for the room
- Room_Availability is the status of availability of the room at a particular point in time

Transaction_Description

- TD_Id is the primary key of the TRANSACTION_DESCRIPTION entity
- Trans_Description is the description on the type of transaction

GROUP 7

1.1.2. Relationship Types

- (Pays) Between **Hotel** Table and **Channel** Table
- (Has) Between **Hotel** Table and **Reservation** Table
- (Has) Between **Hotel** Table and **Room** Table
- (Has) Between **Guest** Table and **Reservation** Table
- (Has) Between **Reservation** Table and **AdditionalGuest** Table
- (Through) Between **Reservation** Table and **Channel** Table
- (Makes) Between **Guest Table** and **GuestTransaction** Table
- (Gets) Between **Guest Table** and **Invoice** Table
- (Has) Between **GuestTransaction** Table and **Invoice** Table
- (Has) Between **Transaction_Description** Table and **GuestTransaction** Table

1.1.3. Assumptions

- Guests are persons who make the reservation, AdditionalGuest are persons who don't make reservation but contained in reservations
- When a guest makes a transaction (example: having dinner in hotel restaurant), he/she will be a Guest
- When the guest makes a reservation, the hotel will only record the information of that guest even if a group of people checks in, but the hotel will have names, age, date of arrival and departure of all the additional guests in their database
- Not Only guests who book a room in the hotel have the right to book other hotel facilities but additional guests can also book the facilities, first hotel has to register them as a guest and then additional guests can book the services, so that we can differentiate between guests and additional guests properly. The guests are the one who pays for any service at the hotel
- The channel table contains 15 different channels and hotel group's booking system as the 16th channel. No booking fee is charged when a reservation is made through the hotel group's booking system

GROUP 7

- Assume that the guests can pay their bills at the departure day or when they receive any service (going to bar, using seminar hall). The GuestTransaction table will store every transaction of the guest even if it is unpaid. And at the departure day the hotel will calculate all the transactions from GuestTransaction table and make the invoice for the guest. So that the invoice will have every transaction of the Guest. If its paid or unpaid or shows if any balance amount, he/she has to pay.

1.1.4. Assumptions for cardinality

Table 1: Entities, Relationships and cardinalities

Entities	Relationship	Cardinality
Hotel and Channel	One hotel can be active in different channels and there are different hotels in one channel.	M:N
Hotel and Reservation	One hotel has many reservations, but one reservation is just for one hotel.	1:N
Hotel and Room	One hotel has many rooms, but one room belongs to one hotel.	1:N
Guest and Reservation	One guest can make many reservations, but one reservation belongs to one customer.	1:N
Reservation and AdditionalGuest	There can be many additional guests in one reservation, but one guest belongs to one reservation.	1:N
Channel and Reservation	There are many reservations in one channel, but one reservation happens in one channel.	1:N
Guest and GuestTransaction	One guest can have many transactions, but one transaction is made by one guest.	1:N
Guest and Invoice	One guest can have many invoices, but one invoice belongs to one customer.	1:N

Invoice and GuestTransaction	One invoice can contain many transactions, but one transaction just belongs to one invoice.	1:N
Transaction_Description and GuestTransaction	One Transaction_Description can correspond to many transactions, but one transaction just corresponds to one Transaction_Description.	1:N

1.2. Logical Design

1.2.1. Logical Design Description

- The **Reservation** table has Reservation_Id as the Primary Key and uses Hotel_Id which is the Primary Key of **Hotel** table, Guest_Id which is the Primary Key of **Guest** table, Channel_Id which is the Primary Key of **Channel** table and Trans_Id which is the Primary Key of **GuestTransaction** table as Foreign Key, which can be shown in **Figure 2**
- The **Channel** table has Channel_Id as the Primary Key and uses Hotel_Id which is the Primary Key of **Hotel** table as Foreign Key, which can be shown in **Figure 2**
- The **Hotel** table has Hotel_Id as the Primary Key without Foreign Key, which can be shown in **Figure 2**
- The **Room** table has Room_Id as the Primary Key and uses Reservation_Id which is the Primary Key of **Reservation** table and Hotel_Id which is the Primary Key of **Hotel** table as Foreign Key, which can be shown in **Figure 2**
- The **Guest** table has Guest_Id as the Primary Key without Foreign Key, which can be shown in **Figure 2**
- The **AdditionalGuest** table has AddGuest_Id as the Primary Key and uses Reservation_Id which is the Primary Key of **Reservation** table and Room_Id which is the Primary Key of **Room** table as Foreign Key, which can be shown in **Figure 2**
- The **GuestTransaction** table has Trans_Id as the Primary Key and uses Guest_Id which is the Primary Key of **Guest** table and TD_Id which is the Primary Key of **Transaction_Description** table as Foreign Key, which can be shown in **Figure 2**
- The **Invoice** table has Invoice_Id as the Primary Key and uses Trans_Id which is the Primary Key of **GuestTransaction** table and Guest_Id which is the Primary Key of **Guest** table as Foreign Key, which can be shown in **Figure 2**

GROUP 7

- The **Transaction_Description** table has TD_Id as the Primary Key without Foreign Key, which can be shown in **Figure 2**

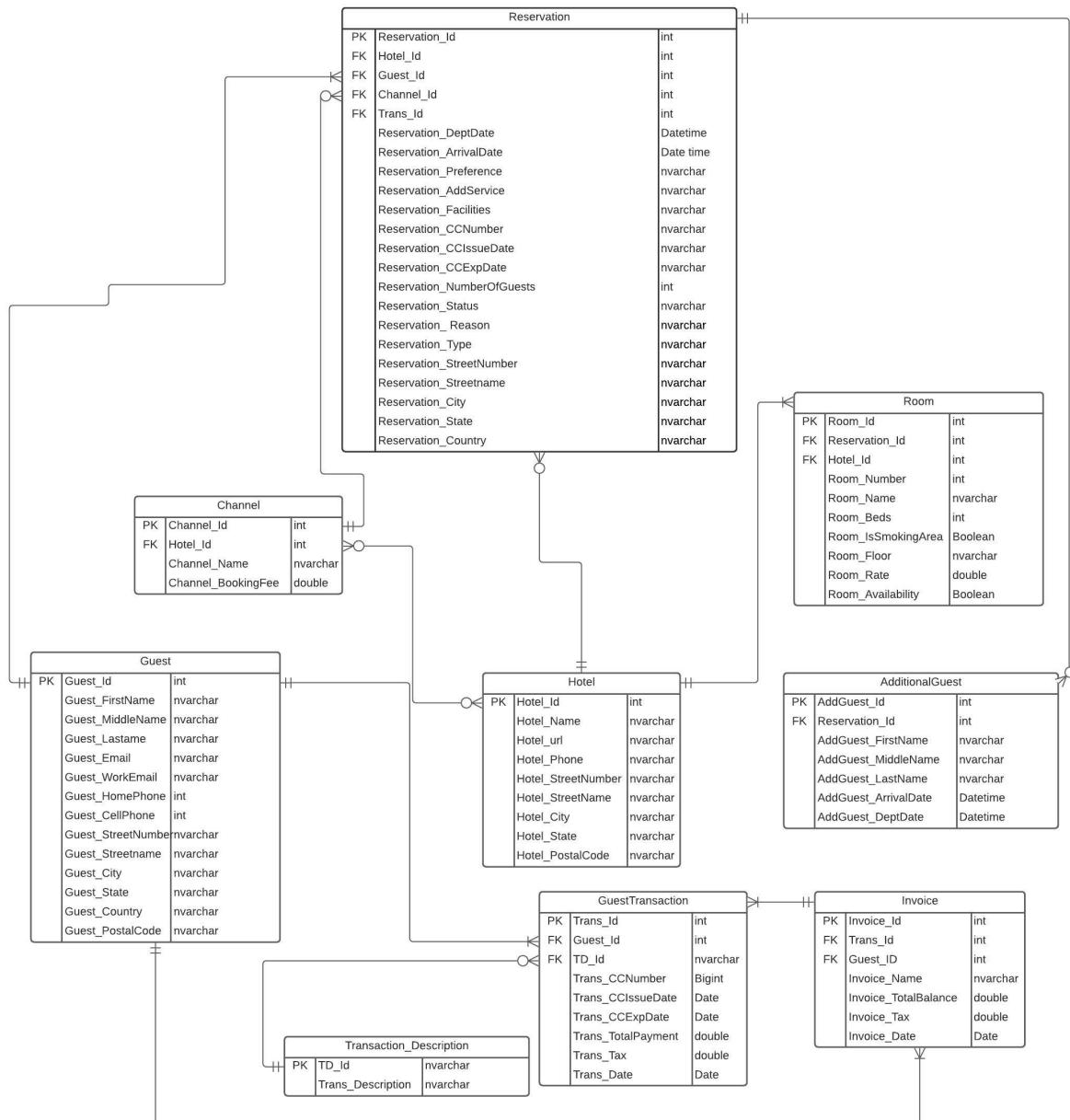


Figure 2: Logical E-R Diagram

GROUP 7

1.3. DDL Queries

```
1 R Markdown
2
3 Installing necessary packages for Data Management to work,
4
5 # install.packages("tidyverse")
6 # install.packages("janitor")
7
8 library("tidyverse")
9 ## — Attaching packages tidyverse
10 1.3.0 —
11
12 ## ✓ ggplot2 3.3.2      ✓ purrr  0.3.4
13 ## ✓ tibble   3.0.4      ✓ dplyr   1.0.2
14 ## ✓ tidyr    1.1.2      ✓ stringr 1.4.0
15 ## ✓ readr    1.4.0      ✓forcats 0.5.0
16
17 ## — Conflicts tidyverse_conflicts() —
18 ## x dplyr::filter() masks stats::filter()
19 ## x dplyr::lag()   masks stats::lag()
20
21 library("dplyr")
22 library("janitor")
23
24 ## — Attaching package: 'janitor'
25
26 ## The following objects are masked from 'package:stats':
27 ##     chisq.test, fisher.test
28
29 library("tidyverse")
30
31 library(plyr)
32
33 ## -----
34
35 ## You have loaded plyr after dplyr - this is likely to cause problems.
36 ## If you need functions from both plyr and dplyr, please load plyr first,
37 ## then dplyr:
38 ## library(plyr); library(dplyr)
39
40 ## -----
41
42 ## — Attaching package: 'plyr'
43
44 ## The following objects are masked from 'package:dplyr':
45 ##     arrange, count, desc, failwith, id, mutate, rename, summarise,
46 ##     summarize
```

GROUP 7

```
41 ## The following object is masked from 'package:purrr':  
42 ##  
43 ##     compact  
44 library(stringr)  
45 library(rebus)  
46 ##  
47 ## Attaching package: 'rebus'  
48 ## The following object is masked from 'package:stringr':  
49 ##  
50 ##     regex  
51 ## The following object is masked from 'package:ggplot2':  
52 ##  
53 ##     alpha  
54 library(readr)  
55 library(RSQLite)  
56 #Connect to SQLite  
57 myconn <- RSQLite::dbConnect(RSQLite::SQLite(),"HotelManagement.db")  
58 #Creating table hotel and dropping the table if it already exists  
59 RSQLite::dbSendQuery(myconn,"DROP TABLE IF EXISTS `Hotel`")  
60 ## <SQLiteResult>  
61 ##   SQL  DROP TABLE IF EXISTS `Hotel`  
62 ##   ROWS Fetched: 0 [complete]  
63 ##       Changed: 0  
64 RSQLite::dbSendQuery(myconn, "CREATE TABLE `Hotel` ("  
65   `Hotel_Id` int(11) NOT NULL,  
66   `Hotel_name` varchar(255) NOT NULL,  
67   `Hotel_WebsiteUrl` varchar(255) NOT NULL,  
68   `Hotel_Phone` varchar(255) NOT NULL,  
69   `Hotel_StreetNumber` varchar(255) NOT NULL,  
70   `Hotel_StreetName` varchar(255) NOT NULL,  
71   `Hotel_City` varchar(255) NOT NULL,  
72   `Hotel_State` varchar(255) NOT NULL,  
73   `Hotel_PostalCode` varchar(255) NOT NULL,  
74   PRIMARY KEY (`Hotel_Id`)  
75 )")  
76 ## <SQLiteResult>  
77 ##   SQL  CREATE TABLE `Hotel` (  
78 ##   `Hotel_Id` int(11) NOT NULL,  
79 ##   `Hotel_name` varchar(255) NOT NULL,  
80 ##   `Hotel_WebsiteUrl` varchar(255) NOT NULL,  
81 ##   `Hotel_Phone` varchar(255) NOT NULL,  
82 ##   `Hotel_StreetNumber` varchar(255) NOT NULL,  
83 ##   `Hotel_StreetName` varchar(255) NOT NULL,  
84 ##   `Hotel_City` varchar(255) NOT NULL,
```

GROUP 7

```
85 ## `Hotel_State` varchar(255) NOT NULL,
86 ## `Hotel_PostalCode` varchar(255) NOT NULL,
87 ## PRIMARY KEY (`Hotel_Id`)
88 ## )
89 ## ROWS Fetched: 0 [complete]
90 ## Changed: 0

91 #Creating table customer and dropping the table if it already exists

92 RSQLite:::dbSendQuery(myconn, "DROP TABLE IF EXISTS `Guest`")

93 ## <SQLiteResult>
94 ##   SQL DROP TABLE IF EXISTS `Guest`
95 ##   ROWS Fetched: 0 [complete]
96 ##     Changed: 0

97 RSQLite:::dbSendQuery(myconn, "CREATE TABLE `Guest` (
98   `Guest_Id` int(11) NOT NULL,
99   `Guest_FirstName` varchar(255) NOT NULL,
100  `Guest_MiddleName` varchar(255) DEFAULT NULL,
101  `Guest_LastName` varchar(255) NOT NULL,
102  `Guest_Email` varchar(255) NOT NULL,
103  `Guest_WorkPhone` varchar(255) DEFAULT NULL,
104  `Guest_HomePhone` varchar(255) DEFAULT NULL,
105  `Guest_CellPhone` varchar(255) DEFAULT NULL,
106  `Guest_StreetNumber` varchar(255) NOT NULL,
107  `Guest_StreetName` varchar(255) NOT NULL,
108  `Guest_City` varchar(255) NOT NULL,
109  `Guest_State` varchar(255) NOT NULL,
110  `Guest_Country` varchar(255) NOT NULL,
111  `Guest_PostalCode` varchar(255) NOT NULL,
112  PRIMARY KEY (`Guest_Id`)
113)")

114 ## <SQLiteResult>
115 ##   SQL CREATE TABLE `Guest` (
116 ##     `Guest_Id` int(11) NOT NULL,
117 ##     `Guest_FirstName` varchar(255) NOT NULL,
118 ##     `Guest_MiddleName` varchar(255) DEFAULT NULL,
119 ##     `Guest_LastName` varchar(255) NOT NULL,
120 ##     `Guest_Email` varchar(255) NOT NULL,
121 ##     `Guest_WorkPhone` varchar(255) DEFAULT NULL,
122 ##     `Guest_HomePhone` varchar(255) DEFAULT NULL,
123 ##     `Guest_CellPhone` varchar(255) DEFAULT NULL,
124 ##     `Guest_StreetNumber` varchar(255) NOT NULL,
125 ##     `Guest_StreetName` varchar(255) NOT NULL,
126 ##     `Guest_City` varchar(255) NOT NULL,
127 ##     `Guest_State` varchar(255) NOT NULL,
128 ##     `Guest_Country` varchar(255) NOT NULL,
129 ##     `Guest_PostalCode` varchar(255) NOT NULL,
130 ##     PRIMARY KEY (`Guest_Id`)
131 ##   )
132 ##   ROWS Fetched: 0 [complete]
133 ##     Changed: 0
```

GROUP 7

```
134 #Creating table channel and dropping the table if it already exists
135 RSQLite:::dbSendQuery(myconn, "DROP TABLE IF EXISTS `Channel`")
136 ## <SQLiteResult>
137 ##   SQL  DROP TABLE IF EXISTS `Channel`
138 ##   ROWS Fetched: 0 [complete]
139 ##       Changed: 0
140 RSQLite:::dbSendQuery(myconn, "CREATE TABLE `Channel` (
141   `Channel_Id` int(11) NOT NULL,
142   `Channel_Name` varchar(255) NOT NULL,
143   `Hotel_Id` int(11) NULL DEFAULT NULL,
144   `Channel_BookingFee` decimal(8, 2) NOT NULL,
145   PRIMARY KEY (`Channel_Id`),
146   CONSTRAINT `fk_1` FOREIGN KEY (`Hotel_Id`) REFERENCES `Hotel` (`Hotel_Id`
147 ``)
148 )")
149 ## <SQLiteResult>
150 ##   SQL  CREATE TABLE `Channel` (
151 ##     `Channel_Id` int(11) NOT NULL,
152 ##     `Channel_Name` varchar(255) NOT NULL,
153 ##     `Hotel_Id` int(11) NULL DEFAULT NULL,
154 ##     `Channel_BookingFee` decimal(8, 2) NOT NULL,
155 ##     PRIMARY KEY (`Channel_Id`),
156 ##     CONSTRAINT `fk_1` FOREIGN KEY (`Hotel_Id`) REFERENCES `Hotel` (`Hotel
157 _Id`)
158 )
159 ##       ROWS Fetched: 0 [complete]
160 ##           Changed: 0
161 #Creating table GuestTransaction and dropping the table if it already exists
162 RSQLite:::dbSendQuery(myconn, "DROP TABLE IF EXISTS `GuestTransaction`");
163 ## <SQLiteResult>
164 ##   SQL  DROP TABLE IF EXISTS `GuestTransaction`
165 ##   ROWS Fetched: 0 [complete]
166 ##       Changed: 0
167 RSQLite:::dbSendQuery(myconn, "CREATE TABLE `GuestTransaction` (
168   `Trans_Id` int(11) NOT NULL,
169   `Guest_Id` int(11) NULL DEFAULT NULL,
170   `Trans_CCNumber` VARCHAR(200) NULL DEFAULT NULL,
171   `Trans_CCIssueDate` VARCHAR(200) NULL DEFAULT NULL,
172   `TD_Id` int(11) NULL DEFAULT NULL,
173   `Trans_CCExpDate` varchar(200) NULL DEFAULT NULL,
174   `Trans_TotalPayment` decimal(8, 2) NULL DEFAULT NULL,
175   `Trans_Date` date NULL DEFAULT NULL,
176   `Trans_Tax` date NULL DEFAULT NULL,
177   PRIMARY KEY (`Trans_Id`),
178   CONSTRAINT `fk1` FOREIGN KEY (`Guest_Id`) REFERENCES `customer` (`Guest_
179 Id`),
180   CONSTRAINT `fk15` FOREIGN KEY (`TD_Id`) REFERENCES `transaction_Descript
```

GROUP 7

```
181 ion` (`TD_Id`)
182 ");
183 ## <SQLiteResult>
184 ##   SQL CREATE TABLE `GuestTransaction` (
185 ##     `Trans_Id` int(11) NOT NULL,
186 ##     `Guest_Id` int(11) NULL DEFAULT NULL,
187 ##     `Trans_CCCNumber` VARCHAR(200) NULL DEFAULT NULL,
188 ##     `Trans_CCIssueDate` VARCHAR(200) NULL DEFAULT NULL,
189 ##     `TD_Id` int(11) NULL DEFAULT NULL,
190 ##     `Trans_CCExpDate` varchar(200) NULL DEFAULT NULL,
191 ##     `Trans_TotalPayment` decimal(8, 2) NULL DEFAULT NULL,
192 ##     `Trans_Date` date NULL DEFAULT NULL,
193 ##     `Trans_Tax` date NULL DEFAULT NULL,
194 ##     PRIMARY KEY (`Trans_Id`),
195 ##     CONSTRAINT `fk1` FOREIGN KEY (`Guest_Id`) REFERENCES `customer` (`Guest_Id`),
196 ##     CONSTRAINT `fk15` FOREIGN KEY (`TD_Id`) REFERENCES `transaction_Description` (`TD_Id`)
197 ##   )
198 ##   ROWS Fetched: 0 [complete]
199 ##           Changed: 0
200 #Creating table Transaction_Description and dropping the table if it already exists
201
202 RSQLite:::dbSendQuery(myconn, "DROP TABLE IF EXISTS `Transaction_Description`");
203 ## <SQLiteResult>
204 ##   SQL DROP TABLE IF EXISTS `Transaction_Description`
205 ##   ROWS Fetched: 0 [complete]
206 ##           Changed: 0
207
208 RSQLite:::dbSendQuery(myconn, "CREATE TABLE `Transaction_Description` (
209   `TD_Id` int(11) NOT NULL,
210   `Trans_Description` varchar(255) NOT NULL,
211   PRIMARY KEY (`TD_Id`)
212 )");
213 ## <SQLiteResult>
214 ##   SQL CREATE TABLE `Transaction_Description` (
215 ##     `TD_Id` int(11) NOT NULL,
216 ##     `Trans_Description` varchar(255) NOT NULL,
217 ##     PRIMARY KEY (`TD_Id`)
218 ##   )
219 ##   ROWS Fetched: 0 [complete]
220 ##           Changed: 0
221 #Creating table invoice and dropping the table if it already exists
222
223 RSQLite:::dbSendQuery(myconn, "DROP TABLE IF EXISTS `Invoice`");
224 ## <SQLiteResult>
225 ##   SQL DROP TABLE IF EXISTS `Invoice`
226 ##   ROWS Fetched: 0 [complete]
227 ##           Changed: 0
```

GROUP 7

```
228 RSQLite:::dbSendQuery(myconn, "CREATE TABLE `Invoice` ("
229   `Invoice_Id` int(11) NOT NULL,
230   `Trans_Id` int(11) NULL DEFAULT NULL,
231   `Guest_ID` int(11) NULL DEFAULT NULL,
232   `Invoice_Name` varchar(255) DEFAULT NULL,
233   `Invoice_TotalAmount` decimal(8, 2) NOT NULL,
234   `Invoice_TotalBalance` decimal(8, 2) NOT NULL,
235   `Invoice_OtherCharges` decimal(8, 2) NULL DEFAULT NULL,
236   `Invoice_Tax` decimal(8, 2) NOT NULL,
237   `Invoice_Date` date NOT NULL,
238   PRIMARY KEY (`Invoice_Id`),
239   CONSTRAINT `fk5` FOREIGN KEY (`Trans_Id`) REFERENCES `transaction` (`Tr
240 ns_Id`),
241   CONSTRAINT `fk6` FOREIGN KEY (`Guest_ID`) REFERENCES `customer` (`Guest_
242 Id`)
243 )");
244 ## <SQLiteResult>
245 ##   SQL  CREATE TABLE `Invoice` (
246 ##     `Invoice_Id` int(11) NOT NULL,
247 ##     `Trans_Id` int(11) NULL DEFAULT NULL,
248 ##     `Guest_ID` int(11) NULL DEFAULT NULL,
249 ##     `Invoice_Name` varchar(255) DEFAULT NULL,
250 ##     `Invoice_TotalAmount` decimal(8, 2) NOT NULL,
251 ##     `Invoice_TotalBalance` decimal(8, 2) NOT NULL,
252 ##     `Invoice_OtherCharges` decimal(8, 2) NULL DEFAULT NULL,
253 ##     `Invoice_Tax` decimal(8, 2) NOT NULL,
254 ##     `Invoice_Date` date NOT NULL,
255 ##     PRIMARY KEY (`Invoice_Id`),
256 ##     CONSTRAINT `fk5` FOREIGN KEY (`Trans_Id`) REFERENCES `transaction` (`
257 Trans_Id`),
258 ##     CONSTRAINT `fk6` FOREIGN KEY (`Guest_ID`) REFERENCES `customer` (`Gue
259 st_Id`)
260 ## )
261 ##   ROWS Fetched: 0 [complete]
262 ##           Changed: 0
263 #Creating table reservation and dropping the table if it already exists
264 RSQLite:::dbSendQuery(myconn, "DROP TABLE IF EXISTS `Reservation`");
265 ## <SQLiteResult>
266 ##   SQL  DROP TABLE IF EXISTS `Reservation`
267 ##   ROWS Fetched: 0 [complete]
268 ##           Changed: 0
269 RSQLite:::dbSendQuery(myconn, "CREATE TABLE `Reservation` (
270   `Reservation_Id` int(11) NOT NULL,
271   `Hotel_Id` int(11) NULL DEFAULT NULL,
272   `Guest_Id` int(11) NULL DEFAULT NULL,
273   `Channel_Id` int(11) NULL DEFAULT NULL,
274   `Reservation_ArrivalDate` date NOT NULL,
275   `Reservation_DeptDate` date NOT NULL,
276   `Trans_Id` int(11) NULL DEFAULT NULL,
277   `Reservation_Preference` varchar(255) DEFAULT NULL,
```

GROUP 7

```
278 `Reservation_AddService` varchar(255) DEFAULT NULL,  
279 `Reservation_Facilities` varchar(255) DEFAULT NULL,  
280 `Reservation_Status` varchar(255) NOT NULL,  
281 `Reservation_CCNumber` bigint(20) NOT NULL,  
282 `Reservation_CCIssueDate` varchar(255) DEFAULT NULL,  
283 `Reservation_CCExpDate` varchar(255) NOT NULL,  
284 `Reservation_NumberOfGuests` int(11) NULL DEFAULT NULL,  
285 PRIMARY KEY (`Reservation_Id`),  
286 CONSTRAINT `fk7` FOREIGN KEY (`Hotel_Id`) REFERENCES `hotel` (`Hotel_Id`)  
287 ),  
288 CONSTRAINT `fk8` FOREIGN KEY (`Guest_Id`) REFERENCES `customer` (`Guest_`  
289 Id`),  
290 CONSTRAINT `fk9` FOREIGN KEY (`Channel_Id`) REFERENCES `channel` (`Chann  
291 el_Id`),  
292 CONSTRAINT `fk10` FOREIGN KEY (`Trans_Id`) REFERENCES `transaction` (`Tr  
293 ans_Id`)  
294 );  
  
295 ## <SQLiteResult>  
296 ##   SQL  CREATE TABLE `Reservation` (  
297 ##     `Reservation_Id` int(11) NOT NULL,  
298 ##     `Hotel_Id` int(11) NULL DEFAULT NULL,  
299 ##     `Guest_Id` int(11) NULL DEFAULT NULL,  
300 ##     `Channel_Id` int(11) NULL DEFAULT NULL,  
301 ##     `Reservation_ArrivalDate` date NOT NULL,  
302 ##     `Reservation_DeptDate` date NOT NULL,  
303 ##     `Trans_Id` int(11) NULL DEFAULT NULL,  
304 ##     `Reservation_Preference` varchar(255) DEFAULT NULL,  
305 ##     `Reservation_AddService` varchar(255) DEFAULT NULL,  
306 ##     `Reservation_Facilities` varchar(255) DEFAULT NULL,  
307 ##     `Reservation_Status` varchar(255) NOT NULL,  
308 ##     `Reservation_CCNumber` bigint(20) NOT NULL,  
309 ##     `Reservation_CCIssueDate` varchar(255) DEFAULT NULL,  
310 ##     `Reservation_CCExpDate` varchar(255) NOT NULL,  
311 ##     `Reservation_NumberOfGuests` int(11) NULL DEFAULT NULL,  
312 ##     PRIMARY KEY (`Reservation_Id`),  
313 ##     CONSTRAINT `fk7` FOREIGN KEY (`Hotel_Id`) REFERENCES `hotel` (`Hotel_`  
314 Id`),  
315 ##     CONSTRAINT `fk8` FOREIGN KEY (`Guest_Id`) REFERENCES `customer` (`Gue  
316 st_Id`),  
317 ##     CONSTRAINT `fk9` FOREIGN KEY (`Channel_Id`) REFERENCES `channel` (`Ch  
318 annel_Id`),  
319 ##     CONSTRAINT `fk10` FOREIGN KEY (`Trans_Id`) REFERENCES `transaction` (  
320 `Trans_Id`)  
321 ## )  
322 ##   ROWS Fetched: 0 [complete]  
323 ##           Changed: 0  
  
324 #Creating table room and dropping the table if it already exists  
  
325 RSQLite:::dbSendQuery(myconn, "DROP TABLE IF EXISTS `Room`");  
  
326 ## <SQLiteResult>  
327 ##   SQL  DROP TABLE IF EXISTS `Room`
```

GROUP 7

```
328 ##  ROWS Fetched: 0 [complete]
329 ##      Changed: 0
330 RSQLite:::dbSendQuery(myconn, "CREATE TABLE `Room` (
331   `Room_Id` int(11) NOT NULL,
332   `Reservation_Id` int(11) NULL DEFAULT NULL,
333   `Room_No` int(11) NULL DEFAULT NULL,
334   `Room_Name` varchar(255) DEFAULT NULL,
335   `Room_Beds` int(11) NOT NULL,
336   `Room_IsSmokingArea` Boolean,
337   `Room_Floor` varchar(255) NOT NULL,
338   `Room_Rate` decimal(8, 2) NOT NULL,
339   `Room_Availability` Boolean,
340   `Hotel_Id` int(11) NULL DEFAULT NULL,
341   PRIMARY KEY (`Room_Id`),
342   CONSTRAINT `fk11` FOREIGN KEY (`Reservation_Id`) REFERENCES `reservation
343   (`Reservation_Id`),
344   CONSTRAINT `fk12` FOREIGN KEY (`Hotel_Id`) REFERENCES `hotel` (`Hotel_Id
345   `)
346   )");
347 ## <SQLiteResult>
348 ##   SQL CREATE TABLE `Room` (
349 ##     `Room_Id` int(11) NOT NULL,
350 ##     `Reservation_Id` int(11) NULL DEFAULT NULL,
351 ##     `Room_No` int(11) NULL DEFAULT NULL,
352 ##     `Room_Name` varchar(255) DEFAULT NULL,
353 ##     `Room_Beds` int(11) NOT NULL,
354 ##     `Room_IsSmokingArea` Boolean,
355 ##     `Room_Floor` varchar(255) NOT NULL,
356 ##     `Room_Rate` decimal(8, 2) NOT NULL,
357 ##     `Room_Availability` Boolean,
358 ##     `Hotel_Id` int(11) NULL DEFAULT NULL,
359 ##     PRIMARY KEY (`Room_Id`),
360 ##     CONSTRAINT `fk11` FOREIGN KEY (`Reservation_Id`) REFERENCES `reservation
361   (`Reservation_Id`),
362 ##     CONSTRAINT `fk12` FOREIGN KEY (`Hotel_Id`) REFERENCES `hotel` (`Hotel
363   _Id`)
364   )
365 ##  ROWS Fetched: 0 [complete]
366 ##      Changed: 0
367 #Creating table guests and dropping the table if it already exists
368 RSQLite:::dbSendQuery(myconn, "DROP TABLE IF EXISTS `AdditionalGuest`");
369 ## <SQLiteResult>
370 ##   SQL DROP TABLE IF EXISTS `AdditionalGuest`
371 ##   ROWS Fetched: 0 [complete]
372 ##      Changed: 0
373 RSQLite:::dbSendQuery(myconn, "CREATE TABLE `AdditionalGuest` (
374   `AddGuest_Id` int(11) NOT NULL,
375   `Reservation_Id` int(11) NULL DEFAULT NULL,
376   `AddGuest_FirstName` varchar(255) DEFAULT NULL,
```

GROUP 7

```
377 `AddGuest_MiddleName` varchar(255) DEFAULT NULL,  
378 `AddGuest_LastName` varchar(255) DEFAULT NULL,  
379 `Room_id` int(11) NULL DEFAULT NULL,  
380 `AddGuest_DOA` varchar(255) DEFAULT NULL,  
381 `AddGuest_DOD` varchar(255) DEFAULT NULL,  
382 PRIMARY KEY (`AddGuest_Id`),  
383 CONSTRAINT `fk13` FOREIGN KEY (`Reservation_Id`) REFERENCES `reservation`  
(`Reservation_Id`),  
384 CONSTRAINT `fk14` FOREIGN KEY (`Room_Id`) REFERENCES `room`(`Room_Id`)  
");  
  
387 ## <SQLiteResult>  
388 ##   SQL  CREATE TABLE `AdditionalGuest`  (  
389 ##     `AddGuest_Id` int(11) NOT NULL,  
390 ##     `Reservation_Id` int(11) NULL DEFAULT NULL,  
391 ##     `AddGuest_FirstName` varchar(255) DEFAULT NULL,  
392 ##     `AddGuest_MiddleName` varchar(255) DEFAULT NULL,  
393 ##     `AddGuest_LastName` varchar(255) DEFAULT NULL,  
394 ##     `Room_id` int(11) NULL DEFAULT NULL,  
395 ##     `AddGuest_DOA` varchar(255) DEFAULT NULL,  
396 ##     `AddGuest_DOD` varchar(255) DEFAULT NULL,  
397 ##     PRIMARY KEY (`AddGuest_Id`),  
398 ##     CONSTRAINT `fk13` FOREIGN KEY (`Reservation_Id`) REFERENCES `reservation`  
399 ##     (`Reservation_Id`),  
400 ##     CONSTRAINT `fk14` FOREIGN KEY (`Room_Id`) REFERENCES `room`(`Room_Id`)  
401 ##   )  
402 ##   ROWS Fetched: 0 [complete]  
403 ##           Changed: 0  
  
405 ##list number of tables in the hotel management database  
  
406 RSQLite:::dbListTables(myconn)  
  
407 ## [1] "AdditionalGuest"          "Channel"  
408 ## [3] "Guest"                  "GuestTransaction"  
409 ## [5] "Hotel"                  "Invoice"  
410 ## [7] "Reservation"           "Room"  
411 ## [9] "Transaction_Description"  
  
412 ##Insert into Channel table  
  
413 RSQLite:::dbSendQuery(myconn, "INSERT INTO Channel VALUES  
414 (1,'booking.com',1,'50.00')")  
  
415 ## <SQLiteResult>  
416 ##   SQL  INSERT INTO Channel VALUES  
417 ## (1,'booking.com',1,'50.00')  
418 ##   ROWS Fetched: 0 [complete]  
419 ##           Changed: 1  
  
420 RSQLite:::dbSendQuery(myconn, "INSERT INTO Channel VALUES  
421 (2,'alibaba.com',1,'60.00')")  
  
422 ## <SQLiteResult>  
423 ##   SQL  INSERT INTO Channel VALUES
```

GROUP 7

```
424 ## (2,'alibaba.com',1,'60.00')
425 ##   ROWS Fetched: 0 [complete]
426 ##     Changed: 1
427 RSQLite:::dbSendQuery(myconn, "INSERT INTO Channel VALUES
428 (3,'paytm.com',1,'55.00')")
429 ## <SQLiteResult>
430 ##   SQL  INSERT INTO Channel VALUES
431 ## (3,'paytm.com',1,'55.00')
432 ##   ROWS Fetched: 0 [complete]
433 ##     Changed: 1
434 RSQLite:::dbSendQuery(myconn, "INSERT INTO Channel  VALUES
435 (4,'goibibo',1,'45.00')")
436 ## <SQLiteResult>
437 ##   SQL  INSERT INTO Channel  VALUES
438 ## (4,'goibibo',1,'45.00')
439 ##   ROWS Fetched: 0 [complete]
440 ##     Changed: 1
441 RSQLite:::dbSendQuery(myconn, "INSERT INTO Channel VALUES
442 (5,'makemytrip',1,'90.00')")
443 ## <SQLiteResult>
444 ##   SQL  INSERT INTO Channel VALUES
445 ## (5,'makemytrip',1,'90.00')
446 ##   ROWS Fetched: 0 [complete]
447 ##     Changed: 1
448 RSQLite:::dbSendQuery(myconn, "INSERT INTO Channel VALUES
449 (6,'wizz',1,'59.00')")
450 ## <SQLiteResult>
451 ##   SQL  INSERT INTO Channel VALUES
452 ## (6,'wizz',1,'59.00')
453 ##   ROWS Fetched: 0 [complete]
454 ##     Changed: 1
455 ##Insert into Guest table
456 RSQLite:::dbSendQuery(myconn, "INSERT INTO Guest VALUES
457 (1,'Dushant','H.', 'Gohri', 'abc@gmail.com', '634872932', '', '', 'heronbank', 'g
458 ibbet hill road', 'coventry', 'coventry', 'United Kingdom', 'CV47ES')")
459 ## <SQLiteResult>
460 ##   SQL  INSERT INTO Guest VALUES
461 ## (1,'Dushant','H.', 'Gohri', 'abc@gmail.com', '634872932', '', '', 'heronbank'
462 ## , 'gibbet hill road', 'coventry', 'coventry', 'United Kingdom', 'CV47ES')
463 ##   ROWS Fetched: 0 [complete]
464 ##     Changed: 1
465 RSQLite:::dbSendQuery(myconn, "INSERT INTO Guest VALUES
466 (2,'Rashi','H.', 'singh', 'singh@gmail.com', '634832932', '', '', 'House Number
467 -1256', 'Sector -14', 'Sonipat', 'Haryana', 'India', 'CV47AL')")
```

GROUP 7

```
468 ## <SQLiteResult>
469 ##   SQL INSERT INTO Guest VALUES
470 ## (2,'Rashi','H.', 'singh', 'singh@gmail.com', '634832932', '', '', 'House Num-
471 ## ber -1256', 'Sector -14', 'Sonipat', 'Haryana', 'India', 'CV47AL')
472 ##   ROWS Fetched: 0 [complete]
473 ##       Changed: 1

474 RSQLite:::dbSendQuery(myconn, "INSERT INTO Guest VALUES
475 (3,'Tanvi','H.', 'Sharan', 'sharan@gmail.com', '634823321', '', '', 'heronbank',
476 'gibbet hill road', 'coventry', 'coventry', 'United Kingdom', 'FV47GS')")

477 ## <SQLiteResult>
478 ##   SQL INSERT INTO Guest VALUES
479 ## (3,'Tanvi','H.', 'Sharan', 'sharan@gmail.com', '634823321', '', '', 'heronban-
480 ## k', 'gibbet hill road', 'coventry', 'coventry', 'United Kingdom', 'FV47GS')
481 ##   ROWS Fetched: 0 [complete]
482 ##       Changed: 1

483 RSQLite:::dbSendQuery(myconn, "INSERT INTO Guest VALUES
484 (4,'rohan','G.', 'harkara', 'BIY@gmail.com', '6348729812', '', '', 'LAKESIDE', '5
485 TH AVENUE', 'coventry', 'coventry', 'Australia', 'HU7ES')")

486 ## <SQLiteResult>
487 ##   SQL INSERT INTO Guest VALUES
488 ## (4,'rohan','G.', 'harkara', 'BIY@gmail.com', '6348729812', '', '', 'LAKESIDE'
489 ## , '5TH AVENUE', 'coventry', 'coventry', 'Australia', 'HU7ES')
490 ##   ROWS Fetched: 0 [complete]
491 ##       Changed: 1

492 RSQLite:::dbSendQuery(myconn, "INSERT INTO Guest VALUES
493 (5,'MANSI','H.', 'MAYANI', 'MAYANI@gmail.com', '634872932', '', '', 'BLUEBELL',
494 'COUNTRYSIDE', 'coventry', 'coventry', 'AUSTRALIA', 'AS47ES')")

495 ## <SQLiteResult>
496 ##   SQL INSERT INTO Guest VALUES
497 ## (5,'MANSI','H.', 'MAYANI', 'MAYANI@gmail.com', '634872932', '', '', 'BLUEBELL
498 ## ', 'COUNTRYSIDE', 'coventry', 'coventry', 'AUSTRALIA', 'AS47ES')
499 ##   ROWS Fetched: 0 [complete]
500 ##       Changed: 1

501 ##Insert into GuestTransaction table

502 RSQLite:::dbSendQuery(myconn, "insert into GuestTransaction values (1, 1,
503 '1575778667654323', '5/29/2020', 1, '12/4/2021', '408889.53', '12/16/2019',
504 '200');")

505 ## <SQLiteResult>
506 ##   SQL insert into GuestTransaction values (1, 1, '1575778667654323', '1
507 ## 5/29/2020', 1, '12/4/2021', '408889.53', '12/16/2019', '200');
508 ##   ROWS Fetched: 0 [complete]
509 ##       Changed: 1

510 RSQLite:::dbSendQuery(myconn, "insert into GuestTransaction values (2, 2,
511 '1575778667254323', '5/29/2020', 1, '12/4/2022', '1089.3', '13/16/2019',
512 '310');")
```

GROUP 7

```
513 ## <SQLiteResult>
514 ##   SQL insert into GuestTransaction values (2, 2, '1575778667254323', '5/29/2020',1, '12/4/2022', '1089.3', '13/16/2019', '310');
515 ##   ROWS Fetched: 0 [complete]
516 ##           Changed: 1
517
518 RSQLite:::dbSendQuery(myconn, "insert into GuestTransaction values (3, 3, '5475778667654323', '5/29/2020',1, '18/3/2022', '75889.51', '14/16/2019', '30');");
519
520
521 ## <SQLiteResult>
522 ##   SQL insert into GuestTransaction values (3, 3, '5475778667654323', '5/29/2020',1, '18/3/2022', '75889.51', '14/16/2019', '30');
523 ##   ROWS Fetched: 0 [complete]
524 ##           Changed: 1
525
526 RSQLite:::dbSendQuery(myconn, "insert into GuestTransaction values (4, 4, '9875778667654323', '5/29/2020',1, '19/5/2030', '8889', '15/16/2019', '360');");
527
528
529 ## <SQLiteResult>
530 ##   SQL insert into GuestTransaction values (4, 4, '9875778667654323', '5/29/2020',1, '19/5/2030', '8889', '15/16/2019', '360');
531 ##   ROWS Fetched: 0 [complete]
532 ##           Changed: 1
533
534 RSQLite:::dbSendQuery(myconn, "insert into GuestTransaction values (5, 1, '1575778667654323', '5/29/2020',1, '30/4/2020', '43889.53', '12/16/2019', '700');");
535
536
537 ## <SQLiteResult>
538 ##   SQL insert into GuestTransaction values (5, 1, '1575778667654323', '5/29/2020',1, '30/4/2020', '43889.53', '12/16/2019', '700');
539 ##   ROWS Fetched: 0 [complete]
540 ##           Changed: 1
541
542 RSQLite:::dbSendQuery(myconn, "insert into GuestTransaction values (6, 1, '1575778667654323', '5/29/2020',1, '30/4/2020', '30888.53', '13/16/2019', '1000');");
543
544
545 ## <SQLiteResult>
546 ##   SQL insert into GuestTransaction values (6, 1, '1575778667654323', '5/29/2020',1, '30/4/2020', '30888.53', '13/16/2019', '1000');
547 ##   ROWS Fetched: 0 [complete]
548 ##           Changed: 1
549
550 ##Insert into hotel table
551 RSQLite:::dbSendQuery(myconn, "INSERT INTO Hotel VALUES
552 (1,'abc hotels','https://www.abchotels.com/','6786785432','4th avenue','hill road','new street','sulihull','bhc g09')")
553
554 ## <SQLiteResult>
555 ##   SQL INSERT INTO Hotel VALUES
556 ## (1,'abc hotels','https://www.abchotels.com/','6786785432','4th avenue','hill road','new street','sulihull','bhc g09')
557
```

GROUP 7

```
558 ##     ROWS Fetched: 0 [complete]
559 ##             Changed: 1
560 RSQLite:::dbSendQuery(myconn, "INSERT INTO Hotel VALUES
561 (2,'def hotels','https://www.defhotels.com/','678672312','4th kingston','h
562 ill road','city center','spa','bhg g34')")
563 ## <SQLiteResult>
564 ##   SQL  INSERT INTO Hotel VALUES
565 ## (2,'def hotels','https://www.defhotels.com/','678672312','4th kingston'
566 ## , 'hill road','city center','spa','bhg g34')
567 ##   ROWS Fetched: 0 [complete]
568 ##             Changed: 1
569 RSQLite:::dbSendQuery(myconn, "INSERT INTO Hotel VALUES
570 (3,'geh hotels','https://www.gehhotels.com/','6786784534','4th algebra','h
571 ill road','wbs','london','bhg g84')")
572 ## <SQLiteResult>
573 ##   SQL  INSERT INTO Hotel VALUES
574 ## (3,'geh hotels','https://www.gehhotels.com/','6786784534','4th algebra'
575 ## , 'hill road','wbs','london','bhg g84')
576 ##   ROWS Fetched: 0 [complete]
577 ##             Changed: 1
578 RSQLite:::dbSendQuery(myconn, "INSERT INTO Hotel VALUES
579 (4,'hij hotels','https://www.hijhotels.com/','6786897898','4th road','hill
580 road','art center','warwick','bhg g56')")
581 ## <SQLiteResult>
582 ##   SQL  INSERT INTO Hotel VALUES
583 ## (4,'hij hotels','https://www.hijhotels.com/','6786897898','4th road','h
584 ## ill road','art center','warwick','bhg g56')
585 ##   ROWS Fetched: 0 [complete]
586 ##             Changed: 1
587 RSQLite:::dbSendQuery(myconn, "INSERT INTO Hotel VALUES
588 (5,'klm hotels','https://www.klmhotels.com/','678623421','4th block','hill
589 road','sulihull','coventry','bhg g23')")
590 ## <SQLiteResult>
591 ##   SQL  INSERT INTO Hotel VALUES
592 ## (5,'klm hotels','https://www.klmhotels.com/','678623421','4th block','h
593 ## ill road','sulihull','coventry','bhg g23')
594 ##   ROWS Fetched: 0 [complete]
595 ##             Changed: 1
596 ##Insert into Transaction_Description table
597 RSQLite:::dbSendQuery(myconn, "INSERT INTO Transaction_Description VALUES (
598 1,'Cash')")
599 ## <SQLiteResult>
600 ##   SQL  INSERT INTO Transaction_Description VALUES (1,'Cash')
601 ##   ROWS Fetched: 0 [complete]
602 ##             Changed: 1
```

GROUP 7

```
603 RSQLite:::dbSendQuery(myconn, "INSERT INTO Transaction_Description VALUES (2, 'Bar')")  
604 ## <SQLiteResult>  
605 ##   SQL  INSERT INTO Transaction_Description VALUES (2, 'Bar')  
606 ##   ROWS Fetched: 0 [complete]  
607 ##           Changed: 1  
608  
609 RSQLite:::dbSendQuery(myconn, "INSERT INTO Transaction_Description VALUES (3, 'Seminar Hall')")  
610 ## <SQLiteResult>  
611 ##   SQL  INSERT INTO Transaction_Description VALUES (3, 'Seminar Hall')  
612 ##   ROWS Fetched: 0 [complete]  
613 ##           Changed: 1  
614  
615 RSQLite:::dbSendQuery(myconn, "INSERT INTO Transaction_Description VALUES (4, 'Banquet')")  
616 ## <SQLiteResult>  
617 ##   SQL  INSERT INTO Transaction_Description VALUES (4, 'Banquet')  
618 ##   ROWS Fetched: 0 [complete]  
619 ##           Changed: 1  
620  
621 RSQLite:::dbSendQuery(myconn, "INSERT INTO Transaction_Description VALUES (5, 'Swimming pool')")  
622 ## <SQLiteResult>  
623 ##   SQL  INSERT INTO Transaction_Description VALUES (5, 'Swimming pool')  
624 ##   ROWS Fetched: 0 [complete]  
625 ##           Changed: 1  
626  
627 RSQLite:::dbSendQuery(myconn, "INSERT INTO Transaction_Description VALUES (6, 'Spa')")  
628 ## <SQLiteResult>  
629 ##   SQL  INSERT INTO Transaction_Description VALUES (6, 'Spa')  
630 ##   ROWS Fetched: 0 [complete]  
631 ##           Changed: 1  
632  
633 ##Insert into Invoice table  
634 RSQLite:::dbSendQuery(myconn, "INSERT INTO Invoice VALUES (1,1,1,'dushant gohri',40,0,4,6,'2020-11-15')");  
635  
636 ## <SQLiteResult>  
637 ##   SQL  INSERT INTO Invoice VALUES  
638 ## (1,1,1,'dushant gohri',40,0,4,6,'2020-11-15')  
639 ##   ROWS Fetched: 0 [complete]  
640 ##           Changed: 1  
641  
642 RSQLite:::dbSendQuery(myconn, "INSERT INTO Invoice VALUES (2,2,2,'rashi gohri',140,0,4,6,'2020-11-15')");  
643 ## <SQLiteResult>  
644 ##   SQL  INSERT INTO Invoice VALUES  
645 ## (2,2,2,'rashi gohri',140,0,4,6,'2020-11-15')
```

GROUP 7

```
646 ##  ROWS Fetched: 0 [complete]
647 ##          Changed: 1
648 RSQLite:::dbSendQuery(myconn, "INSERT INTO Invoice VALUES
649 (3,1,1,'dushant gohri',40,0,4,6,'2020-11-15')");
650 ## <SQLiteResult>
651 ##   SQL  INSERT INTO Invoice VALUES
652 ## (3,1,1,'dushant gohri',40,0,4,6,'2020-11-15')
653 ##   ROWS Fetched: 0 [complete]
654 ##          Changed: 1
655 RSQLite:::dbSendQuery(myconn, "INSERT INTO Invoice VALUES
656 (4,1,1,'dushant gohri',40,0,4,6,'2020-11-15')");
657 ## <SQLiteResult>
658 ##   SQL  INSERT INTO Invoice VALUES
659 ## (4,1,1,'dushant gohri',40,0,4,6,'2020-11-15')
660 ##   ROWS Fetched: 0 [complete]
661 ##          Changed: 1
662 RSQLite:::dbSendQuery(myconn, "INSERT INTO Invoice VALUES
663 (5,1,1,'dushant gohri',40,0,4,6,'2020-11-15')");
664 ## <SQLiteResult>
665 ##   SQL  INSERT INTO Invoice VALUES
666 ## (5,1,1,'dushant gohri',40,0,4,6,'2020-11-15')
667 ##   ROWS Fetched: 0 [complete]
668 ##          Changed: 1
669 RSQLite:::dbSendQuery(myconn, "INSERT INTO Invoice VALUES
670 (6,1,1,'dushant gohri',40,0,4,6,'2020-11-15')");
671 ## <SQLiteResult>
672 ##   SQL  INSERT INTO Invoice VALUES
673 ## (6,1,1,'dushant gohri',40,0,4,6,'2020-11-15')
674 ##   ROWS Fetched: 0 [complete]
675 ##          Changed: 1
676 ##Insert into Reservation table
677 RSQLite:::dbSendQuery(myconn, "insert into Reservation values (1, 1, 1, 1,
678 '28/8/2020', '30/8/2020', 1, 'ut volutpat', 'non ligula', 'eu est congue',
679 'eget semper', 5576864946411916, '8/25/2020', '7/13/2020', 3)");
680 ## <SQLiteResult>
681 ##   SQL  insert into Reservation values (1, 1, 1, 1, '28/8/2020', '30/8/2
682 ## 020', 1, 'ut volutpat', 'non ligula', 'eu est congue', 'eget semper', 5576
683 ## 864946411916, '8/25/2020', '7/13/2020', 3)
684 ##   ROWS Fetched: 0 [complete]
685 ##          Changed: 1
686 RSQLite:::dbSendQuery(myconn, "insert into Reservation values (2, 2, 2, 2,
687 '15/8/2020', '18/8/2020', 2, 'posuere cubilia', '', 'metus', 'magna at nun
688 c', 1061500478900525, '3/4/2020', '12/16/2019', 4)")
```

GROUP 7

```
689 ## <SQLiteResult>
690 ##   SQL insert into Reservation values (2, 2, 2, 2, '15/8/2020', '18/8/
691 2020', 2, 'posuere cubilia', '', 'metus', 'magna at nunc', 106150047890052
692 5, '3/4/2020', '12/16/2019', 4)
693 ##   ROWS Fetched: 0 [complete]
694 ##       Changed: 1

695 RSQLite:::dbSendQuery(myconn, "insert into Reservation values (3, 3, 3, 3,
696 '25/8/2020', '28/8/2020', 3, 'posuere cubilia', '', 'metus', 'magna at nun
697 c', 1061500478900525, '3/4/2020', '12/16/2019', 4)")

698 ## <SQLiteResult>
699 ##   SQL insert into Reservation values (3, 3, 3, 3, '25/8/2020', '28/8/
700 2020', 3, 'posuere cubilia', '', 'metus', 'magna at nunc', 106150047890052
701 5, '3/4/2020', '12/16/2019', 4)
702 ##   ROWS Fetched: 0 [complete]
703 ##       Changed: 1

704 RSQLite:::dbSendQuery(myconn, "insert into Reservation values (4, 4, 4, 4,
705 '17/8/2020', '15/9/2020', 4, 'posuere cubilia', '', 'metus', 'magna at nun
706 c', 1061500478900525, '3/4/2020', '12/16/2019', 4)")

707 ## <SQLiteResult>
708 ##   SQL insert into Reservation values (4, 4, 4, 4, '17/8/2020', '15/9/
709 2020', 4, 'posuere cubilia', '', 'metus', 'magna at nunc', 106150047890052
710 5, '3/4/2020', '12/16/2019', 4)
711 ##   ROWS Fetched: 0 [complete]
712 ##       Changed: 1

713 RSQLite:::dbSendQuery(myconn, "insert into Reservation values (5, 5, 5, 5,
714 '18/8/2020', '25/3/2021', 5, 'posuere cubilia', '', 'metus', 'magna at nun
715 c', 1061500478900525, '3/4/2020', '12/16/2019', 4)")

716 ## <SQLiteResult>
717 ##   SQL insert into Reservation values (5, 5, 5, 5, '18/8/2020', '25/3/
718 2021', 5, 'posuere cubilia', '', 'metus', 'magna at nunc', 106150047890052
719 5, '3/4/2020', '12/16/2019', 4)
720 ##   ROWS Fetched: 0 [complete]
721 ##       Changed: 1

722 ##Insert into Room table

723 RSQLite:::dbSendQuery(myconn, "insert into Room values (1, 1, 12709804140,
724 'platea dictumst', 1, TRUE, 14, 36648, TRUE, 1)");

725 ## <SQLiteResult>
726 ##   SQL insert into Room values (1, 1, 12709804140, 'platea dictumst',
727 1, TRUE, 14, 36648, TRUE, 1)
728 ##   ROWS Fetched: 0 [complete]
729 ##       Changed: 1

730 RSQLite:::dbSendQuery(myconn, "insert into Room values (2, 2, 1270980414,
731 'platea dictumst', 2, TRUE, 14, 3665, TRUE, 2)");

732 ## <SQLiteResult>
733 ##   SQL insert into Room values (2, 2, 1270980414, 'platea dictumst', 2
```

GROUP 7

```
734 , TRUE, 14, 3665, TRUE, 2)
735 ##   ROWS Fetched: 0 [complete]
736 ##       Changed: 1

737 RSQLite:::dbSendQuery(myconn, "insert into Room values (3, 3, 127098040, 'platea dictumst', 2, TRUE, 14, 3623, TRUE, 3)");
738
739 ## <SQLiteResult>
740 ##   SQL insert into Room values (3, 3, 127098040, 'platea dictumst', 2, TRUE, 14, 3623, TRUE, 3)
741 ##   ROWS Fetched: 0 [complete]
742 ##       Changed: 1

744 RSQLite:::dbSendQuery(myconn, "insert into Room values (4, 4, 12709804, 'platea dictumst', 1, FALSE, 14, 3876, TRUE, 4)");
745
746 ## <SQLiteResult>
747 ##   SQL insert into Room values (4, 4, 12709804, 'platea dictumst', 1, FALSE, 14, 3876, TRUE, 4)
748 ##   ROWS Fetched: 0 [complete]
749 ##       Changed: 1

751 RSQLite:::dbSendQuery(myconn, "insert into Room values (5, 5, 12709840, 'platea dictumst', 1, TRUE, 14, 36975, TRUE, 5)");
752
753 ## <SQLiteResult>
754 ##   SQL insert into Room values (5, 5, 12709840, 'platea dictumst', 1, TRUE, 14, 36975, TRUE, 5)
755 ##   ROWS Fetched: 0 [complete]
756 ##       Changed: 1

758 ##Insert into AdditionalGuest table

759 RSQLite:::dbSendQuery(myconn, "insert into AdditionalGuest values (1, 1, 'Nickey', 'enim', 'Chaize', 1, '12/16/2019', '12/18/2019')");
760
761 ## <SQLiteResult>
762 ##   SQL insert into AdditionalGuest values (1, 1, 'Nickey', 'enim', 'Chaize', 1, '12/16/2019', '12/18/2019')
763 ##   ROWS Fetched: 0 [complete]
764 ##       Changed: 1

766 RSQLite:::dbSendQuery(myconn, "insert into AdditionalGuest values (2, 2, 'Nickey', 'enim', 'Chaize', 2, '12/16/2019', '12/18/2019')");
767
768 ## <SQLiteResult>
769 ##   SQL insert into AdditionalGuest values (2, 2, 'Nickey', 'enim', 'Chaize', 2, '12/16/2019', '12/18/2019')
770 ##   ROWS Fetched: 0 [complete]
771 ##       Changed: 1

773 RSQLite:::dbSendQuery(myconn, "insert into AdditionalGuest values (3, 3, 'Nickey', 'enim', 'Chaize', 3, '12/16/2019', '12/18/2019')");
774
775 ## <SQLiteResult>
776 ##   SQL insert into AdditionalGuest values (3, 3, 'Nickey', 'enim', 'Chaize', 3, '12/16/2019', '12/18/2019')
777
```

GROUP 7

```
778 ##  ROWS Fetched: 0 [complete]
779 ##          Changed: 1
780 RSQLite:::dbSendQuery(myconn, "insert into AdditionalGuest values (4, 4, 'N
781 ickey', 'enim', 'Chaize',4, '12/16/2019', '12/18/2019')");
782 ## <SQLiteResult>
783 ##   SQL insert into AdditionalGuest values (4, 4, 'Nickey', 'enim', 'Cha
784 ize',4, '12/16/2019', '12/18/2019')
785 ##  ROWS Fetched: 0 [complete]
786 ##          Changed: 1
787 RSQLite:::dbSendQuery(myconn, "insert into AdditionalGuest values (5, 5, 'N
788 ickey', 'enim', 'Chaize',5, '12/16/2019', '12/18/2019')");
789 ## <SQLiteResult>
790 ##   SQL insert into AdditionalGuest values (5, 5, 'Nickey', 'enim', 'Cha
791 ize',5, '12/16/2019', '12/18/2019')
792 ##  ROWS Fetched: 0 [complete]
793 ##          Changed: 1
794 RSQLite:::dbSendQuery(myconn, "insert into AdditionalGuest values (6, 6, 'N
795 ickey', 'enim', 'Chaize',5, '12/16/2019', '12/18/2019')");
796 ## <SQLiteResult>
797 ##   SQL insert into AdditionalGuest values (6, 6, 'Nickey', 'enim', 'Cha
798 ize',5, '12/16/2019', '12/18/2019')
799 ##  ROWS Fetched: 0 [complete]
800 ##          Changed: 1
801 #using select command on hotel table to see if the insert command works properly and
802 connection is established
803 RSQLite:::dbGetQuery(myconn, "SELECT * FROM `hotel`")
804 ##   Hotel_Id Hotel_name          Hotel_WebsiteUrl Hotel_Phone Hotel_Stre
805 etNumber
806 ## 1      1 abc hotels https://www.abchotels.com/  6786785432      4t
807 h avenue
808 ## 2      2 def hotels https://www.defhotels.com/  678672312      4th
809 kingston
810 ## 3      3 geh hotels https://www.gehhotels.com/  6786784534      4th
811 algebra
812 ## 4      4 hij hotels https://www.hijhotels.com/  6786897898
813 4th road
814 ## 5      5 klm hotels https://www.klmhotels.com/  678623421      4
815 th block
816 ##   Hotel_StreetName Hotel_City Hotel_State Hotel_PostalCode
817 ## 1      hill road new street    sulihull      bhg g09
818 ## 2      hill road city center    spa          bhg g34
819 ## 3      hill road      wbs    london      bhg g84
820 ## 4      hill road art center    warwick      bhg g56
821 ## 5      hill road    sulihull    coventry      bhg g23
822 #Disconnect to SQLite
```

GROUP 7

```
823 RSQLite:::dbDisconnect(myconn)
824
825 SQL queries
826 #Connect to SQLite
827 myconn <- RSQLite:::dbConnect(RSQLite:::SQLite(),"HotelManagement.db")
828 ##The total spent for the customer for a particular stay (checkout invoice). • we have used
829 two tables for storing transaction data, first the trnsaction table to store the customer
830 transaction in hotel while he stays, after that if he check out he/she will get the final invoice
831 containg all the transactions made at hotel. This sql query shows the total spent for each
832 customer for a particular stay, which is defined as a specific payment date. The name
833 combines first, midlle and last name of each customer. For example,
834 RSQLite:::dbGetQuery(myconn, "Select
835 g.Guest_FirstName,g.Guest_MiddleName,g.Guest_LastName,(select sum(Invoice_
836 TotalAmount) from Invoice where Guest_Id = g.Guest_Id) as totalSpent
837 from Guest g");
838 ##   Guest_FirstName Guest_MiddleName Guest_LastName totalSpent
839 ## 1          Dushant           H.        Gohri      200
840 ## 2          Rashi            H.       singh      140
841 ## 3          Tanvi            H.       Sharan      NA
842 ## 4          rohan           G.      harkara     NA
843 ## 5          MANSI           H.      MAYANI      NA
844 The most valuable customers in (a) the last two months, (b) past year and (c) from the
845 beginning of the records.
846 • Assuming the most valuable customers are the top 5 customers who spent most. The
847 name combines first, middle, and last name of each customer along with their spent
848 over past 2 months .
849 RSQLite:::dbGetQuery(myconn, "select
850 g.Guest_FirstName,g.Guest_MiddleName,g.Guest_LastName,sum(i.Invoice_TotalA
851 mount) as totalSpent
852 from Guest g inner join Invoice i on g.Guest_Id = i.Guest_Id
853 where i.Invoice_Date BETWEEN '2020-10-10' and '2020-12-10'
854 group by g.Guest_Id
855 order by sum(i.Invoice_TotalAmount) desc limit 5")
856 ##   Guest_FirstName Guest_MiddleName Guest_LastName totalSpent
857 ## 1          Dushant           H.        Gohri      200
858 ## 2          Rashi            H.       singh      140
859 b) Assuming the most valuable customers are the top 5 customers who spent most. The
860 name combines first, middle, and last name of each customer along with their spent
861 over 1 year
862 RSQLite:::dbGetQuery(myconn, "select
863 g.Guest_FirstName,g.Guest_MiddleName,g.Guest_LastName,sum(i.Invoice_TotalA
864 mount) as totalSpent
```

GROUP 7

```
865 from Guest g inner join Invoice i on g.Guest_Id = i.Guest_Id
866 where i.Invoice_Date BETWEEN '2019-12-10' and '2020-12-10'
867 group by g.Guest_Id
868 order by sum(i.Invoice_TotalAmount) desc limit 5")
869 ## Guest_FirstName Guest_MiddleName Guest_LastName totalSpent
870 ## 1 Dushant H. Gohri 200
871 ## 2 Rashi H. singh 140
872 (c) Assuming the most valuable customers are the top 5 customers who spent most. The
873 name combines first, middle, and last name of each customer along with their spent
874 from the beginning of the records..
875 RSQLite:::dbGetQuery(myconn, "
876 select
877 g.Guest_FirstName,g.Guest_MiddleName,g.Guest_LastName,sum(i.Invoice_TotalA
878 mount) as totalSpent
879 from Guest g inner join Invoice i on g.Guest_Id = i.Guest_Id
880 group by g.Guest_Id
881 order by sum(i.Invoice_TotalAmount) desc limit 5")
882 ## Guest_FirstName Guest_MiddleName Guest_LastName totalSpent
883 ## 1 Dushant H. Gohri 200
884 ## 2 Rashi H. singh 140
885 ##Which are the top countries where our customers come from ? • This SQL Query shows
886 top 10 countries where customers come from, ordered by number of people travelled from
887 these countries, Most of customers traveled from .....
888 RSQLite:::dbGetQuery(myconn, "
889 select Guest_Country, count(Guest_Id) from Guest group by Guest_Country
890 order by count(Guest_Id) desc limit 10")
891 ## Guest_Country count(Guest_Id)
892 ## 1 United Kingdom 2
893 ## 2 India 1
894 ## 3 Australia 1
895 ## 4 AUSTRALIA 1
896 ##How much did the hotel pay in referral fees for each of the platforms that we have
897 contracted with?
898 RSQLite:::dbGetQuery(myconn, "
899
900 select h.Hotel_name,c.Channel_Name,c.Channel_BookingFee
901 from Hotel h inner join Channel c on h.Hotel_Id = c.Hotel_Id")
902 ## Hotel_name Channel_Name Channel_BookingFee
903 ## 1 abc hotels booking.com 50
904 ## 2 abc hotels alibaba.com 60
905 ## 3 abc hotels paytm.com 55
906 ## 4 abc hotels goibibo 45
907 ## 5 abc hotels makemytrip 90
908 ## 6 abc hotels wizz 59
```

GROUP 7

```
909 ##What is the utilization rate for each hotel (that is the average billable days of a hotel  
910 specified as the average utilization of room bookings for the last 12 months)  
  
911 RSQLite:::dbGetQuery(myconn, "  
912 select h.Hotel_name,r.Room_No,r.Room_Name,r.Room_Rate  
913 from Room r inner join Hotel h on r.Hotel_Id = h.Hotel_Id")  
  
914 ##   Hotel_name      Room_No      Room_Name Room_Rate  
915 ## 1 abc hotels 12709804140 platea dictumst    36648  
916 ## 2 def hotels 1270980414 platea dictumst    3665  
917 ## 3 geh hotels 127098040 platea dictumst    3623  
918 ## 4 hij hotels 12709804 platea dictumst    3876  
919 ## 5 klm hotels 12709840 platea dictumst    36975  
  
920 ##Calculate the Customer Value in terms of total spent for each customer before the  
921 current booking.  
  
922 RSQLite:::dbGetQuery(myconn, "select g.Guest_FirstName,g.Guest_MiddleName,g  
923 .Guest_LastName,sum(t.Trans_TotalPayment) as totalSpent  
924 from Guest g inner join `GuestTransaction` t on g.Guest_Id = t.Guest_Id")  
  
925 ##   Guest_FirstName Guest_MiddleName Guest_LastName totalSpent  
926 ## 1           Dushant             H.          Gohri  569535.4
```

2. PART B -Semi Structured Data (Files)

```

927 Installing necessary packages for Data Management to work,
928 #install.packages("tidyverse")
929 #install.packages("janitor")
930
931 library("tidyverse")
932
933 library("readxl")
934 library("dplyr")
935 library("janitor")
936
937
938 library(plyr)
939
940 library(xml2)
941 library(stringr)
942 library(rebus)
943 library(XML)
944 library(readr)

945 #Listing all the directory inside the group_assignment folder
946 dirs<- list.dirs( "./group_assignment")

947
948 #intialise empty data frame
949
950 AllFilePaths = data.frame()

951
952 #using for Loop to get all the files from one directory at a time and then
953 combining that file paths in one data frame
954
955 for(i in 1:length(dirs)){
  file_names <- list.files(dirs[i], pattern = "\\.xlsx$")
956
957   #as we have one folder empty in the group_assignment folder so we have to
958 ignore that file path, that's why we used If condition
959
960   if(length(file_names)!=0){
961     paste(dirs[i],file_names, sep="/") %>%
962       data.frame() %>%
963       rbind(AllFilePaths,. ) -> AllFilePaths
964   }
965 }
966
967
968 #intialise empty data frame
969 allFiles = data.frame()

```

GROUP 7

```

970 #using for Loop for every excel file, we will clean the file and bind the
971 file in allFiles data frame
972 #using dplyr library we are using pipes so that we dont have to use so many variables
973
974
975 for ( ExcelFilePath in AllFilePath$.){
976   excel <- read_excel(ExcelFilePath, col_names = FALSE) %>%
977     #removing 2nd column which is an empty column
978   .[,-2] %>%
979   mutate(., Country=paste(.[5,2]),Flow=paste(.[3,2])) %>%
980     #removing extra rows, so that it will be easy to bind all the excel files together using Rbind
981   .[-c(1,2,3,4,5,7),] %>%
982     #removing NA values from data frame
983   na.omit() %>%
984     #changing to NA values if .. is in the excel data frame
985   na_if(., "...") %>%
986   rbind(allFiles,.) -> allFiles
987 }
988
989 #now we are going to use row_to_names so that columns will have a proper name as per we need
990
991
992 allFiles <- row_to_names(allFiles,1)
993 #renaming columns names for 1, 67, 68nd index
994 names(allFiles)[1] <- "year"
995 names(allFiles)[67] <- "country"
996 names(allFiles)[68] <- "flow"
997
998 #now we are gonna use pivot_longer, so that every value is considered individual for every country , every year, every flow and every product
1000
1001 allFiles<- allFiles[allFiles$BKB != "BKB",] %>% pivot_longer(-c(country,year,flow),names_to ="product") %>%
1002   select(country, everything())
1003
1004 head(allFiles)
1005
1006 ## # A tibble: 6 x 5
1007 ##   country year   flow   product      value
1008 ##   <chr>    <chr> <chr>   <chr>      <chr>
1009 ## 1 Albania 1971 Imports Additives/blending components 0
1010 ## 2 Albania 1971 Imports Anthracite                <NA>
1011 ## 3 Albania 1971 Imports Aviation gasoline          0
1012 ## 4 Albania 1971 Imports BKB                      0
1013 ## 5 Albania 1971 Imports Biodiesels              0
1014 ## 6 Albania 1971 Imports Biogases                 0
1015 nrow(allFiles)
1016 ## [1] 573950

```

GROUP 7

```
1017 #Initialise data_result dataframe from allFiles
1018 data_result_new<-allFiles
1019
1020 #the total number of records on the dataset and the total number of record
1021 s for each product across countries #across years using Group by and remov
1022 ing NA values which are unique
1023
1024 data_product<-data_result_new %>%
1025   filter(!is.na(.)) %>%
1026   group_by(country, year, product) %>%
1027   mutate(total_value=sum(as.numeric(as.character(value)))) %>%
1028   select(country, year, product, total_value) %>%
1029   distinct()
1030
1031 head(data_product)

1032 ## # A tibble: 6 x 4
1033 ## # Groups:   country, year, product [6]
1034 ##   country year  product          total_value
1035 ##   <chr>    <chr> <chr>           <dbl>
1036 ## 1 Albania 1971 Additives/blending components     NA
1037 ## 2 Albania 1971 Anthracite                      NA
1038 ## 3 Albania 1971 Aviation gasoline                NA
1039 ## 4 Albania 1971 BKB                            NA
1040 ## 5 Albania 1971 Biodiesels                     NA
1041 ## 6 Albania 1971 Biogases                      NA

1042 nrow(data_product)
1043 ## [1] 101335
```

3. PART C – Semi Structured Data (Web)

1044 Installing necessary packages for Data Management to work,

```

1045 #install.packages("tidyverse")
1046 #install.packages("janitor")
1047 #install.packages("rvest")
1048 #install.packages("httr")
1049 #install.packages("XML")
1050
1051 library("tidyverse")

1052 ## — Attaching packages tidyverse
1053 1.3.0 —

1054 ## ✓ ggplot2 3.3.2      ✓ purrr   0.3.4
1055 ## ✓ tibble  3.0.4      ✓ dplyr    1.0.2
1056 ## ✓ tidyr   1.1.2      ✓ stringr  1.4.0
1057 ## ✓ readr   1.4.0      ✓ forcats  0.5.0

1058 ## — Conflicts tidyverse_conflict
1059 icts() —
1060 ## x dplyr::filter() masks stats::filter()
1061 ## x dplyr::lag()   masks stats::lag()

1062 library("dplyr")
1063 library("janitor")

1064 ##
1065 ## Attaching package: 'janitor'

1066 ## The following objects are masked from 'package:stats':
1067 ##
1068 ##     chisq.test, fisher.test

1069 library("tidyverse")
1070 library("readxl")
1071
1072 library(leaflet)
1073 library(plyr)

1074 ##
1075 ----

1076 ## You have loaded plyr after dplyr - this is likely to cause problems.
1077 ## If you need functions from both plyr and dplyr, please load plyr first,
1078 ## then dplyr:
1079 ## library(plyr); library(dplyr)

1080 ##
1081 ----

1082 ##
1083 ## Attaching package: 'plyr'
```

GROUP 7

```
1084 ## The following objects are masked from 'package:dplyr':  
1085 ##  
1086 ##     arrange, count, desc, failwith, id, mutate, rename, summarise,  
1087 ##     summarise  
1088 ## The following object is masked from 'package:purrr':  
1089 ##  
1090 ##     compact  
1091 library(rvest)  
1092 ## Loading required package: xml2  
1093 ##  
1094 ## Attaching package: 'rvest'  
1095 ## The following object is masked from 'package:purrr':  
1096 ##  
1097 ##     pluck  
1098 ## The following object is masked from 'package:readr':  
1099 ##  
1100 ##     guess_encoding  
1101 library(httr)  
1102 library(stringr)  
1103 library(rebus)  
1104 ##  
1105 ## Attaching package: 'rebus'  
1106 ## The following object is masked from 'package:stringr':  
1107 ##  
1108 ##     regex  
1109 ## The following object is masked from 'package:ggplot2':  
1110 ##  
1111 ##     alpha  
1112 library(readr)  
1113 library(xml2)  
1114 library(XML)  
1115 ##  
1116 ## Attaching package: 'XML'  
1117 ## The following object is masked from 'package:rvest':  
1118 ##  
1119 ##     xml  
1120 Created a for loop again for reading the excel file individually as all the excel files are similar  
1121 we can also apply necessary operations like removing a column/row in the same loop, so  
1122 that we can have a structured data frame of all the data  
1123 now adding the country and flow column in the dataframe and removing duplicate row instances from  
1124 the data frame.
```

GROUP 7

```
1125 finally, we have the data frame of our need, we need to apply the pivot  
1126 Longer function on the data #frame and count the actual rows produced with  
1127 all the excel data  
1128 #using pivot_wider function to find out the number of records for each  
1129 product across countries across #years  
1130 # With the given Link in the assignment, scrapping all the Links for XML d  
1131 ata  
1132 url <- "https://www.food.gov.uk/uk-food-hygiene-rating-data-api"  
1133 # store all xml Link  
1134 url %>%  
1135 read_html() %>%  
1136 html_nodes(xpath = "//*/article/div/div/p[5]/a") %>%  
1137 html_attr("href") %>%  
1138 read_html() %>%  
1139 html_nodes(xpath = "//*/article/dl[2]/dd/strong/a") %>%  
1140 html_attr("href") -> all.xml.links  
1141  
1142 head(all.xml.links)  
1143 ## [1] "http://ratings.food.gov.uk/"  
1144 ## [2] "http://ratings.food.gov.uk/OpenDataFiles/FHRS760en-GB.xml"  
1145 ## [3] "http://ratings.food.gov.uk/OpenDataFiles/FHRS761en-GB.xml"  
1146 ## [4] "http://ratings.food.gov.uk/OpenDataFiles/FHRS323en-GB.xml"  
1147 ## [5] "http://ratings.food.gov.uk/OpenDataFiles/FHRS055en-GB.xml"  
1148 ## [6] "http://ratings.food.gov.uk/OpenDataFiles/FHRS062en-GB.xml"  
1149 Download all XML files through link for every row (city)  
1150 #Checking if the directory exists  
1151 output_dir <- file.path("./xml_files")  
1152  
1153 if (!dir.exists(output_dir)){  
1154 dir.create(output_dir)  
1155 } else {  
1156 print("Dir already exists!")  
1157 }  
1158 ## [1] "Dir already exists!"  
1159 #Initialise empty data frame for storing all the xml file path in current  
1160 directory  
1161 allxmlfilepath = data.frame()  
1162  
1163 for (i in 2:length(all.xml.links)) {  
1164 url <- all.xml.links[i]  
1165 #splitting the Link to get the filename  
1166 filename_vector <- strsplit(all.xml.links[i], split="/")[[1]]  
1167 filename <- filename_vector[length(filename_vector)]  
1168 filePath <- paste0("xml_files/", filename)  
1169 wholefilePath <- paste0("./", filePath)  
1170 #checking if the file already exists in the directory then don't downloa
```

GROUP 7

```
1171 d the file again
1172 if(!file.exists(wholefilePath)) {
1173   print(wholefilePath)
1174   #downloading XML file to Local directory
1175   download.file(url,filePath)
1176 }
1177 allxmlfilepath <- rbind(allxmlfilepath,wholefilePath)
1178 #renaming the colName of dataframe to more meaningful name
1179 names(allxmlfilepath)[1] <- "wholeFilePath"
1180 }

1181 #initialise an empty Data Frames
1182
1183 dfHygiene<- data.frame()
1184 dfStructural<- data.frame()
1185 dfConfidenceInManagement<- data.frame()
1186 total<- data.frame()
1187 result <- data.frame()
1188
1189 append_xml_nodes <- function(xml, node_name) {
1190   nodes = getNodeSet(xml, str_interp("//EstablishmentDetail/Scores[not(${node_name})]"))
1191   sapply(nodes, function(node) node[[str_interp("${node_name}")]]) = ""
1192 }
1193
1194
1195 for(wholeFilePath in allxmlfilepath$wholeFilePath) {
1196   # LOADING TRANSFORMED XML INTO R DATA FRAME
1197
1198   # (xpathSApply(wholeFilePath, "//ItemCount", xmlValue)) -> count
1199   # total <- rbind(total,count)
1200
1201 xml<-xmlParse(wholeFilePath)
1202   append_xml_nodes(xml, "Hygiene")
1203   append_xml_nodes(xml, "Structural")
1204   append_xml_nodes(xml, "ConfidenceInManagement")
1205
1206 #extracting data from xml using XmlToDataFrame
1207
1208 xmldf <- xmlToDataFrame(nodes = getNodeSet(xml, "//EstablishmentDetail"))
1209 %>%
1210   select(., -Geocode) %>% select(., -Scores)
1211
1212 dfGeocode <- xmlToDataFrame(nodes = getNodeSet(xml, "//Geocode") ) %>%
1213   bind_cols(xmldf, .) %>%
1214   bind_rows(.) -> result
1215
1216 colHygiene <- xmlToDataFrame(nodes = getNodeSet(xml, "//Hygiene") ) %>%
1217   bind_rows(dfHygiene, .) -> dfHygiene
1218
1219 colStructural <- xmlToDataFrame(nodes = getNodeSet(xml, "//Structural") )
1220 %>%
1221   bind_rows(dfStructural, .) -> dfStructural
1222
1223 colConfidenceInManagement <- xmlToDataFrame(nodes = getNodeSet(xml, "//Con
```

GROUP 7

```

1224 fidenceInManagement") ) %>%
1225   bind_rows(dfConfidenceInManagement, .) -> dfConfidenceInManagement
1226
1227
1228 }
1229
1230 #changing the column names to more meaningful name
1231 colnames(dfHygiene)[1] <- "Hygiene"
1232 colnames(dfStructural)[1] <- "Structural"
1233 colnames(dfConfidenceInManagement)[1] <- "ConfidenceInManagement"
1234 #colnames(total)[1] <- "Observations"
1235
1236 # binding all the dataframe to result dataframe, so that we can have more
1237 cleaner look
1238 result<- bind_cols(list(result, dfHygiene, dfStructural, dfConfidenceInManagement)) %>%
1239   na_if(., "")
1240
1241
1242 #converting Longitude and Latitude to Numeric data type
1243 result$Latitude <- as.numeric(as.character(result$Latitude))
1244 result$Longitude <- as.numeric(as.character(result$Longitude))
1245
1246 #saving the Xml data in Rdata file for Part D continuation, so that we don
1247 t have to render the data again from web
1248 save(result,file="data_result_C.RData")
1249
1250
1251 #To check ItemCount in all XML ratings Link matches number of rows
1252 #numberOfObservation <- sum(as.numeric(total$Observations))
1253
1254
1255
1256 head(result)

1257 ##      FHRSID LocalAuthorityBusinessID          BusinessName
1258 ## 1  982849                      EHDC12463 1 & 30 DONALD DEWAR COURT
1259 ## 2  592387                      EHDC10723 1906 RESTAURANT AT HMT
1260 ## 3  1017312                     EHDC12570 2 BROTHERS PIZZA
1261 ## 4  593681                      EHDC9793 210 BISTRO
1262 ## 5  997643                      EHDC12441 210 CABOOSE
1263 ## 6  593037                     EHDC4774 22 CLUB
1264 ##                               BusinessType BusinessTypeID          Addre
1265 ssLine2
1266 ## 1 Hospitals/Childcare/Caring Premises           5 1 & 30 Donald Dewa
1267 r Court
1268 ## 2                               Restaurant/Cafe/Canteen        1          Rosemount
1269 Viaduct
1270 ## 3                               Takeaway/sandwich shop    7844      Great Northe
1271 rn Road
1272 ## 4                               Restaurant/Cafe/Canteen        1          210 Market
1273 Street
1274 ## 5                               Restaurant/Cafe/Canteen        1          37 Waterl
1275 oo Quay
1276 ## 6                               Pub/bar/nightclub       7843          55 Rose

```

GROUP 7

```

1277 Street
1278 ## AddressLine3 AddressLine4 PostCode RatingValue
1279 ## 1 Provost Fraser Drive Aberdeen AB16 5JB Pass
1280 ## 2 Aberdeen <NA> AB25 1GL Pass
1281 ## 3 Aberdeen <NA> AB24 2DU Pass
1282 ## 4 Aberdeen <NA> AB11 5PQ Improvement Required
1283 ## 5 Aberdeen <NA> AB11 5BS Pass
1284 ## 6 Aberdeen <NA> AB10 1UB Pass
1285 ## RatingKey RatingDate LocalAuthorityCode
1286 ## 1 fhis_pass_en-GB 2017-05-15 760
1287 ## 2 fhis_pass_en-GB 2020-02-20 760
1288 ## 3 fhis_pass_en-GB 2019-10-30 760
1289 ## 4 fhis_improvement_required_en-GB 2018-12-18 760
1290 ## 5 fhis_pass_en-GB 2019-01-09 760
1291 ## 6 fhis_pass_en-GB 2014-03-11 760
1292 ## LocalAuthorityName LocalAuthorityWebSite
1293 ## 1 Aberdeen City http://www.aberdeencity.gov.uk
1294 ## 2 Aberdeen City http://www.aberdeencity.gov.uk
1295 ## 3 Aberdeen City http://www.aberdeencity.gov.uk
1296 ## 4 Aberdeen City http://www.aberdeencity.gov.uk
1297 ## 5 Aberdeen City http://www.aberdeencity.gov.uk
1298 ## 6 Aberdeen City http://www.aberdeencity.gov.uk
1299 ## LocalAuthorityEmailAddress SchemeType NewRatingPending Address
1300 Line1
1301 ## 1 commercial@aberdeencity.gov.uk FHIS False
1302 <NA>
1303 ## 2 commercial@aberdeencity.gov.uk FHIS False
1304 <NA>
1305 ## 3 commercial@aberdeencity.gov.uk FHIS False
1306 699B
1307 ## 4 commercial@aberdeencity.gov.uk FHIS False
1308 <NA>
1309 ## 5 commercial@aberdeencity.gov.uk FHIS False Provender
1310 House
1311 ## 6 commercial@aberdeencity.gov.uk FHIS False
1312 <NA>
1313 ## Longitude Latitude RightToReply Hygiene Structural ConfidenceInManagement
1314 ## 1 -2.167495 57.16131 <NA> <NA> <NA>
1315 <NA>
1316 ## 2 -2.104965 57.14816 <NA> <NA> <NA>
1317 <NA>
1318 ## 3 -2.138175 57.17238 <NA> <NA> <NA>
1319 <NA>
1320 ## 4 -2.092258 57.14228 <NA> <NA> <NA>
1321 <NA>
1322 ## 5 -2.084367 57.14587 <NA> <NA> <NA>
1323 <NA>
1324 ## 6 -2.111912 57.14456 <NA> <NA> <NA>
1325 <NA>
1326 <NA>
1327 nrow(result)
1328 ## [1] 596285

```

4. PART D – Dashboard with R/Shiny

1329 Installing necessary packages for Data Management to work,

```

1330 #install.packages("tidyverse")
1331 #install.packages("plotly")
1332 #install.packages("shinydashboard")
1333 #install.packages("anchors")
1334 #install.packages("shiny")
1335 #install.packages("ggplot2")
1336 #install.packages('devtools')
1337 library(devtools)

1338 #install_github("ramnathv/rCharts")
1339
1340 library("plyr")
1341 library(plotly)

1342 library(shinydashboard)

1343 library(anchors)

1344 library(shiny)
1345 library(ggplot2)
1346 library(rCharts)
1347 library(leaflet)
```

1348 A graphical representation of the rating values as obtained from the parsing of the XML document.

```

1349 #data_result_C <- get(Load("./data_result_C.RData"))
1350 data_result_C <- readRDS("./data_result_C.rds")
1351
1352
1353 bb_data <- data_result_C
1354 ##Change the class of LocalAuthorityName an RatingValue to prepare for the
1355 ##following operation and to make it more proper way to show in the table
1356
1357 bb_data$LocalAuthorityName <- as.factor(bb_data$LocalAuthorityName)
1358 bb_data$RatingValue <- as.factor(bb_data$RatingValue)
1359
1360 ui <- dashboardPage(
1361   dashboardHeader(title = "Food Hygiene Ratings Scheme Data"),
1362   dashboardSidebar(
1363     sidebarMenu(
1364       menuItem("Data", tabName = "dataset"),
1365       menuItem("Presence in UK", tabName = "chart_1"),
1366       menuItem("Location Wise Ratings", tabName = "chart_2")
1367     )),
1368   dashboardBody(
1369     tabItems(
1370       # first tab named "dataset"
1371       tabItem(tabName = "dataset",
1372               fluidRow(h1("DATASET")),
1373               fluidRow(column(width = 12, DT::dataTableOutput("resulttable"))))
```

GROUP 7

```

1374 })) ,
1375 ),
1376
1377     # second tab named "chart_1"
1378     tabItem(tabName = "chart_1",
1379             fluidRow(column(width=12, h1("Map"))),
1380             fluidPage( navbarPage("", id="main", tabPanel("Map", leaflet
1381 Output("bbmap", height=1000))))
1382         ),
1383     #third tab named "chart_2"
1384     tabItem(tabName = "chart_2",
1385             fluidRow(column(width = 12, h1("Local Authority Wise Ratings"))
1386 )),
1387             fluidRow(column(width = 10, selectInput("LocalAuthorityName",
1388 select Local Authority", choices = c("Select", levels(bb_data$LocalAuthorit
1389 yName)), selected = "Select")
1390             )),
1391             fluidRow(column(width = 12,plotlyOutput("ratingsdata")))
1392         )
1393     )
1394   )
1395 )
1396
1397 server<-function(input, output) {
1398
1399   changefactor <- c("AwaitingInspection" = "Awaiting Inspection", "Awaitin
1400 gPublication" = "Awaiting Publication")
1401   bb_data$RatingValue <- revalue(bb_data$RatingValue, changefactor)
1402   bb_data<-bb_data %>% filter(!is.na(Latitude)) %>% filter(!is.na(Longitud
1403 e))
1404   sapply(bb_data, class)
1405
1406   output$resulttable = DT:::renderDataTable(bb_data)
1407
1408   # create a color for different rating value
1409   Col <- colorFactor(topo.colors(12), bb_data$RatingValue)
1410
1411   output$bbmap <- renderLeaflet({
1412     leaflet(bb_data) %>%
1413       addTiles() %>%
1414       addCircles(lng = ~Longitude, lat = ~Latitude,
1415                 color = ~Col(RatingValue),
1416                 radius = 2,
1417                 stroke = T,
1418                 fillOpacity = 1,
1419                 popup=paste(
1420                   "Rating:", bb_data$RatingValue, "<br>",
1421                   "Business:", bb_data$BusinessName, "<br>",
1422                   "Area:", bb_data$LocalAuthorityName, "<br>"))
1423   })
1424
1425   output$ratingsdata <- renderPlotly({
1426     if(input$LocalAuthorityName=="Select"){
1427       output<-ggplot((bb_data),aes(x=RatingValue))+geom_bar()

```

```

1428     scale_fill_hue(l=40, c=35)
1429     ggplotly(output)
1430   }
1431 else{
1432   to_filter <- input$LocalAuthorityName
1433   output <- ggplot(subset(bb_data, LocalAuthorityName==to_filter), aes
1434 (x=RatingValue))+geom_bar()
1435   scale_fill_hue(l=40, c=35)
1436   ggplotly(output)}
1437 )
1438 )
1439 }
1440 }
1441 }
1442 shinyApp(ui = ui, server = server)

```

Shiny applications not supported in static R Markdown documents

4.1. Shiny Dashboard

Dashboard creation involved libraries like shiny, leaflet, DT and ggplot2. We have created three tabs on our dashboard. The three tabs depict the dataset of hygiene ratings obtained from various parts of United Kingdom, geographical location of rated places, plot between ratings and location wise.

As shown in **Figure 3**, the figure depicts the map of the restaurant location spread across UK, describing multiple colors for different ratings for better Human understanding, furthermore on click of the Business's location, we can see the rating, business name and area of the particular Business

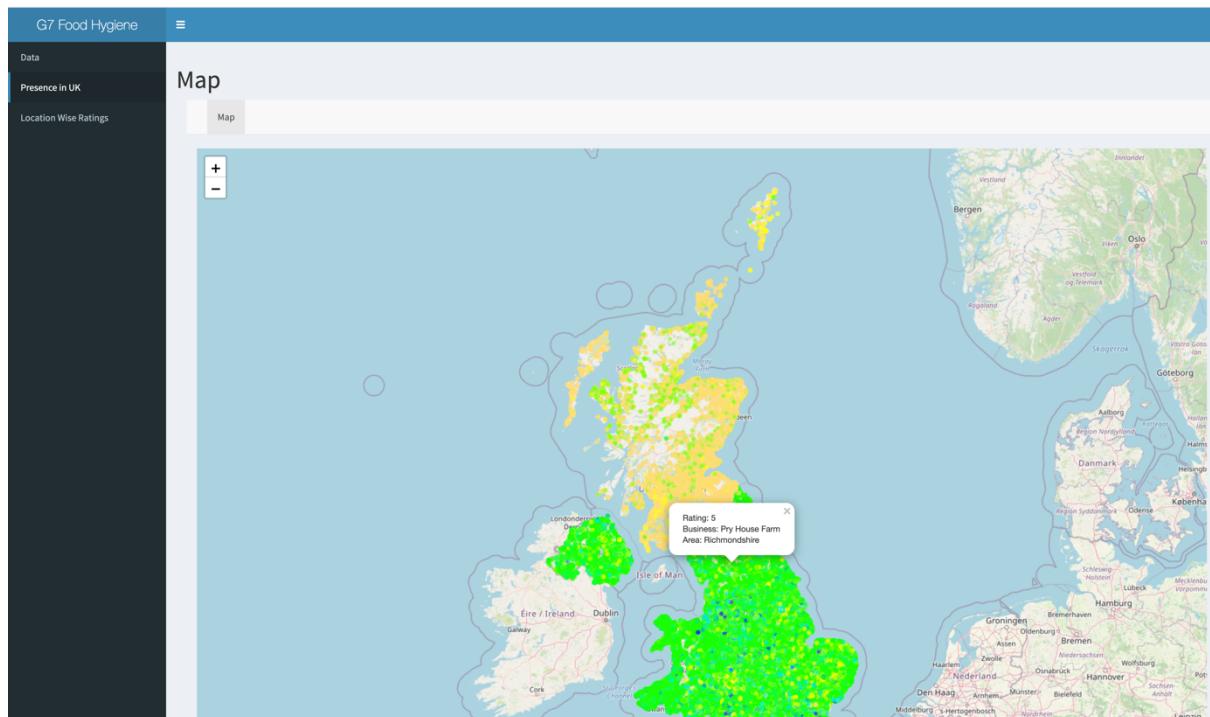


Figure 3: Presence in UK

GROUP 7

As per given **Figure 4**, we can see the whole dataset of the UK food hygiene rating data.

FHRSID	LocalAuthorityBusinessID	BusinessName	BusinessType	BusinessTypeID	AddressLine1	AddressLine2	AddressLine3	AddressLine4	PostCode
1	982849	1 & 30 DONALD DEWAR COURT	Hospitals/Childcare/Caring Premises	5	1 & 30 Donald Dewar Court	Provost Fraser Drive	Aberdeen	AB16 5JB	
2	592387	1906 RESTAURANT AT HMT	Restaurant/Cafe/Canteen	1	Rosemount Viaduct	Aberdeen		AB25 1GL	
3	1017312	2 BROTHERS PIZZA	Takeaway/sandwich shop	7844	Great Northern Road	Aberdeen		AB24 2DU	
4	593681	210 BISTRO	Restaurant/Cafe/Canteen	1	210 Market Street	Aberdeen		AB11 5PQ	
5	997643	210 CABOOSE	Restaurant/Cafe/Canteen	1	37 Waterloo Quay	Aberdeen		AB11 5BS	
6	593037	22 CLUB	Pub/bar/nightclub	7843	55 Rose Street	Aberdeen		AB10 1UB	
7	1194249	3 STAR FISH BAR	Takeaway/sandwich shop	7844	Lewis Road	Sheddoxley	Aberdeen		
8	593410	3rd HOUSE GUESTHOUSE	Hotel/bed & breakfast/guest house	7842	406 Great Western Road	Aberdeen		AB10 6NR	
9	593690	52 REGENT QUAY APARTMENTS	Hotel/bed & breakfast/guest house	7842	52 Regent Quay	Aberdeen		AB11 5AQ	
10	1106585	524 BAR	Pub/bar/nightclub	7843	524 George Street	Aberdeen		AB25 3XJ	

Showing 1 to 10 of 506,602 entries

Previous 1 2 3 4 5 ... 50661 Next

Figure 4: Food Hygiene Dataset

As shown in **Figure 5**, Rating by location wise, the bar chart depicts hygiene rating values of that businesses as per their location. We can filter out Location to see the trend. With this plot, viewers can see the distribution of rating across location by count and rating value.



Figure 5: Location Wise Ratings

GROUP 7

As shown in **Figure 6**, viewers can see the Bar chart of the location Coventry.

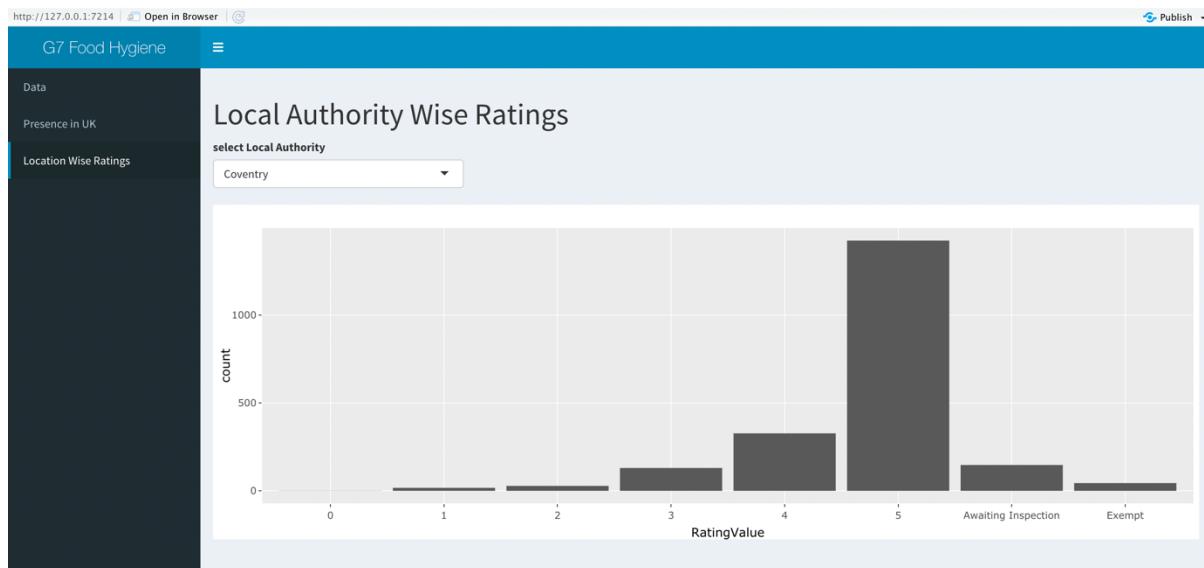


Figure 6: LocationWise Data Coventry

REFERENCES

RSQLite

(Wickham *et al.*, 2015)

Wickham, H. *et al.* (2015) ‘Package “ RSSQLite ”’, *CRAN Reference manual*.

Tidyverse

(Wickham, Averick, *et al.*, 2019)

Wickham, H., Averick, M., *et al.* (2019) ‘Welcome to the Tidyverse’, *Journal of Open Source Software*.

doi: 10.21105/joss.01686.

Plotly

Sievert, C. (2020) *Interactive Web-Based Data Visualization with R, plotly, and shiny, Interactive Web-Based Data Visualization with R, plotly, and shiny*. doi: 10.1201/9780429447273.

Lubridate

(Grolemund and Wickham, 2011)

Grolemund, G. and Wickham, H. (2011) ‘Dates and times made easy with lubridate’, *Journal of Statistical Software*. doi: 10.18637/jss.v040.i03.

Readxl

(Wickham, Bryan, *et al.*, 2019)

Wickham, H., Bryan, J., *et al.* (2019) ‘Read Excel Files’, *Package ‘readxl’ Version 1.3.1*.

Dplyr

(Wickham, François, *et al.*, 2019)

Wickham, H., François, R., *et al.* (2019) ‘*dplyr: A Grammar of Data Manipulation. R package version*’, *Media*.

Tidyr

(Wickham and Henry, 2019)

Wickham, H. and Henry, L. (2019) ‘tidyr: Tidy Messy Data’, *R package version 1.0.0*.

GROUP 7

Shiny

(Chang *et al.*, 2018)

Stringr

Hadley Wickham (2019). stringr: Simple, Consistent Wrappers for Common String Operations. R package version 1.4.0. <https://CRAN.R-project.org/package=stringr>

Ggplot2

(Ginestet, 2011)

Ginestet, C. (2011) ‘ggplot2: Elegant Graphics for Data Analysis’, *Journal of the Royal Statistical Society: Series A (Statistics in Society)*. doi: 10.1111/j.1467-985x.2010.00676_9.x.

Leaflet

(Cheng, Karambelkar and Xie, 2018)

Cheng, J., Karambelkar, B. and Xie, Y. (2018) ‘Package ‘leaflet’ - Create Interactive Web Maps with the JavaScript “Leaflet” Library’, *CRAN Repository*

Janitor

(Firke, 2020)

Firke, S. (2020) ‘Simple Tools for Examining and Cleaning Dirty Data [R package janitor version 2.0.1]’. Available at: <https://cran.r-project.org/package=janitor> (Accessed: 9 December 2020).

Rebus

Richard Cotton (2017). rebus: Build Regular Expressions in a Human Readable Way. R package version 0.1-3. <https://CRAN.R-project.org/package=rebus>

Readr

(‘Read Rectangular Text Data [R package readr version 1.4.0]’, 2020)

‘Read Rectangular Text Data [R package readr version 1.4.0]’ (2020). Available at: <https://cran.r-project.org/package=readr> (Accessed: 9 December 2020).{Formatting Citation}

Anchor

(CRAN - Package anchors, 2016)

GROUP 7

CRAN - Package anchors (2016). Available at: <https://cran.r-project.org/web/packages/anchors/index.html> (Accessed: 9 December 2020).

Devtools

('Tools to Make Developing R Packages Easier [R package devtools version 2.3.2]', 2020)

'Tools to Make Developing R Packages Easier [R package devtools version 2.3.2]' (2020). Available at: <https://cran.r-project.org/package=devtools> (Accessed: 9 December 2020).

Shinydashboard

Winston Chang and Barbara Borges Ribeiro (2018). shinydashboard:

Create Dashboards with 'Shiny'. R package version 0.7.1. <https://CRAN.R-project.org/package=shinydashboard>

Rcharts

rCharts

rCharts (no date). Available at: <https://ramnathv.github.io/rCharts/> (Accessed: 9 December 2020).

Rvest

(Wickham, 2020)

Wickham, H. (2020) 'Easily Harvest (Scrape) Web Pages [R package rvest version 0.3.6]'. Available at: <https://cran.r-project.org/package=rvest> (Accessed: 10 December 2020).

Xml

(Lang, 2020)

Lang, D. T. (2020) '*Tools for Parsing and Generating XML Within R and S-Plus* [R package XML version 3.99-0.5]'. Available at: <https://cran.r-project.org/package=XML> (Accessed: 10 December 2020).

Httr

(Wickham, 2020)

Wickham, H. (2020) 'Tools for Working with URLs and HTTP [R package httr version 1.4.2]'. Available at: <https://cran.r-project.org/package=httr> (Accessed: 10 December 2020).