COMP1204: Database Theory and Practice Coursework

Name: Vedant Chokshi ID: 27748456 Email: vc4g15@ecs.soton.ac.uk

1 THE RELATIONAL MODEL AND ERD

1.1 EX1 Relations

R1(**INT** ReviewID, **INT** HotelID, **TEXT** URL, **INT** Overall, **INT** AvgPrice, **TEXT** Author, **TEXT**
↪ Content, **TEXT** DateCreated, **INT** NumOfReaders, **INT** NumFoundHelpful, **INT**
↪ OverallReview, **INT** MoneyValue, **INT** Rooms, **INT** Location, **INT** Cleanliness, **INT**
↪ CheckIn, **INT** Service, **INT** BusinessService)

1.2 EX2 Functional Dependencies

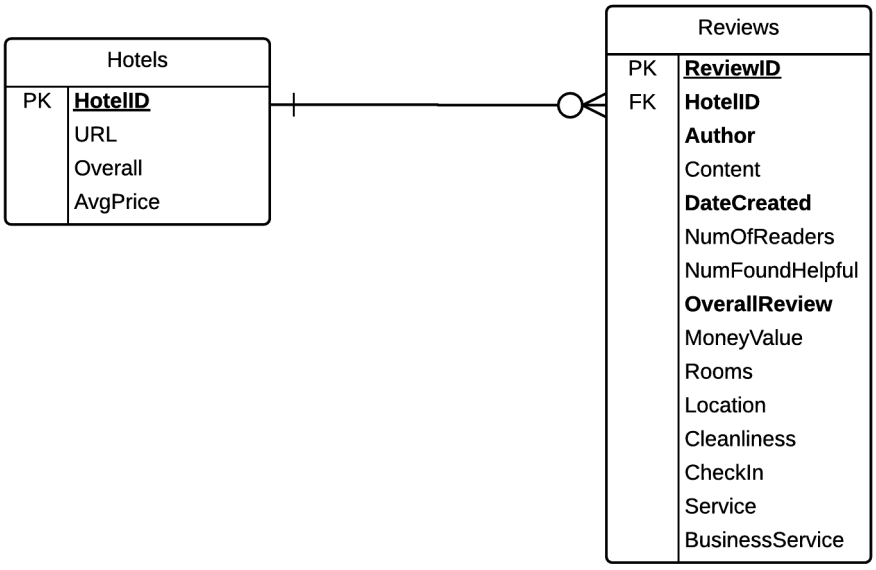
HotelID → *URL, Overall, AvgPrice*
ReviewID → *HotelID, Author, Content, DateCreated, NumOfReaders, NumFoundHelp, OverallReview, MoneyValue, Rooms, Location, Cleanliness, CheckIn, Service, BusinessService*

1.3 EX3 Normalisation

Hotels(HotelID, URL, Overall, AvgPrice)

Reviews(ReviewID, HotelID, Author, Content, DateCreated, NumOfReaders, NumFoundHelpful,
↪ OverallReview, MoneyValue, Rooms, Location, Cleanliness, CheckIn, Service,
↪ BusinessService)

1.4 EX4 ERD Model



2 RELATIONAL ALGEBRA

2.1 EX5

$$\sigma_{(Author="Tom")} Reviews$$

2.2 EX6

$$\pi_{Author,NumOfReviews} \sigma_{NumOfReviews>2} \gamma_{Author,Count(Author)\rightarrow NumOFReviews}(Reviews)$$

2.3 EX7

$$\pi_{HotelID,NumOfReviews} \sigma_{NumOfReviews>10} \gamma_{HotelID,Count(ReviewID)\rightarrow NumOFReviews}(Reviews)$$

2.4 EX8

$$R_1 = (\gamma_{(HotelID,AVG(Cleanliness)\rightarrow AvgCleanliness)}(Reviews))$$
$$R_2 = (\pi_{(HotelID,OverallReview)}(Hotels)$$
$$R = R_1 \bowtie_{R2.OverallReview>3 \ AND \ R1.AvgCleanliness\geq 5} R_2$$

3 SQL QUERIES

3.1 EX9 Creating the table

```
CREATE TABLE HotelReviews(ReviewID INTEGER PRIMARY KEY, HotelID INT, URL TEXT,
    ↪ Overall INT, AvgPrice INT, Author TEXT, Content TEXT, DateCreated TEXT,
    ↪ NumOfReaders INT, NumFoundHelpful INT, OverallReview INT, MoneyValue INT,
    ↪ Rooms INT, Location INT, Cleanliness INT, CheckIn INT, Service INT,
    ↪ BusinessService INT);
```

3.2 EX10 generatesql.sh

Script is written in the appendix. This script is written for UNIX-formatted data, run `dos2unix [reviews folder]/*` to convert data from DOS text formatting.

3.3 EX11 Creating normalised tables

```
CREATE TABLE Hotels(HotelID INTEGER PRIMARY KEY, URL TEXT, Overall INT, AvgPrice INT);

CREATE TABLE Reviews(ReviewID INTEGER PRIMARY KEY, HotelID INT, Author TEXT, Content
    ↪ TEXT, DateCreated TEXT, NumOfReaders INT, NumFoundHelpful INT,
    ↪ OverallReview INT, MoneyValue INT, Rooms INT, Location INT, Cleanliness INT,
    ↪ CheckIn INT, Service INT, BusinessService INT);
```

3.4 EX12 Populating the normalised tables

```
INSERT or REPLACE INTO Hotels (HotelID, URL, Overall, AvgPrice)
SELECT HotelID, URL, Overall, AvgPrice FROM HotelReviews;

INSERT or REPLACE INTO Reviews (ReviewID, HotelID, Author, DateCreated, Content,
    ↪ NumOfReaders, NumFoundHelpful, OverallReview, MoneyValue, Rooms, Location,
    ↪ Cleanliness, CheckIn, Service, BusinessService)
SELECT ReviewID, HotelID, Author, DateCreated, Content, NumOfReaders, NumFoundHelpful,
    ↪  OverallReview, MoneyValue, Rooms, Location, Cleanliness, CheckIn, Service,
    ↪ BusinessService FROM HotelReviews;
```

3.5 EX13 Indexes

For the table 'Hotels', I would index to be HotelID and AvgPrice. I have chosen HotelID as this value is unique and is a functional dependancy for all the other attributes in the table. I have chosen AvgPrice as I assume that the AvgPrice will be queried frequently.

For the table 'Reviews' I would index to be ReviewID as this is a unique value and is a functional dependancy for all the other attributes in the table

3.6 EX14 SQL Queries

```
With reference to EX5:
    SELECT * FROM Reviews WHERE Author = "A␣TripAdvisor␣Member";
With reference to EX6:
    SELECT Author, Count(Author) as NumOfReviews FROM Reviews GROUP BY Author HAVING
        ↪ Count(Author)>2;
With reference to EX7:
    SELECT HotelID, Count(ReviewID) as NumOfReviews FROM Reviews GROUP BY HotelID
        ↪ HAVING Count(ReviewID)>10;
With reference to EX8:
    SELECT Table1.HotelID, Overall, AvgCleanliness FROM (SELECT HotelID, AVG(
        ↪ Cleanliness) AS AvgCleanliness FROM Reviews GROUP BY HotelID) AS Table1 JOIN
        ↪  (SELECT HotelID, Overall FROM Hotels) AS Table2 ON Table1.HotelID=Table2.
        ↪ HotelID WHERE Overall>3 AND AvgCleanliness>=5;
```

4 CONCLUSION

Firstly, I started the coursework by going through the dataset to identify its structure and attributes. I realised that the order of the attributed was consistent and hence I used `getline`. This meant that when my if statements returned true, `awk` moves on to the next line. The data was retrieved by splitting each attribute using the ">" symbol.
When running and testing my script, I found that there were ratings with the value of `'-1'` and that one hotel had it's `"Avg.Price"` as `"Unkonwn"`. I replaced those ratings and hotel with `"NULL"` so they do not have an effect on the averages and sum. I also found that some authors were using quotation marks in their reviews and so I replaced quotation marks with \" so SQL does not get confused.

5 APPENDIX

5.1 generatesql.sh

```
#!/bin/bash

#Create empty file
> hotelreviews.sql

#Write 'create table' command into file
echo "CREATE TABLE HotelReviews(ReviewID INTEGER PRIMARY KEY, HotelID INT, URL TEXT, Overall INT,
    ↪ AvgPrice INT, Author TEXT,
Content TEXT, DateCreated TEXT, NumOfReaders INT, NumFoundHelpful INT, OverallReview INT, MoneyValue INT,
    ↪  Rooms INT, Location INT, Cleanliness INT,
CheckIn INT, Service INT, BusinessService INT);" >> hotelreviews.sql

for hotel in $1/*;
do
   # get hotel name
   hotelName=$(basename $hotel .dat)
   # pass bash variables into awk command
   awk -v hotelName=$hotelName 'BEGIN {
      #Split line
      FS=">";
      #Intialise variables
      Overall=NULL;
      AvgPrice=NULL;
      URL=NULL;
      Author=NULL;
      Content=NULL;
      DateCreated=NULL;
      NumOfReaders=NULL;
      NumFoundHelpful=NULL;
      OverallReview=NULL;
      MoneyValue=NULL;
      Rooms=NULL;
      Location=NULL;
      Cleanliness=NULL;
      CheckIn=NULL;
      Service=NULL;
      BusinessService=NULL;
      #Remove "hotel_" so only HotelID is entered into SQL
      gsub("hotel_","",hotelName)
      linestart=sprintf("INSERT INTO HotelReviews(HotelID,URL,Overall,AvgPrice,Author,Content,DateCreated
    ↪ ,NumOfReaders,NumFoundHelpful,OverallReview,MoneyValue,Rooms,Location,Cleanliness,CheckIn,Service,
    ↪ BusinessService) VALUES(%s,", hotelName)
   } {
      if ($1 == "<Overall Rating"){ #Store Overall Rating
         Overall=$2
         getline
      }

      if ($1 == "<Avg. Price"){ #Store Average Price
         AvgPrice=$2
         gsub("Unkonwn","NULL",AvgPrice)
         gsub(/[$]/,"",AvgPrice)
         gsub(",","",AvgPrice)
         getline
      }

      if ($1 == "<URL"){ #Store URL
         URL=$2
```

```
        getline
    }

    if ($1 == "<Author"){ #Store Author
        Author=$2
        getline
    }

    if ($1 == "<Content"){ #Store Content
        Content=$2
        gsub("\"",/[\"]/,Content)
        getline
    }

    if ($1 == "<Date"){ #Store Date
        DateCreated=$2
        gsub(",","",DateCreated)
        getline
    }

    if ($1 == "<No. Reader"){ #Store Number of Readers
        if ($2 == -1) {
            NumOfReaders="NULL"
        } else {
            NumOfReaders=$2
        }
        getline
    }

    if ($1 == "<No. Helpful"){ #Store Number Of Helpful
        if ($2 == -1) {
            NumFoundHelpful="NULL"
        } else {
            NumFoundHelpful=$2
        }
        getline
    }

    if ($1 == "<Overall"){ #Store Overall
        if ($2 == -1) {
            OverallReview="NULL"
        } else {
            OverallReview=$2
        }
        getline
    }

    if ($1 == "<Value"){ #Store Value for Money
        if ($2 == -1) {
            MoneyValue="NULL"
        } else {
            MoneyValue=$2
        }
        getline
    }

    if ($1 == "<Rooms"){ #Store Room Rating
        if ($2 == -1) {
            Rooms="NULL"
        } else {
            Rooms=$2
        }
        getline
```

```
        }

        if ($1 == "<Location"){ #Store Location Rating
            if ($2 == -1) {
                Location="NULL"
            } else {
                Location=$2
            }
            getline
        }

        if ($1 == "<Cleanliness"){ #Store Cleanliness Rating
            if ($2 == -1) {
                Cleanliness="NULL"
            } else {
                Cleanliness=$2
            }
            getline
        }

        if ($1 == "<Check in / front desk"){ #Store Check in / front desk Rating
            if ($2 == -1) {
                CheckIn="NULL"
            } else {
                CheckIn=$2
            }
            getline
        }

        if ($1 == "<Service"){ #Store Service Rating
            if ($2 == -1) {
                Service="NULL"
            } else {
                Service=$2
            }
            getline
        }

        if ($1 == "<Business service"){ #Store Business service Rating
            if ($2 == -1) {
                BusinessService="NULL"
            } else {
                BusinessService=$2
            }
            #Print the formatted line
            printf("%s \"%s\", %s, %s, \"%s\", \"%s\", \"%s\", %s, %s, %s, %s, %s, %s, %s, %s, %s, %s);\n",
    ↪ linestart,URL,Overall,AvgPrice,Author,Content,DateCreated,NumOfReaders,NumFoundHelpful,
    ↪ OverallReview,MoneyValue,Rooms,Location,Cleanliness,CheckIn,Service,BusinessService)
        }
    }' $hotel >> hotelreviews.sql
done
```