Databases Project – Spring 2019

Team No:

Names:

Contents

Contents 1

Deliverable 1 2

Assumptions 2

Entity Relationship Schema 2

Schema 2

Description 2

Relational Schema 2

ER schema to Relational schema 2

DDL 3

General Comments 3

Deliverable 2 4

Assumptions 4

Data Loading 4

Query Implementation 4

Query a: 4

Description of logic: 4

SQL statement 4

Interface 4

Design logic Description 4

Screenshots 4

General Comments 4

Deliverable 3 5

Assumptions 5

Query Implementation 5

Query a: 5

Description of logic: 5

SQL statement 5

Query Analysis 5

Selected Queries (and why) 5

Query 1 5

Query 2 5

Query 3 5

Interface 6

Design logic Description 6

Screenshots 6

General Comments 6

# Deliverable 1

## Assumptions

<In this section write down the assumptions you made about the data. Write a sentence for each assumption you made>

## Entity Relationship Schema

<In this section you should have figure of the ER schema as well as descriptions about entities and relations>

### Schema

<Add the figure of the ER schema>

### Description

<Describe all the choices you made for Entities and Relationships>

## Relational Schema

### ER schema to Relational schema

<Describe the transition from ER schema to Relational schema>

### DDL

## CREATE TABLE Listing

## (

## listing\_id INTEGER(10),

## listing\_url VARCHAR(255) NOT NULL,

## name VARCHAR(255),

## summary VARCHAR(255),

## space VARCHAR(255),

## description VARCHAR(255),

## neighborhood\_overview VARCHAR(255),

## notes VARCHAR(255),

## transit VARCHAR(255),

## access VARCHAR(255),

## interaction VARCHAR(255),

## house\_rules VARCHAR(255),

## pricture\_url VARCHAR(255),

host\_id INTEGER(10) NOT NULL,

venue\_id INTEGER(10) NOT NULL***,***

## PRIMARY KEY(listing\_id),

FOREIGN KEY(host\_id) REFERENCES Host(host\_id),

FOREIGN KEY(venue\_id) REFERENCES Venue(venue\_id),

UNIQUE(listing\_url)

)

The Listing Entity is translated into a table with one primary key, listing\_id, and two foreign keys, host\_id and venue\_id. With the two foreign keys, we combined the relationship Owned\_by and Located\_at with Listing since each listing will have one unique host and one unique venue. As a result, these two columns have NOT NULL constraint on their entry values.

Also, the listing\_url is set to be unique for all instances to ensure proper display of the listing items on the website.

## CREATE TABLE Host

## (

## host\_id INTEGER(10),

## host\_url VARCHAR(255) NOT NULL,

## host\_name VARCHAR(255) NOT NULL,

## host\_since DATE,

## host\_about VARCHAR(255),

## host\_response\_rate FLOAT(10),

## host\_response\_time VARCHAR(255),

## host\_thumbnail\_url VARCHAR(255),

## host\_neighborhood VARCHAR(255),

## host\_verifications VARCHAR(255),

## PRIMARY KEY(host\_id),

## UNIQUE(host\_url),

## )

## CREATE TABLE Venue

## (

## venue\_id INTEGER(10),

## neighborhood VARCHAR(255) ,

## city VARCHAR(255),

## country\_code VARCHAR(255),

## latitude DOUBLE(20),

## longtitude DOUBLE(20),

## PRIMARY KEY(venue\_id)

## )

## CREATE TABLE Review

## (

## review\_id INTERGER(10),

## listing\_id INTEGER(10) NOT NULL,

## date DATE NOT NULL,

## reviewer\_id INTERGER(10),

## reviewer\_name VARCHAR(255),

## comments VARCHAR(255),

PRIMARY KEY(review\_id, listing\_id),

FOREIGN KEY (listing\_id) REFERENCES Listing(listing\_id)

ON DELETE CASCADE

## )

The relationship Receives is combined with the Reviews table as each Reviews corresponds to a unique listing item. As Reviews is designed to be a weak entity of Listing, when one instance of Listing is deleted, the corresponding Reviews instances will be deleted as well.

## CREATE TABLE Score

## (

## score\_id INTERGER(10),

listing\_id INTEGER(10) NOT NULL,

## review\_scores\_accuracy INTEGER(10),

## review\_scores\_clean INTEGER(10),

## reciew\_scores\_checkin INTERGER(10),

## review\_scores\_communication INTERGER(10),

## review\_scores\_location INTERGER(10),

## review\_scores\_value INTERGER(10),

PRIMARY KEY(score\_id),

FOREIGN KEY (listing\_id) REFERENCES Listing(listing\_id)

)

The relationship Evaluated is combined with the Score table as each Score corresponds to a unique listing item.

## CREATE TABLE Policy

## (

## policy\_id INTEGER(10),

## is\_business\_travel\_ready BIT,

## cancellation\_policy VARCHAR(255),

## require\_guest\_profile\_picture BIT,

## require\_guest\_phone\_verification BIT,

listing\_id INTEGER(10) NOT NULL,

## PRIMARY KEY(policy\_id),

## FOREIGN KEY (listing\_id) REFERENCES Listing(listing\_id)

## )

The relationship Has is combined with the Policy table as each Policy instance corresponds to a unique Listing instance.

## CREATE TABLE Price

## (

## price\_id INTEGER(10),

## price FLOAT(10),

## weekly\_price FLOAT(10),

## monthly\_price FLOAT(10),

## security\_deposit FLOAT(10),

## cleaning\_fee FLOAT(10),

## guests\_included INTEGER(10),

## extra\_people INTERGER(10),

listing\_id INTEGER(10) NOT NULL,

## PRIMARY KEY(price\_id),

## FOREIGN KEY (listing\_id) REFERENCES Listing(listing\_id)

## )

The relationship Incurs is combined with the Price table as each Price instance corresponds to a unique Listing instance.

## CREATE TABLE House\_Details

## (

## detail\_id INTEGER(10),

## property\_type VARCHAR(255),

## room-type VARCHAR(255),

## accommodates VARCHAR(255),

## bathrooms INTEGER(10),

## bedrooms INTEGER(10),

## beds INTERGER(10),

## bed\_type VARCHAR(255),

## amenities VARCHAR(255),

## square\_feet INTEGER(10),

listing\_id INTEGER(10) NOT NULL,

## PRIMARY KEY(detail\_id),

FOREIGN KEY (listing\_id) REFERENCES Listing(listing\_id)

)

The relationship Pocesses is combined with the House\_Details table as each House\_Details instance corresponds to a unique Listing instance.

## CREATE TABLE Amenities

## (

## amenity\_id INTERGER(10),

## amenity\_name varchar(255),

## PRIMARY KEY(amenity\_id)

## )

## CREATE TABLE Contains

## (

## listing\_id INTEGER(10),

## amenitty\_id INTEGER(10),

## PRIMARY KEY(amenity\_id),

## FOREIGN KEY(listing\_id) REFERENCES Listing(listing\_id)

## )

The table Contains is created to store the many-to-many relationship between Amenities and Listing.

CREATE TABLE Calender

{

listing\_id INTEGER(10),

date DATE,

available BIT,

price FLOAT(10),

PRIMARY KEY(listing\_id, date),

FOREIGN KEY(listing\_id) REFERENCES Listing(listing\_id)

}

The relationship Reserves is combined with the Calendar table as each Calendar instance corresponds to exactly one Listing instance.

## General Comments

<In this section write general comments about your deliverable (comments and work allocation between team members>

# Deliverable 2

## Assumptions

<In this section write down the assumptions you made about the data. Write a sentence for each assumption you made>

## Data Loading

## Query Implementation

<For each query>

### Query a:

#### Description of logic:

<What does the query do and how do I decide to solve it>

#### SQL statement

<The SQL statement>

## Interface

### Design logic Description

<Describe the general logic of your design as well as the technology you decided to use>

### Screenshots

<Provide some initial screen shots of your interface>

## General Comments

<In this section write general comments about your deliverable (comments and work allocation between team members>

# Deliverable 3

# Assumptions

<In this section write down the assumptions you made about the data. Write a sentence for each assumption you made>

## Query Implementation

<For each query>

### Query a:

#### Description of logic:

<What does the query do and how do I decide to solve it>

#### SQL statement

<The SQL statement>

## Query Analysis

### Selected Queries (and why)

#### Query 1

<Initial Running time:

Optimized Running time:

Explain the improvement:

Initial plan

Improved plan>

#### Query 2

<Initial Running time:

Optimized Running time:

Explain the improvement:

Initial plan

Improved plan>

#### Query 3

<Initial Running time:

Optimized Running time:

Explain the improvement:

Initial plan

Improved plan>

# Interface

### Design logic Description

<Describe the general logic of your design as well as the technology you decided to use>

### Screenshots

<Provide some initial screen shots of your interface>

# General Comments

<In this section write general comments about your deliverable (comments and work allocation between team members>