

Data Management for Analytics

Database for Airbnb, Inc.



Authors:

Swati Kohli

Divya Raghunathan

Soham Milind Chaudhari

Sai Punith Godasi

Syed Asim

Table of Contents

1. Introduction	3
2. Entities	5
3. Business Rules	7
4. ER Diagram	8
5. Normalization	9
6. SQL Statements	11
6.1 Database Table creation	11
6.2 Data Loading	12
6.3 Retrieval Queries	13
7. Conclusion	19
7.1 Benefits:	19
7.2 Challenges:	208
8. Appendix	21

1. Introduction

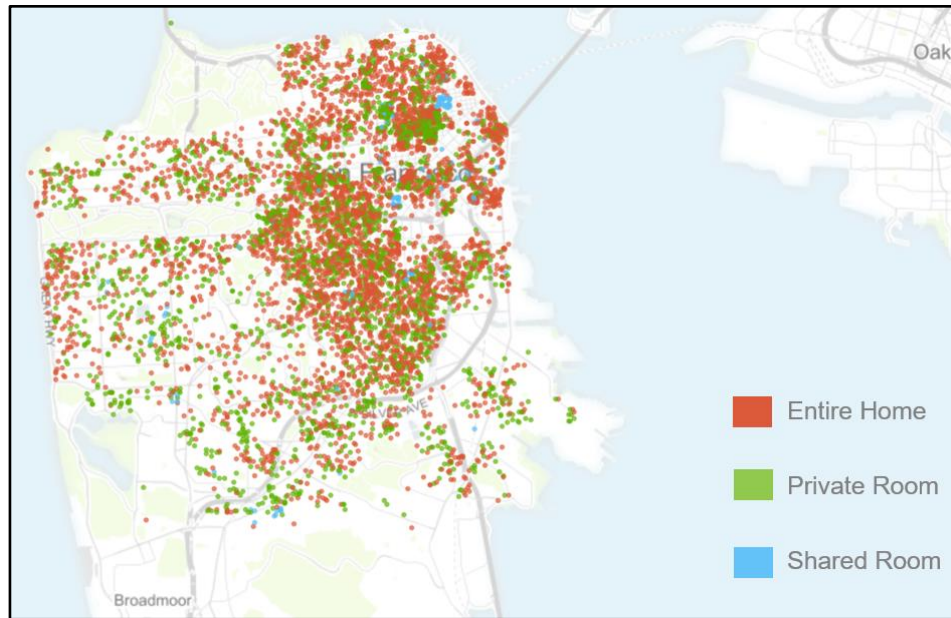
1.1. Project Name: Airbnb Relational Database Management System

1.2. Business Sector: Hospitality/ Travel Lodging

1.3.a Description: Airbnb has one of the most successful accommodation-providing services around the globe as an online rental platform. One can list, discover and book unique accommodations worldwide. Airbnb now has over four million listings in 65,000 cities across 191 countries. This accounts for a huge data repository which requires systematic management and running of the company.

1.3.b Area Focus : Our project focus is to understand how a company as an online marketplace offering homestays utilizes a digital platform with their Database and Relational Database Management System model to manage its property listings and stakeholders (Guest and Host only in this case). Through this data repository, we explore how AirBnB enables both Hosts and Guests to make informed, data-backed decisions for interaction choices in both online and offline world and also track their organization's periodic and overall performance. Our focus is to make scalable data by designing RDBMS for a city (San Francisco, California, USA) that can be expanded further in a similar fashion into states and countries to cover their area of presence.

1.4 Data Collection: The data has been sourced from InsideAirBnB website which is further sourced from publicly available information from the Airbnb site. The project focuses on a part of the data available- San Francisco, California. The link to the source can be referred to from <http://insideairbnb.com/get-the-data.html>.



1.5 Intent:

- On the front-end, for Airbnb, the user interface enables the search, collaboration and discovery of metadata related to accommodation rentals. We aim to illustrate how integral it is to incorporate a schema design that can make core processes related to these tasks, like populating new data records or maintaining databases more systematic even for non-experts to perform analytics.
- Furthermore, we seek to explore for a real-world fast-growing organization as AirBnB, what variety of metadata information is captured and how database model design captures this information about different data assets wherein their connectivity reflects the relationships of consumption, production, association, etc.
- By making a good schema and design of the database we can search, retrieve and analyze data to gather insights which would be extremely helpful in answering relevant questions like what is the occupancy rate of a listing? Or what is the income per month of each listing? Or evaluate discrete time slots available for a property or find a superhost based on some eligibility criteria.

2. Entities

Our design reflects seven main entities for Airbnb database and two multivalued entities.

1. User

User entity stores values specific to a single user. A user is someone who creates an account on the Airbnb website. The attributes stored under each user include User ID, Name, Address, Email ID and many more. The user entity is identified by its unique primary key- UserID. It can be of two types - Guest or host. A User is a supertype for Subtype Guest and Host following total specialization disjoint rule where he/she can be either and not both at the same time. So if a user is a host and wants to be a guest to use AirBnB for booking, it will be a separate record entry with a unique Guest ID assigned.

User attributes include User ID, Name, Address, Date of Birth, Gender. The attributes are general information about the User.

2. Guest

Guest entity stores values specific to a single guest. Guest is a type of User. A guest is someone who can make a booking on the Airbnb website. The attributes stored under each guest include GuestID, Rating, Verification and Emergency contact. The guest entity is identified by its unique primary key- Guest ID. Guest Rating is the rating received for each guest and the guest verification is whether the guest is verified or not.

3. Host

Host entity stores values specific to a single host. Host is a type of User. A host is someone who can list their properties on the Airbnb website for the guests to book. Some of the attributes stored under each host include HostID, Super host details, Host Since and Verification. The host entity is identified by its unique primary key- Host ID. Response Rate is the rate at which the host responds to the property booking. Host Since is the date since the user has been a host. Host Verification describes whether the

host has been verified or not. Acceptance Rate is the rate of bookings the host accepts. "Is Super host" describes whether the host is a super host i.e a status earned by a host through time and good reviews.

4. Review

Review entity stores values specific to one guest and one property. A guest leaves a review for a specific property. Some of the attributes stored under each review include Date of review, rating, recommendation. The review entity is identified by its unique primary key - Review ID. "Review date" is the date of review and "Rating" is the rating value given by the guest - 1,2,3,4,5.

5. Property

Property entity stores values specific to a single property. Each property is owned/managed by a single host. Properties are listings on the Airbnb website available for a guest to book. Hosts can have more than one listing. Some of the attributes stored under each property are Address, Charges for extra people, pet friendly, availability of kitchen. The property entity is identified by its unique primary key - Property ID. "Space" is the summary of the type of property. "Guests included" is the number of guests that are allowed to stay within the published price. "Extra people rate" is the Extra charge per person. Kitchen and pet friendly describe whether the property has a kitchen and if or not the property is pet friendly respectively.

6. Booking order

Booking entity stores values specific to a single booking made by a guest. When a guest makes a booking for a property, he receives a booking order ID. Some of the attributes of the booking order are date of arrival, number of nights, extra guests. Each booking order is identified by its unique primary key- Order ID. "Extra guests" is the number of additional people who would be staying at the property.

7. Calendar slot

Calendar slot entity stores values specific to a property. Each property may have many slots available in a year. Each property available slot is identified in the calendar slot entity by the Slot ID. Some of the attributes included in the calendar slots are availability details, price, minimum number of nights, maximum number of nights. "Available" describes the property's availability - "Y" for available, "N" for not available and "Adj price" is the Price after discounts.

8. Bed

Bed table stores type of bed for each room of a property available. This is a result of the multivalued attribute of Property entity where a property with more than one room for rent can have more than one type of bed. The type of beds are King, Queen and Twin.

9. Credit/Coupon

A guest's credit or coupon balance is stored in this table. This is a result of the multivalued attribute of Guest entity where a guest can have none, either or both through various holiday/festival coupons or refer to friend programs.

3. Business Rules

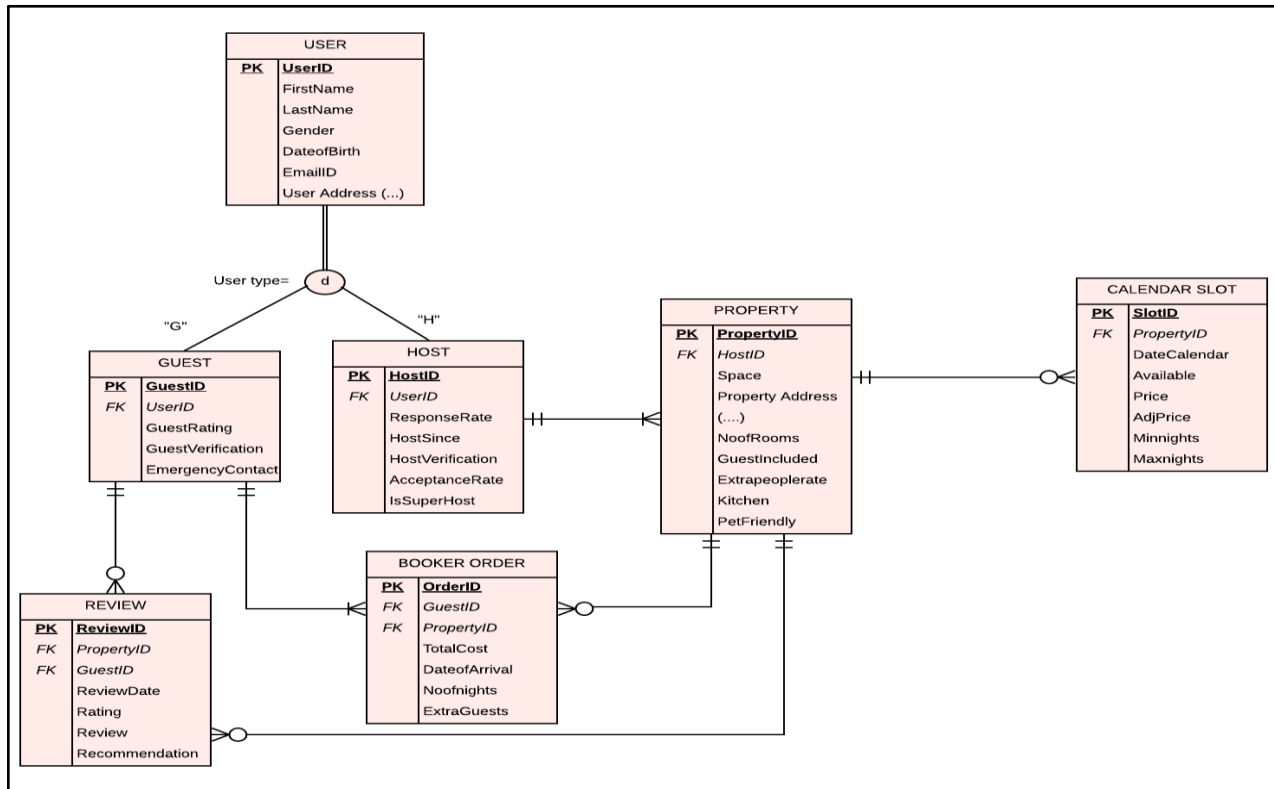
There are **10 relationships and 20 business rules** starting with a registered User.

- A User may be a Guest.
- A Guest must be a registered User.
- Each Guest must have at least one Booking (or is only a User until then)
- Each Booking must correspond to only one Guest.
- A Guest can give a Review for the stay experience with respect to every booking order he/she makes.
- Each Review is given by a single Guest.
- A Property can have one or more Reviews.
- Each Review associates only with a single Property.
- A User may be a Host.

-
- A Host must be a User.
 - A Host should have at least one Property listed (or is only a User until then).
 - Each Property is associated with only one Host.
 - A Property may or may not have a Booking. Also, a property can be booked multiple times.
 - Each Booking corresponds to one Property only.
 - A Property may have none or multiple Calendar_Slots of availability.
 - A Calendar_Slot must correspond to one Property only.
 - Each Property must have at least one Bed. Also, a property can have more than one Bed.
 - One Bed is associated with one Property only.
 - A Guest may have one or multiple credit/coupons.
 - Each credit/coupon can only be for a single Guest.

4. ER Diagram

Initial ERD for DBMS design basis



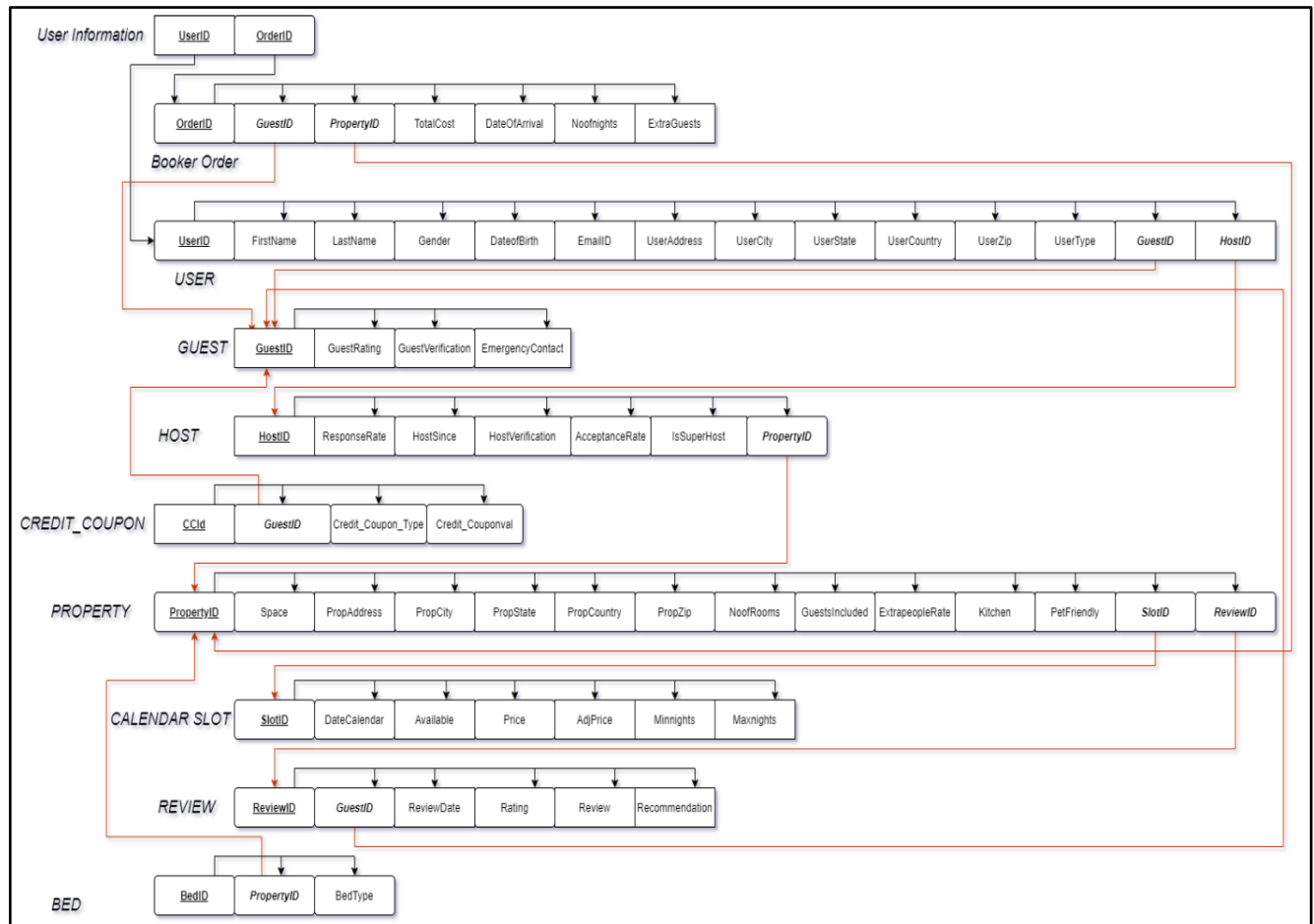
5. Normalization

Through data source, we got Host, property related entities in 1NF form. Further data tables on Review and Calendar Slot were individually available. However all other relevant entities for the RDBMS were brainstormed, studied and researched with fictional data created for the same. Therefore, the data and tables were a mix of both 1NF and 3NF. Additionally, some entity columns have been reduced for simplicity.

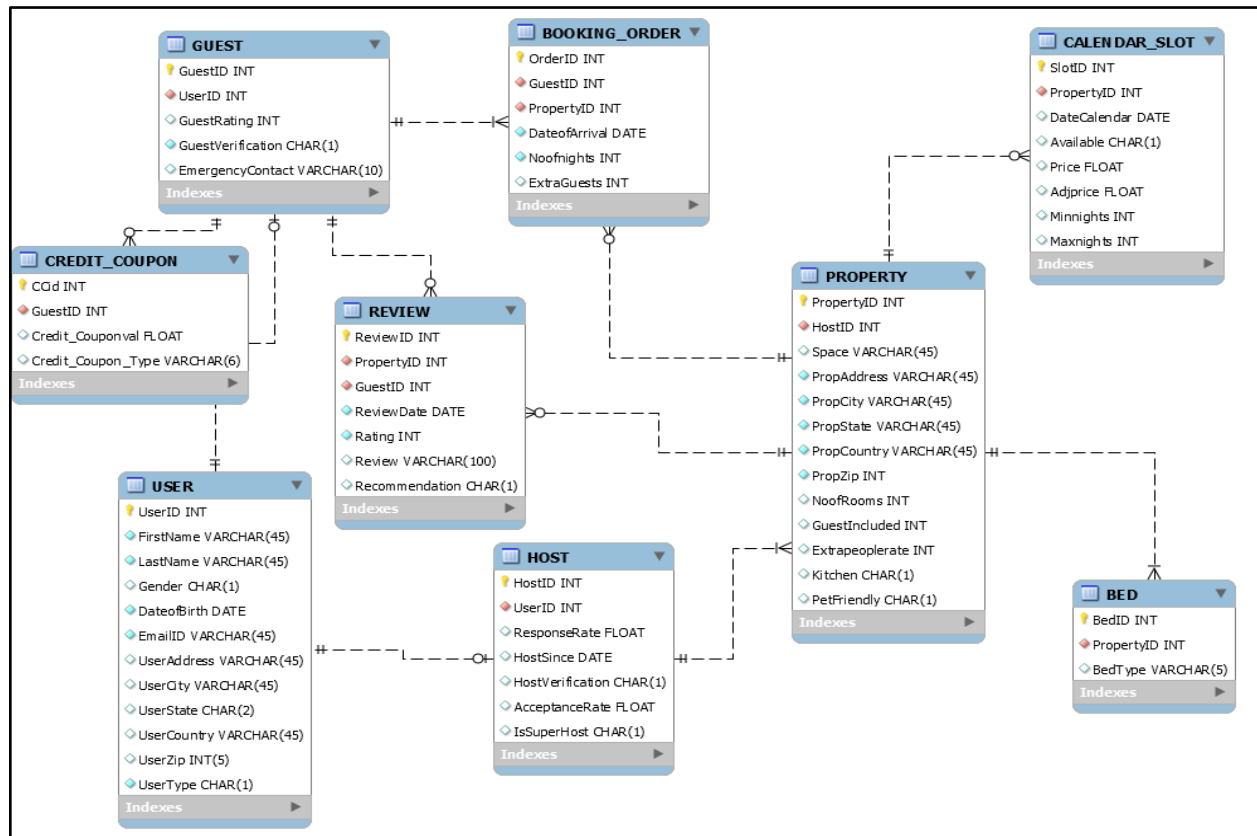
Following steps were followed for normalization.

1. Make every attribute value atomic (1NF)
2. Reflect requirement of relation (1NF)
3. Remove multivalued attributes to create entities (1NF)
4. Remove full, partial and transitive dependencies (2NF)

Complete 3NF relational model (Also as 2NF)



Entity Relation Diagram (From 3NF)



6. SQL Statements

6.1 Database Table creation

Now that the data model is designed and normalized, we can define the columns needed for each table. A schema called "Airbnb" was created which is the structure that contains descriptions of objects created. The entire table creation MySQL code is given in Appendix 1. In this section, we shall go over some of the important steps made while creating the tables.

- 1) Each table has its unique primary key. Other non primary key columns in the table that are required to be not null are designated as NOT NULL or decided for default values.
- 2) Metadata is defined to describe the data records and relationships within entities.
- 3) Referential integrity is maintained by defining a parent-child relationship in the form of foreign keys to avoid anomalies related to deletion, updation and insertion.

-
- 4) Validation rules have been established to constraint the values that are inserted for some of the columns. For example, in the USER table, the gender column is CHAR(1) data type and can allow only characters "M" or "F" to be inserted into the column or choose to have Null if a User does not wish to disclose.

6.2 Data Loading

After creating Entities, records are created for each entity to build a database. SQL Code to create the first record for each entity is given below. Rest can be referred to in Appendix 2.

```
INSERT INTO USER
(UserID,FirstName,LastName,Gender,DateofBirth,EmailID,UserAddress,UserCity,UserState,UserCountry,UserZip,UserType) VALUES (101,'Mitzi','Clein','F','1987-05-01','mz12@gmail.com','123 Floribunda Ave','San Francisco','CA','USA',94016,'G');
```

```
INSERT INTO GUEST (GuestID,UserID,GuestRating,GuestVerification,EmergencyContact) VALUES (201,101,4,'Y','5105331221');
```

```
INSERT INTO HOST
(HostID,UserID,ResponseRate,HostSince,HostVerification,AcceptanceRate,IsSuperHost) VALUES (301,111,0.86,'2008-07-31','N',0.8,'Y');
```

```
INSERT INTO PROPERTY
(PropertyID,HostID,SpaceType,PropAddress,PropCity,PropState,PropCountry,PropZip,NoofRooms,GuestIncluded,ExtrapeopleRate,Kitchen,Petfriendly) VALUES (401,301,'Apartment','121 Alma St','San Francisco','CA','USA',94117,1,2,25,NULL,'Y');
```

```
INSERT INTO BOOKING_ORDER
(OrderID,GuestID,PropertyID,BookingDate,DateofArrival,Noofnights,ExtraGuests) VALUES (501,201,401,'2018-12-01','2018-12-15',3,1);
```

```
INSERT INTO REVIEW (ReviewID,PropertyID,GuestID,ReviewDate,Rating,Review,Recommendation) VALUES (601,401,201,'2019-01-02',3,'stay was okish','N');
```

```
INSERT INTO CALENDAR_SLOT
(SlotID,PropertyID,DateCalendar,Available,Price,Adjprice,Minnights,Maxnights) VALUES
(701,401,'2019-01-01','Y',135.00,130.00,1,30);

INSERT INTO BED (BedID,PropertyID,BedType) VALUES (801,401,'King');

INSERT INTO CREDIT_COUPON (Ccid,GuestID,Credit_couponval,Credit_Coupon_Type) VALUES
(901,201,50,'Credit');
```

6.3 Retrieval Queries

Various SQL elements are used in the queries below to demonstrate element variety and application for the organization. Specifically, the elements covered are:

1. **Select Clauses:** SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY
2. **Functions** - COUNT, ROUND, AVERAGE, SUM, DATE_FORMAT
3. **Boolean Operators:** AND , NOT, LIKE
4. **Comparison Operators:** =, >, <
5. **Multiple Tables Queries:** INNER JOIN, ON,
6. **SubQuery** : IN, NOT IN, NOT EXISTS, Derived table, UNION,
7. **Conditional:** CASE (WHEN,THEN) with Derived Table

6.3.1 First Query

6.3.1.a Purpose of Query: Occupancy rate of each property based on availability.

What is the percentage availability of the properties in 2019,? In other words, in a year(365 days), out of the total days when the property is available (has a calendar slot), what percentage of days is the property rented?

6.3.1.b SQL Command:

```
SELECT b.Property_ID, ROUND((b.counts/a.total)*100,2) AS Availability_Rate
```

```

FROM
    (SELECT calendar_slot.propertyid AS Property_ID, COUNT(calendar_slot.slotid) AS
    counts
    FROM calendar_slot
    WHERE
        calendar_slot.available='Y'
        AND CALENDAR_SLOT.DateCalendar> DATE_FORMAT('2018-12-31','%y-
        %m-%d')
        AND CALENDAR_SLOT.DateCalendar < DATE_FORMAT('2020-01-01','%y-
        %m-%d')
        GROUP BY calendar_slot.PropertyID)b,

    (SELECT calendar_slot.propertyid AS p2, COUNT(calendar_slot.slotid) AS
    total
    FROM calendar_slot
    WHERE calendar_slot.available='Y'
    AND CALENDAR_SLOT.DateCalendar> DATE_FORMAT('2018-12-31','%y-
    %m-%d')
    AND CALENDAR_SLOT.DateCalendar < DATE_FORMAT('2020-01-01','%y-
    %m-%d') )a;

```

6.3.1.c Result of Query:

	Property_ID	Availability_Rate
▶	401	25.00
	402	25.00
	403	37.50
	404	12.50

6.3.2 Second Query

6.3.2.a Purpose of Query: To list the available properties from cheapest to most expensive in San Francisco for the month of August 2019

6.3.2.b SQL Command:

```

SELECT Calendar_slot.PropertyID, CALENDAR_SLOT.Price, CALENDAR_SLOT.DateCalendar,
PROPERTY.PropCity
FROM CALENDAR_SLOT, Property
WHERE
    CALENDAR_SLOT.Available="Y"
    AND PROPERTY.PropertyID=CALENDAR_SLOT.PropertyID

```

```

AND PROPERTY.PropCity LIKE '%San Francisco%'
AND CALENDAR_SLOT.DateCalendar > DATE_FORMAT('2019-07-31','%y-%m-%d')
AND CALENDAR_SLOT.DateCalendar < DATE_FORMAT('2019-09-01','%y-%m-%d')
ORDER BY CALENDAR_SLOT.PRICE ;

```

6.3.2.c Result of Query:

PropertyID	Price	DateCalendar	PropCity
402	77	2019-08-01	San Francisco
403	196	2019-08-01	San Francisco

6.3.3 Third Query

6.3.3.a Purpose of Query: Guest's generic search based

Find the top hot properties (property IDs) in San Francisco, their original price and associated Host ID ,name and IsSuperhost based on the average review rating of that property to be greater than 3. Display their average rating as well.

6.3.3.b SQL Command:

```

USE Airbnb ;
SELECT p.PropertyID, h.HostID, u.FirstName, u.LastName, h.IsSuperhost, AVG(r.Rating),
c.Price
FROM user u, review r, host h, property p, calender_slot c
WHERE
    p.HostID = h.HostID
    AND p.PropCity = 'San Francisco'
    AND h.UserID = u.UserID
    AND r.PropertyID = p.PropertyID
    AND p.PropertyID = c.PropertyID
GROUP BY r.PropertyID
Having AVG(r.Rating) > 3

```

6.3.3.c Result of Query:

PropertyID	HostID	FirstName	LastName	IsSuperhost	AVG(r.Rating)	Price
401	301	Holly	Lopez	Y	3.5000	135
403	303	Ivan & Wendy	Baker	Y	3.6667	129

6.3.4 Fourth Query

6.3.4.a Purpose of Query: Guest's specific search based

List properties in an area of San Francisco that have at least one King Bed, come with a kitchen and are Petfriendly.

6.3.4.b SQL Command:

```
SELECT PROPERTY.PropertyID, PROPERTY.Kitchen, PROPERTY.PetFriendly, BED.BedType,
PROPERTY.PropAddress
FROM PROPERTY INNER JOIN BED
WHERE
    PROPERTY.PropertyID = BED.PropertyID
    AND PROPERTY.PropAddress like '%22nd st%'
    AND PROPERTY.Kitchen = 'Y'
    AND PROPERTY.PetFriendly = 'Y'
    AND BED.BedType LIKE '%King%';
```

6.3.4.c Result of Query:



The screenshot shows a 'Result Grid' window with a toolbar containing 'Filter Rows' and 'Export' buttons. The grid has five columns: PropertyID, Kitchen, PetFriendly, BedType, and PropAddress. A single row of data is displayed with the following values: 403, Y, Y, King, and 22nd st.

PropertyID	Kitchen	PetFriendly	BedType	PropAddress
403	Y	Y	King	22nd st

6.3.5 Fifth Query

6.3.5.a Purpose of Query: Specific search based

Display the properties in the area of San Francisco that are 2 bedroom houses with a rating of 3 or above.

6.3.5.b SQL Command:

```
USE Airbnb ;
SELECT PROPERTY.PropertyID, PROPERTY.NoofRooms, PROPERTY.SpaceType,
REVIEW.RATING, PROPERTY.PropAddress, PROPERTY.PropCity, PROPERTY.PropState
FROM PROPERTY INNER JOIN REVIEW
ON PROPERTY.PropertyID = REVIEW.PropertyID
WHERE
    PROPERTY.SpaceType LIKE 'House'
    AND PROPERTY.NoofRooms = 2
```

AND REVIEW.RATING = 3;

6.3.5.c Result of Query:



The screenshot shows a database query result grid with the following columns: PropertyID, NoofRooms, SpaceType, RATING, PropAddress, PropCity, and PropState. There are two rows of data, both for PropertyID 402, NoofRooms 2, SpaceType House, and RATING 3. The PropAddress is Berkeley Way, PropCity is San Francisco, and PropState is CA.

PropertyID	NoofRooms	SpaceType	RATING	PropAddress	PropCity	PropState
402	2	House	3	Berkeley Way	San Francisco	CA
402	2	House	3	Berkeley Way	San Francisco	CA

6.3.6 Sixth Query

6.3.6.a Purpose of Query: To find the total value of credit and coupon owned by each guest.

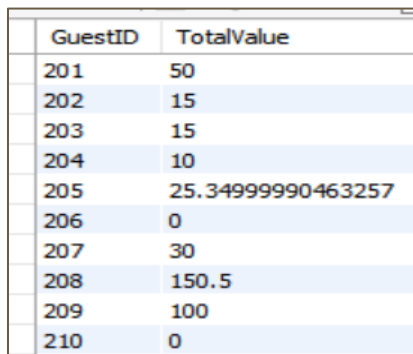
6.3.6.b SQL Command:

```
SELECT GuestID, SUM(Credit_couponval)as TotalValue
FROM CREDIT_COUPON
GROUP BY GuestID

UNION

SELECT GuestID, 0
FROM GUEST
WHERE NOT EXISTS
      (SELECT * FROM CREDIT_COUPON
       WHERE CREDIT_COUPON.GuestID = Guest.GuestID);
```

6.3.6.c Result of Query:



The screenshot shows a database query result grid with the following columns: GuestID and TotalValue. There are 10 rows of data, showing GuestIDs from 201 to 210 and their corresponding TotalValues.

GuestID	TotalValue
201	50
202	15
203	15
204	10
205	25.34999990463257
206	0
207	30
208	150.5
209	100
210	0

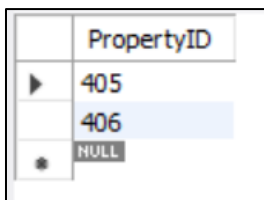
6.3.7 Seventh Query

6.3.7.a Purpose of Query: Search based query to list properties that have never been booked by a guest

6.3.7.b SQL Command:

```
SELECT PropertyID
FROM PROPERTY
WHERE PROPERTY.PropertyID NOT IN
      (SELECT PropertyID from BOOKING_ORDER);
```

6.3.7.c Result of Query:



PropertyID
405
406
NULL

6.3.8 Eighth Query

6.3.8.a Purpose of Query: Case Query



List out the full name, Guest ID and total sum of credit and coupon values of the guests that have both credit and coupon values in their account.

6.3.8.b SQL Command:

```
SELECT u.firstname, u.lastname, g.guestid, sum(Credit_Couponval)
FROM credit_coupon c, guest g, User u
WHERE
      g.guestid = c.guestid
      AND g.userid = u.userid
      AND g.guestid IN
            (SELECT c. guestid
             FROM Credit_Coupon c
             GROUP BY c.guestid
             HAVING SUM(CASE
                   WHEN c.Credit_Coupon_Type = 'Credit' THEN 1
                   WHEN c.Credit_Coupon_Type = 'Coupon' THEN 1
                   END) = 2)
```

GROUP BY c.guestid;

6.3.8.c Result of Query:

Result Grid   Filter Rows: <input type="text"/> Export:				
	firstname	lastname	guestid	sum(Credit_Couponval)
▶	Tyler	Jones	205	25.34999990463257
	Steve	Williams	208	150.5

7. Conclusion

7.1 Benefits:

Schema based design with a database repository is a critical component to a fast and smart growth for an organization. Through this project we gained an understanding of how to map a business through its database because of its self describing nature. This is because a DBMS not only contains the database itself, but also metadata to define and describe the data and relationships between tables in the database.

Additionally, the management system prevents data redundancy or duplication of data and any change in it is reflected immediately.

Another advantage we learnt is how helpful is a scalable database system design for a growing business especially across countries. To make the database scalable, we understood the importance of RDBMS principles of data integrity, security, privacy, consistency and backup and recovery.

Particularly, the enforcement of integrity constraints gives the benefit of adding restriction or rule that dictates what can be entered or edited in a table such as phone number is 10 digit long or adding valid user type (Guest or Host) only. This is to ensure that users enter valid information and maintain data integrity.

Data independence is another advantage through DBMS wherein the data describing data (metadata) are separated from the application programs because the changes to the data structure are handled by the DBMS and not through embedding in the program.

7.2 Challenges:

We faced errors and inefficiencies with our initial ERD model and after evaluating all of them stepwise as per the database management process, we set out to systematically resolve those problems.

It was a challenge to decide the approach of the humongous AirBnB data of myriad countries with the perspective of DBMS. In other words, should we take a top down or bottom up approach. However, designing a base ERD and taking a scalable approach i.e. to start from a city that can be scaled to a state and in turn a country level data worked effectively.

Designing for data integrity was another challenge which took quite a lot of brainstorming amongst the group members. For example, how to code uniquely for various IDs of entities like Guest, Host, User etc. so that they don't have a conflict of overlap from the perspective of organization and also facilitate entering of records. For this we explored various options like auto prefix or auto increments and found that adding a unique code in front of each entity to make it unique was effective.

Another challenge was to make the data entries as error free as possible. For this adding checks in metadata descriptions proved successful. For example, in Host entity, a Host is a Superhost can only have Y or N or Null as a record and not anything else.

After many evolutionary steps, it gave us as a group the confidence that the AirBnB project using RDBMS for data infrastructure stack design is stable, reliable, and scalable.

8. Appendix

Appendix 1 : Tables creation

```
CREATE SCHEMA IF NOT EXISTS Airbnb;  
USE Airbnb ;
```

```
DROP TABLE IF EXISTS BOOKING_ORDER;  
DROP TABLE IF EXISTS REVIEW;  
DROP TABLE IF EXISTS CREDIT_COUPON;  
DROP TABLE IF EXISTS GUEST;  
DROP TABLE IF EXISTS CALENDAR_SLOT;  
DROP TABLE IF EXISTS BED;  
DROP TABLE IF EXISTS PROPERTY;  
DROP TABLE IF EXISTS HOST;  
DROP TABLE IF EXISTS USER;
```

```
CREATE TABLE IF NOT EXISTS USER (  
  UserID INT NOT NULL,  
  FirstName VARCHAR(45) NOT NULL,  
  LastName VARCHAR(45) NOT NULL,  
  Gender CHAR(1) CHECK(Gender IN ('M','F')),  
  DateofBirth DATE NOT NULL,  
  EmailID VARCHAR(45) NOT NULL,  
  UserAddress VARCHAR(45),  
  UserCity VARCHAR(45),  
  UserState CHAR(2),  
  UserCountry VARCHAR(45),  
  UserZip INT(5),  
  UserType CHAR(1) NOT NULL CHECK(UserType IN ('G','H')),  
  CONSTRAINT User_pk PRIMARY KEY (UserID));
```

```
CREATE TABLE IF NOT EXISTS GUEST (  
  GuestID INT NOT NULL,  
  UserID INT NOT NULL,  
  GuestRating INT CHECK(GuestRating IN ('1','2','3','4','5')),  
  GuestVerification CHAR(1) CHECK(GuestVerification IN ('Y','N')),  
  EmergencyContact VARCHAR(10),  
  CONSTRAINT GUEST_pk PRIMARY KEY (GuestID),  
  CONSTRAINT GUEST_fk FOREIGN KEY (UserID) REFERENCES USER(UserID));
```

```
CREATE TABLE IF NOT EXISTS HOST (  
HostID INT NOT NULL,  
UserID INT NOT NULL,  
ResponseRate FLOAT ,  
HostSince DATE ,  
HostVerification CHAR(1) CHECK(HostVerification IN ('Y','N')),  
AcceptanceRate FLOAT ,  
IsSuperHost CHAR(1) CHECK(IsSuperHost IN ('Y','N')),  
CONSTRAINT HOST_pk PRIMARY KEY (HostID),  
CONSTRAINT HOST_fk FOREIGN KEY (UserID) REFERENCES USER(UserID));
```

```
CREATE TABLE IF NOT EXISTS PROPERTY (  
PropertyID INT NOT NULL,  
HostID INT NOT NULL,  
SpaceType VARCHAR(45),  
PropAddress VARCHAR(45) NOT NULL,  
PropCity VARCHAR(45) NOT NULL,  
PropState VARCHAR(45) NOT NULL,  
PropCountry VARCHAR(45) NOT NULL,  
PropZip INT NOT NULL,  
NoofRooms INT ,  
GuestIncluded INT ,  
Extrapeoplerate FLOAT ,  
Kitchen CHAR(1) CHECK(Kitchen IN ('Y','N')),  
PetFriendly CHAR(1) CHECK(PetFriendly IN ('Y','N')),  
CONSTRAINT PROPERTY_pk PRIMARY KEY (PropertyID),  
CONSTRAINT PROPERTY_fk FOREIGN KEY (HostID) REFERENCES HOST(HostID));
```

```
CREATE TABLE IF NOT EXISTS BOOKING_ORDER (  
OrderID INT NOT NULL,  
GuestID INT NOT NULL,  
PropertyID INT NOT NULL,  
BookingDate DATE NOT NULL,  
DateofArrival DATE NOT NULL,  
Noofnights INT NOT NULL,  
ExtraGuests INT,  
CONSTRAINT BOOKING_ORDER_pk PRIMARY KEY (OrderID),  
CONSTRAINT BOOKING_ORDER_fk1 FOREIGN KEY (GuestID) REFERENCES GUEST(GuestID),  
CONSTRAINT BOOKING_ORDER_fk2 FOREIGN KEY (PropertyID) REFERENCES  
PROPERTY(PropertyID));
```

```
CREATE TABLE IF NOT EXISTS REVIEW (
```

```
ReviewID INT NOT NULL,
PropertyID INT NOT NULL,
GuestID INT NOT NULL,
ReviewDate DATE NOT NULL,
Rating INT NOT NULL,
Review VARCHAR(100),
Recommendation CHAR(1) CHECK(Recommendation IN ('Y','N')),
CONSTRAINT REVIEW_pk PRIMARY KEY (ReviewID),
CONSTRAINT REVIEW_fk1 FOREIGN KEY (PropertyID) REFERENCES PROPERTY(PropertyID),
CONSTRAINT REVIEW_fk2 FOREIGN KEY (GuestID) REFERENCES GUEST(GuestID));
```

```
CREATE TABLE IF NOT EXISTS CALENDAR_SLOT (
SlotID INT NOT NULL,
PropertyID INT NOT NULL,
DateCalendar DATE ,
Available CHAR(1) CHECK(Available IN ('Y','N')),
Price FLOAT ,
Adjprice FLOAT ,
Minnights INT ,
Maxnights INT ,
CONSTRAINT CALENDAR_SLOT_pk PRIMARY KEY (SlotID),
CONSTRAINT CALENDAR_SLOT_fk FOREIGN KEY (PropertyID) REFERENCES
PROPERTY(PropertyID));
```

```
CREATE TABLE IF NOT EXISTS BED (
BedID INT NOT NULL,
PropertyID INT NOT NULL,
BedType VARCHAR(5) CHECK(BedType IN ('King','Queen','Twin')),
CONSTRAINT BED_pk PRIMARY KEY (BedID),
CONSTRAINT BED_fk FOREIGN KEY (PropertyID) REFERENCES PROPERTY(PropertyID));
```

```
CREATE TABLE IF NOT EXISTS CREDIT_COUPON (
CCid INT NOT NULL,
GuestID INT NOT NULL,
Credit_Couponval FLOAT ,
Credit_Coupon_Type VARCHAR(6) CHECK(Credit_Coupon_Type IN ('Credit','Coupon')),
CONSTRAINT CREDIT_COUPON_pk PRIMARY KEY (CCid),
CONSTRAINT CREDIT_COUPON_fk FOREIGN KEY (GuestID) REFERENCES GUEST(GuestID));
```

Appendix 2 : Data Loading

USER Data Loading

INSERT INTO USER

(UserID,FirstName,LastName,Gender,DateofBirth,EmailID,UserAddress,UserCity,UserState,UserCountry,User
Zip,UserType)

VALUES (101,'Mitzi','Clein','F','1987-05-01','mz12@gmail.com','123 Floribunda Ave','San
Francisco','CA','USA',94016,'G');

INSERT INTO USER

(UserID,FirstName,LastName,Gender,DateofBirth,EmailID,UserAddress,UserCity,UserState,UserCountry,User
Zip,UserType)

VALUES (102,'Renu','Chopra','F','1990-10-10','choprar@gmail.com','123 abc','San Jose','ST','USA',94088,'G');

INSERT INTO USER

(UserID,FirstName,LastName,Gender,DateofBirth,EmailID,UserAddress,UserCity,UserState,UserCountry,User
Zip,UserType)

VALUES (103,'Matthew','Smith','M','1980-03-02','ms0302@gmail.com','1211 johnson drive','Buffalo
Grove','IL','USA',60089,'G');

INSERT INTO USER

(UserID,FirstName,LastName,Gender,DateofBirth,EmailID,UserAddress,UserCity,UserState,UserCountry,User
Zip,UserType)

VALUES (104,'Tina','Johnson','F','1975-12-20','tinaj@gmail.com','87 adams
lake','atlanta','GA','USA',30339,'G');

INSERT INTO USER

(UserID,FirstName,LastName,Gender,DateofBirth,EmailID,UserAddress,UserCity,UserState,UserCountry,User
Zip,UserType)

VALUES (105,'Tyler','Jones','M','1976-11-23','Tylerjones@gmail.com','123 abc','Palo
Alto','ST','USA',94020,'G');

INSERT INTO USER

(UserID,FirstName,LastName,Gender,DateofBirth,EmailID,UserAddress,UserCity,UserState,UserCountry,User
Zip,UserType)

VALUES (106,'Jessica','Davis','F','1982-06-14','Jessd@gmail.com','67 hathaway
ave','woodstock','GA','USA',30188,'G');

INSERT INTO USER

(UserID,FirstName,LastName,Gender,DateofBirth,EmailID,UserAddress,UserCity,UserState,UserCountry,User
Zip,UserType)

VALUES (107,'Anthony','Miller','M','1990-01-01','Millerann@gmail.com','123 abc','Test
City','ST','USA',12345,'G');

INSERT INTO USER

(UserID,FirstName,LastName,Gender,DateofBirth,EmailID,UserAddress,UserCity,UserState,UserCountry,User
Zip,UserType)

VALUES (108,'Steve','Williams','F','1974-11-23','Wills@gmail.com','45 1st
drive','everett','WA','USA',98208,'G');

```

INSERT INTO USER
(UserID,FirstName,LastName,Gender,DateofBirth,EmailID,UserAddress,UserCity,UserState,UserCountry,User
Zip,UserType)
VALUES (109,'Kalev','Brown','F','1980-05-17','Brownboo@gmail.com','345 camellia
lane','atlanta','GA','USA',30324,'G');
INSERT INTO USER
(UserID,FirstName,LastName,Gender,DateofBirth,EmailID,UserAddress,UserCity,UserState,UserCountry,User
Zip,UserType)
VALUES (110,'Aaron','Cony','M','1987-05-29','Connron@gmail.com','77 lady fern
street','ashburn','VA','USA',12345,'G');
INSERT INTO USER
(UserID,FirstName,LastName,Gender,DateofBirth,EmailID,UserAddress,UserCity,UserState,UserCountry,User
Zip,UserType)
VALUES (111,'Holly','Lopez','M','1964-05-17','Hollylop@gmail.com','123 abc','Test City','ST','USA',12345,'H');
INSERT INTO USER
(UserID,FirstName,LastName,Gender,DateofBirth,EmailID,UserAddress,UserCity,UserState,UserCountry,User
Zip,UserType)
VALUES (112,'Kevin','Green','M','1987-01-04','green4187@gmail.com','123 abc','Test
City','ST','USA',12345,'H');
INSERT INTO USER
(UserID,FirstName,LastName,Gender,DateofBirth,EmailID,UserAddress,UserCity,UserState,UserCountry,User
Zip,UserType)
VALUES (113,'Ivan & Wendy','Baker','M','1989-12-25','Newbaker@gmail.com','123 abc','Test
City','ST','USA',12345,'H');
INSERT INTO USER
(UserID,FirstName,LastName,Gender,DateofBirth,EmailID,UserAddress,UserCity,UserState,UserCountry,User
Zip,UserType)
VALUES (114,'Bernat','Powell','M','1997-11-27','powellprop@gmail.com','123 abc','Test
City','ST','USA',12345,'H');
INSERT INTO USER
(UserID,FirstName,LastName,Gender,DateofBirth,EmailID,UserAddress,UserCity,UserState,UserCountry,User
Zip,UserType)
VALUES (115,'Aaron','Diaz','F','1973-02-18','Aaronprop@gmail.com','123 abc','Test
City','ST','USA',12345,'H');
INSERT INTO USER
(UserID,FirstName,LastName,Gender,DateofBirth,EmailID,UserAddress,UserCity,UserState,UserCountry,User
Zip,UserType)
VALUES (116,'Olivia','Jolie','F','1975-12-27','OJ2712@gmail.com','123 abc','Test City','ST','USA',12345,'H');

```

GUEST Data Loading

```

INSERT INTO GUEST (GuestID,UserID,GuestRating,GuestVerification,EmergencyContact) VALUES
(201,101,4,'Y','5105331221');

```

```
INSERT INTO GUEST (GuestID,UserID,GuestRating,GuestVerification,EmergencyContact) VALUES
(202,102,3,'Y','9167796688');
INSERT INTO GUEST (GuestID,UserID,GuestRating,GuestVerification,EmergencyContact) VALUES
(203,103,3,'N','9495559494');
INSERT INTO GUEST (GuestID,UserID,GuestRating,GuestVerification,EmergencyContact) VALUES
(204,104,5,'Y','6508883663');
INSERT INTO GUEST (GuestID,UserID,GuestRating,GuestVerification,EmergencyContact) VALUES
(205,105,5,'Y','9259564338');
INSERT INTO GUEST (GuestID,UserID,GuestRating,GuestVerification,EmergencyContact) VALUES
(206,106,5,'Y','6693450050');
INSERT INTO GUEST (GuestID,UserID,GuestRating,GuestVerification,EmergencyContact) VALUES
(207,107,1,'N','9493437783');
INSERT INTO GUEST (GuestID,UserID,GuestRating,GuestVerification,EmergencyContact) VALUES
(208,108,5,'Y','6502235615');
INSERT INTO GUEST (GuestID,UserID,GuestRating,GuestVerification,EmergencyContact) VALUES
(209,109,5,'Y','4157128422');
INSERT INTO GUEST (GuestID,UserID,GuestRating,GuestVerification,EmergencyContact) VALUES
(210,110,4,'Y','2249054643');
```

HOST Data Loading

```
INSERT INTO HOST (HostID,UserID,ResponseRate,HostSince,HostVerification,AcceptanceRate,IsSuperHost)
VALUES (301,111,0.86,'2008-07-31','N',0.8,'Y');
INSERT INTO HOST (HostID,UserID,ResponseRate,HostSince,HostVerification,AcceptanceRate,IsSuperHost)
VALUES (302,112,1,'2008-12-08','Y',1,'N');
INSERT INTO HOST (HostID,UserID,ResponseRate,HostSince,HostVerification,AcceptanceRate,IsSuperHost)
VALUES (303,113,1,'2009-01-27','Y',1,'Y');
INSERT INTO HOST (HostID,UserID,ResponseRate,HostSince,HostVerification,AcceptanceRate,IsSuperHost)
VALUES (304,114,1,'2009-09-15','Y',0.9,'N');
INSERT INTO HOST (HostID,UserID,ResponseRate,HostSince,HostVerification,AcceptanceRate,IsSuperHost)
VALUES (305,115,0.83,'2009-06-17','N',1,'Y');
INSERT INTO HOST (HostID,UserID,ResponseRate,HostSince,HostVerification,AcceptanceRate,IsSuperHost)
VALUES (306,116,1,'2009-11-18','Y',0.95,'Y');
```

PROPERTY Data Loading

```
INSERT INTO PROPERTY
(PropertyID,HostID,SpaceType,PropAddress,PropCity,PropState,PropCountry,PropZip,NoofRooms,GuestIncluded,Extrapeoplerate,Kitchen,Petfriendly) VALUES (401,301,'Apartment','121 Alma St','San Francisco','CA','USA',94117,1,2,25,NULL,'Y');
INSERT INTO PROPERTY
(PropertyID,HostID,SpaceType,PropAddress,PropCity,PropState,PropCountry,PropZip,NoofRooms,GuestIncluded,Extrapeoplerate,Kitchen,Petfriendly) VALUES (402,302,'House','Berkeley Way','San Francisco','CA','USA',94131,2,2,20,'Y','Y');
```

INSERT INTO PROPERTY

(PropertyID,HostID,SpaceType,PropAddress,PropCity,PropState,PropCountry,PropZip,NoofRooms,GuestIncluded,Extrapeoplerate,Kitchen,Petfriendly) VALUES (403,303,'Condominium','22nd st','San Francisco','CA','USA',94110,1,2,60,'Y','Y');

INSERT INTO PROPERTY

(PropertyID,HostID,SpaceType,PropAddress,PropCity,PropState,PropCountry,PropZip,NoofRooms,GuestIncluded,Extrapeoplerate,Kitchen,Petfriendly) VALUES (404,304,'Apartment','Dodge St','San Francisco','CA','USA',94102,3,1,0,'N','N');

INSERT INTO PROPERTY

(PropertyID,HostID,SpaceType,PropAddress,PropCity,PropState,PropCountry,PropZip,NoofRooms,GuestIncluded,Extrapeoplerate,Kitchen,Petfriendly) VALUES (405,305,'Apartment','Belmont Ave','San Francisco','CA','USA',94117,1,1,12,'N','N');

INSERT INTO PROPERTY

(PropertyID,HostID,SpaceType,PropAddress,PropCity,PropState,PropCountry,PropZip,NoofRooms,GuestIncluded,Extrapeoplerate,Kitchen,Petfriendly) VALUES (406,306,'Apartment','27th St','San Francisco','CA','USA',94110,1,2,35,'N','N');

BOOKING_ORDER Data Loading

INSERT INTO BOOKING_ORDER

(OrderID,GuestID,PropertyID,BookingDate,DateofArrival,Noofnights,ExtraGuests) VALUES (501,201,401,'2018-12-01','2018-12-15',3,1);

INSERT INTO BOOKING_ORDER

(OrderID,GuestID,PropertyID,BookingDate,DateofArrival,Noofnights,ExtraGuests) VALUES (502,203,402,'2019-05-05','2019-06-28',2,1);

INSERT INTO BOOKING_ORDER

(OrderID,GuestID,PropertyID,BookingDate,DateofArrival,Noofnights,ExtraGuests) VALUES (503,204,402,'2019-02-02','2019-05-16',5,2);

INSERT INTO BOOKING_ORDER

(OrderID,GuestID,PropertyID,BookingDate,DateofArrival,Noofnights,ExtraGuests) VALUES (504,205,403,'2018-11-15','2018-12-24',10,0);

INSERT INTO BOOKING_ORDER

(OrderID,GuestID,PropertyID,BookingDate,DateofArrival,Noofnights,ExtraGuests) VALUES (505,208,404,'2019-04-03','2019-08-01',2,0);

REVIEW Data Loading

INSERT INTO REVIEW (ReviewID,PropertyID,GuestID,ReviewDate,Rating,Review,Recommendation) VALUES (601,401,201,'2019-01-02',3,'stay was okish','N');

INSERT INTO REVIEW (ReviewID,PropertyID,GuestID,ReviewDate,Rating,Review,Recommendation) VALUES (602,401,202,'2019-01-06',4,'Fantastic and highly recommend!!','Y');

INSERT INTO REVIEW (ReviewID,PropertyID,GuestID,ReviewDate,Rating,Review,Recommendation) VALUES (603,402,203,'2019-07-04',3,'Clean and courteous','Y');

```

INSERT INTO REVIEW (ReviewID,PropertyID,GuestID,ReviewDate,Rating,Review,Recommendation) VALUES
(604,402,204,'2019-07-07',3,'Kevin is a very hospitable host. ','Y');
INSERT INTO REVIEW (ReviewID,PropertyID,GuestID,ReviewDate,Rating,Review,Recommendation) VALUES
(605,403,205,'2019-01-01',5,'amazing place to stay! ','Y');
INSERT INTO REVIEW (ReviewID,PropertyID,GuestID,ReviewDate,Rating,Review,Recommendation) VALUES
(606,403,206,'2019-01-07',1,'Bugs and not clean!','N');
INSERT INTO REVIEW (ReviewID,PropertyID,GuestID,ReviewDate,Rating,Review,Recommendation) VALUES
(607,403,207,'2019-01-10',5,'Nice room, great hosts.','Y');
INSERT INTO REVIEW (ReviewID,PropertyID,GuestID,ReviewDate,Rating,Review,Recommendation) VALUES
(608,404,208,'2019-08-18',2,'Bad experience','N');

```

CALENDAR SLOT Data Loading

```

INSERT INTO CALENDAR_SLOT
(SlotID,PropertyID,DateCalendar,Available,Price,Adjprice,Minnights,Maxnights) VALUES (701,401,'2019-01-
01','Y',135.00,130.00,1,30);
INSERT INTO CALENDAR_SLOT
(SlotID,PropertyID,DateCalendar,Available,Price,Adjprice,Minnights,Maxnights) VALUES (702,401,'2019-03-
01','Y',148.00,148.00,1,30);
INSERT INTO CALENDAR_SLOT
(SlotID,PropertyID,DateCalendar,Available,Price,Adjprice,Minnights,Maxnights) VALUES (703,402,'2019-07-
01','Y',48.00,40.00,1,10);
INSERT INTO CALENDAR_SLOT
(SlotID,PropertyID,DateCalendar,Available,Price,Adjprice,Minnights,Maxnights) VALUES (704,402,'2019-08-
01','Y',77.00,77.00,1,10);
INSERT INTO CALENDAR_SLOT
(SlotID,PropertyID,DateCalendar,Available,Price,Adjprice,Minnights,Maxnights) VALUES (705,403,'2019-01-
01','Y',129.00,120.00,1,14);
INSERT INTO CALENDAR_SLOT
(SlotID,PropertyID,DateCalendar,Available,Price,Adjprice,Minnights,Maxnights) VALUES (706,403,'2019-08-
01','Y',196.00,196.00,1,14);
INSERT INTO CALENDAR_SLOT
(SlotID,PropertyID,DateCalendar,Available,Price,Adjprice,Minnights,Maxnights) VALUES (707,403,'2019-12-
04','Y',129.00,129.00,1,14);
INSERT INTO CALENDAR_SLOT
(SlotID,PropertyID,DateCalendar,Available,Price,Adjprice,Minnights,Maxnights) VALUES (708,404,'2019-12-
04','Y',194.00,194.00,30,1125);

```

BED Data Loading

```

INSERT INTO BED (BedID,PropertyID,BedType) VALUES (801,401,'King');
INSERT INTO BED (BedID,PropertyID,BedType) VALUES (802,402,'Twin');
INSERT INTO BED (BedID,PropertyID,BedType) VALUES (803,402,'Queen');
INSERT INTO BED (BedID,PropertyID,BedType) VALUES (804,403,'King');

```

```
INSERT INTO BED (BedID,PropertyID,BedType) VALUES (805,404,'King');
INSERT INTO BED (BedID,PropertyID,BedType) VALUES (806,404,'Queen');
INSERT INTO BED (BedID,PropertyID,BedType) VALUES (807,404,'Queen');
INSERT INTO BED (BedID,PropertyID,BedType) VALUES (808,405,'Twin');
INSERT INTO BED (BedID,PropertyID,BedType) VALUES (809,406,'King');
```

CREDIT_COUPON Data Loading

```
INSERT INTO CREDIT_COUPON (Ccid,GuestID,Credit_couponval,Credit_Coupon_Type) VALUES
(901,201,50,'Credit');
INSERT INTO CREDIT_COUPON (Ccid,GuestID,Credit_couponval,Credit_Coupon_Type) VALUES
(902,202,15,'Coupon');
INSERT INTO CREDIT_COUPON (Ccid,GuestID,Credit_couponval,Credit_Coupon_Type) VALUES
(903,203,15,'Coupon');
INSERT INTO CREDIT_COUPON (Ccid,GuestID,Credit_couponval,Credit_Coupon_Type) VALUES
(904,204,10,'Coupon');
INSERT INTO CREDIT_COUPON (Ccid,GuestID,Credit_couponval,Credit_Coupon_Type) VALUES
(905,205,5.35,'Credit');
INSERT INTO CREDIT_COUPON (Ccid,GuestID,Credit_couponval,Credit_Coupon_Type) VALUES
(906,205,20,'Coupon');
INSERT INTO CREDIT_COUPON (Ccid,GuestID,Credit_couponval,Credit_Coupon_Type) VALUES
(907,206,0,NULL);
INSERT INTO CREDIT_COUPON (Ccid,GuestID,Credit_couponval,Credit_Coupon_Type) VALUES
(908,207,30,'Coupon');
INSERT INTO CREDIT_COUPON (Ccid,GuestID,Credit_couponval,Credit_Coupon_Type) VALUES
(909,208,135.5,'Credit');
INSERT INTO CREDIT_COUPON (Ccid,GuestID,Credit_couponval,Credit_Coupon_Type) VALUES
(910,208,15,'Coupon');
INSERT INTO CREDIT_COUPON (Ccid,GuestID,Credit_couponval,Credit_Coupon_Type) VALUES
(911,209,100,'Credit');
INSERT INTO CREDIT_COUPON (Ccid,GuestID,Credit_couponval,Credit_Coupon_Type) VALUES
(912,210,0,NULL);
```