PROJECT REPORT

IB9HP0 - DATA MANAGEMENT

MODULE CONVENOR:
DR NIKOLAOS KORFIATIS

GROUP 7

Tables of Contents

Index of figures

Index of tables

1. Part A – Structured Database Design

1.1. Conceptual Design

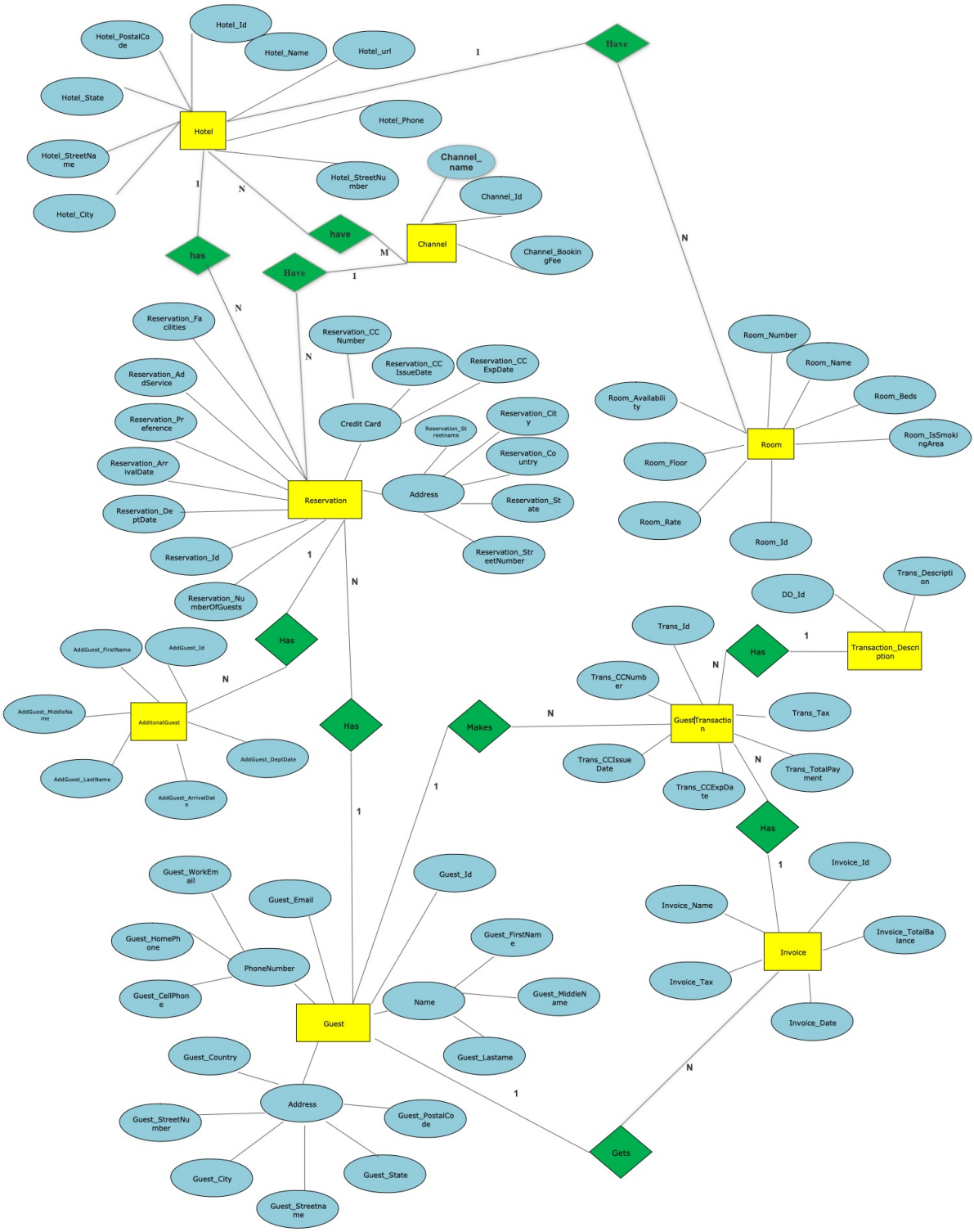

Figure 1: E-R Diagram

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

- 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

- 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

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

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

- 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 form 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 belong 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**

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

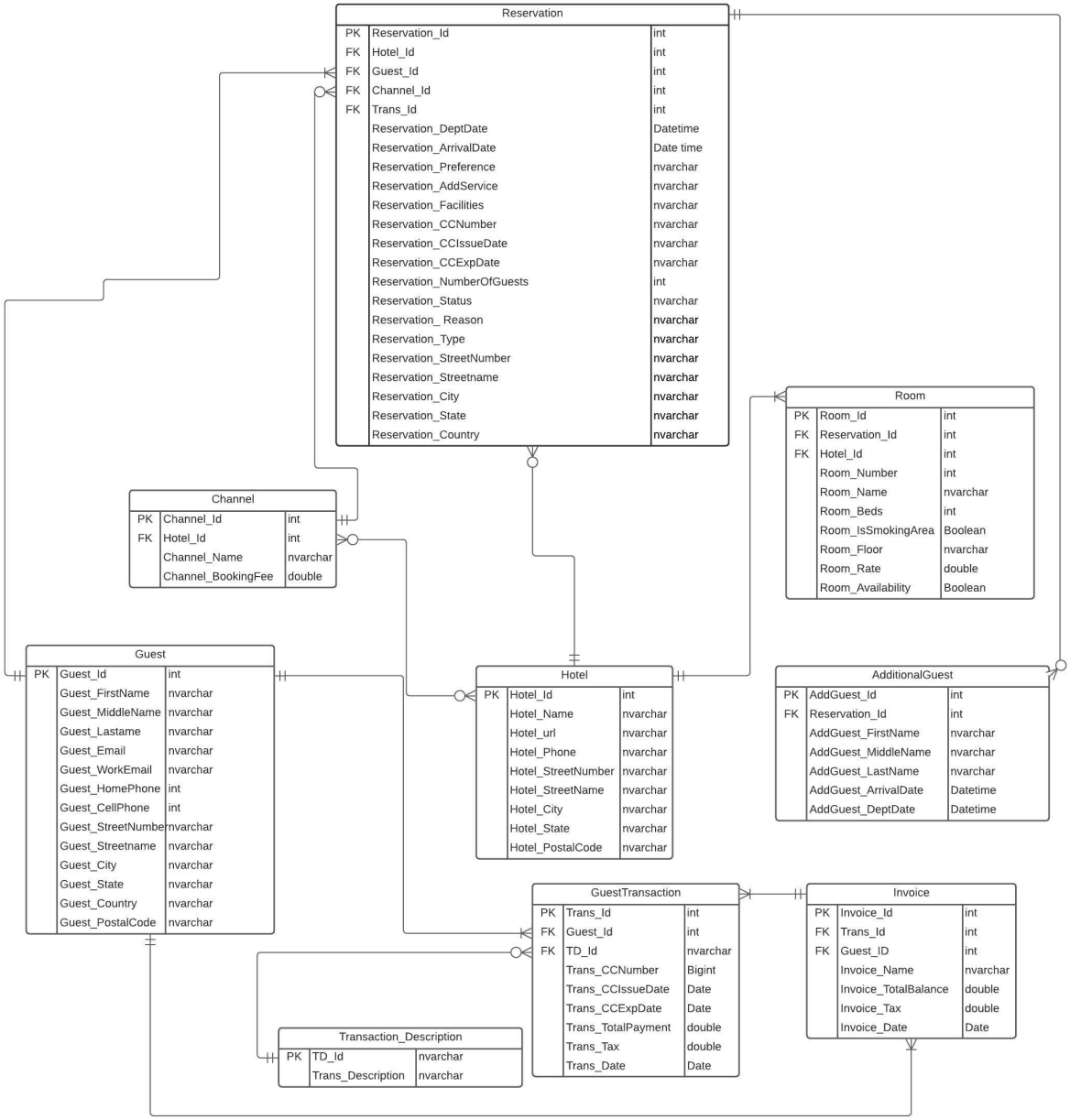

Figure 2: Logical E-R Diagram

1.3. DDL Queries

R Markdown

Installing necessary packages for Data Management to work,

```
# install.packages("tidyverse")
# install.packages("janitor")

library("tidyverse")

## ── Attaching packages ─────────────────────────────────────── tidyverse
1.3.0 ──

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

## ── Conflicts ────────────────────────────────────────── tidyverse_confl
icts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library("dplyr")
library("janitor")

##
## Attaching package: 'janitor'

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

library("tidyr")

library(plyr)

## ------------------------------------------------------------------------
-------

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

## ------------------------------------------------------------------------
-------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following object is masked from 'package:purrr':
##
##     compact

library(stringr)
library(rebus)

##
## Attaching package: 'rebus'

## The following object is masked from 'package:stringr':
##
##     regex

## The following object is masked from 'package:ggplot2':
##
##     alpha

library(readr)
library(RSQLite)
```

#Connect to SQLite

```
myconn <- RSQLite::dbConnect(RSQLite::SQLite(),"HotelManagement.db")
```

#Creating table hotel and dropping the table if it already exists

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

## <SQLiteResult>
##   SQL  DROP TABLE IF EXISTS `Hotel`
##   ROWS Fetched: 0 [complete]
##        Changed: 0

RSQLite::dbSendQuery(myconn, "CREATE TABLE `Hotel`  (
  `Hotel_Id` int(11) NOT NULL,
  `Hotel_name` varchar(255)  NOT NULL,
  `Hotel_WebsiteUrl` varchar(255) NOT NULL,
  `Hotel_Phone` varchar(255) NOT NULL,
  `Hotel_StreetNumber` varchar(255) NOT NULL,
  `Hotel_StreetName` varchar(255) NOT NULL,
  `Hotel_City` varchar(255) NOT NULL,
  `Hotel_State` varchar(255) NOT NULL,
  `Hotel_PostalCode` varchar(255) NOT NULL,
  PRIMARY KEY (`Hotel_Id`)
)")

## <SQLiteResult>
##   SQL  CREATE TABLE `Hotel`  (
##   `Hotel_Id` int(11) NOT NULL,
##   `Hotel_name` varchar(255)  NOT NULL,
##   `Hotel_WebsiteUrl` varchar(255) NOT NULL,
##   `Hotel_Phone` varchar(255) NOT NULL,
##   `Hotel_StreetNumber` varchar(255) NOT NULL,
##   `Hotel_StreetName` varchar(255) NOT NULL,
##   `Hotel_City` varchar(255) NOT NULL,
```

```
##   `Hotel_State` varchar(255) NOT NULL,
##   `Hotel_PostalCode` varchar(255) NOT NULL,
##   PRIMARY KEY (`Hotel_Id`)
## )
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

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

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

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

```
RSQLite::dbSendQuery(myconn, "CREATE TABLE `Guest`  (
  `Guest_Id` int(11) NOT NULL,
  `Guest_FirstName` varchar(255) NOT NULL,
  `Guest_MiddleName` varchar(255) DEFAULT NULL,
  `Guest_LastName` varchar(255) NOT NULL,
  `Guest_Email` varchar(255) NOT NULL,
  `Guest_WorkPhone` varchar(255) DEFAULT NULL,
  `Guest_HomePhone` varchar(255) DEFAULT NULL,
  `Guest_CellPhone` varchar(255) DEFAULT NULL,
  `Guest_StreetNumber` varchar(255) NOT NULL,
  `Guest_StreetName` varchar(255) NOT NULL,
  `Guest_City` varchar(255) NOT NULL,
  `Guest_State` varchar(255) NOT NULL,
  `Guest_Country` varchar(255) NOT NULL,
  `Guest_PostalCode` varchar(255) NOT NULL,
  PRIMARY KEY (`Guest_Id`)
)")
```

```
## <SQLiteResult>
##   SQL  CREATE TABLE `Guest`  (
##   `Guest_Id` int(11) NOT NULL,
##   `Guest_FirstName` varchar(255) NOT NULL,
##   `Guest_MiddleName` varchar(255) DEFAULT NULL,
##   `Guest_LastName` varchar(255) NOT NULL,
##   `Guest_Email` varchar(255) NOT NULL,
##   `Guest_WorkPhone` varchar(255) DEFAULT NULL,
##   `Guest_HomePhone` varchar(255) DEFAULT NULL,
##   `Guest_CellPhone` varchar(255) DEFAULT NULL,
##   `Guest_StreetNumber` varchar(255) NOT NULL,
##   `Guest_StreetName` varchar(255) NOT NULL,
##   `Guest_City` varchar(255) NOT NULL,
##   `Guest_State` varchar(255) NOT NULL,
##   `Guest_Country` varchar(255) NOT NULL,
##   `Guest_PostalCode` varchar(255) NOT NULL,
##   PRIMARY KEY (`Guest_Id`)
## )
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
#Creating table channel and dropping the table if it already exists
RSQLite::dbSendQuery(myconn,"DROP TABLE IF EXISTS `Channel`")

## <SQLiteResult>
##   SQL  DROP TABLE IF EXISTS `Channel`
##   ROWS Fetched: 0 [complete]
##        Changed: 0

RSQLite::dbSendQuery(myconn, "CREATE TABLE `Channel`  (
  `Channel_Id` int(11) NOT NULL,
  `Channel_Name` varchar(255) NOT NULL,
  `Hotel_Id` int(11) NULL DEFAULT NULL,
  `Channel_BookingFee` decimal(8, 2) NOT NULL,
  PRIMARY KEY (`Channel_Id`) ,
  CONSTRAINT `fk_1` FOREIGN KEY (`Hotel_Id`) REFERENCES `Hotel` (`Hotel_Id
`)
)")

## <SQLiteResult>
##   SQL  CREATE TABLE `Channel`  (
##   `Channel_Id` int(11) NOT NULL,
##   `Channel_Name` varchar(255) NOT NULL,
##   `Hotel_Id` int(11) NULL DEFAULT NULL,
##   `Channel_BookingFee` decimal(8, 2) NOT NULL,
##   PRIMARY KEY (`Channel_Id`) ,
##   CONSTRAINT `fk_1` FOREIGN KEY (`Hotel_Id`) REFERENCES `Hotel` (`Hotel
_Id`)
## )
##   ROWS Fetched: 0 [complete]
##        Changed: 0

#Creating table GuestTransaction and dropping the table if it already exists
RSQLite::dbSendQuery(myconn, "DROP TABLE IF EXISTS `GuestTransaction`");

## <SQLiteResult>
##   SQL  DROP TABLE IF EXISTS `GuestTransaction`
##   ROWS Fetched: 0 [complete]
##        Changed: 0

RSQLite::dbSendQuery(myconn, "CREATE TABLE `GuestTransaction`  (
  `Trans_Id` int(11) NOT NULL,
  `Guest_Id` int(11) NULL DEFAULT NULL,
  `Trans_CCNumber` VARCHAR(200) NULL DEFAULT NULL,
  `Trans_CCIssueDate` VARCHAR(200) NULL DEFAULT NULL,
  `TD_Id` int(11) NULL DEFAULT NULL,
  `Trans_CCExpDate` varchar(200) NULL DEFAULT NULL,
  `Trans_TotalPayment` decimal(8, 2) NULL DEFAULT NULL,
  `Trans_Date` date NULL DEFAULT NULL,
  `Trans_Tax` date NULL DEFAULT NULL,
  PRIMARY KEY (`Trans_Id`),
  CONSTRAINT `fk1` FOREIGN KEY (`Guest_Id`) REFERENCES `customer` (`Guest_
Id`),
  CONSTRAINT `fk15` FOREIGN KEY (`TD_Id`) REFERENCES `transaction_Descript
```

```
ion` (`TD_Id`)
)");

## <SQLiteResult>
##   SQL  CREATE TABLE `GuestTransaction`  (
##   `Trans_Id` int(11) NOT NULL,
##   `Guest_Id` int(11) NULL DEFAULT NULL,
##   `Trans_CCNumber` VARCHAR(200) NULL DEFAULT NULL,
##   `Trans_CCIssueDate` VARCHAR(200) NULL DEFAULT NULL,
##   `TD_Id` int(11) NULL DEFAULT NULL,
##   `Trans_CCExpDate` varchar(200) NULL DEFAULT NULL,
##   `Trans_TotalPayment` decimal(8, 2) NULL DEFAULT NULL,
##   `Trans_Date` date NULL DEFAULT NULL,
##   `Trans_Tax` date NULL DEFAULT NULL,
##   PRIMARY KEY (`Trans_Id`),
##   CONSTRAINT `fk1` FOREIGN KEY (`Guest_Id`) REFERENCES `customer` (`Gue
st_Id`),
##   CONSTRAINT `fk15` FOREIGN KEY (`TD_Id`) REFERENCES `transaction_Descr
iption` (`TD_Id`)
## )
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

#Creating table Transaction_Description and dropping the table if it already exists

```
RSQLite::dbSendQuery(myconn, "DROP TABLE IF EXISTS `Transaction_Descriptio
n`");

## <SQLiteResult>
##   SQL  DROP TABLE IF EXISTS `Transaction_Description`
##   ROWS Fetched: 0 [complete]
##        Changed: 0

RSQLite::dbSendQuery(myconn, "CREATE TABLE `Transaction_Description`  (
  `TD_Id` int(11) NOT NULL,
  `Trans_Description` varchar(255) NOT NULL,
  PRIMARY KEY (`TD_Id`)
)");

## <SQLiteResult>
##   SQL  CREATE TABLE `Transaction_Description`  (
##   `TD_Id` int(11) NOT NULL,
##   `Trans_Description` varchar(255) NOT NULL,
##   PRIMARY KEY (`TD_Id`)
## )
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

#Creating table invoice and dropping the table if it already exists

```
RSQLite::dbSendQuery(myconn, "DROP TABLE IF EXISTS `Invoice`");

## <SQLiteResult>
##   SQL  DROP TABLE IF EXISTS `Invoice`
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
RSQLite::dbSendQuery(myconn, "CREATE TABLE `Invoice`  (
  `Invoice_Id` int(11) NOT NULL,
  `Trans_Id` int(11) NULL DEFAULT NULL,
  `Guest_ID` int(11) NULL DEFAULT NULL,
  `Invoice_Name` varchar(255) DEFAULT NULL,
  `Invoice_TotalAmount` decimal(8, 2) NOT NULL,
  `Invoice_TotalBalance` decimal(8, 2) NOT NULL,
  `Invoice_OtherCharges` decimal(8, 2) NULL DEFAULT NULL,
  `Invoice_Tax` decimal(8, 2) NOT NULL,
  `Invoice_Date` date NOT NULL,
  PRIMARY KEY (`Invoice_Id`),
  CONSTRAINT `fk5` FOREIGN KEY (`Trans_Id`) REFERENCES `transaction` (`Tra
ns_Id`),
  CONSTRAINT `fk6` FOREIGN KEY (`Guest_ID`) REFERENCES `customer` (`Guest_
Id`)
)");
```

```
## <SQLiteResult>
##   SQL  CREATE TABLE `Invoice`  (
##   `Invoice_Id` int(11) NOT NULL,
##   `Trans_Id` int(11) NULL DEFAULT NULL,
##   `Guest_ID` int(11) NULL DEFAULT NULL,
##   `Invoice_Name` varchar(255) DEFAULT NULL,
##   `Invoice_TotalAmount` decimal(8, 2) NOT NULL,
##   `Invoice_TotalBalance` decimal(8, 2) NOT NULL,
##   `Invoice_OtherCharges` decimal(8, 2) NULL DEFAULT NULL,
##   `Invoice_Tax` decimal(8, 2) NOT NULL,
##   `Invoice_Date` date NOT NULL,
##   PRIMARY KEY (`Invoice_Id`),
##   CONSTRAINT `fk5` FOREIGN KEY (`Trans_Id`) REFERENCES `transaction` (`
Trans_Id`),
##   CONSTRAINT `fk6` FOREIGN KEY (`Guest_ID`) REFERENCES `customer` (`Gue
st_Id`)
## )
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

#Creating table reservation and dropping the table if it already exists

```
RSQLite::dbSendQuery(myconn, "DROP TABLE IF EXISTS `Reservation`");
```

```
## <SQLiteResult>
##   SQL  DROP TABLE IF EXISTS `Reservation`
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
RSQLite::dbSendQuery(myconn, "CREATE TABLE `Reservation` (
  `Reservation_Id` int(11) NOT NULL,
  `Hotel_Id` int(11) NULL DEFAULT NULL,
  `Guest_Id` int(11) NULL DEFAULT NULL,
  `Channel_Id` int(11) NULL DEFAULT NULL,
  `Reservation_ArrivalDate` date NOT NULL,
  `Reservation_DeptDate` date NOT NULL,
  `Trans_Id` int(11) NULL DEFAULT NULL,
  `Reservation_Preference` varchar(255) DEFAULT NULL,
```

```
  `Reservation_AddService` varchar(255) DEFAULT NULL,
  `Reservation_Facilities` varchar(255) DEFAULT NULL,
  `Reservation_Status` varchar(255) NOT NULL,
  `Reservation_CCNumber` bigint(20) NOT NULL,
  `Reservation_CCIssueDate` varchar(255) DEFAULT NULL,
  `Reservation_CCExpDate` varchar(255) NOT NULL,
  `Reservation_NumberOfGuests` int(11) NULL DEFAULT NULL,
  PRIMARY KEY (`Reservation_Id`),
  CONSTRAINT `fk7` FOREIGN KEY (`Hotel_Id`) REFERENCES `hotel` (`Hotel_Id`
),
  CONSTRAINT `fk8` FOREIGN KEY (`Guest_Id`) REFERENCES `customer` (`Guest_
Id`),
  CONSTRAINT `fk9` FOREIGN KEY (`Channel_Id`) REFERENCES `channel` (`Chann
el_Id`),
  CONSTRAINT `fk10` FOREIGN KEY (`Trans_Id`) REFERENCES `transaction` (`Tr
ans_Id`)
)");

## <SQLiteResult>
##   SQL  CREATE TABLE `Reservation` (
##   `Reservation_Id` int(11) NOT NULL,
##   `Hotel_Id` int(11) NULL DEFAULT NULL,
##   `Guest_Id` int(11) NULL DEFAULT NULL,
##   `Channel_Id` int(11) NULL DEFAULT NULL,
##   `Reservation_ArrivalDate` date NOT NULL,
##   `Reservation_DeptDate` date NOT NULL,
##   `Trans_Id` int(11) NULL DEFAULT NULL,
##   `Reservation_Preference` varchar(255) DEFAULT NULL,
##   `Reservation_AddService` varchar(255) DEFAULT NULL,
##   `Reservation_Facilities` varchar(255) DEFAULT NULL,
##   `Reservation_Status` varchar(255) NOT NULL,
##   `Reservation_CCNumber` bigint(20) NOT NULL,
##   `Reservation_CCIssueDate` varchar(255) DEFAULT NULL,
##   `Reservation_CCExpDate` varchar(255) NOT NULL,
##   `Reservation_NumberOfGuests` int(11) NULL DEFAULT NULL,
##   PRIMARY KEY (`Reservation_Id`),
##   CONSTRAINT `fk7` FOREIGN KEY (`Hotel_Id`) REFERENCES `hotel` (`Hotel_
Id`),
##   CONSTRAINT `fk8` FOREIGN KEY (`Guest_Id`) REFERENCES `customer` (`Gue
st_Id`),
##   CONSTRAINT `fk9` FOREIGN KEY (`Channel_Id`) REFERENCES `channel` (`Ch
annel_Id`),
##   CONSTRAINT `fk10` FOREIGN KEY (`Trans_Id`) REFERENCES `transaction` (
`Trans_Id`)
## )
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

#Creating table room and dropping the table if it already exists

```
RSQLite::dbSendQuery(myconn, "DROP TABLE IF EXISTS `Room`");

## <SQLiteResult>
##   SQL  DROP TABLE IF EXISTS `Room`
```

```
##   ROWS Fetched: 0 [complete]
##        Changed: 0

RSQLite::dbSendQuery(myconn, "CREATE TABLE `Room`  (
  `Room_Id` int(11) NOT NULL,
  `Reservation_Id` int(11) NULL DEFAULT NULL,
  `Room_No` int(11) NULL DEFAULT NULL,
  `Room_Name` varchar(255) DEFAULT NULL,
  `Room_Beds` int(11) NOT NULL,
  `Room_IsSmokingArea` Boolean,
  `Room_Floor` varchar(255) NOT NULL,
  `Room_Rate` decimal(8, 2) NOT NULL,
  `Room_Availability` Boolean,
  `Hotel_Id` int(11) NULL DEFAULT NULL,
  PRIMARY KEY (`Room_Id`) ,
  CONSTRAINT `fk11` FOREIGN KEY (`Reservation_Id`) REFERENCES `reservation
` (`Reservation_Id`),
  CONSTRAINT `fk12` FOREIGN KEY (`Hotel_Id`) REFERENCES `hotel` (`Hotel_Id
`)
)");

## <SQLiteResult>
##   SQL  CREATE TABLE `Room`  (
##   `Room_Id` int(11) NOT NULL,
##   `Reservation_Id` int(11) NULL DEFAULT NULL,
##   `Room_No` int(11) NULL DEFAULT NULL,
##   `Room_Name` varchar(255) DEFAULT NULL,
##   `Room_Beds` int(11) NOT NULL,
##   `Room_IsSmokingArea` Boolean,
##   `Room_Floor` varchar(255) NOT NULL,
##   `Room_Rate` decimal(8, 2) NOT NULL,
##   `Room_Availability` Boolean,
##   `Hotel_Id` int(11) NULL DEFAULT NULL,
##   PRIMARY KEY (`Room_Id`) ,
##   CONSTRAINT `fk11` FOREIGN KEY (`Reservation_Id`) REFERENCES `reservat
ion` (`Reservation_Id`),
##   CONSTRAINT `fk12` FOREIGN KEY (`Hotel_Id`) REFERENCES `hotel` (`Hotel
_Id`)
## )
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

#Creating table guests and dropping the table if it already exists

```
RSQLite::dbSendQuery(myconn, "DROP TABLE IF EXISTS `AdditionalGuest`");

## <SQLiteResult>
##   SQL  DROP TABLE IF EXISTS `AdditionalGuest`
##   ROWS Fetched: 0 [complete]
##        Changed: 0

RSQLite::dbSendQuery(myconn, "CREATE TABLE `AdditionalGuest`  (
  `AddGuest_Id` int(11) NOT NULL,
  `Reservation_Id` int(11) NULL DEFAULT NULL,
  `AddGuest_FirstName` varchar(255) DEFAULT NULL,
```

```
  `AddGuest_MiddleName` varchar(255) DEFAULT NULL,
  `AddGguest_LastName` varchar(255) DEFAULT NULL,
  `Room_id` int(11) NULL DEFAULT NULL,
  `AddGuest_DOA` varchar(255) DEFAULT NULL,
  `AddGuest_DOD` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`AddGuest_Id`),
  CONSTRAINT `fk13` FOREIGN KEY (`Reservation_Id`) REFERENCES `reservation
` (`Reservation_Id`),
  CONSTRAINT `fk14` FOREIGN KEY (`Room_Id`) REFERENCES `room` (`Room_Id`)
)");

## <SQLiteResult>
##   SQL  CREATE TABLE `AdditionalGuest`  (
##   `AddGuest_Id` int(11) NOT NULL,
##   `Reservation_Id` int(11) NULL DEFAULT NULL,
##   `AddGuest_FirstName` varchar(255) DEFAULT NULL,
##   `AddGuest_MiddleName` varchar(255) DEFAULT NULL,
##   `AddGguest_LastName` varchar(255) DEFAULT NULL,
##   `Room_id` int(11) NULL DEFAULT NULL,
##   `AddGuest_DOA` varchar(255) DEFAULT NULL,
##   `AddGuest_DOD` varchar(255) DEFAULT NULL,
##   PRIMARY KEY (`AddGuest_Id`),
##   CONSTRAINT `fk13` FOREIGN KEY (`Reservation_Id`) REFERENCES `reservat
ion` (`Reservation_Id`),
##   CONSTRAINT `fk14` FOREIGN KEY (`Room_Id`) REFERENCES `room` (`Room_Id
`)
## )
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

##list number of tables in the hotel management database

```
RSQLite::dbListTables(myconn)

## [1] "AdditionalGuest"         "Channel"
## [3] "Guest"                   "GuestTransaction"
## [5] "Hotel"                   "Invoice"
## [7] "Reservation"             "Room"
## [9] "Transaction_Description"
```

##Insert into Channel table

```
RSQLite::dbSendQuery(myconn, "INSERT INTO Channel VALUES
(1,'booking.com',1,'50.00')")

## <SQLiteResult>
##   SQL  INSERT INTO Channel VALUES
## (1,'booking.com',1,'50.00')
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "INSERT INTO Channel VALUES
(2,'alibaba.com',1,'60.00')")

## <SQLiteResult>
##   SQL  INSERT INTO Channel VALUES
```

```
## (2,'alibaba.com',1,'60.00')
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "INSERT INTO Channel VALUES
(3,'paytm.com',1,'55.00')")

## <SQLiteResult>
##   SQL  INSERT INTO Channel VALUES
## (3,'paytm.com',1,'55.00')
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "INSERT INTO Channel  VALUES
(4,'goibibo',1,'45.00')")

## <SQLiteResult>
##   SQL  INSERT INTO Channel  VALUES
## (4,'goibibo',1,'45.00')
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "INSERT INTO Channel VALUES
(5,'makemytrip',1,'90.00')")

## <SQLiteResult>
##   SQL  INSERT INTO Channel VALUES
## (5,'makemytrip',1,'90.00')
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "INSERT INTO Channel VALUES
(6,'wizz',1,'59.00')")

## <SQLiteResult>
##   SQL  INSERT INTO Channel VALUES
## (6,'wizz',1,'59.00')
##   ROWS Fetched: 0 [complete]
##        Changed: 1
```

##Insert into Guest table

```
RSQLite::dbSendQuery(myconn, "INSERT INTO Guest VALUES
(1,'Dushant','H.','Gohri','abc@gmail.com','634872932','','','heronbank','g
ibbet hill road','coventry','coventry','United Kingdom','CV47ES')")

## <SQLiteResult>
##   SQL  INSERT INTO Guest VALUES
## (1,'Dushant','H.','Gohri','abc@gmail.com','634872932','','','heronbank'
,'gibbet hill road','coventry','coventry','United Kingdom','CV47ES')
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "INSERT INTO Guest VALUES
(2,'Rashi','H.','singh','singh@gmail.com','634832932','','','House Number
-1256','Sector -14','Sonipat','Haryana','India','CV47AL')")
```

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

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

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

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

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

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

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

##Insert into GuestTransaction table

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

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

RSQLite::dbSendQuery(myconn, "insert into GuestTransaction values (2, 2, '
1575778667254323', '5/29/2020',1, '12/4/2022', '1089.3', '13/16/2019', '31
0');")
```

```
## <SQLiteResult>
##   SQL  insert into GuestTransaction values (2, 2, '1575778667254323', '
5/29/2020',1, '12/4/2022', '1089.3', '13/16/2019', '310');
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "insert into GuestTransaction values (3, 3, '
5475778667654323', '5/29/2020',1, '18/3/2022', '75889.51', '14/16/2019', '
30');")

## <SQLiteResult>
##   SQL  insert into GuestTransaction values (3, 3, '5475778667654323', '
5/29/2020',1, '18/3/2022', '75889.51', '14/16/2019', '30');
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "insert into GuestTransaction values (4, 4, '
9875778667654323', '5/29/2020',1, '19/5/2030', '8889', '15/16/2019', '360'
);")

## <SQLiteResult>
##   SQL  insert into GuestTransaction values (4, 4, '9875778667654323', '
5/29/2020',1, '19/5/2030', '8889', '15/16/2019', '360');
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "insert into GuestTransaction values (5, 1, '
1575778667654323', '5/29/2020',1, '30/4/2020', '43889.53', '12/16/2019', '
700');")

## <SQLiteResult>
##   SQL  insert into GuestTransaction values (5, 1, '1575778667654323', '
5/29/2020',1, '30/4/2020', '43889.53', '12/16/2019', '700');
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "insert into GuestTransaction values (6, 1, '
1575778667654323', '5/29/2020',1, '30/4/2020', '30888.53', '13/16/2019', '
1000');")

## <SQLiteResult>
##   SQL  insert into GuestTransaction values (6, 1, '1575778667654323', '
5/29/2020',1, '30/4/2020', '30888.53', '13/16/2019', '1000');
##   ROWS Fetched: 0 [complete]
##        Changed: 1

##Insert into hotel table

RSQLite::dbSendQuery(myconn,"INSERT INTO Hotel VALUES
(1,'abc hotels','https://www.abchotels.com/','6786785432','4th avenue','hi
ll road','new street','sulihull','bhg g09')")

## <SQLiteResult>
##   SQL  INSERT INTO Hotel VALUES
## (1,'abc hotels','https://www.abchotels.com/','6786785432','4th avenue',
'hill road','new street','sulihull','bhg g09')
```

```
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn,"INSERT INTO Hotel VALUES
(2,'def hotels','https://www.defhotels.com/','678672312','4th kingston','h
ill road','city center','spa','bhg g34')")

## <SQLiteResult>
##   SQL  INSERT INTO Hotel VALUES
## (2,'def hotels','https://www.defhotels.com/','678672312','4th kingston'
,'hill road','city center','spa','bhg g34')
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn,"INSERT INTO Hotel VALUES
(3,'geh hotels','https://www.gehhotels.com/','6786784534','4th algebra','h
ill road','wbs','london','bhg g84')")

## <SQLiteResult>
##   SQL  INSERT INTO Hotel VALUES
## (3,'geh hotels','https://www.gehhotels.com/','6786784534','4th algebra'
,'hill road','wbs','london','bhg g84')
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn,"INSERT INTO Hotel VALUES
(4,'hij hotels','https://www.hijhotels.com/','6786897898','4th road','hill
road','art center','warwick','bhg g56')")

## <SQLiteResult>
##   SQL  INSERT INTO Hotel VALUES
## (4,'hij hotels','https://www.hijhotels.com/','6786897898','4th road','h
ill road','art center','warwick','bhg g56')
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn,"INSERT INTO Hotel VALUES
(5,'klm hotels','https://www.klmhotels.com/','678623421','4th block','hill
road','sulihull','coventry','bhg g23')")

## <SQLiteResult>
##   SQL  INSERT INTO Hotel VALUES
## (5,'klm hotels','https://www.klmhotels.com/','678623421','4th block','h
ill road','sulihull','coventry','bhg g23')
##   ROWS Fetched: 0 [complete]
##        Changed: 1
```

##Insert into Transaction_Description table

```
RSQLite::dbSendQuery(myconn, "INSERT INTO Transaction_Description VALUES (
1,'Cash')")

## <SQLiteResult>
##   SQL  INSERT INTO Transaction_Description VALUES (1,'Cash')
##   ROWS Fetched: 0 [complete]
##        Changed: 1
```

```
RSQLite::dbSendQuery(myconn, "INSERT INTO Transaction_Description VALUES (
2,'Bar')")
```

```
## <SQLiteResult>
##   SQL  INSERT INTO Transaction_Description VALUES (2,'Bar')
##   ROWS Fetched: 0 [complete]
##        Changed: 1
```

```
RSQLite::dbSendQuery(myconn, "INSERT INTO Transaction_Description VALUES (
3,'Seminar Hall')")
```

```
## <SQLiteResult>
##   SQL  INSERT INTO Transaction_Description VALUES (3,'Seminar Hall')
##   ROWS Fetched: 0 [complete]
##        Changed: 1
```

```
RSQLite::dbSendQuery(myconn, "INSERT INTO Transaction_Description VALUES (
4,'Banquet')")
```

```
## <SQLiteResult>
##   SQL  INSERT INTO Transaction_Description VALUES (4,'Banquet')
##   ROWS Fetched: 0 [complete]
##        Changed: 1
```

```
RSQLite::dbSendQuery(myconn, "INSERT INTO Transaction_Description VALUES (
5,'Swimming pool')")
```

```
## <SQLiteResult>
##   SQL  INSERT INTO Transaction_Description VALUES (5,'Swimming pool')
##   ROWS Fetched: 0 [complete]
##        Changed: 1
```

```
RSQLite::dbSendQuery(myconn, "INSERT INTO Transaction_Description VALUES (
6,'Spa')")
```

```
## <SQLiteResult>
##   SQL  INSERT INTO Transaction_Description VALUES (6,'Spa')
##   ROWS Fetched: 0 [complete]
##        Changed: 1
```

##Insert into Invoice table

```
RSQLite::dbSendQuery(myconn, "INSERT INTO Invoice VALUES
(1,1,1,'dushant gohri',40,0,4,6,'2020-11-15')");
```

```
## <SQLiteResult>
##   SQL  INSERT INTO Invoice VALUES
## (1,1,1,'dushant gohri',40,0,4,6,'2020-11-15')
##   ROWS Fetched: 0 [complete]
##        Changed: 1
```

```
RSQLite::dbSendQuery(myconn, "INSERT INTO Invoice VALUES
(2,2,2,'rashi gohri',140,0,4,6,'2020-11-15')");
```

```
## <SQLiteResult>
##   SQL  INSERT INTO Invoice VALUES
## (2,2,2,'rashi gohri',140,0,4,6,'2020-11-15')
```

```
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "INSERT INTO Invoice VALUES
(3,1,1,'dushant gohri',40,0,4,6,'2020-11-15')");

## <SQLiteResult>
##   SQL  INSERT INTO Invoice VALUES
## (3,1,1,'dushant gohri',40,0,4,6,'2020-11-15')
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "INSERT INTO Invoice VALUES
(4,1,1,'dushant gohri',40,0,4,6,'2020-11-15')");

## <SQLiteResult>
##   SQL  INSERT INTO Invoice VALUES
## (4,1,1,'dushant gohri',40,0,4,6,'2020-11-15')
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "INSERT INTO Invoice VALUES
(5,1,1,'dushant gohri',40,0,4,6,'2020-11-15')");

## <SQLiteResult>
##   SQL  INSERT INTO Invoice VALUES
## (5,1,1,'dushant gohri',40,0,4,6,'2020-11-15')
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "INSERT INTO Invoice VALUES
(6,1,1,'dushant gohri',40,0,4,6,'2020-11-15')");

## <SQLiteResult>
##   SQL  INSERT INTO Invoice VALUES
## (6,1,1,'dushant gohri',40,0,4,6,'2020-11-15')
##   ROWS Fetched: 0 [complete]
##        Changed: 1
```

##Insert into Reservation table

```
RSQLite::dbSendQuery(myconn, "insert into Reservation values (1, 1, 1, 1,
'28/8/2020', '30/8/2020', 1, 'ut volutpat', 'non ligula', 'eu est congue',
'eget semper', 5576864946411916, '8/25/2020', '7/13/2020', 3)");

## <SQLiteResult>
##   SQL  insert into Reservation values (1, 1, 1, 1, '28/8/2020', '30/8/2
020', 1, 'ut volutpat', 'non ligula', 'eu est congue', 'eget semper', 5576
864946411916, '8/25/2020', '7/13/2020', 3)
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "insert into Reservation  values (2, 2, 2, 2,
'15/8/2020', '18/8/2020', 2, 'posuere cubilia', '', 'metus', 'magna at nun
c', 1061500478900525, '3/4/2020', '12/16/2019', 4)")
```

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

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

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

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

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

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

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

##Insert into Room table

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

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

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

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

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

RSQLite::dbSendQuery(myconn, "insert into Room  values (3, 3, 127098040, '
platea dictumst', 2, TRUE, 14, 3623, TRUE, 3)");

## <SQLiteResult>
##   SQL  insert into Room  values (3, 3, 127098040, 'platea dictumst', 2,
TRUE, 14, 3623, TRUE, 3)
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "insert into Room  values (4, 4, 12709804, 'p
latea dictumst', 1, FALSE, 14, 3876, TRUE, 4)");

## <SQLiteResult>
##   SQL  insert into Room  values (4, 4, 12709804, 'platea dictumst', 1,
FALSE, 14, 3876, TRUE, 4)
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "insert into Room  values (5, 5, 12709840, 'p
latea dictumst', 1, TRUE, 14, 36975, TRUE, 5)");

## <SQLiteResult>
##   SQL  insert into Room  values (5, 5, 12709840, 'platea dictumst', 1,
TRUE, 14, 36975, TRUE, 5)
##   ROWS Fetched: 0 [complete]
##        Changed: 1
```

##Insert into AdditionalGuest table

```
RSQLite::dbSendQuery(myconn, "insert into AdditionalGuest values (1, 1, 'N
ickey', 'enim', 'Chaize',1, '12/16/2019', '12/18/2019')");

## <SQLiteResult>
##   SQL  insert into AdditionalGuest values (1, 1, 'Nickey', 'enim', 'Cha
ize',1, '12/16/2019', '12/18/2019')
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "insert into AdditionalGuest values (2, 2, 'N
ickey', 'enim', 'Chaize',2, '12/16/2019', '12/18/2019')");

## <SQLiteResult>
##   SQL  insert into AdditionalGuest values (2, 2, 'Nickey', 'enim', 'Cha
ize',2, '12/16/2019', '12/18/2019')
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "insert into AdditionalGuest values (3, 3, 'N
ickey', 'enim', 'Chaize',3, '12/16/2019', '12/18/2019')");

## <SQLiteResult>
##   SQL  insert into AdditionalGuest values (3, 3, 'Nickey', 'enim', 'Cha
ize',3, '12/16/2019', '12/18/2019')
```

```
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "insert into AdditionalGuest values (4, 4, 'N
ickey', 'enim', 'Chaize',4, '12/16/2019', '12/18/2019')");

## <SQLiteResult>
##   SQL  insert into AdditionalGuest values (4, 4, 'Nickey', 'enim', 'Cha
ize',4, '12/16/2019', '12/18/2019')
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "insert into AdditionalGuest values (5, 5, 'N
ickey', 'enim', 'Chaize',5, '12/16/2019', '12/18/2019')");

## <SQLiteResult>
##   SQL  insert into AdditionalGuest values (5, 5, 'Nickey', 'enim', 'Cha
ize',5, '12/16/2019', '12/18/2019')
##   ROWS Fetched: 0 [complete]
##        Changed: 1

RSQLite::dbSendQuery(myconn, "insert into AdditionalGuest values (6, 6, 'N
ickey', 'enim', 'Chaize',5, '12/16/2019', '12/18/2019')");

## <SQLiteResult>
##   SQL  insert into AdditionalGuest values (6, 6, 'Nickey', 'enim', 'Cha
ize',5, '12/16/2019', '12/18/2019')
##   ROWS Fetched: 0 [complete]
##        Changed: 1
```

#using select command on hotel table to see if the insert command works properly and connection is established

```
RSQLite::dbGetQuery(myconn, "SELECT * FROM `hotel`")

##   Hotel_Id Hotel_name           Hotel_WebsiteUrl Hotel_Phone Hotel_Stre
etNumber
## 1        1 abc hotels https://www.abchotels.com/  6786785432         4t
h avenue
## 2        2 def hotels https://www.defhotels.com/   678672312       4th
kingston
## 3        3 geh hotels https://www.gehhotels.com/  6786784534        4th
algebra
## 4        4 hij hotels https://www.hijhotels.com/  6786897898
4th road
## 5        5 klm hotels https://www.klmhotels.com/   678623421          4
th block
##   Hotel_StreetName  Hotel_City Hotel_State Hotel_PostalCode
## 1        hill road  new street    sulihull          bhg g09
## 2        hill road city center         spa          bhg g34
## 3        hill road         wbs      london          bhg g84
## 4        hill road  art center     warwick          bhg g56
## 5        hill road    sulihull    coventry          bhg g23
```

#Disconnect to SQLite

RSQLite::dbDisconnect(myconn)

SQL queries

#Connect to SQLite

```
myconn <- RSQLite::dbConnect(RSQLite::SQLite(),"HotelManagement.db")
```

##The total spent for the customer for a particular stay (checkout invoice). • we have used two tables for storing transaction data, first the trnsaction table to store the customer transaction in hotel while he stays, after that if he check out he/she will get the final invoice containg all the transactions made at hotel. This sql query shows the total spent for each customer for a particular stay, which is defined as a specific payment date. The name combines first, midlle and last name of each customer. For example,

```
RSQLite::dbGetQuery(myconn, "Select
g.Guest_FirstName,g.Guest_MiddleName,g.Guest_LastName,(select sum(Invoice_
TotalAmount) from Invoice where Guest_Id = g.Guest_Id) as totalSpent
from Guest g");
```

```
##   Guest_FirstName Guest_MiddleName Guest_LastName totalSpent
## 1         Dushant               H.          Gohri        200
## 2           Rashi               H.          singh        140
## 3           Tanvi               H.         Sharan         NA
## 4           rohan               G.        harkara         NA
## 5           MANSI               H.         MAYANI         NA
```

The most valuable customers in (a) the last two months, (b) past year and (c) from the beginning of the records.

- Assuming the most valuable customers are the top 5 customers who spent most. The name combines first, middle, and last name of each customer along with their spent over past 2 months .

```
RSQLite::dbGetQuery(myconn, "select
g.Guest_FirstName,g.Guest_MiddleName,g.Guest_LastName,sum(i.Invoice_TotalA
mount) as totalSpent
from Guest g inner join Invoice i on g.Guest_Id = i.Guest_Id
where i.Invoice_Date  BETWEEN '2020-10-10' and '2020-12-10'
group by g.Guest_Id
order by sum(i.Invoice_TotalAmount) desc limit 5")
```

```
##   Guest_FirstName Guest_MiddleName Guest_LastName totalSpent
## 1         Dushant               H.          Gohri        200
## 2           Rashi               H.          singh        140
```

b) Assuming the most valuable customers are the top 5 customers who spent most. The name combines first, middle, and last name of each customer along with their spent over 1 year

```
RSQLite::dbGetQuery(myconn, "select
g.Guest_FirstName,g.Guest_MiddleName,g.Guest_LastName,sum(i.Invoice_TotalA
mount) as totalSpent
```

```
from Guest g inner join Invoice i on g.Guest_Id = i.Guest_Id
where i.Invoice_Date  BETWEEN '2019-12-10' and '2020-12-10'
group by g.Guest_Id
order by sum(i.Invoice_TotalAmount) desc limit 5")

##   Guest_FirstName Guest_MiddleName Guest_LastName totalSpent
## 1         Dushant               H.          Gohri        200
## 2           Rashi               H.          singh        140
```

(c) Assuming the most valuable customers are the top 5 customers who spent most. The name combines first, middle, and last name of each customer along with their spent from the beginning of the records..

```
RSQLite::dbGetQuery(myconn, "
select
g.Guest_FirstName,g.Guest_MiddleName,g.Guest_LastName,sum(i.Invoice_TotalA
mount) as totalSpent
from Guest g inner join Invoice i on g.Guest_Id = i.Guest_Id
group by g.Guest_Id
order by sum(i.Invoice_TotalAmount) desc limit 5")

##   Guest_FirstName Guest_MiddleName Guest_LastName totalSpent
## 1         Dushant               H.          Gohri        200
## 2           Rashi               H.          singh        140
```

##Which are the top countries where our customers come from ? • This SQL Query shows top 10 countries where customers come from, ordered by number of people travelled from these countries, Most of customers traveled from …..

```
RSQLite::dbGetQuery(myconn, "
select Guest_Country, count(Guest_Id) from Guest group by Guest_Country
order by count(Guest_Id) desc limit 10")

##    Guest_Country count(Guest_Id)
## 1 United Kingdom               2
## 2          India               1
## 3      Australia               1
## 4      AUSTRALIA               1
```

##How much did the hotel pay in referral fees for each of the platforms that we have contracted with?

```
RSQLite::dbGetQuery(myconn, "

select h.Hotel_name,c.Channel_Name,c.Channel_BookingFee
from Hotel h inner join Channel c on h.Hotel_Id = c.Hotel_Id")

##   Hotel_name Channel_Name Channel_BookingFee
## 1 abc hotels  booking.com                50
## 2 abc hotels  alibaba.com                60
## 3 abc hotels    paytm.com                55
## 4 abc hotels      goibibo                45
## 5 abc hotels   makemytrip                90
## 6 abc hotels         wizz                59
```

##What is the utilization rate for each hotel (that is the average billable days of a hotel specified as the average utilization of room bookings for the last 12 months)

```
RSQLite::dbGetQuery(myconn, "
select h.Hotel_name,r.Room_No,r.Room_Name,r.Room_Rate
from Room r inner join Hotel h on r.Hotel_Id = h.Hotel_Id")
```

```
##   Hotel_name     Room_No       Room_Name Room_Rate
## 1 abc hotels 12709804140 platea dictumst     36648
## 2 def hotels  1270980414 platea dictumst      3665
## 3 geh hotels   127098040 platea dictumst      3623
## 4 hij hotels    12709804 platea dictumst      3876
## 5 klm hotels    12709840 platea dictumst     36975
```

##Calculate the Customer Value in terms of total spent for each customer before the current booking.

```
RSQLite::dbGetQuery(myconn, "select g.Guest_FirstName,g.Guest_MiddleName,g
.Guest_LastName,sum(t.Trans_TotalPayment) as totalSpent
from Guest g inner join `GuestTransaction` t on g.Guest_Id = t.Guest_Id")
```

```
##   Guest_FirstName Guest_MiddleName Guest_LastName totalSpent
## 1         Dushant               H.          Gohri   569535.4
```

2. PART B -Semi Structured Data (Files)

Installing necessary packages for Data Management to work,

```
#install.packages("tidyverse")
#install.packages("janitor")

library("tidyverse")

library("readxl")
library("dplyr")
library("janitor")

library("tidyr")


library(plyr)

library(xml2)
library(stringr)
library(rebus)

library(XML)
library(readr)

#Listing all the directory inside the group_assignment folder

dirs<- list.dirs( "./group_assignment")


#intialise empty data frame

AllFilesPath = data.frame()


#using for loop to get all the files from one directory at a time and then
combining that file paths in one data frame

for(i in 1:length(dirs)){
  file_names <- list.files(dirs[i], pattern = "\\.xlsx$")


  #as we have one folder empty in the group_assignment folder so we have t
o ignore that file path, that's why we used If condition

    if(length(file_names)!=0){
      paste(dirs[i],file_names, sep="/") %>%
        data.frame() %>%
        rbind(AllFilesPath,.) -> AllFilesPath
    }
}


#intialise empty data frame
allFiles = data.frame()
```

```
#using for loop for every excel file, we will clean the file and bind the
file in  allFiles data frame
#using dyplr library we are using pipes so that we dont have to use so man
y variables

for ( ExcelFilePath in AllFilesPath$.){
   excel <- read_excel(ExcelFilePath, col_names = FALSE) %>%
     #removing 2nd column which is an empty column
.[,-2] %>%
 mutate(., Country=paste(.[5,2]),Flow=paste(.[3,2])) %>%
     #removing extra rows, so that it will be easy to bind all the excel f
iles together using Rbind
 .[-c(1,2,3,4,5,7),] %>%
     #removing NA values from data frame
 na.omit() %>%
     #changing to NA values if .. is in the excel data frame
  na_if(., "..") %>%
 rbind(allFiles,.) -> allFiles
}

#now we are going to use row_to_names so that columns will have a proper n
ame as per we need


allFiles <- row_to_names(allFiles,1)
#renaming columns names for 1, 67, 68nd index
names(allFiles)[1] <- "year"
names(allFiles)[67] <- "country"
names(allFiles)[68] <- "flow"


#now we are gonna use pivot_longer, so that every value is considered indi
vidual for every country , every year, every flow and every product

allFiles<- allFiles[allFiles$BKB != "BKB",] %>%  pivot_longer(-c(country,y
ear,flow),names_to ="product") %>%
  select(country, everything())

head(allFiles)

## # A tibble: 6 x 5
##   country year  flow    product                       value
##   <chr>   <chr> <chr>   <chr>                         <chr>
## 1 Albania 1971  Imports Additives/blending components 0
## 2 Albania 1971  Imports Anthracite                    <NA>
## 3 Albania 1971  Imports Aviation gasoline             0
## 4 Albania 1971  Imports BKB                           0
## 5 Albania 1971  Imports Biodiesels                    0
## 6 Albania 1971  Imports Biogases                      0

nrow(allFiles)

## [1] 573950
```

```r
#Initialise data_result dataframe from allFiles
data_result_new<-allFiles

#the total number of records on the dataset and the total number of record
s for each product across countries #across years using Group by and remov
ing NA values which are unique

data_product<-data_result_new %>%
  filter(!is.na(.)) %>%
  group_by(country, year, product) %>%
  mutate(total_value=sum(as.numeric(as.character(value)))) %>%
  select(country, year, product, total_value) %>%
  distinct()

head(data_product)
```

```
## # A tibble: 6 x 4
## # Groups:   country, year, product [6]
##   country year  product                        total_value
##   <chr>   <chr> <chr>                                <dbl>
## 1 Albania 1971  Additives/blending components          NA
## 2 Albania 1971  Anthracite                             NA
## 3 Albania 1971  Aviation gasoline                      NA
## 4 Albania 1971  BKB                                    NA
## 5 Albania 1971  Biodiesels                             NA
## 6 Albania 1971  Biogases                              NA
```

```r
nrow(data_product)
```

```
## [1] 101335
```

3. PART C – Semi Structured Data (Web)

Installing necessary packages for Data Management to work,

```
#install.packages("tidyverse")
#install.packages("janitor")
#install.packages("rvest")
#install.packages("httr")
#install.packages("XML")

library("tidyverse")
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse
1.3.0 ──
```

```
## ✓ ggplot2 3.3.2     ✓ purrr   0.3.4
## ✓ tibble  3.0.4     ✓ dplyr   1.0.2
## ✓ tidyr   1.1.2     ✓ stringr 1.4.0
## ✓ readr   1.4.0     ✓ forcats 0.5.0
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_confl
icts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library("dplyr")
library("janitor")
```

```
##
## Attaching package: 'janitor'
```

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

```
library("tidyr")
library("readxl")

library(leaflet)
library(plyr)
```

```
## ----------------------------------------------------------------------
-------
```

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

```
## ----------------------------------------------------------------------
-------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following object is masked from 'package:purrr':
##
##     compact

library(rvest)

## Loading required package: xml2

##
## Attaching package: 'rvest'

## The following object is masked from 'package:purrr':
##
##     pluck

## The following object is masked from 'package:readr':
##
##     guess_encoding

library(httr)
library(stringr)
library(rebus)

##
## Attaching package: 'rebus'

## The following object is masked from 'package:stringr':
##
##     regex

## The following object is masked from 'package:ggplot2':
##
##     alpha

library(readr)
library(xml2)
library(XML)

##
## Attaching package: 'XML'

## The following object is masked from 'package:rvest':
##
##     xml
```

Created a for loop again for reading the excel file individually as all the excel files are similar we can also apply necessary operations like removing a column/row in the same loop, so that we can have a structured data frame of all the data

now adding the country and flow column in the dataframe and removing duplicate row instances from the data frame.

```
finally, we have the data frame of our need, we need to apply the pivot longer function on the data #frame and count the actual rows produced with all the excel data

#using pivot_wider function to find out the number of records for each product across countries across #years

# With the given link in the assignment, scrapping all the links for XML data
url <- "https://www.food.gov.uk/uk-food-hygiene-rating-data-api"
# store all xml link
url %>%
  read_html() %>%
  html_nodes(xpath = "//*/article/div/div/p[5]/a") %>%
  html_attr("href") %>%
  read_html() %>%
  html_nodes(xpath = "//*/article/dl[2]/dd/strong/a") %>%
  html_attr("href") -> all.xml.links

head(all.xml.links)

## [1] "http://ratings.food.gov.uk/"
## [2] "http://ratings.food.gov.uk/OpenDataFiles/FHRS760en-GB.xml"
## [3] "http://ratings.food.gov.uk/OpenDataFiles/FHRS761en-GB.xml"
## [4] "http://ratings.food.gov.uk/OpenDataFiles/FHRS323en-GB.xml"
## [5] "http://ratings.food.gov.uk/OpenDataFiles/FHRS055en-GB.xml"
## [6] "http://ratings.food.gov.uk/OpenDataFiles/FHRS062en-GB.xml"
```

Download all XML files through link for every row (city)

```
#Checking if the directory exists
output_dir <- file.path("./xml_files")

if (!dir.exists(output_dir)){
dir.create(output_dir)
} else {
    print("Dir already exists!")
}

## [1] "Dir already exists!"

#Initialise empty data frame for storing all the xml file path in current directory
allxmlfilespath = data.frame()

for (i in 2:length(all.xml.links)) {
  url <- all.xml.links[i]
  #splitting the link to get the filename
  filename_vector <- strsplit(all.xml.links[i],split="/")[[1]]
  filename <- filename_vector[length(filename_vector)]
  filePath <- paste0("xml_files/",filename)
  wholefilePath <- paste0("./", filePath)
  #checking if the file already exists in the directory then don't downloa
```

```
d the file again
  if(!file.exists(wholefilePath)) {
    print(wholefilePath)
    #downloading XML file to local directory
    download.file(url,filePath)
  }
  allxmlfilespath <- rbind(allxmlfilespath,wholefilePath)
  #renaming the colName of dataframe to more meaningful name
  names(allxmlfilespath)[1] <- "wholeFilePath"
}

#initialise an empty Data Frames

dfHygiene<- data.frame()
dfStructural<- data.frame()
dfConfidenceInManagement<- data.frame()
total<- data.frame()
result <- data.frame()

append_xml_nodes <- function(xml, node_name) {
  nodes = getNodeSet(xml, str_interp("//EstablishmentDetail/Scores[not(${n
ode_name})]"))
  sapply(nodes, function(node) node[[str_interp("${node_name}")]] = "")
}

for(wholeFilePath in allxmlfilespath$wholeFilePath) {
# LOADING TRANSFORMED XML INTO R DATA FRAME

  # (xpathSApply(wholeFilePath, "//ItemCount", xmlValue)) -> count
  # total <- rbind(total,count)

xml<-xmlParse(wholeFilePath)
  append_xml_nodes(xml, "Hygiene")
  append_xml_nodes(xml, "Structural")
  append_xml_nodes(xml, "ConfidenceInManagement")

#extracting data from xml using XmltoDataFrame

xmldf <- xmlToDataFrame(nodes = getNodeSet(xml, "//EstablishmentDetail"))
%>%
  select(., -Geocode) %>% select(., -Scores)

dfGeocode <- xmlToDataFrame(nodes = getNodeSet(xml, "//Geocode") ) %>%
  bind_cols(xmldf, .) %>%
  bind_rows(result,.) -> result

colHygiene <- xmlToDataFrame(nodes = getNodeSet(xml, "//Hygiene") ) %>%
  bind_rows(dfHygiene, .) -> dfHygiene

colStructural <- xmlToDataFrame(nodes = getNodeSet(xml, "//Structural") )
%>%
  bind_rows(dfStructural, .) -> dfStructural

colConfidenceInManagement <- xmlToDataFrame(nodes = getNodeSet(xml, "//Con
```

```
fidenceInManagement") ) %>%
  bind_rows(dfConfidenceInManagement, .) -> dfConfidenceInManagement


}

#changing the column names to more meaningful name
colnames(dfHygiene)[1] <- "Hygiene"
colnames(dfStructural)[1] <- "Structural"
colnames(dfConfidenceInManagement)[1] <- "ConfidenceInManagement"
#colnames(total)[1] <- "Observations"

# binding all the dataframe to result dataframe, so that we can have more
cleaner look
result<- bind_cols(list(result, dfHygiene, dfStructural, dfConfidenceInMan
agement)) %>%
  na_if(.,"")

#converting Longitude and latitude to Numeric data type
result$Latitude <- as.numeric(as.character(result$Latitude))
result$Longitude <- as.numeric(as.character(result$Longitude))

#saving the Xml data in Rdata file for Part D continuation, so that we don
t have to render the data again from web
save(result,file="data_result_C.RData")


#To check ItemCount in all XML ratings link matches number of rows
#numberOfObservation <- sum(as.numeric(total$Observations))



head(result)
```

```
##    FHRSID LocalAuthorityBusinessID              BusinessName
## 1  982849                EHDC12463 1 & 30 DONALD DEWAR COURT
## 2  592387                EHDC10723    1906 RESTAURANT AT HMT
## 3 1017312                EHDC12570          2 BROTHERS PIZZA
## 4  593681                 EHDC9793                210 BISTRO
## 5  997643                EHDC12441               210 CABOOSE
## 6  593037                 EHDC4774                  22 CLUB
##                          BusinessType BusinessTypeID              Addre
ssLine2
## 1 Hospitals/Childcare/Caring Premises             5 1 & 30 Donald Dewa
r Court
## 2            Restaurant/Cafe/Canteen             1         Rosemount
Viaduct
## 3             Takeaway/sandwich shop          7844       Great Northe
rn Road
## 4            Restaurant/Cafe/Canteen             1         210 Market
Street
## 5            Restaurant/Cafe/Canteen             1          37 Waterl
oo Quay
## 6                  Pub/bar/nightclub          7843            55 Rose
```

```
Street
##           AddressLine3 AddressLine4 PostCode          RatingValue
## 1 Provost Fraser Drive     Aberdeen AB16 5JB                 Pass
## 2             Aberdeen         <NA> AB25 1GL                 Pass
## 3             Aberdeen         <NA> AB24 2DU                 Pass
## 4             Aberdeen         <NA> AB11 5PQ Improvement Required
## 5             Aberdeen         <NA> AB11 5BS                 Pass
## 6             Aberdeen         <NA> AB10 1UB                 Pass
##                        RatingKey RatingDate LocalAuthorityCode
## 1                fhis_pass_en-GB 2017-05-15                760
## 2                fhis_pass_en-GB 2020-02-20                760
## 3                fhis_pass_en-GB 2019-10-30                760
## 4 fhis_improvement_required_en-GB 2018-12-18                760
## 5                fhis_pass_en-GB 2019-01-09                760
## 6                fhis_pass_en-GB 2014-03-11                760
##   LocalAuthorityName          LocalAuthorityWebSite
## 1      Aberdeen City http://www.aberdeencity.gov.uk
## 2      Aberdeen City http://www.aberdeencity.gov.uk
## 3      Aberdeen City http://www.aberdeencity.gov.uk
## 4      Aberdeen City http://www.aberdeencity.gov.uk
## 5      Aberdeen City http://www.aberdeencity.gov.uk
## 6      Aberdeen City http://www.aberdeencity.gov.uk
##       LocalAuthorityEmailAddress SchemeType NewRatingPending    Address
Line1
## 1 commercial@aberdeencity.gov.uk       FHIS            False
<NA>
## 2 commercial@aberdeencity.gov.uk       FHIS            False
<NA>
## 3 commercial@aberdeencity.gov.uk       FHIS            False
699B
## 4 commercial@aberdeencity.gov.uk       FHIS            False
<NA>
## 5 commercial@aberdeencity.gov.uk       FHIS            False Provender
House
## 6 commercial@aberdeencity.gov.uk       FHIS            False
<NA>
##   Longitude Latitude RightToReply Hygiene Structural ConfidenceInManage
ment
## 1 -2.167495 57.16131         <NA>    <NA>       <NA>
<NA>
## 2 -2.104965 57.14816         <NA>    <NA>       <NA>
<NA>
## 3 -2.138175 57.17238         <NA>    <NA>       <NA>
<NA>
## 4 -2.092258 57.14228         <NA>    <NA>       <NA>
<NA>
## 5 -2.084367 57.14587         <NA>    <NA>       <NA>
<NA>
## 6 -2.111912 57.14456         <NA>    <NA>       <NA>
<NA>
```

```
nrow(result)
```

```
## [1] 596285
```

4. PART D – Dashboard with R/Shiny

Installing necessary packages for Data Management to work,

```
#install.packages("tidyverse")
#install.packages("plotly")
#install.packages("shinydashboard")
#install.packages("anchors")
#install.packages("shiny")
#install.packages("ggplot2")
#install.packages('devtools')
library(devtools)

#install_github("ramnathv/rCharts")

library("plyr")
library(plotly)

library(shinydashboard)

library(anchors)

library(shiny)
library(ggplot2)
library(rCharts)
library(leaflet)
```

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

```
#data_result_C <- get(load("./data_result_C.RData"))
data_result_C <- readRDS("./data_result_C.rds")


bb_data <- data_result_C
##Change the class of LocalAuthorityName an RatingValue to prepare for the
following operation and to make it more proper way to show in the table

bb_data$LocalAuthorityName <- as.factor(bb_data$LocalAuthorityName)
bb_data$RatingValue <- as.factor(bb_data$RatingValue)

ui <- dashboardPage(
  dashboardHeader(title = "Food Hygiene Ratings Scheme Data"),
    dashboardSidebar(
    sidebarMenu(
      menuItem("Data", tabName = "dataset"),
      menuItem("Presence in UK",tabName = "chart_1"),
      menuItem("Location Wise Ratings",tabName = "chart_2")
    )),
  dashboardBody(
    tabItems(
      # first tab named "dataset"
      tabItem(tabName = "dataset",
              fluidRow(h1("DATASET")),
              fluidRow(column(width = 12,DT::dataTableOutput("resulttable"
```

```
)))
              ),

      # second tab named "chart_1"
      tabItem(tabName = "chart_1",
              fluidRow(column(width=12, h1("Map"))),
              fluidPage( navbarPage("", id="main", tabPanel("Map", leaflet
Output("bbmap", height=1000))))
              ),
      #third tab named "chart_2"
    tabItem(tabName = "chart_2",
            fluidRow(column(width = 12, h1("Local Authority Wise Ratings")
)),
            fluidRow(column(width = 10, selectInput("LocalAuthorityName","
select Local Authority",choices = c("Select", levels(bb_data$LocalAuthorit
yName)), selected = "Select")
                                ),
            fluidRow(column(width = 12,plotlyOutput("ratingsdata")))
            )
    )
    )
  )

server<-function(input, output) {

  changefactor <- c("AwaitingInspection" = "Awaiting Inspection", "Awaitin
gPublication" = "Awaiting Publication")
  bb_data$RatingValue <- revalue(bb_data$RatingValue, changefactor)
  bb_data<-bb_data %>% filter(!is.na(Latitude)) %>% filter(!is.na(Longitud
e))
  sapply(bb_data, class)

  output$resulttable = DT::renderDataTable(bb_data)

  # create a color for different rating value
 Col <- colorFactor(topo.colors(12), bb_data$RatingValue)

  output$bbmap <- renderLeaflet({
    leaflet(bb_data) %>%
      addTiles() %>%
      addCircles(lng = ~Longitude, lat = ~Latitude,
          color = ~Col(RatingValue),
          radius = 2,
          stroke = T,
          fillOpacity = 1,
          popup=paste(
            "Rating:", bb_data$RatingValue, "<br>",
            "Business:", bb_data$BusinessName, "<br>",
            "Area:", bb_data$LocalAuthorityName, "<br>"))
  })

    output$ratingsdata <- renderPlotly({
      if(input$LocalAuthorityName=="Select"){
        output<-ggplot((bb_data),aes(x=RatingValue))+geom_bar()
```

```
        scale_fill_hue(l=40, c=35)
        ggplotly(output)
      }
      else{
        to_filter <- input$LocalAuthorityName
        output <- ggplot(subset(bb_data,LocalAuthorityName==to_filter),aes
(x=RatingValue))+geom_bar()
        scale_fill_hue(l=40, c=35)
        ggplotly(output)}

  })

}

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

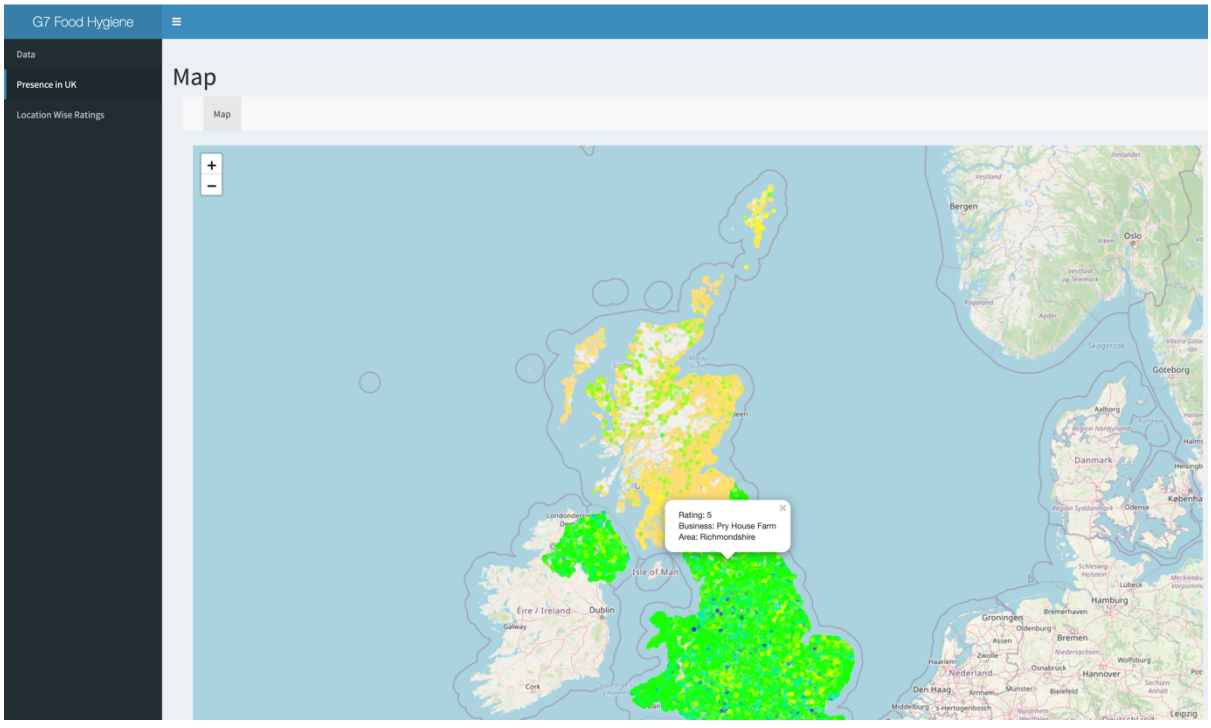

Figure 3: Presence in UK

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

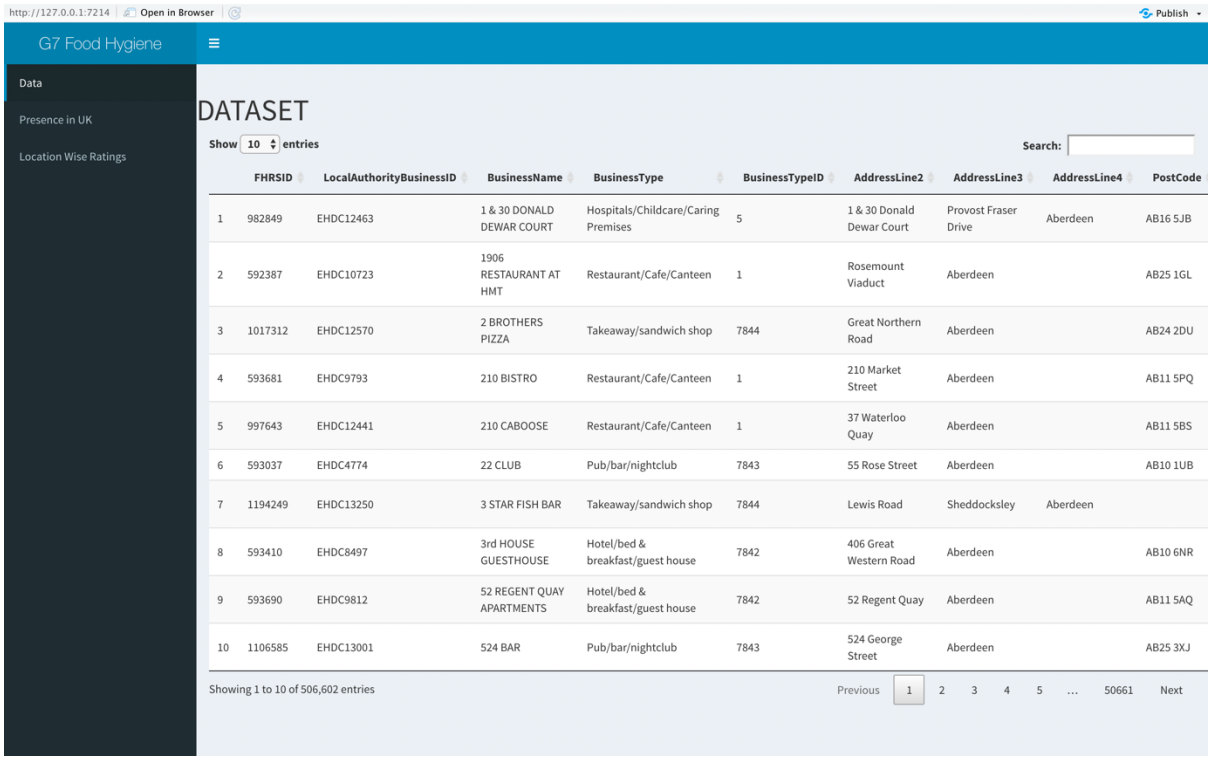

Figure 4: Food Hygine Dataset

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.

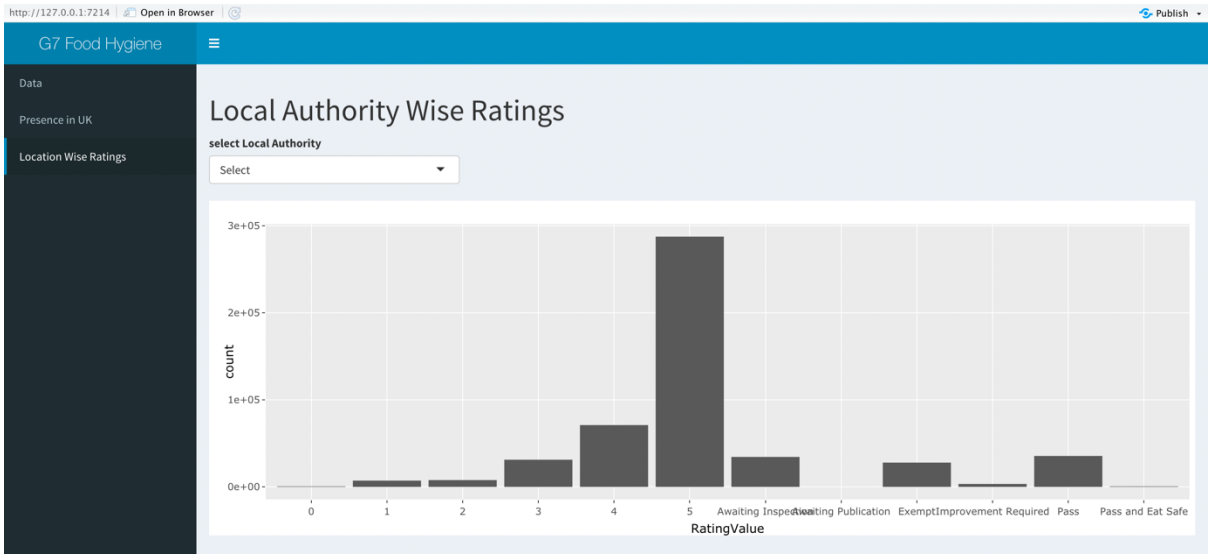

Figure 5:Location Wise Ratings

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

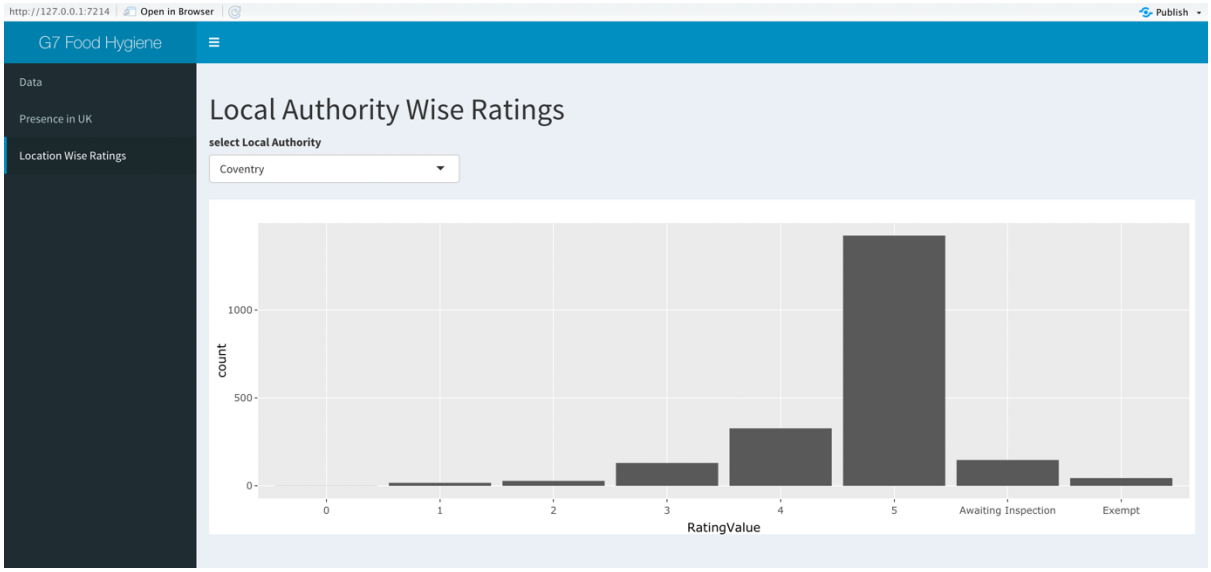

Figure 6:LocationWise Data Coventry

REFERENCES

RSQLite

(Wickham *et al.*, 2015)

Wickham, H. *et al.* (2015) 'Package " RSQLite "', *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*.

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)

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).

Shihnydashboard

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

Rchars

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).