Module IB9HP0
Data Management
Group Assignment

MSBA Group 28 Student ID(s):

2117597

2135793

2138027

2152449

2135744

Word Count: 1989 (excluding codes)

This is to certify that the work I am submitting is my own. All external references and sources are clearly acknowledged and identified within the contents. I am aware of the University of Warwick regulation concerning plagiarism and collusion.

No substantial part(s) of the work submitted here has also been submitted by me in other assessments for accredited courses of study, and I acknowledge that if this has been done an appropriate reduction in the mark I might otherwise have received will be made.

Part A1

This is to certify that the work I am submitting is my own. All external references and sources are clearly acknowledged and identified within the contents. I am aware of the University of Warwick regulation concerning plagiarism and collusion.

No substantial part(s) of the work submitted here has also been submitted by me in other assessments for accredited courses of study, and I acknowledge that if this has been done an appropriate reduction in the mark I might otherwise have received will be made.

Part A1

MSBA Group 28

18/11/2021

```
WNB_HOTEL_GROUP <- dbConnect(RSQLite::SQLite(), "WNB_HOTEL_GROUP.sqlite")
```

WNB HOTEL GROUP

Given WNB HOTEL GROUP request on providing a system for optimizing its revenue. The Entity-relationship diagram illustrates how every entity to which we want to compile data such as hotels, guests, rooms, types of rooms, services, invoices, items_invoices, reservations, and booking channels are related to each other within the hotel chain. More details are as follows:

- *hotels*
  - Information about hotels of the chain
- *services*
  - Each hotel provides additional services
- *service_by_hotel*
  - Services provided by each hotel
- *rooms*
  - General information about the rooms provided for each hotel including whether it's occupied or not(is_occupied where, 1=yes 0=no)
- *room_class*
  - Specific characteristics for each room #beds as size and smoking area (room_smoking where, 1=yes 0=no)
- *reservations*
  - Details of reservations(including credit card details, preferences(such as, smoking / non-smoking, number of beds, high or low floor), and reservation status where, 1=pending and 0=cancelled)
- *bookings*
  - Reservations already confirmed
- *booking_channels*
  - Information about booking channels (commissions, fee, payment due date)
- *invoices*

  - Information about services used and paid invoices (invoice_paid where, 1=yes 0=no)

- *payment_details*
  - Information about services printed on each invoice, amount_items(cost+tax-payment) which should be =0 once paid when checking out.

- **Context-ER DIAGRAM EXPLANATION**

Each hotel owns a variety of rooms (double/single) 1:N relationship (A hotel can own many rooms, but a room can not be owned by many hotels). It also offers the possibility to use different services (rental of phone, use of facilities) to its customers (M:N pointing out that many hotels offer many services, and a service can be provided by many hotels). As a result, we would like to maintain records on which services are available at which hotel (creating a relationship-table called service_by_hotels) after having assumed the M: N cardinality between hotels and services. Following that, a weak entity connected to rooms called "room-class", a 1:M cardinality, to which we will store information regarding the size (number of beds 1 or 2), and whether it's smoking(indicated as numeric, 1) or no-smoking area(indicated as numeric, 0) to eliminate redundancy from rooms table which will have room-class_ID as a foreign key. Financially speaking, it is also cost-effective to know whether the room is occupied or not, so that records of utilization can be tracked (this is indicated as an additional column in the rooms entity). It enables the business to make management decisions for future plans, and keep on track how popular the service is that is being offered.

For every guest who is visiting and using additional services, all charges (including accommodation, additional services, use of facilities) are charged to the room, then reflected in the check-out invoice. Payment_details acts as the relationship linking Invoices and Services. At the same time, invoices holds each unique booking ID. Therefore, the total invoice will have a full description of the services used by the customer, and methods of payment when checking out. Payment_details relationship table was generated after having established a M:N cardinatily of services and invoices (a service can be reflected on multiple invoices and an invoice can display different services). At the same time, invoices entity contains the balance of charges, taxes and the amount paid (charges + taxes - amount paid) which is reflected in a column called, item amount specifying the details of the invoice.

NOTE: all amount fees are paid when checking out fully. Since every hotel can offer different services, the amount of fee per service will depend on each hotel fee whereas for rooms all costs are equal across hotels.

Regarding guests entity, it is represented as another principal connection in which we are keeping all personal details of visitors. This data is tracked historically by each hotel. It provides crucial attributes such as full name, residence address, and email and phone number records that are non-compulsory to fill out during the registration process. Following that, the cardinality 1:M reflects that one guest can make multiple reservations, but a reservation can only be processed by one guest ID. Likewise, reservations status can be either pending or cancelled. Additionally,it's mandatory to provide credit card details to secure the reservation (NOT NULL). Finally, once the hotel checks all requirements, chooses floor, smoking area according to the client's preferences, then it's reflected on bookings.

Moving on through the ER diagram, those reservations pass to become bookings (1:1 it's just a record transaction). Every booking has a unique ID (booking ID). Subsequently, an invoice is generated for each booking id, regardless of whether it belongs to the same customer. As a result of the M:N relationship within guests and rooms a bookings table tracks historical information of all guests who was accommodated in the the whole hotel's rooms.

The hotel has to record the booking channels to which every customer processed their reservations (M:1 many reservations are secured through one booking channel, and one booking channel can have many reservations), each reservation will have booking_channel_ID as a foreign key. Then,we have the percentage of commission and the channel fee for each channel (Booking.com, Hotels.com, Tripadvisor.com, etc.). The total amount that will be transferred to each third party is calculated based on the total charges excluding taxes, multiplied by the percentage of commission of each booking channel.

Please refer to Appendix Part A1

- **SQL QUERIES**

1.The total spent for the customer for a particular stay (checkout invoice).

As we have made the following assumption: we have item_amount which contains the balance (charges+taxes-amountpaid) , then item_amount column from invoice items entity contains these three values on different rows. We applied sum function for item_amount and multiply by -1 to generate a positive value.

```
SELECT p1.guest_id , p1.departure_date,
SUM(p3.item_amount)*-1 AS "Total_Spent"
FROM invoices AS p2
INNER JOIN payment_details AS p3
ON p2.invoice_id = p3.invoice_id
INNER JOIN bookings AS p1
ON p2.booking_id = p1.booking_id
WHERE p3.item_amount<0
GROUP BY guest_Id
```

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

Here, we have the same explanation as above for total spent. Assuming that Today's date is 2021-11-19 00:00:00 , we can use BETWEEN to calculate the total spent in the last two months. Also,we have to consider bookings instead of reservations since that is the real revenue (all booking are confirmed, not longer reservations).Finally, everything has been ordered by total spent descendingly.

```
-- most valuable customers in (a) the last two months TOP 20

----Today's date 2021-11-19 00:00:00

SELECT p3.guest_id,
 (sum(p1.item_amount)*-1) AS "TOTAL_SPENT"
FROM payment_details AS p1
INNER JOIN invoices  AS p2
ON p1.invoice_id = p2.invoice_id
INNER JOIN bookings AS p3
ON p2.booking_id= p3.booking_id
WHERE p2.invoice_paid = 1 AND p3.arrival_date BETWEEN DATETIME("2021-09-19 00:00:00") AND DATETIME("202
GROUP BY  p3.guest_id
ORDER BY TOTAL_SPENT desc
LIMIT 20 ;
```

```
-- most valuable customers in (b) the past year  TOP 20

SELECT p3.guest_id,
 (sum(p1.item_amount)*-1) AS "TOTAL_SPENT"
FROM payment_details AS p1
INNER JOIN invoices  AS p2
ON p1.invoice_id = p2.invoice_id
```

```
INNER JOIN bookings AS p3
ON p2.booking_id= p3.booking_id
WHERE p2.invoice_paid = 1 AND p3.arrival_date BETWEEN DATETIME("2020-11-19 00:00:00") AND DATETIME("202
GROUP BY  p3.guest_id
ORDER BY TOTAL_SPENT desc
LIMIT 20
;
```

```
-- most valuable customers (c) beginning of the records  TOP 20

SELECT p3.guest_id,
 (sum(p1.item_amount)*-1) AS "TOTAL_SPENT"
FROM payment_details AS p1
INNER JOIN invoices  AS p2
ON p1.invoice_id = p2.invoice_id
INNER JOIN bookings AS p3
ON p2.booking_id= p3.booking_id
WHERE p2.invoice_paid = 1 AND item_amount <0
GROUP BY  p3.guest_id
ORDER BY TOTAL_SPENT desc
LIMIT 20
;
```

3.Which are the top countries where our customers come from ?

This query can be run by selecting columns from the guests entity: guest_country and guest_id, counting each country, and finally ordering descendingly.

```
---TOP 10 COUNTRIES

SELECT p1.guest_country AS "Country" ,COUNT(p1.guest_id) AS "total_guests"
FROM guests AS p1
GROUP BY p1.guest_country
ORDER BY total_guests desc
LIMIT 10;
```

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

On our database referral fees are in the column called booking_channel_commission % from the booking channels entity.

We are assuming that we pay the % of commissions based on the item amount without taxes.Therefore, we have to multiply the % of commission (booking_channel_commission) by the total item amount. Finally, as the balance item_amount is <0, we have to multiply the sum by -1 so as to generate a positive value.

```
SELECT p1.booking_channel_id,
p1.booking_channel_name,
(SUM (p1.booking_channel_commission) *(p4.item_amount))*-1 AS Referral_Fees
```

```
FROM booking_channels AS p1
INNER JOIN reservations AS p2
ON p2.booking_channel_id  = p1.booking_channel_id
INNER JOIN invoices AS p3
ON p3.booking_id = p2.booking_id
INNER JOIN payment_details AS p4
ON p3.invoice_id = p4.invoice_id
WHERE p3.invoice_paid =1 AND p4.is_tax=0 AND p4.item_amount<0
GROUP BY p1.booking_channel_id
ORDER BY Referral_Fees desc
;
```

5. 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)

The utilization rate will be # of rooms which are occupied (room_occupied='1') divided by the total of rooms either occupied or not (room_occupied)

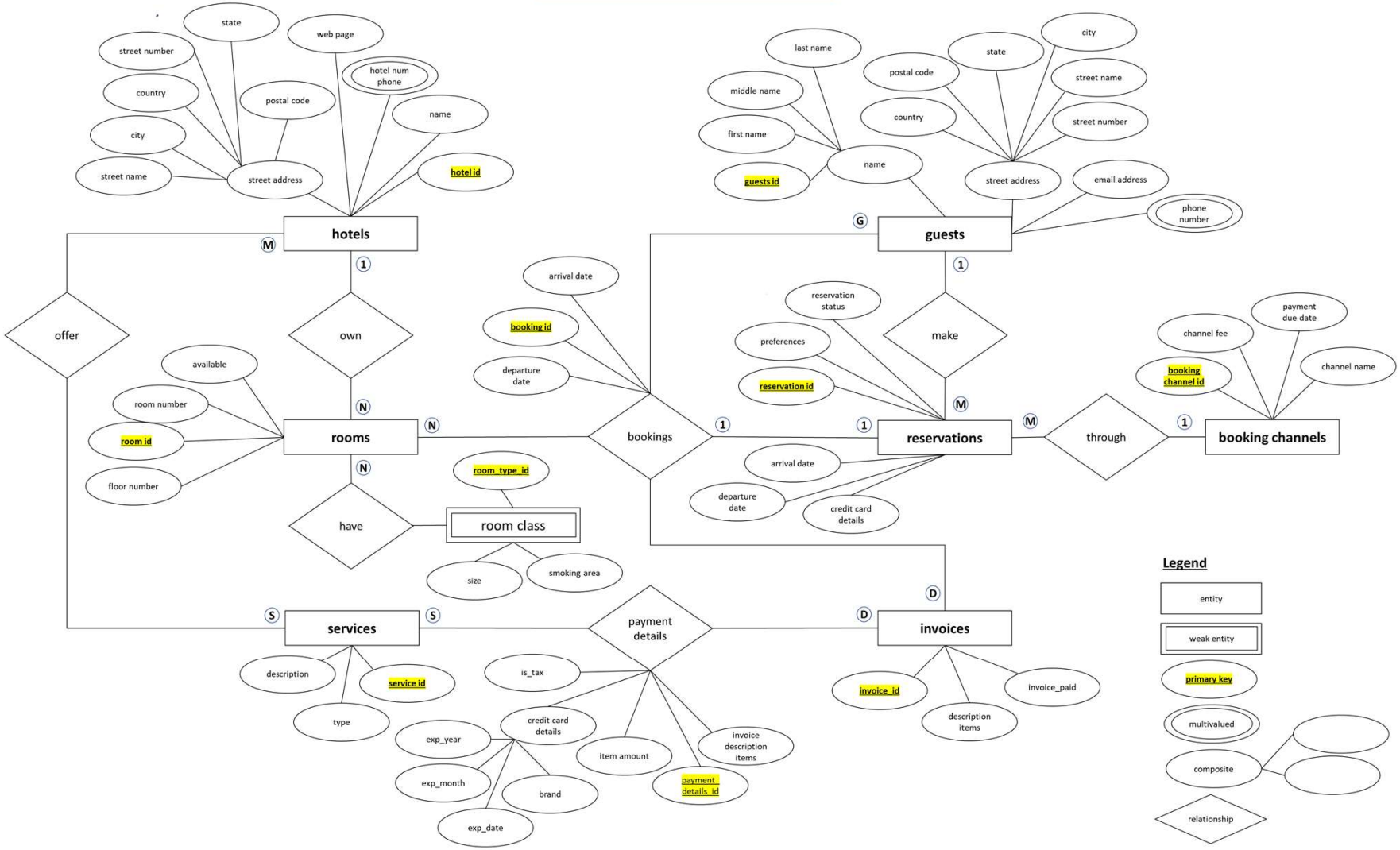
```
----Today's date 2021-11-19 00:00:00

SELECT p2.hotel_id,(count(p2.room_occupied='1'))/(count(p2.room_occupied)) AS "RATE_UTILIZATION"
FROM rooms AS p2
INNER JOIN bookings AS p1
ON p2.room_id= p1.room_id
WHERE p1.arrival_date BETWEEN DATETIME("2020-11-19 00:00:00") AND DATETIME("2021-11-19 00:00:00")
GROUP BY p2.hotel_id
ORDER BY RATE_UTILIZATION DESC
;
```

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

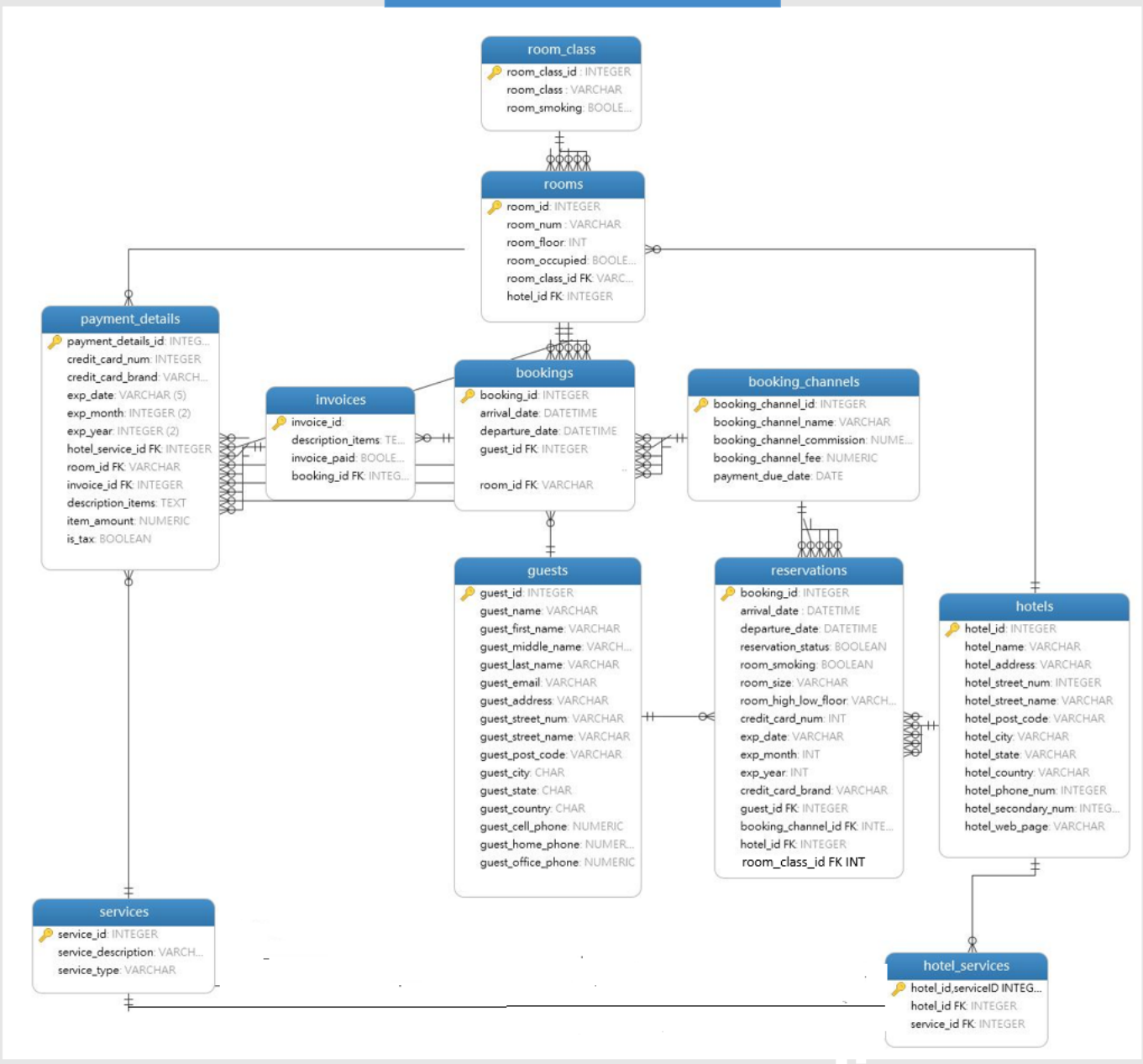
To calculate this particular task we calculated the total item amount (the balance described previously) and filter by the arrival dates before the current date which is "now". It takes all previous visits from each customer without the current booking.

```
SELECT p3.guest_id,
(sum(p1.item_amount)*-1) AS "TOTAL_SPENT"
FROM payment_details AS p1
INNER JOIN invoices AS p2
ON p1.invoice_id = p2.invoice_id
INNER JOIN bookings AS p3
ON p2.booking_id= p3.booking_id
WHERE p2.invoice_paid = 1 AND  p3.arrival_date < DATETIME("now") AND p1.item_amount <0
GROUP BY p3.guest_id
ORDER BY TOTAL_SPENT DESC
;
```

WNB_HOTEL_GROUP
RELATIONAL MODEL



WNB_HOTEL_GROUP
PHYSICAL SCHEMA



Part A2

This is to certify that the work I am submitting is my own. All external references and sources are clearly acknowledged and identified within the contents. I am aware of the University of Warwick regulation concerning plagiarism and collusion.

No substantial part(s) of the work submitted here has also been submitted by me in other assessments for accredited courses of study, and I acknowledge that if this has been done an appropriate reduction in the mark I might otherwise have received will be made.

PART A2

```
A2 <- dbConnect(RSQLite::SQLite(), "A2.sqlite")
```

A2 Logical data Schema

- **customers**(*cust_id*,customer_name, customer_first_name, customer_middle_name,customer_last_name, customer_cell_phone)
- **cars**(*car_id*,car_brand, car_is_new, car_color, car_mileage, car_age)
- **dealers**(*dealer_id*,dealer_name, dealer_address)
- **car_purchase_orders**(*po_id* ,*customer_id*, *car_id*, *dealer_id*,purchase_at, payment)
- **package_purchases**(*package_id*, *po_id*,package_fee, start_at, end_at)
- **repair_records**(*repair_id*, *dealer_id*, *po_id*, mileage, repair_at, labor_cost, spare_cost)

Like all other businesses, we need to record customer information, such as customer names and contact information. Thus, we need to create a 'customer' entity. Each car information, including brand, model, color, and so on, should be recorded in the 'car' entity. There are many dealers with different names and addresses available for customers to choose. So we include a 'dealer' entity to record different dealers' information. The dealer provides two kinds of service - car sales and car repair. Thus, we need two entities, 'car_purchase_order' and 'repair record' to record information for these two businesses. For car purchases, we can use customer_id, car_id, and dealer_id to locate a specific purchasing order. Package purchase is attached with the purchasing order. Thus, we use po_id as a foreign key for 'package_purchase entity'. It records the price and valid period of a service package. We assume we can get all dealers' information, then, for the 'repair_record' entity, a repair action must be after a purchase action. Thus, we use 'dealer_id' and 'po_id' as foreign keys to locate a repair record.

How many customers have stopped bringing their cars after the first encounter with the dealer?

Using po_id and dealer_id to combine the car_purchase_order and repair_record. We would like to count how many customers have stopped bringing their cars after the first encounter with the dealer. In other words, it includes two groups of customers: 1) Customers who purchased the car but never repair the car. 2) Customers who purchased cars from one dealer but repaired cars in another dealer. Since we use the LEFT JOIN command, these two kinds of customers will have the "NULL" value for repair_id. Thus, we use 'WHERE repair_id IS NULL' to select the needed values. A customer may purchase multiple cars, which will lead to a cutomer_id attached with many po_id. Thus, we use 'DISTINCT customer_id' to ensure we only count each customer once.

```sql

SELECT count(DISTINCT customer_id) AS "#CustomerChurn"
FROM (
SELECT customer_id, rr.dealer_id, car_id
FROM car_purchase_order AS cp
JOIN repair_record AS rr
ON cp.po_id = rr.po_id
GROUP BY customer_id, rr.dealer_id, car_id
HAVING count(DISTINCT repair_id)=1 )
```

Table 1: 1 records

#CustomerChurn
0

What is the relationship between the price of the service and the age of the car in terms of (a) actual car age (e.g., mileage) and (b) time with thecurrent owner?

Service_cost is composited of labor_cost and spare_cost. The car's mileage is computed through the initial mileage from the car record and the mileage when repairing the car. We choose 'DATE' data type for 'repair_at'(repair date) and 'purchase_at'(purchase date). Thus, we use julianday function to calculate the duration and divide 365.25 to convert unit from 'days' to 'year'. We use 'car_id' and 'po_id' join three entities together to select the information we needed.

```sql

SELECT cp.car_id,labor_cost + spare_cost AS service_cost, rr.mileage - c.car_mileage AS mileage, (julia
FROM repair_record AS rr
JOIN car_purchase_order AS cp
ON rr.po_id = cp.po_id
JOIN car AS c
ON cp.car_id = c.car_id;
```

Table 2: 0 records

car_id	service_cost	mileage	years

Please refer to Appendix Part A2

Part B

This is to certify that the work I am submitting is my own. All external references and sources are clearly acknowledged and identified within the contents. I am aware of the University of Warwick regulation concerning plagiarism and collusion.

No substantial part(s) of the work submitted here has also been submitted by me in other assessments for accredited courses of study, and I acknowledge that if this has been done an appropriate reduction in the mark I might otherwise have received will be made.

Part B

MSBA Group 28

17/11/2021

```
#install.packages("readxl")

#Checking all countries we have on the database
OCDE_COUNTRIES <-"partB_data_files/"
#Listing all files (each country)
folders <-list.files(OCDE_COUNTRIES)
OCDE_COUNTRIES
```

```
## [1] "partB_data_files/"
```

```
folders
```

```
##  [1] "Albania"            "Algeria"            "Armenia"
##  [4] "Austria"            "Belarus"            "Belgium"
##  [7] "Bulgaria"           "Cyprus"             "Egypt"
## [10] "France"             "Germany"            "Hondura"
## [13] "Hungary"            "Iran"               "Iraq"
## [16] "Israel"             "Italy"              "Jordan"
## [19] "Kosovo"             "Kuwait"             "Lebanon"
## [22] "Libya"              "Malta"              "Morocco"
## [25] "Oman"               "Poland"             "Portugal"
## [28] "Quatar"             "Romania"            "Russian Federetion"
## [31] "Saudi"              "Serbia"             "Slovak Republic"
## [34] "Slovania"           "Spain"              "Syria"
## [37] "Tunisia"            "Ukraine"
```

- list the files

```
#IMPORTANT: Set working directory to the partB_data_files folder in local computer

setwd("/Users/Tati/Desktop/MSC BUSINESS ANALYTICS/IB9HP0 Data Management/assigment/Group assignment/par

knitr::opts_chunk$set(echo = TRUE)
```

Step 1: Given the list of .xlsx. Creating a function to import automatically the required columns

```
  # Creating a function into a new object CLEAN TABLE,
  #this table will have the structure required to present
  #the final results For this part.(Country,Year,Flow
  #,Product(),Value)

clean_table <- function(df) {

  df %>%

     # Since we will be parsing range A6:BO50 of each excel files,
    #this will remove the first row and second column of
    #the second column of the parsed .xlsx file.
    slice(-1) %>%
    select(-2)%>%

    # Using pivot_longer function, we can transpose
    #the table (setting columns as rows)
    pivot_longer(2:66, names_to = "product", values_to = "value") %>%

    # From Seminar 6, Replacing NA'S with 0
    #Extra comment=From the original database all
    #cells which did not contain a value (NA'S) was 0. Taking
    #this information from the original excel tables.
    mutate(value = replace(value, value == "..", 0)) %>%

    # Adding country name and flow variables
    #to the table
    mutate(country = countryname,
           flow = flowname) %>%

     # Rename and rearranging the columns to desired output
    rename(year = Product) %>%
    select(country, year, flow, product, value)

}
```

Step 2: From every sub-folder, importing all .xlsx files into country tables.

```
setwd("/Users/Tati/Desktop/MSC BUSINESS ANALYTICS/IB9HP0 Data Management/assigment/Group assignment/par

#  Extract list of country file names in the directory
files <- list.files()

# Use nested for-loop function to screen through
#every folders and every .xlsx files in the directory
for (j in files) {

  xlsfilepath <- dir_ls(j, glob = "*.xlsx")
```

```
  # Create an empty list to contain all the
  #subtables in respective country's folder
  country_table <- list()

  # For every .xlsx files in the folder,
  for (i in seq_along(xlsfilepath)) {

    # extract flow name, country name,and table range
    flowname <- names(read_excel(path = xlsfilepath[i], range = "C3"))

    countryname <- names(read_excel(path = xlsfilepath[i], range = "C5"))

    assign(paste0(countryname,"_table_",i), read_excel(path = xlsfilepath[i],
                                                       range = "A6:BO50",
                                                       col_types = "text"))

    # Here , we have to use the function from step 1 to clean the current table
    country_table[[i]] <- clean_table(get(paste0(countryname,"_table_",i)))

  }


  # Join all the flow tables into a full country
  #table using the reduce() function
  assign(paste0(countryname,"_full_table"),
         country_table %>% reduce(full_join))

}
```

Step 3 :Joining all countries tables into a one single table so that we can manipulate all data.

```
# Extract all country tables from the global environment into a new list
new_list <- as.list(.GlobalEnv)

table_list <- new_list[grep("_full_table", names(new_list))]

# Join all country-specific table into a single table
single_table = table_list %>% reduce(full_join)


#Viewing dataset and treating anomalous data

# Setting the "value" column type into numeric instead of character

single_table$value <- as.numeric(single_table$value)
```

```
#saving the final result into a csv file

write.csv(single_table,"C:\\Users\\Tati\\Desktop\\
```

```
          MSC BUSINESS ANALYTICS
          \\IB9HP0 Data Management\\assigment\\Group
          assignment\\single_table.csv", row.names = FALSE)
```

```
#Checking the final result
single_table1 <- read.csv("single_table.csv")

head(single_table1)
```

```
##   country year    flow                         product value
## 1 Albania 1971 Imports Additives/blending components     0
## 2 Albania 1971 Imports                     Anthracite     0
## 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
```

```
#Checking the structure of our data variables,
#all variables are characters except for value column.

#The total of records are 531,050 observations and 5 variables
str(single_table1)

#Saving the single table as a dataframe into a new object called dataset

dataset <- data.frame(single_table1)
```

Total number of records for each product (Numerofproductrecords) across years and countries.

To approach this result, we can select the columns product, year, and country. Then, by using mutate function to create a new column "Numerofproductrecords" as well as counting the number of each product across year and countries.

The highest was in Elec/heat output from Elec/heat output from non-specified manufactured gases in 1971 in Albania-Imports flow which accounted for 54 records

```
productsrecords <- dataset %>% select(product, year,country, flow)
%>% mutate(Numerofproductrecords= str_count(product))
```

Data-Understanding

The database contains statistics on energy as well as the energy balances of OECD non-OECD countries such as Albania,Cyprus,Egypt,France,Germany,Qatar ,Spain, Ukraine among others (38 countries in total).

```
dataset %>% select(country) %>% count(country)
```

The years of data compilation were between 1960 and 2013

```
dataset %>% select(year) %>% count(year)
```

Flows categories involved exports,imports,losses,production and total primary energy supply. It alludes to what was the main source which the energy products were supplied from during each year per country.

```
dataset %>% select(flow) %>% count(flow)
```

On the other hand, energy products were additives/blending components,anthracite,aviation gasoline,biodiesels etc

```
dataset %>% select(product) %>% count(product)
```

Part C

This is to certify that the work I am submitting is my own. All external references and sources are clearly acknowledged and identified within the contents. I am aware of the University of Warwick regulation concerning plagiarism and collusion.

No substantial part(s) of the work submitted here has also been submitted by me in other assessments for accredited courses of study, and I acknowledge that if this has been done an appropriate reduction in the mark I might otherwise have received will be made.

Part c

MSBA Group 28

17/11/2021

Importing the XML's from the website https://data.food.gov.uk/catalog/datasets/38dd8d6a-5ab1-4f50-b753-ab33288e3200

```
# source page
MAIN_PAGE = "https://data.food.gov.uk/catalog/datasets/38dd8d6a-5ab1-4f50-b753-ab33288e3200"

MAIN_HTML = read_html(MAIN_PAGE) # read url to R
LINKS = MAIN_HTML %>% html_nodes("a") # get links
LINKS %>% html_attr("data-mime-type") -> xml_id # get xml links indicator
which(xml_id == "application/xml") -> xml_id # get xml id
LINKS %>% html_attr('href') -> links # get links
LINKS %>% html_attr('title') -> titles # get titles
links_xml = links[xml_id] # xml links
title_xml = titles[xml_id] # xml titles
```

```
Xml2Df = function(i){
  tmp_xml_url =  links_xml[i]
  tmp_city = title_xml[i]
  tmp_info = read_xml(tmp_xml_url)
  parents <-xml_find_all(tmp_info, ".//EstablishmentDetail")


  dfs<-lapply(parents, function(node){
    #Find all children
    nodes <- xml_children(node)

    #get node name and value
    nodenames<-  xml_name(nodes)
    values <- xml_text(nodes)

    #made data frame with results
    df<- as.data.frame(t(values), stringsAsFactors=FALSE)
    names(df)<-nodenames
    df$city = tmp_city
    df
  })
  tmp_res <-bind_rows(dfs)
  return(tmp_res)
```

```
}
```

```
# collect all xml
res = NULL # empty object

for(i in 1:length(links_xml)){ # loop each xml

  tmp_df = Xml2Df(i) # get table from ith xml
  res = bind_rows(res, tmp_df) # merge results

}
```

```
#Making sure the new values for RatingValue column are distinct

res %>% count(RatingValue)

res %>% count(LocalAuthorityName)

#removing rows which contain (Welsh language)
#since it's repeated information

res <- res[!grepl("Welsh",res$city),]

#removing duplicates

res1 <- unique(res)
```

```
# separate geocode column into latitude and longitude

res1 <- res1 %>%
        separate(.,Geocode,into = c("longitude","latitude"),-17)

# saving the dataframe created into the current
#directory as a csv file

write.csv(res,"C:\\Users\\Tati\\Desktop\\MSC BUSINESS ANALYTICS\\IB9HP0 Data Management\\assigment\\PAR

write.csv(res1,"C:\\Users\\Tati\\Desktop\\MSC BUSINESS ANALYTICS\\IB9HP0 Data Management\\assigment\\PA
```

596,773 observations x 24 variables

```
dataExcel <- read.csv("food2.csv")
```

```
#Checking every character we have on RatingValue column

dataExcel %>% count(RatingValue,SchemeType)
```

```
##             RatingValue SchemeType      n
## 1                     0       FHRS    735
## 2                     1       FHRS   6271
## 3                     2       FHRS   6748
```

```
## 4                     3       FHRS  29480
## 5                     4       FHRS  70784
## 6                     5       FHRS 340283
## 7   Awaiting Inspection       FHIS   8545
## 8    AwaitingInspection       FHRS  58167
## 9   AwaitingPublication       FHRS     23
## 10               Exempt       FHIS   1704
## 11               Exempt       FHRS  30702
## 12 Improvement Required       FHIS   3352
## 13                 Pass       FHIS  39314
## 14    Pass and Eat Safe       FHIS    665
```

```
#Identify the number of NA's for each variable.
#To return the names of all R data frame columns
#and a sum of NA values in them

cbind(
   lapply(
     lapply(dataExcel, is.na)
     , sum)
   )
```

```
##                             [,1]
## FHRSID                      0
## LocalAuthorityBusinessID    0
## BusinessName                0
## BusinessType                0
## BusinessTypeID              0
## AddressLine2                148241
## AddressLine3                212183
## AddressLine4                336193
## PostCode                    99407
## RatingValue                 0
## RatingKey                   0
## RatingDate                  0
## LocalAuthorityCode          0
## LocalAuthorityName          0
## LocalAuthorityWebSite       0
## LocalAuthorityEmailAddress 1556
## Scores                      145942
## SchemeType                  0
## NewRatingPending            0
## longitude                   100578
## latitude                    100578
## city                        0
## AddressLine1                205699
## RightToReply                596712
```

Data-Understanding

- Rating value: The measures (scores) vary depending on whether it's in Scotland or the rest of United of Kingdom.

Scotland (FHIS) Food Hygiene Information Scheme.

Rating Score	
Awaiting Inspection FHIS	(SPACE)
Exempt FHIS	
Improvement Required FHIS	
Pass FHIS	
Pass and Eat Safe FHIS	

The Food Hygiene Rating Scheme (FHRS) in the rest of the United Kingdom including Northern Ireland. (England, Wales and Northern Ireland)

Rating Score	Description
0	Urgent improvement necessary
1	Major improvement necessary
2	Improvement necessary
3	Generally satisfactory
4	Good
5	Very good
Exempt	FHRS
AwaitingPublication	FHRS (NO SPACE)
AwaitingInspection	FHRS (NO SPACE)

Part c

MSBA Group 28

17/11/2021

Normalization PART C

Defining a connection to RSQlite

```
my_connection <- RSQLite::dbConnect(RSQLite::SQLite(),"deletemater.db")
```

Reading the lastest database generated

```
Masterdatabase <- readr::read_csv("food2.csv")

str(Masterdatabase)
```

Creating a unique primary key for every business and overwritting the database

```
Masterdatabase$BusinessID <- random_id(596773)

Masterdatabase$BusinessID <- Masterdatabase$BusinessID
```

Saving the new database modified from the previous steps

```
write.csv(Masterdatabase,"C:\\Users\\Tati\\Desktop\\MSC BUSINESS ANALYTICS\\IB9HP0 Data Management\\ass
```

```
Masterdatabase1 <- readr::read_csv("foodUPDATED.csv")
```

```
## Rows: 596773 Columns: 25

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (18): LocalAuthorityBusinessID, BusinessName, BusinessType, AddressLine...
## dbl   (4): FHRSID, BusinessTypeID, longitude, latitude
## lgl   (2): NewRatingPending, RightToReply
## date  (1): RatingDate

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Writting the file.csv foodUPDATED.csv into mysqlite

```
RSQLite::dbWriteTable(my_connection,"Masterdatabase1",Masterdatabase1,overwrite=TRUE)
```

Creating ratings table with attributes (‘rating_id’,‘rating’, and region)

```
CREATE TABLE'ratings_table' (
  'rating_id' PRIMARY KEY,
  'rating' VARCHAR,
  'region'
);
```

Inserting data into rating _table from the Masterdatabase1

```
insert into ratings_table(rating_id,rating,region)
SELECT DISTINCT RatingKey,RatingValue,SchemeType
from Masterdatabase1
```

Verifying the content of the ratings_table

Region= FHIS for Scotland or FHRS for the rest of the UK

```
select *
from ratings_table
```

Table 1: Displaying records 1 - 10

rating_id	rating	region
fhis_pass_en-GB	Pass	FHIS
fhis_awaiting_inspection_en-GB	Awaiting Inspection	FHIS
fhis_improvement_required_en-GB	Improvement Required	FHIS
fhis_pass_and_eat_safe_en-GB	Pass and Eat Safe	FHIS
fhis_exempt_en-GB	Exempt	FHIS
fhrs_awaitinginspection_en-GB	AwaitingInspection	FHRS
fhrs_5_en-GB	5	FHRS
fhrs_4_en-GB	4	FHRS
fhrs_3_en-GB	3	FHRS
fhrs_exempt_en-GB	Exempt	FHRS

Creating business_details table with attributes (‘business_ID’,‘business_name’,‘city’,‘latitude’,‘longitud’ , ‘addressline2’,‘addressline3’, ‘POSTCODE’,‘Rating_date_taken’). Additionally, foreign keys such as’rating_id’,‘local_authority_ID, and ’business_type_ID’ as a result of the relationship M:1 between business and three entities which are local_authority,Ratings and business type entity.

```
CREATE TABLE'business_details' (
  'business_ID' varchar PRIMARY KEY ,
   'business_name'VARCHAR,
  'city' VARCHAR,
  'latitude' NUMERIC,
  'longitud' NUMERIC,
  'addressline2' VARCHAR,
```

```
  'addressline3' VARCHAR,
  'POSTCODE' VARCHAR,
  'Rating_date_taken' DATE,
  'rating_id' ,
  'local_authority_ID',
  'business_type_ID',
  FOREIGN KEY ('rating_id')
    REFERENCES ratings_table('rating_id'),
      FOREIGN KEY ('local_authority_ID')
    REFERENCES local_authorities('local_authority_ID'),
     FOREIGN KEY ('business_type_ID')
    REFERENCES business_types('business_type_ID')

);
```

Inserting data into business_details from the Masterdatabase1

```
insert into business_details(business_ID,business_name,
city,latitude,longitud,addressline2,addressline3,POSTCODE,
Rating_date_taken,rating_id,local_authority_ID,business_type_ID)

SELECT BusinessID,BusinessName,city,latitude,longitude,
AddressLine2,AddressLine3,PostCode,RatingDate,RatingKey,
LocalAuthorityCode,BusinessTypeID

from Masterdatabase1
```

Verifying the content of business_details table

```
select business_ID,business_name
from business_details
limit 5
```

Table 2: 5 records

business_ID	business_name
dab2aac2ef916cf6712fb73aea76dc11	1 & 30 DONALD DEWAR COURT
5a8dd23ba9276fb347fdafceae22b096	1906 RESTAURANT AT HMT
7fbbc19bcf1c65c195dd4bedb19a2cf6	1DS
318ce1750b299b469f7e14d382c84e35	2 BROTHERS PIZZA
0a4174778b9cb08ec2709a00da10b002	210 BISTRO

Creating local_authorities table which indicates 'local_authority_ID', 'local_authority_name' ,'local_authority_WEBSITE','local_authority_email'

```
CREATE TABLE'local_authorities' (
  'local_authority_ID' PRIMARY KEY ,
  'local_authority_name' VARCHAR,
  'local_authority_WEBSITE' VARCHAR,
  'local_authority_email' VARCHAR
);
```

Inserting data into local_authorities from the Masterdatabase1

```
insert into local_authorities(local_authority_ID,
local_authority_name,local_authority_WEBSITE,local_authority_email)

SELECT distinct  LocalAuthorityCode,
LocalAuthorityName, LocalAuthorityWebSite,
LocalAuthorityEmailAddress

from Masterdatabase1
```

Verifying the content of local_authorities table

```
select *
from local_authorities
limit 5
```

Table 3: 5 records

local_authority_ID	local_authority_name	local_authority_WEBSITE	local_authority_email
760	Aberdeen City	http://www.aberdeencity.gov.uk	commercial@aberdeencity.gov.uk
761	Aberdeenshire	http://www.aberdeenshire.gov.uk/	environmental@aberdeenshire.gov.uk
323	Adur	http://www.adur-worthing.gov.uk	publichealth.regulation@adur-worthing.gov.uk
055	Allerdale	http://www.allerdale.gov.uk	environmental.health@allerdale.gov.uk
062	Amber Valley	http://www.ambervalley.gov.uk	envhealth@ambervalley.gov.uk

Creating business_types table which indicates 'business_type_ID', 'business_type_name'

```
CREATE TABLE'business_types' (
  'business_type_ID' PRIMARY KEY ,
  'business_type_name' VARCHAR
);
```

Inserting data into business_types from the Masterdatabase1

```
insert into business_types(business_type_ID,business_type_name)

SELECT distinct  BusinessTypeID,BusinessType
from Masterdatabase1
```

Verifying the content of business_types table

```
select *
from business_types
limit 5
```

Table 4: 5 records

business_type_ID	business_type_name
5	Hospitals/Childcare/Caring Premises
1	Restaurant/Cafe/Canteen
7843	Pub/bar/nightclub
7844	Takeaway/sandwich shop
7842	Hotel/bed & breakfast/guest house

Lets drop the table Masterdatabase1 from the lsit of tables to save storage.

```
drop table Masterdatabase1
```

Listing all the tables and database from the sqlite environment

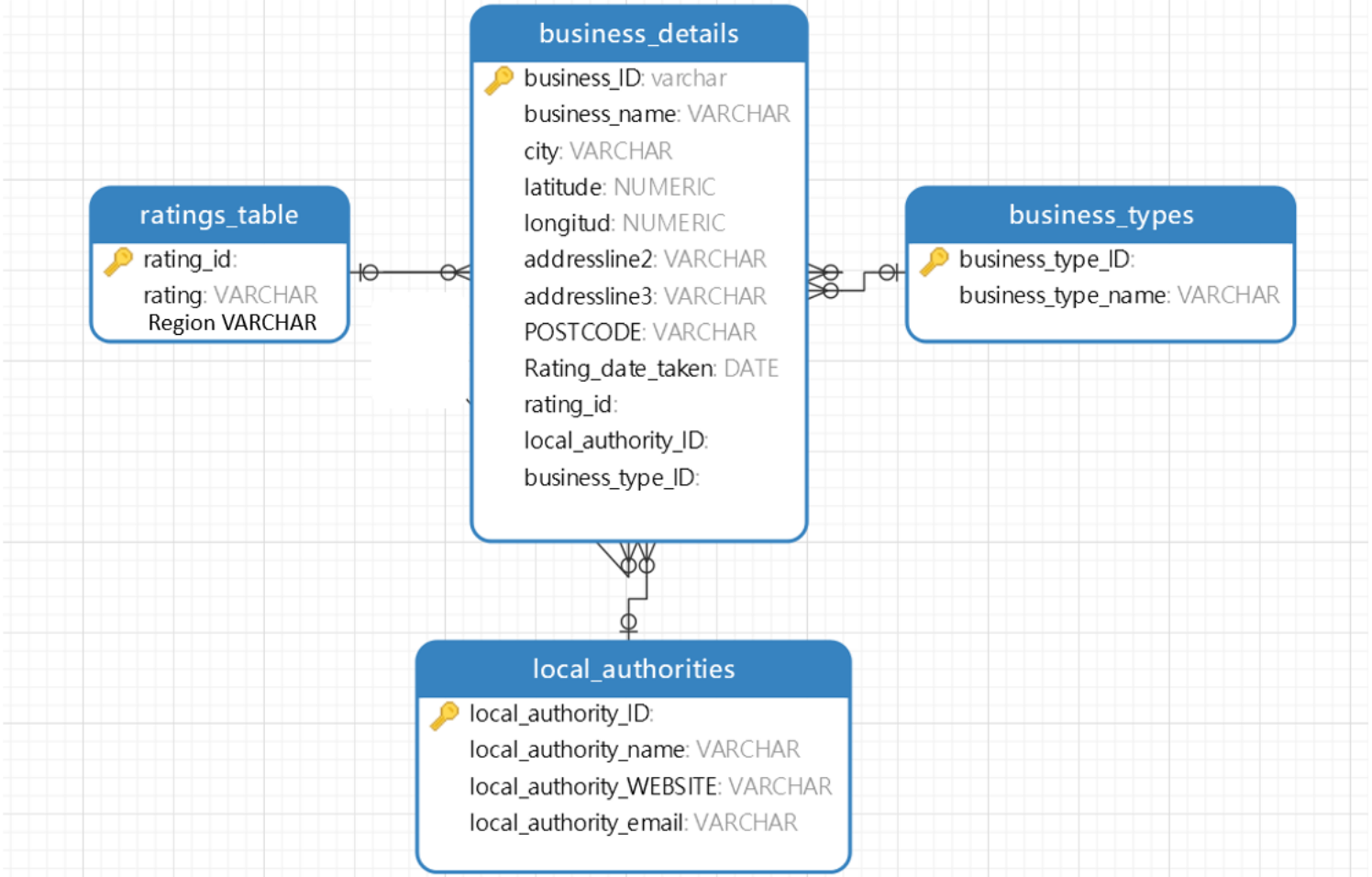
```
# Get a list of tables from the database that we already
# created
RSQLite::dbListTables(my_connection)
```

```
## [1] "business_details"  "business_types"    "local_authorities"
## [4] "ratings_table"
```

Disconnect from sqlite

```
# Disconnect from the database using the connection variable that we setup
# before
RSQLite::dbDisconnect(my_connection)
```

FOOD_HYGIENE_RATING PHYSICAL SCHEMA



Part D

This is to certify that the work I am submitting is my own. All external references and sources are clearly acknowledged and identified within the contents. I am aware of the University of Warwick regulation concerning plagiarism and collusion.

No substantial part(s) of the work submitted here has also been submitted by me in other assessments for accredited courses of study, and I acknowledge that if this has been done an appropriate reduction in the mark I might otherwise have received will be made.

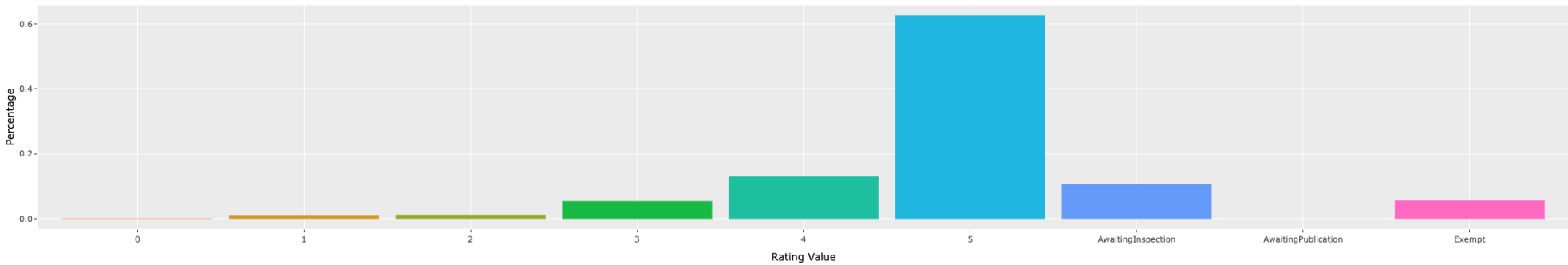
Food Hygiene Rating

Overview of ratings

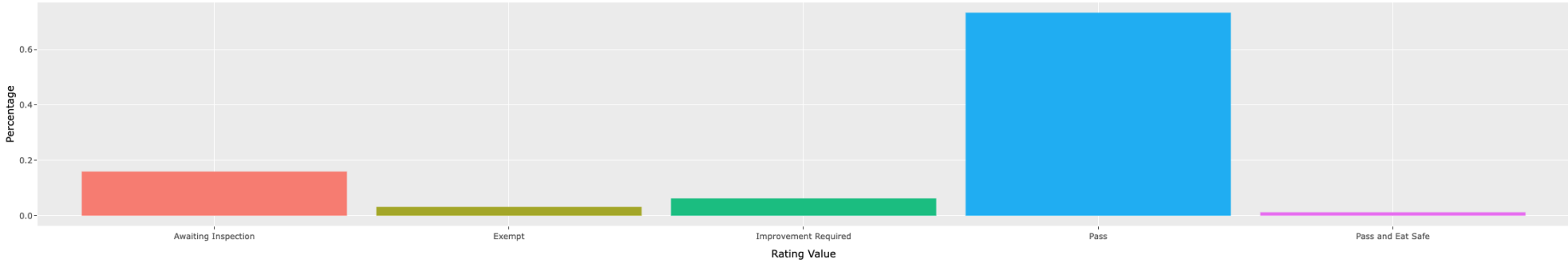
Top & Bottom cities ranking

Overview of ratings

Proportion of Rating Value (Rest of UK)



Proportion of Rating Value (Scotland)



Search information of restaurants

Show 10 entries

Search:

	business_name	business_type_name	POSTCODE	rating	region	Rating_date_taken	local_authority_ID	local_authority_name	local_authority_WEBSITE	local_authority_email	city
1	A breath of fresh air	Mobile caterer		AwaitingInspection	FHRS		323	Adur	http://www.adur-worthing.gov.uk	publichealth.regulation@adur-worthing.gov.uk	Adur
2	A Knead for More	Other catering premises		5	FHRS	18829	323	Adur	http://www.adur-worthing.gov.uk	publichealth.regulation@adur-worthing.gov.uk	Adur
3	ADSushi	Other catering premises		5	FHRS	18792	323	Adur	http://www.adur-worthing.gov.uk	publichealth.regulation@adur-worthing.gov.uk	Adur
4	Adur Express Convenience Store	Retailers - other	BN42 4AR	5	FHRS	17960	323	Adur	http://www.adur-worthing.gov.uk	publichealth.regulation@adur-worthing.gov.uk	Adur
5	Adur Outdoor Activities Centre	Hospitals/Childcare/Caring Premises	BN43 5LT	5	FHRS	18304	323	Adur	http://www.adur-worthing.gov.uk	publichealth.regulation@adur-worthing.gov.uk	Adur
6	Adur Special Needs Project	Hospitals/Childcare/Caring Premises	BN43 6TN	5	FHRS	17756	323	Adur	http://www.adur-worthing.gov.uk	publichealth.regulation@adur-worthing.gov.uk	Adur
7	Age Concern Lunch Club	Other catering premises	BN43 5WB	5	FHRS	17921	323	Adur	http://www.adur-worthing.gov.uk	publichealth.regulation@adur-worthing.gov.uk	Adur
8	Age UK West Sussex	Other catering premises	BN43 6BU	5	FHRS	18156	323	Adur	http://www.adur-worthing.gov.uk	publichealth.regulation@adur-worthing.gov.uk	Adur
9	AGE UK West Sussex and Brighton	Restaurant/Cafe/Canteen	BN41 1QH	5	FHRS	18850	323	Adur	http://www.adur-worthing.gov.uk	publichealth.regulation@adur-worthing.gov.uk	Adur
10	Albion Inn	Pub/bar/nightclub	BN41 1PH	5	FHRS	18061	323	Adur	http://www.adur-worthing.gov.uk	publichealth.regulation@adur-worthing.gov.uk	Adur

Showing 1 to 10 of 543,193 entries

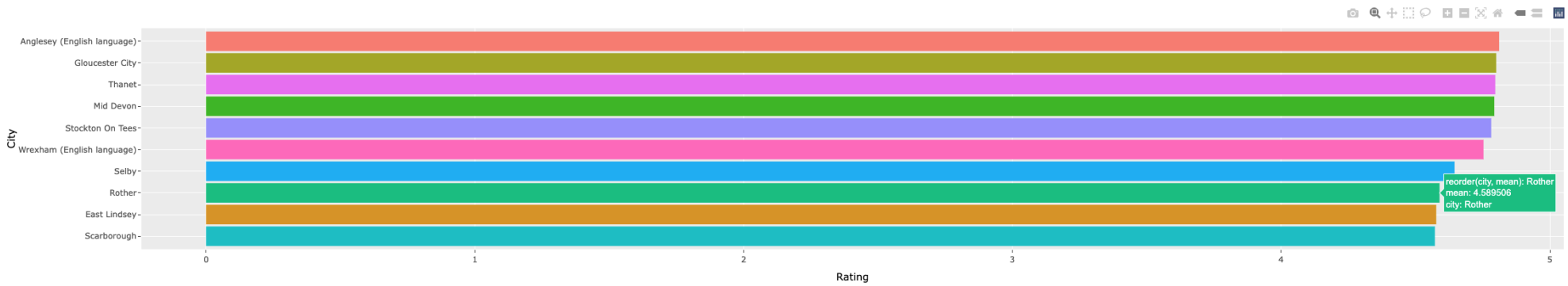
Food Hygiene Rating

Overview of ratings

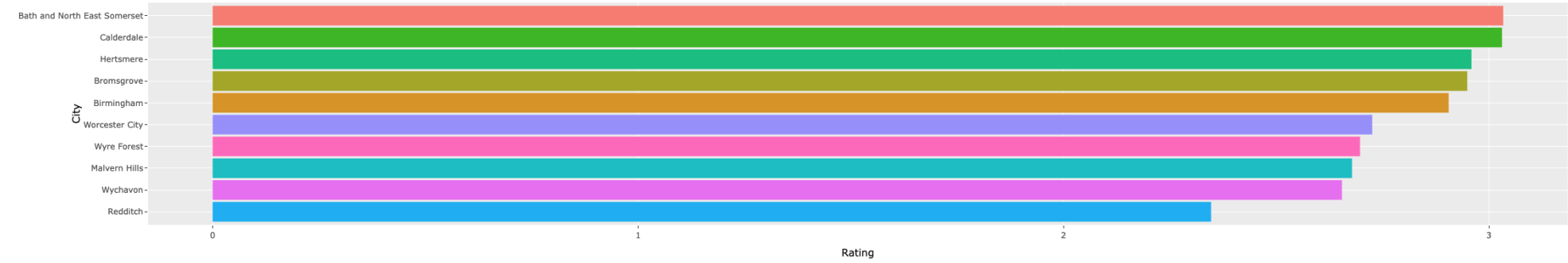
Top & Bottom cities ranking

Rankings

The top ten cities



The bottom ten cities



server.R

2021-12-07

```
library(shiny)
```

```
## Warning: package 'shiny' was built under R version 4.1.2
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.2
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(RSQLite)
library(plotly)
```

```
## Warning: package 'plotly' was built under R version 4.1.2
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
# Establishing connection via SQLite to the database file
conn <- dbConnect(RSQLite::SQLite(), "database.db")

# Creating object "query1" using SQL for a dashboard output
# To extract the ratings from region "FHRS", rest of the UK
query1 = dbGetQuery(conn, "SELECT rating FROM ratings_table AS rt
                    JOIN business_details AS bd on rt.rating_id = bd.rating_id
                    WHERE region = 'FHRS'")

# Creating object "query2" using SQL for a dashboard output
# To extract the ratings from region "FHIS", Scotland
query2 = dbGetQuery(conn, "SELECT rating FROM ratings_table AS rt
                    JOIN business_details AS bd on rt.rating_id = bd.rating_id
                    WHERE region = 'FHIS'")

# Creating object "query3" using SQL for a dashboard output
# Extracting a table of important details for all the food business types
query3 = dbGetQuery(conn, "SELECT business_name,business_type_name,
                    POSTCODE,rating,region,Rating_date_taken,
                    bd.local_authority_ID,
                    local_authority_name,local_authority_WEBSITE,
                    local_authority_email,city
                    FROM business_details AS bd
                    JOIN ratings_table AS rt ON rt.rating_id = bd.rating_id
                    JOIN business_types AS bt ON bt.business_type_ID=bd.business_type_ID
                    JOIN local_authorities AS la ON la.local_authority_ID=bd.local_authority_ID
                    WHERE region = 'FHRS'")

# Creating object "query4" using SQL for a dashboard output
# Extracting top 10 and bottom 10 food business types with rankings of 0 to 5 by cities
query4 = dbGetQuery(conn, "SELECT AVG(rating) AS mean,city
                    FROM business_details AS bd
                    JOIN ratings_table AS rt ON rt.rating_id = bd.rating_id
                    WHERE region = 'FHRS' GROUP BY city")

# Creating visualisations for the dashboard objects
server <- function(input,output){
  output$fooddataproportion_1 <- renderPlotly({
    output <- query1 %>% ggplot(aes(x = factor(rating), fill = rating)) +
      geom_bar(aes(y = (..count..)/sum(..count..))) +
      labs( x = "Rating Value", y="Percentage")+ theme(legend.position = "none")
    ggplotly(output)
  }
  )

  output$fooddataproportion_2 <- renderPlotly({
    output <- query2 %>% ggplot(aes(x = factor(rating), fill = rating)) +
      geom_bar(aes(y = (..count..)/sum(..count..))) +
      labs( x = "Rating Value", y="Percentage")+ theme(legend.position = "none")
    ggplotly(output)
  }
  )
```

```
  output$fooddatatable <- DT::renderDataTable(
    query3, options = list(scrollX = TRUE)

  )
  output$rate_1 <- renderPlotly({
    output1 <- query4 %>%
      top_n(mean, n = 10) %>%
      ggplot(aes(reorder(city,mean),mean,fill=city))+
      geom_bar(stat = "identity") +
      coord_flip()+
      labs( x = "City", y = "Rating")+ theme(legend.position = "none")
    ggplotly(output1)
  }

  )
  output$rate_2 <- renderPlotly({
    output1 <- query4 %>%
      top_n(mean, n = -10) %>%
      ggplot(aes(reorder(city,mean),mean,fill=city))+
      geom_bar(stat = "identity") +
      coord_flip()+
      labs( x = "City", y = "Rating")+ theme(legend.position = "none")
    ggplotly(output1)
  }
  )



}
```

ui.R

2021-12-07

```
# Loading all the necessary packages for creating a dashboard
library(shiny)
```

```
## Warning: package 'shiny' was built under R version 4.1.2
```

```
library(shinydashboard)
```

```
## Warning: package 'shinydashboard' was built under R version 4.1.2
```

```
##
## Attaching package: 'shinydashboard'
```

```
## The following object is masked from 'package:graphics':
##
##     box
```

```
library(DT)
```

```
## Warning: package 'DT' was built under R version 4.1.2
```

```
##
## Attaching package: 'DT'
```

```
## The following objects are masked from 'package:shiny':
##
##     dataTableOutput, renderDataTable
```

```
library(ggplot2)
library(plotly)
```

```
## Warning: package 'plotly' was built under R version 4.1.2
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

```
library(RSQLite)

# Establishing connection via SQLite to the database file
conn <- dbConnect(RSQLite::SQLite(), "last database normalized.db") # create db

# Creating a dashboard with two tabs on the sidebar
# Compiling objects created in the server to specific locations in the dashboard
ui <- dashboardPage(
  dashboardHeader(title="Food Hygiene Rating"),
  dashboardSidebar(

    sidebarMenu(
      menuItem("Overview of ratings", tabName = "fooddata", icon = icon("fas fa-utensils")),
      menuItem("Top & Bottom cities ranking", tabName = "foodplots", icon = icon("fas fa-arrows-alt-v"))


    )),
  dashboardBody(
    tabItems(
      # First tab content
      tabItem(tabName = "fooddata",
              fluidRow( h1("Overview of ratings") ),
              fluidRow(
                column(width = 12,
                       box(title="Proportion of Rating Value (Rest of UK)",
                           width=12,
                           plotlyOutput("fooddataproportion_1"))),
                column(width = 12,
                       box(title="Proportion of Rating Value (Scotland)",
                           width=12,
                           plotlyOutput("fooddataproportion_2"))),
                column(width = 12,
                       box(title="Search information of restaurants",
                           width=12,
                       DT::dataTableOutput("fooddatatable"))
                )
              )

      ),




      # Second tab content
      tabItem(tabName = "foodplots",
```

```
            fluidRow(column(width=12, h1("Rankings"))),
            fluidRow(
              column(width = 12,
                     box(title="The top ten cities",
                         width=12,
                         plotlyOutput("rate_1"))),
              column(width = 12,
                     box(title="The bottom ten cities",
                         width=12,
                         plotlyOutput("rate_2")))
            )
    )
    )
  )
)
```

Appendix

This is to certify that the work I am submitting is my own. All external references and sources are clearly acknowledged and identified within the contents. I am aware of the University of Warwick regulation concerning plagiarism and collusion.

No substantial part(s) of the work submitted here has also been submitted by me in other assessments for accredited courses of study, and I acknowledge that if this has been done an appropriate reduction in the mark I might otherwise have received will be made.

Appendix Part A1

07/12/2021

```
WNB_HOTEL_GROUP <- dbConnect(RSQLite::SQLite(), "WNB_HOTEL_GROUP.sqlite")
```

SQL DDL PHYSICAL DESIGN

- Creating a new database and setting up the tables and specifying the tables (With Foreign Key Constraints) WHEN needed

Hotels

```
CREATE TABLE'hotels' (
  'hotel_id' INTEGER PRIMARY KEY NOT NULL,
  'hotel_name' VARCHAR NOT NULL,
  'hotel_address' VARCHAR NOT NULL,
  'hotel_street_num' INTEGER NOT NULL,
  'hotel_street_name' VARCHAR NOT NULL,
  'hotel_post_code' VARCHAR NOT NULL,
  'hotel_city' VARCHAR NOT NULL,
  'hotel_state' VARCHAR NOT NULL,
  'hotel_country' VARCHAR NOT NULL,
  'hotel_phone_num' INTEGER ,
  'hotel_secondary_num' INTEGER,
  'hotel_web_page' VARCHAR
);
```

Services

```
-- Services
CREATE TABLE 'services' (
  'service_id' INTEGER PRIMARY KEY NOT NULL,
  'service_description' VARCHAR,
  'service_type'VARCHAR
);
```

Services by hotels relationship

```
-- Services_by_hotels

CREATE TABLE 'hotel_services'(
'hotel_id' INTEGER NOT NULL,
'service_id' INTEGER NOT NULL,
'hotel_service_id' INTEGER NOT NULL,
  FOREIGN KEY ('hotel_id')
    REFERENCES  hotels ('hotel_id'),
  FOREIGN KEY ('service_id')
    REFERENCES services ('service_id'),
  PRIMARY KEY ('hotel_id','service_id')
);
```

Rooms

Comments:

*'room_occupied' 1=yes 0=no

```
-- Rooms
CREATE TABLE 'rooms' (
  'room_id' INTEGER PRIMARY KEY NOT NULL,
  'room_num ' VARCHAR,
  'room_floor'INT ,
  'room_occupied' BOOLEAN ,
  'room_class_id' VARCHAR NOT NULL,
  'hotel_id' INTEGER NOT NULL,
  FOREIGN KEY('hotel_id')
    REFERENCES hotels('hotel_id'),
     FOREIGN KEY('room_class_id')
    REFERENCES room_class('room_class_id')
);
```

Room_class

Comments:

'room_class'= Double /single 'room_smoke'= 1= smoking 0= non-smoking

```
-- Room_class
CREATE TABLE 'room_class' (
  'room_class_id ' INTEGER PRIMARY KEY NOT NULL,
  'room_class ' VARCHAR ,
  'room_smoking' BOOLEAN
);
```

Reservations

Comments: When making a reservation, the arrival date,departure date, all credit card details can not be empty (NOT NULL) whereas preferences (smoking, room class,floor) are optional (NULL) as mentioned on the business scenario.

Preferences: *'room_class'* = *single/double* 'room_high_low_floor' = high/ low *'reservation_status' 1=pending , 0=cancelled

```
-- Reservations
CREATE TABLE 'reservations' (
  'booking_id' INTEGER PRIMARY KEY NOT NULL,
  'arrival_date ' DATETIME NOT NULL,
  'departure_date'DATETIME NOT NULL,
  'reservation_status' BOOLEAN,
  'room_smoke' BOOLEAN NULL,
  'room_high_low_floor' VARCHAR NULL,
  'credit_card_num' INT NOT NULL,
  'exp_date' VARCHAR NOT NULL,
  'exp_month' INT NOT NULL,
  'exp_year' INT NOT NULL ,
  'credit_card_brand' VARCHAR NOT NULL,
  'guest_id' INTEGER NOT NULL,
  'booking_channel_id' INTEGER NOT NULL,
  'hotel_id' INTEGER NOT NULL,
  FOREIGN KEY('guest_id')
    REFERENCES guests('guest_id'),
   FOREIGN KEY('booking_channel_id')
    REFERENCES booking_channels('booking_channel_id'),
    FOREIGN KEY('hotel_id')
    REFERENCES hotels('hotel_id'),
);
```

Bookings relationship

Comments:This table reflects the central/main interaction between all entities. Also, it is worth to highlight that all details stored on it are reservations confirmed.

```
-- bookings
CREATE TABLE 'bookings' (
  'booking_id' INTEGER PRIMARY KEY NOT NULL,
  'arrival_date' DATETIME NOT NULL,
  'departure_date'DATETIME NOT NULL,
  'guest_id' INTEGER NOT NULL,
  'room_id' VARCHAR NOT NULL,
  FOREIGN KEY('guest_id')
    REFERENCES guests('guest_id'),
  FOREIGN KEY ('room_id')
    REFERENCES rooms('room_id'),
   FOREIGN KEY('booking_channel_id')
    REFERENCES booking_channels('booking_channel_id')
);
```

Booking_channels

Comments:

'booking_channel_commission'= commission in % per marketplace (channel) 'booking_channel_fee' =invoice item amount * commission %

```
-- booking_channels
CREATE TABLE 'booking_channels' (
  'booking_channel_id' INTEGER PRIMARY KEY NOT NULL,
  'booking_channel_name' VARCHAR,
  'booking_channel_commission' NUMERIC,
  'booking_channel_fee' NUMERIC,
  'payment_due_date' DATETIME
);
```

Guests

Comments: As described on the scenario, all guest's personal details are restricted to not be empty (NOT NULL), except for email addresses and phone numbers (NULL)

```
-- guests
CREATE TABLE 'guests' (
  'guest_id' INTEGER PRIMARY KEY NOT NULL,
  'guest_name' VARCHAR NOT NULL,
  'guest_first_name' VARCHAR NOT NULL,
  'guest_middle_name' VARCHAR NOT NULL,
  'guest_last_name' VARCHAR NOT NULL,
  'guest_email' VARCHAR NULL,
  'guest_address' VARCHAR NOT NULL,
  'guest_street_num' VARCHAR NOT NULL,
  'guest_street_name' VARCHAR NOT NULL,
  'guest_post_code' VARCHAR NOT NULL,
  'guest_city' CHAR NOT NULL,
  'guest_state' CHAR NOT NULL,
  'guest_country' CHAR NOT NULL,
  'guest_cell_phone' NUMERIC NULL,
  'guest_home_phone' NUMERIC NULL,
  'guest_office_phone' NUMERIC NULL
);
```

Invoices

Comments:

'description_items' it concatenates all the items acquired by the guest (room cost, extra services, taxes,and amount paid)

'invoice_paid' 1=payed 0=not paid

```
-- invoices
CREATE TABLE 'invoices' (
  'invoice_id' PRIMARY KEY NOT NULL,
  'description_items' TEXT NOT NULL,
  'invoice_paid' BOOLEAN NOT NULL,
  'booking_id' INTEGER NOT NULL,
    FOREIGN KEY ('booking_id')
    REFERENCES bookings('booking_id')
);
```

Invoice description relationship :

Comments: The 'item_amount' column, which is a balance account, contains the sum of tax+charges-payments. If it is zero (0) means the invoice is totally paid.In this case, we are assuming that the whole amount is paid when checking out, then we will have a negative value as output.

'description_items'= services that will be printed on the invoice

'is_tax' 1=yes 0=no. As a business understanding,we have to consider that we pay commission to each booking channels without taxes

```
--payment_details
CREATE TABLE 'payment_details' (
  'payment_details_id' INTEGER PRIMARY KEY NOT NULL,
  'credit_card_num' INTEGER,
  'credit_card_brand' VARCHAR,
  'exp_date' VARCHAR (5),
  'exp_month' INTEGER (2),
  'exp_year' INTEGER (2) ,
  'hotel_service_id' INTEGER NOT NULL,
  'room_id' VARCHAR NOT NULL,
  'invoice_id' INTEGER NOT NULL,
  'description_items' TEXT,
  'item_amount' NUMERIC,
  'is_tax' BOOLEAN,
  FOREIGN KEY('hotel_service_id')
    REFERENCES services('hotel_service_id'),
    FOREIGN KEY('room_id')
    REFERENCES rooms('room_id'),
    FOREIGN KEY('invoice_id')
    REFERENCES invoices('invoice_id')

);
```

Appendix Part A2

Tatiana Hernandez

07/12/2021

```r
A2 <- DBI::dbConnect(RSQLite::SQLite(),"A2.sqlite")
```

```
-- customer
CREATE TABLE 'customer' (
 'customer_id' NUMERIC PRIMARY KEY,
 'customer_name' VARCHAR,
 'customer_first_name' VARCHAR,
 'customer_middle_name' VARCHAR,
 'customer_last_name' VARCHAR,
 'customer_cell_phone' NUMERIC NULL
);
```

```
-- car
CREATE TABLE 'car' (
 'car_id' NUMERIC PRIMARY KEY,
 'car_brand' VARCHAR,
 'car_model' VARCHAR,
 'car_is_new' INTEGER,
 'car_color' VARCHAR,
 'car_mileage' REAL,
 'car_age' REAL
);
```

```
-- car
CREATE TABLE 'car' (
 'car_id' NUMERIC PRIMARY KEY,
 'car_brand' VARCHAR,
 'car_model' VARCHAR,
 'car_is_new' INTEGER,
 'car_color' VARCHAR,
 'car_mileage' REAL,
 'car_age' REAL
);
```

```
-- dealer
CREATE TABLE 'dealer' (
 'dealer_id' NUMERIC PRIMARY KEY,
 'dealer_name' VARCHAR,
 'dealer_address' VARCHAR
);
```

```sql
-- car purchase order
CREATE TABLE 'car_purchase_order' (
 'po_id' NUMERIC PRIMARY KEY,
 'purchase_at' DATE,
 'payment' REAL,
 'customer_id' NUMERIC,
 'car_id' NUMERIC,
 'dealer_id' NUMERIC,
 FOREIGN KEY('customer_id')
   REFERENCES customer ('customer_id'),
 FOREIGN KEY('car_id')
   REFERENCES car ('car_id'),
 FOREIGN KEY('dealer_id')
   REFERENCES dealer ('dealer_id')
);
```

```sql
-- package purchase
CREATE TABLE 'package_purchase' (
 'package_id' NUMERIC PRIMARY KEY,
 'package_fee' REAL,
 'start_at' DATE,
 'end_at' DATE,
 'po_id' NUMERIC,
 FOREIGN KEY('po_id')
   REFERENCES car_purchase_order ('po_id')
);
```

```sql
-- repair record
CREATE TABLE 'repair_record' (
 'repair_id' NUMERIC PRIMARY KEY,
 'mileage' REAL,
 'repair_at' DATE,
 'labor_cost' REAL,
 'spare_cost' REAL,
 'dealer_id' NUMERIC,
 'po_id' NUMERIC,
 FOREIGN KEY('po_id')
   REFERENCES car_purchase_order ('po_id'),
 FOREIGN KEY('dealer_id')
   REFERENCES dealer ('dealer_id')
);
```