Mahboobeh Hosseini
ProjectMilestone2
12/09/2020
IST 659

The Airbnb Database Project

Project Description

Airbnb is a great opportunity for guests all over the world to find accommodation that fits the best with their needs. Also buying and managing an Airbnb investment property is very popular as it generates a great profit. However, the Airbnb investors should consider some factors in order to choose the right property to buy for investing in short- term rentals.

The purpose of this project is to create a database that holds some Airbnb listings data with their real estate data in Washington State which keeps track of the factors that needs to be considered in order to buy an Airbnb investment property in Washington. It will help the investors to know how location, property and amenity types have correlation with Reviews quality and the rental income in Airbnb listings. Also, investors can consider what kinds of regulation they need to follow in the locations which they prefer to buy properties and how the properties price range would have a role in their decision according to their budget and expenses.

First of all, the stakeholders and some Business Rules for the project are introduced followed with a glossary which can help the reader to have better understanding of the words and their roles in our data and the conclusion. Then the conceptual and logical diagram of the project data are illustrated along with the five main questions which this project is going to focus on answering.

Stakeholders Description

The Stakeholders in Airbnb Database project are the Airbnb property investors, the Airbnb hosts and their business partners, Airbnb Company stakeholders and its employees and real estate agents.

This database will help the Airbnb property investors look for the right properties with the convenient location and property types and amenities that gives them the most profit in their investment considering their expenses and budget. The hosts who can be the property investors themselves with any other business partners are Stakeholders as their profit from their short-term rentals goes higher because the right property is selected for this matter.

Also this database will encourage more people to think of investing on Airbnb properties as it makes the process easier to find a property and it will benefit the real agents because more houses will be sold. In the other hand, there will be more Airbnb listings available which might get good numbers of visits and this increase the profit for Airbnb company stakeholders and finally will benefit the Airbnb employees since there will be a chance of raise in their salary and also having a better job security.

Project Glossary

A **User** is a person who either lists properties on Airbnb or uses Airbnb to book a visit

An **Airbnb Listing** includes a property listed by a user to be available to get booked by another user

A **Location** contains the address of a property which has one or more properties located in it.

A **Location address** contains the city and zip code.

A **visit** is booked by a user and it contains the visit date and check out date and number of guests.

Occupancy regulations are rules that an Airbnb property should follow for short-term rentals in a specific location. Occupancy regulation contains Occupancy Regulation Type.

An Occupancy Regulation Type can be rules such as restricted numbers of nights permitted to rent out annually and city taxes

A **property** is a place which belongs to a location and is listed by a user in an Airbnb listing in order for another user to book a visit to it. The property contains the property type, the property real estate price, nightly occupancy rate, and the capacity.

A **Property type** is either Entire place, Private room or shared room.

Rental expenses includes expense type and expense amount.

An **expense type** is either cleaning service fees, monthly utility, or monthly insurance for the Rental expense.

Capacity means how many people can fit in the property.

An **amenity** is an element of comfort and convenience that is belonged to a property. Amenity contains amenity type.

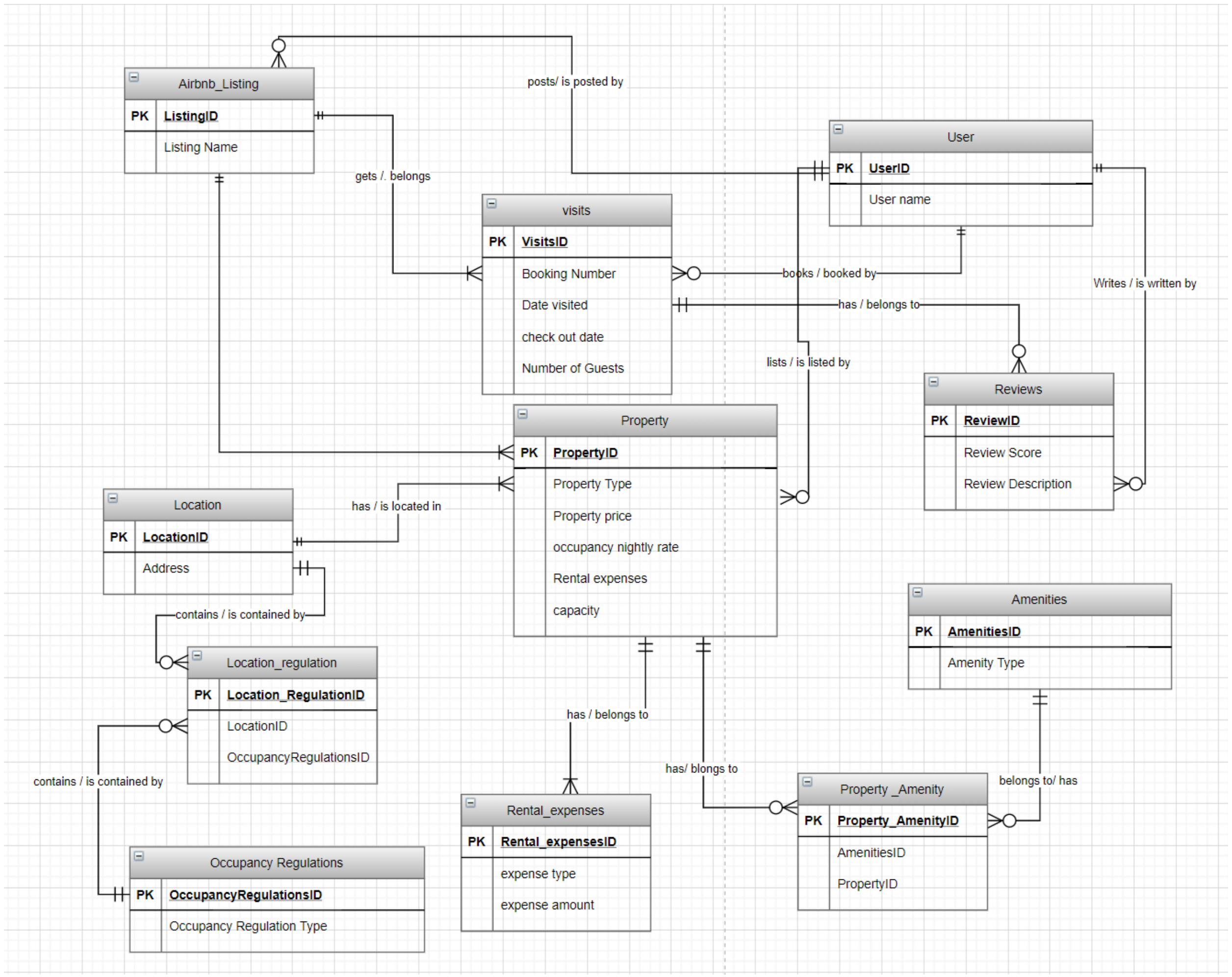
An **Amenity type** can be things such as hot tub, pool, gym, waterfront, ocean front and etc.

A **review** is a feedback that is written and posted by a user and it contains the review score and its optional description text.

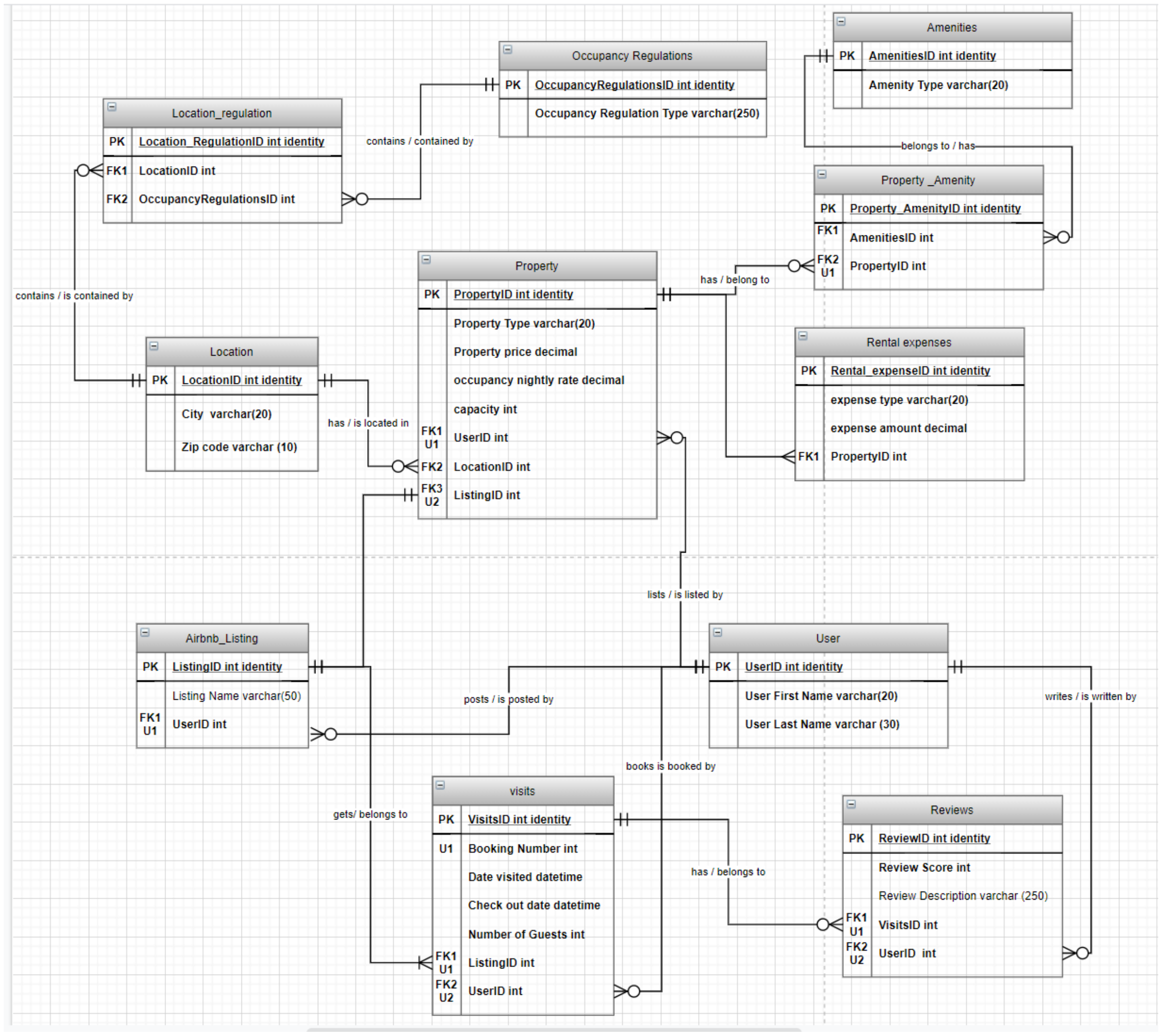
Project Business Rules

- A user can be a person of any age
- An Airbnb Listing is posted by only one user
- A user posts zero or more Airbnb Listing
- A Location has one or more properties in it
- A review is written by a user
- A visit has zero or more reviews
- A listing has one or more visits
- A user either books a visit or lists a property
- A visit is booked by a user
- An Amenity belongs to a property.
- A property is listed by a user
- A property should contain one or more types of rental expenses
- An occupancy regulations belongs to a specific location
- A location can have zero or more occupancy regulation
- A review is written for a visit

Conceptual Data Model



Logical Data Model

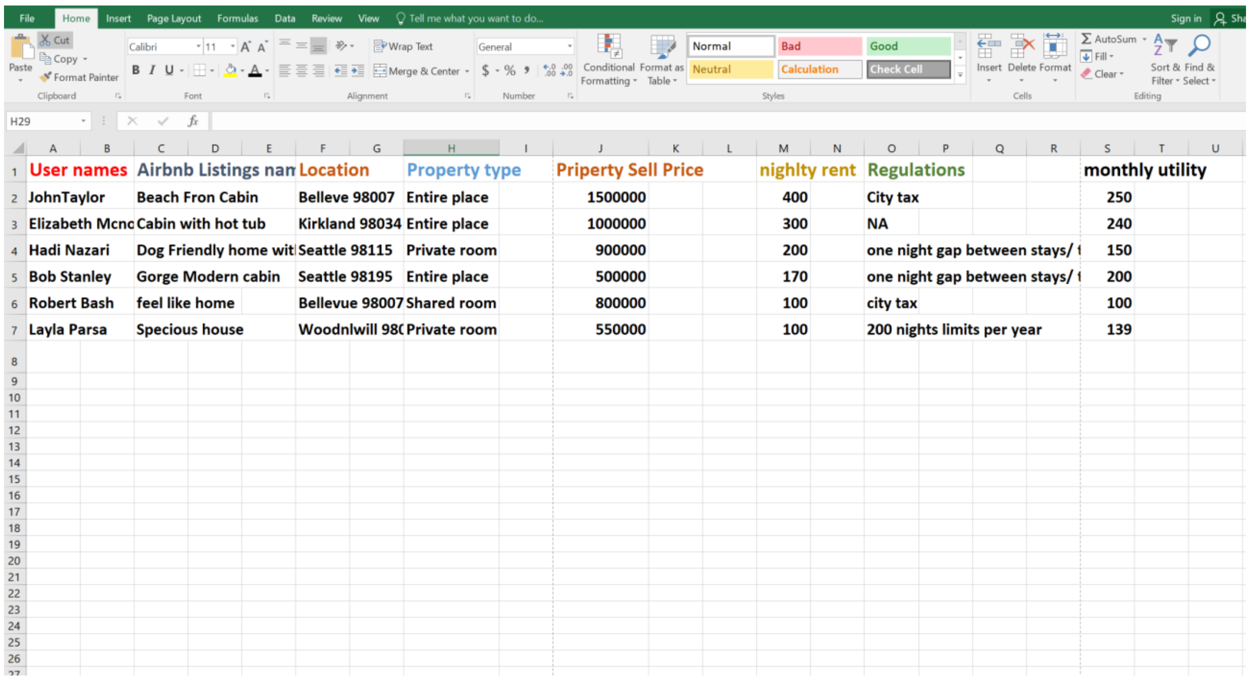


Data Questions

1. What is the average visits in each unique location?
2. What are the top 2 locations with the least number of occupancy regulation rules?
3. What is the average property price range in each unique location?
4. What is the average occupancy nightly rate for each unique city?
5. What is the top 2 highest Occupancy nightly rate by city?
6. what is the average rental expenses in each property with Certain property type ?
7. What are the top 5 amenities that have the most frequent visits/best reviews?
8. What is the top property type most visited in each unique location?
9. What are the top five locations with the most rental income after deducting the rental expenses from it?
10. What is the average capacity for the top 20% properties that had the most visits ?

Raw Data Sample

There have been raw data of Airbnb Listings and their info stored on an excel .



User names	Airbnb Listings nan	Location	Property type	Priperty Sell Price	nighlty rent	Regulations	monthly utility
JohnTaylor	Beach Fron Cabin	Belleve 98007	Entire place	1500000	400	City tax	250
Elizabeth Mcnc	Cabin with hot tub	Kirkland 98034	Entire place	1000000	300	NA	240
Hadi Nazari	Dog Friendly home wit	Seattle 98115	Private room	900000	200	one night gap between stays/ t	150
Bob Stanley	Gorge Modern cabin	Seattle 98195	Entire place	500000	170	one night gap between stays/ t	200
Robert Bash	feel like home	Bellevue 98007	Shared room	800000	100	city tax	100
Layla Parsa	Specious house	Woodnlwill 98(	Private room	550000	100	200 nights limits per year	139

User names	Airbnb Listings names	Location	Property type
JohnTaylor	Beach Fron Cabin	Belleve 98007	Entire place
Elizabeth Mcnon	Cabin with hot tub	Kirkland 98034	Entire place
Hadi Nazari	Dog Friendly home with stunning views	Seattle 98115	Private room
Bob Stanley	Gorge Modern cabin	Seattle 98195	Entire place
Robert Bash	feel like home	Bellevue 98007	Shared room
Layla Parsa	Specious house	Woodnlwill 98034	Private room

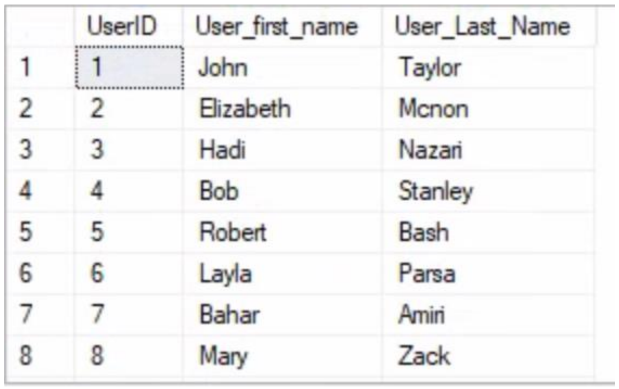
Physical Database

```sql
/*
Author   : Mahboobeh Hosseini
Course   : IST 659 M403
Term     : November 19, 2020
*/

--Drop Procedures
 If OBJECT_ID('dbo.ChangeUserFirstName') IS NOT NULL
      DROP PROCEDURE dbo.ChangeUserFirstName

If OBJECT_ID('dbo.add_visits') IS NOT NULL
      DROP PROCEDURE dbo.add_visits

--Drop Views
If OBJECT_ID('dbo.RegulationForLocation') IS NOT NULL
      DROP VIEW dbo.RegulationForLocation

If OBJECT_ID('dbo.AirbnbListingHost') IS NOT NULL
      DROP VIEW dbo.AirbnbListingHost

If OBJECT_ID('dbo.PropertyAmenities') IS NOT NULL
      DROP VIEW dbo.PropertyAmenities

If OBJECT_ID('dbo.PropertyLocations') IS NOT NULL
      DROP VIEW dbo.PropertyLocations

-- drop all tables in reverse order of their dependencies
IF OBJECT_ID('dbo.Location_Regulation', 'U') IS NOT NULL
            DROP TABLE dbo.Location_Regulation;
go

IF OBJECT_ID('dbo.Occupancy_Regulations', 'U') IS NOT NULL
            DROP TABLE dbo. Occupancy_Regulations;
go

IF OBJECT_ID('dbo.Rental_Expenses', 'U') IS NOT NULL
            DROP TABLE dbo.Rental_Expenses;
go

IF OBJECT_ID('dbo.Property_Amenity', 'U') IS NOT NULL
            DROP TABLE dbo.Property_Amenity;
go

IF OBJECT_ID('dbo.Amenities', 'U') IS NOT NULL
            DROP TABLE dbo.Amenities;
Go
```

```
IF OBJECT_ID('dbo.Reviews', 'U') IS NOT NULL
            DROP TABLE dbo.Reviews;
Go

IF OBJECT_ID('dbo.Visits', 'U') IS NOT NULL
            DROP TABLE dbo.Visits;
Go

IF OBJECT_ID('dbo.Property', 'U') IS NOT NULL
            DROP TABLE dbo.Property;
Go

IF OBJECT_ID('dbo.Locations', 'U') IS NOT NULL
            DROP TABLE dbo.Locations;
Go

IF OBJECT_ID('dbo.Airbnb_Listing', 'U') IS NOT NULL
            DROP TABLE dbo.Airbnb_Listing;
Go

IF OBJECT_ID('dbo.Users', 'U') IS NOT NULL
            DROP TABLE dbo.Users;
Go



-- create all tables in order of their dependencies

-- Creating the User
CREATE TABLE Users (
      -- Columns for the User table
      UserID  int identity
    ,User_first_name varchar(20)  Not Null
      ,User_Last_Name varchar(30) Not Null
      CONSTRAINT PK_Users PRIMARY KEY(UserID)
)
      -- End Creating the User Table
GO


-- Creating the Airbnb Listing
CREATE TABLE Airbnb_Listing (
      -- Columns for the Airbnb Listing table
      ListingID  int identity
      ,Listing_name varchar (50)
      ,UserID int Not Null
      -- Constraints on the Airbnb Listing Table
CONSTRAINT FK1_Airbnb_Listing FOREIGN KEY (UserID) REFERENCES Users(UserID)
CONSTRAINT PK_Airbnb_Listing PRIMARY KEY(ListingID)
)
```

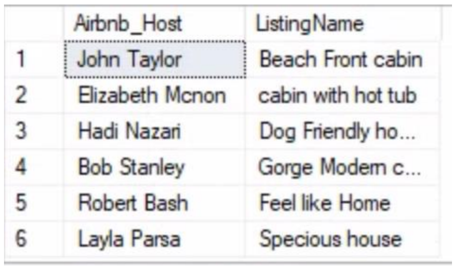
```
	-- End Creating the Airbnb Listing
GO


-- Creating the Location
CREATE TABLE Locations (
	-- Columns for the Location
	LocationID  int identity
	,City varchar(20)    NOT NULL
	,ZipCode varchar(10)   NOT NULL,
	CONSTRAINT PK_Locations PRIMARY KEY(LocationID)
)
	-- End Creating the Location
GO


-- Creating the Property
CREATE TABLE Property (
	-- Columns for the Property table
	PropertyID  int identity
	, Property_Type varchar (20)   NOT NULL
	, Property_Price decimal   NOT NULL
	, Occupancy_Nightly_Rate decimal   NOT NULL
	, Capacity int     NOT NULL
	, UserID int Not Null
	, LocationID int Not Null
	, ListingID int Not Null
	-- Constraints on the Property Table
CONSTRAINT FK1_Property FOREIGN KEY (UserID) REFERENCES Users(UserID)
,CONSTRAINT FK2_Property FOREIGN KEY (LocationID) REFERENCES
Locations(LocationID)
,CONSTRAINT FK3_Propert FOREIGN KEY (ListingID) REFERENCES
Airbnb_Listing(ListingID)
,CONSTRAINT PK_Property PRIMARY KEY(PropertyID)
)
	-- End Creating the Property
GO


-- Creating the Visits table
CREATE TABLE Visits (
	-- Columns for the Visits table
	VisitsID  int identity
	, UserID int Not Null
	, ListingID int Not Null
	-- Constraints on the Visits Table
CONSTRAINT FK1_Visits FOREIGN KEY (UserID) REFERENCES Users(UserID)
,CONSTRAINT FK2_Visits FOREIGN KEY (ListingID) REFERENCES
Airbnb_Listing(ListingID)
,CONSTRAINT PK_Visits PRIMARY KEY(VisitsID)
)
      -- End Creating the Visits

GO
-- Creating the Reviews
CREATE TABLE Reviews (
      -- Columns for the Reviews table
      ReviewsID  int identity
      , Review_Score int Not Null
      , Review_Description varchar(250)
      , UserID int Not Null
      , VisitsID int Not Null
      -- Constraints on the Reviews Table
CONSTRAINT FK1_Reviews FOREIGN KEY (UserID) REFERENCES Users(UserID)
, CONSTRAINT FK2_Reviews FOREIGN KEY (VisitsID) REFERENCES Visits(VisitsID)
,CONSTRAINT PK_Reviews PRIMARY KEY(ReviewsID)
)
      -- End Creating the Reviews

GO


-- Creating the Amenities
CREATE TABLE Amenities (
      -- Columns for the Amenities table
      AmenityID  int identity
      , Amenities_Type varchar(20)  Not Null
      , CONSTRAINT PK_Amenities PRIMARY KEY(AmenityID)
)
      -- End Creating the Amenities
GO


-- Creating the Property Amenity
CREATE TABLE Property_Amenity (
      -- Columns for the Property_Amenity table
      Property_AmenityID  int identity
      , AmenityID int Not Null
      , PropertyID int Not Null
      -- Constraints on the Property Amenity Table
CONSTRAINT FK1_Property_Amenity FOREIGN KEY (AmenityID) REFERENCES
Amenities(AmenityID)
,CONSTRAINT FK2_Property_Amenity FOREIGN KEY (PropertyID) REFERENCES
Property(PropertyID)
,CONSTRAINT PK_Property_Amenity PRIMARY KEY(Property_AmenityID)
)
      -- End Creating the Property Amenity

GO
```

```
-- Creating the Rental Expenses
CREATE TABLE Rental_Expenses (
	-- Columns for the Rental_Expenses table
	Rental_ExpensesID int identity
	, Expense_Type varchar(50)  Not Null
	, Expense_Amount decimal    Not Null
	, PropertyID int Not Null
	-- Constraints on the Rental_Expenses Table
	CONSTRAINT FK1_Rental_Expenses FOREIGN KEY (PropertyID) REFERENCES
Property(PropertyID)
	, CONSTRAINT PK_Rental_Expenses PRIMARY KEY(Rental_ExpensesID)
)
	-- End Creating the Rental Expenses
GO


-- Creating the Occupancy Regulations
CREATE TABLE Occupancy_Regulations (
	-- Columns for the Occupancy Regulations table
	Occupancy_RegulationsID  int identity
	, Occupancy_Regulations_Type varchar(250) Not Null
	,CONSTRAINT PK_Occupancy_Regulations PRIMARY
KEY(Occupancy_RegulationsID)
)
	-- End Creating the Occupancy Regulations
GO



-- Creating the Location Regulation
CREATE TABLE Location_Regulation (
	-- Columns for the Location Regulation table
	Location_RegulationID  int identity
	, LocationID int Not Null
	, Occupancy_RegulationsID int Not Null
-- Constraints on the Location Regulation Table
CONSTRAINT FK1_Location_Regulation FOREIGN KEY (LocationID) REFERENCES
Locations(LocationID)
,CONSTRAINT FK2_Location_Regulation FOREIGN KEY (Occupancy_RegulationsID)
REFERENCES Occupancy_Regulations(Occupancy_RegulationsID)
,CONSTRAINT PK_Location_Regulation PRIMARY KEY(Location_RegulationID)

)
-- End Creating the Location Regulation
-- insert customers' first and last names into the table User
      INSERT INTO  Users(User_first_name, User_Last_Name)
VALUES ('John', 'Taylor'),
         ('Elizabeth', 'Mcnon'),
         ('Hadi', 'Nazari'),
         ('Bob', 'Stanley'),
         ('Robert', 'Bash'),
         ('Layla', 'Parsa'),
         ('Bahar', 'Amiri'),
         ('Mary', 'Zack'),
         ('Harry', 'Neve'),
         ('Jimmie', 'James'),
         ('Rana', 'Naseri'),
         ('Sara','Rashidi')


      -- Look up into the inserts
      SELECT * FROM Users
```



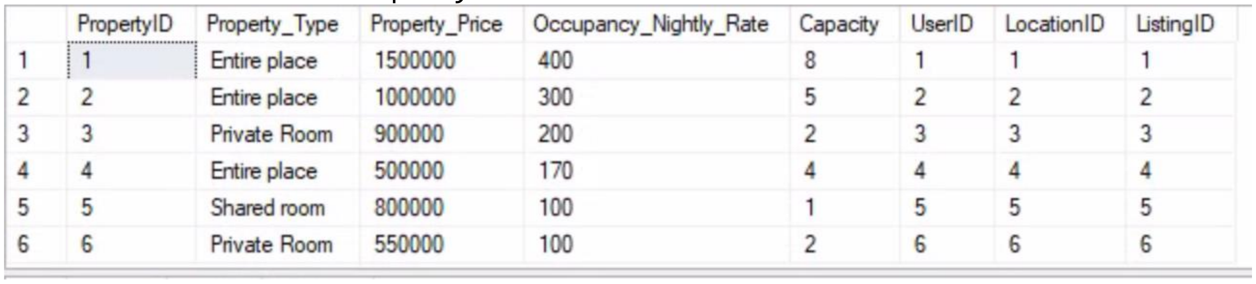
	UserID	User_first_name	User_Last_Name
1	1	John	Taylor
2	2	Elizabeth	Mcnon
3	3	Hadi	Nazari
4	4	Bob	Stanley
5	5	Robert	Bash
6	6	Layla	Parsa
7	7	Bahar	Amiri
8	8	Mary	Zack

```
      -- insert customers' first and last names into the table Airbnb Listing
      INSERT INTO  Airbnb_Listing(Listing_name, UserID)
VALUES ('Beach Front cabin', (SELECT UserID FROM Users WHERE UserID = '1' )),
         ('cabin with hot tub', (SELECT UserID FROM Users WHERE UserID = '2'
)),
         ('Dog Friendly home with stunning views', (SELECT UserID FROM Users
WHERE UserID = '3' )),
         ('Gorge Modern cabin', (SELECT UserID FROM Users WHERE UserID = '4'
)),
         ('Feel like Home', (SELECT UserID FROM Users WHERE UserID = '5' )),
         ('Specious house',(SELECT UserID FROM Users WHERE UserID = '6' ))

      -- Create a view to see th users and their Airbnb Listings
      GO
      CREATE VIEW AirbnbListingHost
      AS
      SELECT
                  Users.User_first_name + ' ' + Users.User_Last_Name as
                  Airbnb_Host,
                  Airbnb_Listing.Listing_name as ListingName
                  FROM Airbnb_Listing
                  Inner JOIN Users ON Airbnb_Listing.UserID = Users.UserID
                  GO
                  SELECT * FROM AirbnbListingHost
```



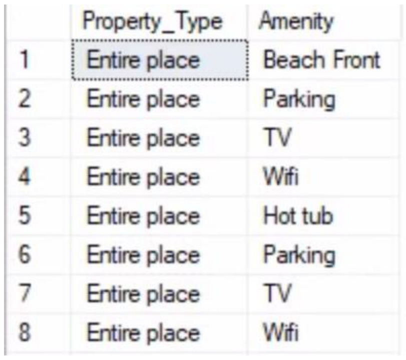
	Airbnb_Host	ListingName
1	John Taylor	Beach Front cabin
2	Elizabeth Mcnon	cabin with hot tub
3	Hadi Nazari	Dog Friendly ho...
4	Bob Stanley	Gorge Modem c...
5	Robert Bash	Feel like Home
6	Layla Parsa	Specious house

```
            -- insert city and zip code for all listings into the table Location
            INSERT INTO  Locations(City, ZipCode)
VALUES ('Bellevue', '98007'),
         ('Kirkland', '98034'),
         ('Seattle', '98115'),
         ('Seattle', '98195'),
         ('Bellevue', '98007'),
         ('Woodnlwill','98034')

             --Lets check our inserts for Locations
             SELECT * FROM Locations

             -- insert city and zip code for all listings into the table Location
            INSERT INTO  Property(Property_Type, Property_Price,
Occupancy_Nightly_Rate, Capacity, UserID, LocationID, ListingID)
VALUES ('Entire place', '1500000', '400' ,'8' , (SELECT UserID FROM Users
WHERE UserID = '1' ),
            (SELECT LocationID FROM Locations WHERE LocationID = '1' ),
(SELECT ListingID FROM Airbnb_Listing WHERE ListingID = '1' ) ),
            ('Entire place', '1000000', '300' ,'5' , (SELECT UserID FROM
Users WHERE UserID = '2' ),
            (SELECT LocationID FROM Locations WHERE LocationID = '2' ),
(SELECT ListingID FROM Airbnb_Listing WHERE ListingID = '2' ) ),
            ('Private Room', '900000', '200' ,'2' , (SELECT UserID FROM Users
WHERE UserID = '3' ),
            (SELECT LocationID FROM Locations WHERE LocationID = '3' ),
(SELECT ListingID FROM Airbnb_Listing WHERE ListingID = '3' ) ),
          ('Entire place', '500000', '170' ,'4' , (SELECT UserID FROM Users
WHERE UserID = '4' ),
            (SELECT LocationID FROM Locations WHERE LocationID = '4' ),
(SELECT ListingID FROM Airbnb_Listing WHERE ListingID = '4' ) ),
         ('Shared room ', '800000', '100' ,'1' , (SELECT UserID FROM Users
WHERE UserID = '5' ),
            (SELECT LocationID FROM Locations WHERE LocationID = '5' ),
(SELECT ListingID FROM Airbnb_Listing WHERE ListingID = '5' ) ),
         ('Private Room', '550000', '100' ,'2' , (SELECT UserID FROM Users
WHERE UserID = '6' ),
            (SELECT LocationID FROM Locations WHERE LocationID = '6' ),
(SELECT ListingID FROM Airbnb_Listing WHERE ListingID = '6' ) )


--Lets check our inserts for Locations
       SELECT * FROM Property
```



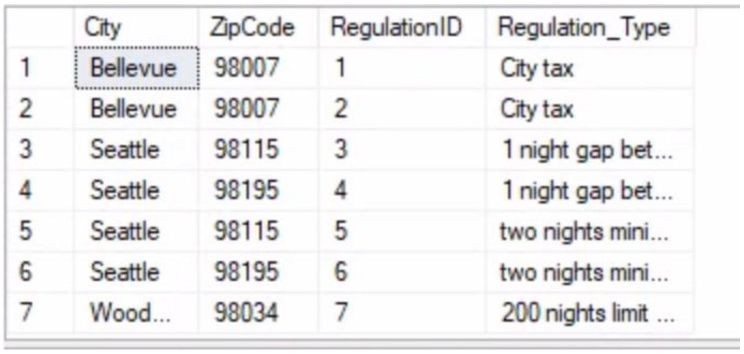
	PropertyID	Property_Type	Property_Price	Occupancy_Nightly_Rate	Capacity	UserID	LocationID	ListingID
1	1	Entire place	1500000	400	8	1	1	1
2	2	Entire place	1000000	300	5	2	2	2
3	3	Private Room	900000	200	2	3	3	3
4	4	Entire place	500000	170	4	4	4	4
5	5	Shared room	800000	100	1	5	5	5
6	6	Private Room	550000	100	2	6	6	6

```
       -- insert UserID and ListingID for all visits into the table Visits
      INSERT INTO  Visits( UserID, ListingID)
VALUES ((SELECT UserID FROM Users WHERE UserID = '7' ), (SELECT ListingID
FROM Airbnb_Listing WHERE ListingID = '1' )),
             ((SELECT UserID FROM Users WHERE UserID = '8' ), (SELECT
ListingID FROM Airbnb_Listing WHERE ListingID = '2' )),
              ((SELECT UserID FROM Users WHERE UserID = '9' ), (SELECT
ListingID FROM Airbnb_Listing WHERE ListingID = '3' )),
             ((SELECT UserID FROM Users WHERE UserID = '10' ), (SELECT
ListingID FROM Airbnb_Listing WHERE ListingID = '4' )),
              ((SELECT UserID FROM Users WHERE UserID = '11' ), (SELECT
ListingID FROM Airbnb_Listing WHERE ListingID = '5' )),
             ((SELECT UserID FROM Users WHERE UserID = '12' ), (SELECT
ListingID FROM Airbnb_Listing WHERE ListingID = '6' ))


       --Lets check our inserts for Visits
       SELECT * FROM Visits

       -- insert Review Score and possible Review Description matching User
ID and Visits ID into the table Reviews
       INSERT INTO  Reviews( Review_Score, Review_Description, UserID,
VisitsID)
VALUES (5,NULL, (SELECT UserID FROM Users WHERE UserID = '7' ), (SELECT
VisitsID FROM Visits WHERE VisitsID = '1' )),
             (4.5,NULL,  (SELECT UserID FROM Users WHERE UserID = '8' ),
(SELECT VisitsID FROM Visits WHERE VisitsID = '2' )),
              (3.75, 'Need better cleaning service', (SELECT UserID FROM
Users WHERE UserID = '9' ),
              (SELECT VisitsID FROM Visits WHERE VisitsID = '3' )),
             (4.4, NULL, (SELECT UserID FROM Users WHERE UserID = '10' ),
(SELECT VisitsID FROM Visits WHERE VisitsID = '4' )),
              (4.8, 'Excellent experience', (SELECT UserID FROM Users WHERE
UserID = '11' ), (SELECT VisitsID FROM Visits WHERE VisitsID = '5' )),
             (4.2, NULL, (SELECT UserID FROM Users WHERE UserID = '12' ),
(SELECT VisitsID FROM Visits WHERE VisitsID = '6' ))

      -- Check the inserts for Reviews table
             SELECT * FROM Reviews

         -- insertAmenity type for table Amenities
        INSERT INTO  Amenities( Amenities_Type)
VALUES ('Beach Front'),
            ('Hot tub'),
            ('Parking'),
            ('AirConditioning'),
            ('TV'),
            ('Wifi')

       -- Check the insert for Amenities
          SELECT * FROM Amenities


        -- insert Amenity ID and Property ID  into the table Property_Amenity
             INSERT INTO  Property_Amenity( AmenityID, PropertyID)
VALUES ((SELECT AmenityID FROM Amenities WHERE AmenityID = '1' ), (SELECT
PropertyID FROM Property WHERE PropertyID = '1' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '3' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '1' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '5' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '1' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '6' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '1' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '2' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '2' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '3' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '2' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '5' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '2' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '6' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '2' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '3' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '3' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '5' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '3' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '6' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '3' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '3' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '4' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '5' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '4' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '6' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '4' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '4' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '4' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '3' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '5' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '5' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '5' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '6' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '5' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '3' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '6' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '4' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '6' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '5' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '6' )),
            ((SELECT AmenityID FROM Amenities WHERE AmenityID = '6' ),
(SELECT PropertyID FROM Property WHERE PropertyID = '6' ))



--Check the inserts for Property_Amenity Table
          SELECT * FROM Property_Amenity
      GO


      -- Create a view of Properties and its Amenities
      CREATE VIEW PropertyAmenities
      AS
      SELECT
                  Property.Property_Type ,
                  Amenities.Amenities_Type as Amenity

                  FROM Property_Amenity
                  FULL OUTER JOIN Amenities ON Amenities.AmenityID =
Property_Amenity.AmenityID
                  inner JOIN Property ON Property.PropertyID =
Property_Amenity.PropertyID
      GO

      -- Check the output for PropertyAmenities view
            SELECT * FROM PropertyAmenities
      GO

      -- Create a view of Property type and its location
      CREATE VIEW PropertyLocations
      AS
      SELECT
                  Property.Property_Type,
                  Locations.City as City,
                  Locations.ZipCode

            FROM Locations
                  inner JOIN Property ON Property.LocationID =
Locations.LocationID
            GO

                  -- Check the output for PropertyLocations view
                  SELECT * FROM PropertyLocations
            GO
```



	Property_Type	Amenity
1	Entire place	Beach Front
2	Entire place	Parking
3	Entire place	TV
4	Entire place	Wifi
5	Entire place	Hot tub
6	Entire place	Parking
7	Entire place	TV
8	Entire place	Wifi

```
       -- insert Expense Type, Expense amount and Property ID
       --for all listings into the table Rental Expenses
      INSERT INTO  Rental_Expenses( Expense_Type , Expense_Amount,
PropertyID)
VALUES ('cleaning service fees each visit ', 120, (SELECT PropertyID FROM
Property WHERE PropertyID = '1' )),
         ('cleaning service fees each visit ', 120, (SELECT PropertyID FROM
Property WHERE PropertyID = '2' )),
         ('cleaning service fees each visit ', 30, (SELECT PropertyID FROM
Property WHERE PropertyID = '3' )),
         ('cleaning service fees each visit ', 95, (SELECT PropertyID FROM
Property WHERE PropertyID = '4' )),
         ('cleaning service fees each visit ', 20, (SELECT PropertyID FROM
Property WHERE PropertyID = '5' )),
         ('cleaning service fees each visit ', 25, (SELECT PropertyID FROM
Property WHERE PropertyID = '6' )),
          ('Monthly Utility ', 250, (SELECT PropertyID FROM Property WHERE
PropertyID = '1' )),
         ('Monthly Utility ', 240, (SELECT PropertyID FROM Property WHERE
PropertyID = '2' )),
         ('Monthly Utility' , 150, (SELECT PropertyID FROM Property WHERE
PropertyID = '3' )),
         ('Monthly Utility' , 200, (SELECT PropertyID FROM Property WHERE
PropertyID = '4' )),
         ('Monthly Utility' , 100, (SELECT PropertyID FROM Property WHERE
PropertyID = '5' )),
         ('Monthly Utility ' , 139, (SELECT PropertyID FROM Property WHERE
PropertyID = '6' )),
         ('Monthly Insurance' , 50, (SELECT PropertyID FROM Property WHERE
PropertyID = '1' )),
         ('Monthly Insurance' , 50, (SELECT PropertyID FROM Property WHERE
PropertyID = '2' )),
         ('Monthly Insurance' , 40, (SELECT PropertyID FROM Property WHERE
PropertyID = '3' )),
         ('Monthly Insurance' ,50, (SELECT PropertyID FROM Property WHERE
PropertyID = '4' )),
         ('Monthly Insurance ' ,20, (SELECT PropertyID FROM Property WHERE
PropertyID = '5' )),
         ('Monthly Insurance' ,30, (SELECT PropertyID FROM Property WHERE
PropertyID = '6' ))
      GO

       --Lets check our inserts for Rental expenses
       SELECT * FROM Rental_Expenses

      GO

       -- insert Occupancy Regulations Type  for Occupancy_Regulations table
      INSERT INTO  Occupancy_Regulations( Occupancy_Regulations_Type)
VALUES ('City tax'),
         ('200 nights limit per year '),
         ('two nights minimum '),
         ('1 night gap between each visit due to corona')

          --Lets check our inserts for Occupancy_Regulations table
         SELECT * FROM Occupancy_Regulations
      GO

 -- insert LocationID and Occupancy RegulationsID into Location Regulation
table
      INSERT INTO  Location_Regulation( LocationID,Occupancy_RegulationsID  )
VALUES ((SELECT LocationID FROM Locations WHERE LocationID = '1' ),
      (SELECT Occupancy_RegulationsID FROM Occupancy_Regulations WHERE
Occupancy_RegulationsID = '1' )),
      ((SELECT LocationID FROM Locations WHERE LocationID = '5' ),
      (SELECT Occupancy_RegulationsID FROM Occupancy_Regulations WHERE
Occupancy_RegulationsID = '1' )),
      ((SELECT LocationID FROM Locations WHERE LocationID = '3' ),
      (SELECT Occupancy_RegulationsID FROM Occupancy_Regulations WHERE
Occupancy_RegulationsID = '4' )),
      ((SELECT LocationID FROM Locations WHERE LocationID = '4' ),
      (SELECT Occupancy_RegulationsID FROM Occupancy_Regulations WHERE
Occupancy_RegulationsID = '4' )),
      ((SELECT LocationID FROM Locations WHERE LocationID = '3' ),
      (SELECT Occupancy_RegulationsID FROM Occupancy_Regulations WHERE
Occupancy_RegulationsID = '3' )),
      ((SELECT LocationID FROM Locations WHERE LocationID = '4' ),
      (SELECT Occupancy_RegulationsID FROM Occupancy_Regulations WHERE
Occupancy_RegulationsID = '3' )),
      ((SELECT LocationID FROM Locations WHERE LocationID = '6' ),
      (SELECT Occupancy_RegulationsID FROM Occupancy_Regulations WHERE
Occupancy_RegulationsID = '2' ))
GO
      --Lets check our inserts for Location_Regulation table
       SELECT * FROM Location_Regulation
GO



-- Create a view of Location citis and zipcodes and their regulations
      CREATE VIEW RegulationForLocation
      AS
      SELECT
                  Locations.city as City,
                  Locations.ZipCode as ZipCode,
                  Location_RegulationID as RegulationID,
      Occupancy_Regulations.Occupancy_Regulations_Type as Regulation_Type

            FROM Location_Regulation
                  inner JOIN Locations ON location_Regulation.LocationID =
Locations.LocationID

                  inner JOIN Occupancy_Regulations ON
location_Regulation.Occupancy_RegulationsID =
Occupancy_Regulations.Occupancy_RegulationsID

            GO
             -- Check the output for RegulationForLocation view
             SELECT * FROM RegulationForLocation
            GO
```

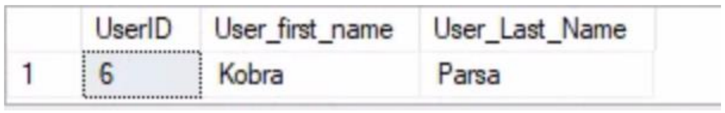


	City	ZipCode	RegulationID	Regulation_Type
1	Bellevue	98007	1	City tax
2	Bellevue	98007	2	City tax
3	Seattle	98115	3	1 night gap bet...
4	Seattle	98195	4	1 night gap bet...
5	Seattle	98115	5	two nights mini...
6	Seattle	98195	6	two nights mini...
7	Wood...	98034	7	200 nights limit ...

```
            --We forgot to add some of our visits
            --We need to Create a procedure so it lets us add more inserts
into the table visits
             CREATE PROCEDURE add_visits(@username varchar(20), @listingID
int)
      AS
      BEGIN
            -- We have the user LastName, but we need the ID for the visits
            -- First, declare a variable to hold the ID
            DECLARE @userID int

            --Get the UserID for the User LastName provided and store it in
@userID
            SELECT @userID = UserID FROM Users WHERE User_Last_Name =
@username

            -- Now we can add the row using an INSERT statement
            INSERT INTO Visits(UserID, ListingID)
            VALUES(@userID, @listingID)

            -- Now return the @@identity so the calling code knows where
            -- the data ended up
            RETURN @@identity

            END
            GO

--Declare a variable to store the new data in it and add it to Visits table
DECLARE @addedValue int
            EXEC @addedValue = add_visits 'Taylor' , '2'

            SELECT
                  Users.UserID
                  , Users.User_Last_Name
                  , Visits.ListingID
                  , visits.UserID
FROM Users
JOIN Visits on Users.UserID = Visits.UserID
WHERE VisitsID = @addedValue


--Declare a variable to store the new data in it and add it to Visits table
DECLARE @addedValue2 int
            EXEC @addedValue2 = add_visits 'Bash' , '2'

            SELECT
                  Users.UserID
                  , Users.User_Last_Name
                  , Visits.ListingID
                  , visits.UserID
```

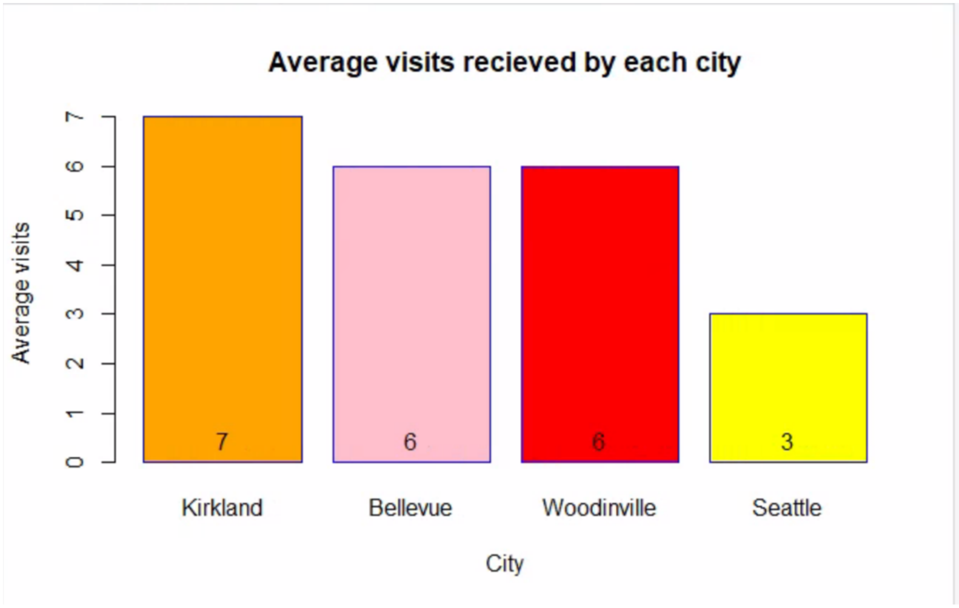
```
FROM Users
JOIN Visits on Users.UserID = Visits.UserID
WHERE VisitsID = @addedValue2


--Declare a variable to store the new data in it and add it to Visits table
DECLARE @addedValue3 int
            EXEC @addedValue3 = add_visits 'Mcnon' , '1'

            SELECT
                  Users.UserID
                  , Users.User_Last_Name
                  , Visits.ListingID
                  , visits.UserID
FROM Users
JOIN Visits on Users.UserID = Visits.UserID
WHERE VisitsID = @addedValue3



--Declare a variable to store the new data in it and add it to Visits table
DECLARE @addedValue4 int
            EXEC @addedValue4 = add_visits 'Stanley' , '5'

            SELECT
                  Users.UserID
                  , Users.User_Last_Name
                  , Visits.ListingID
                  , visits.UserID
FROM Users
JOIN Visits on Users.UserID = Visits.UserID
WHERE VisitsID = @addedValue4

--Check how the visits table  changed
SELECT * FROM visits
GO

--Declare a variable to store the new data in it and add it to Visits table
DECLARE @addedValue4 int
            EXEC @addedValue4 = add_visits 'Mcnon' , '2'

            SELECT
                  Users.UserID
                  , Users.User_Last_Name
                  , Visits.ListingID
                  , visits.UserID
FROM Users
JOIN Visits on Users.UserID = Visits.UserID
WHERE VisitsID = @addedValue4
GO
```

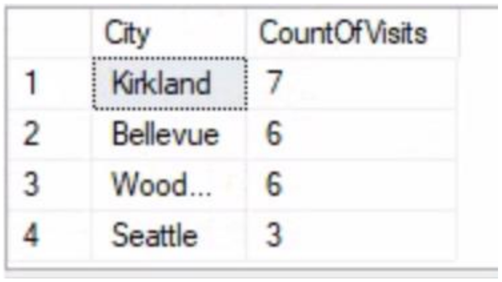
```
-- Create a procedure to update a Users First Name
-- the first parameter is the user ID for the user to change
-- the second is the new First Name
CREATE PROCEDURE ChangeUserFirstName(@userID int, @newFirstName varchar(50))
AS
BEGIN
      UPDATE Users SET User_first_name = @newFirstName
      WHERE UserID = @userID
      END
      GO
      EXEC ChangeUserFirstName'6' , 'Kobra'
      --Check the First name update
SELECT * FROM Users WHERE UserID = '6'
```



	UserID	User_first_name	User_Last_Name
1	6	Kobra	Parsa

```sql
--Data Question 1
--What are the numbers of visits for each location ?
SELECT
              Locations.City as City
             ,AVG(Visits.VisitsID) as  CountOfVisits
FROM Property

INNER JOIN Visits ON Property.ListingID = Visits.ListingID
INNER JOIN Airbnb_Listing ON Property.ListingID = Visits.ListingID
INNER JOIN Locations ON Locations.LocationID = Property.LocationID

group by
      Locations.City
ORDER BY
      CountOfVisits DESC
```

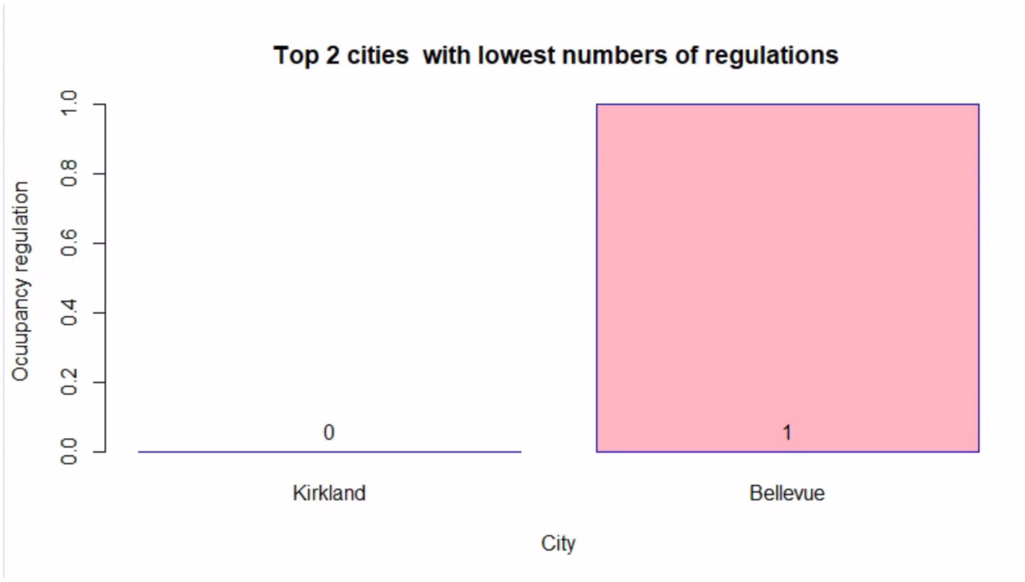


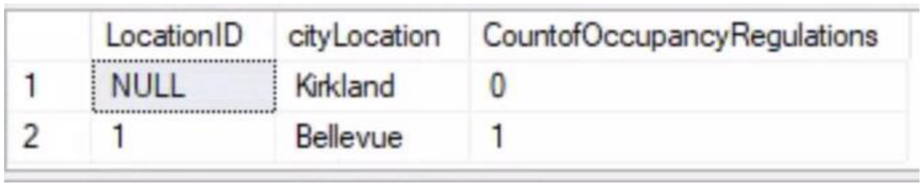


	City	CountOfVisits
1	Kirkland	7
2	Bellevue	6
3	Wood...	6
4	Seattle	3

```sql
--Question 2
```

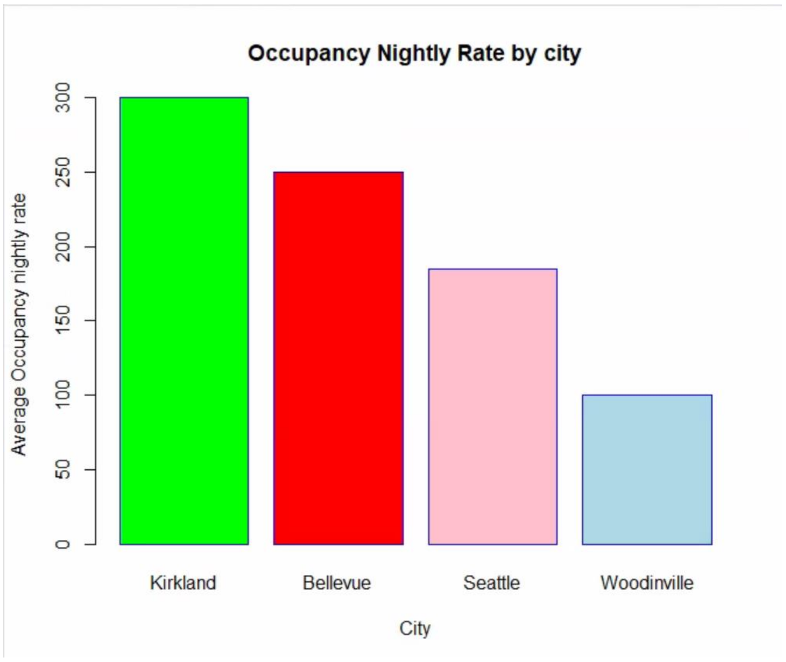
```
--What are the top 2 locations with the least number of occupancy regulation
rules?


SELECT
            Top 2  Location_Regulation.LocationID
,Locations.City as cityLocation
,COUNT(Location_Regulation.LocationID) as  CountofOccupancyRegulations

FROM Location_Regulation
right JOIN Locations ON Location_Regulation.LocationID = locations.LocationID

group by
locations.City
,Location_Regulation.LocationID
ORDER BY
Location_Regulation.LocationID ASC
```

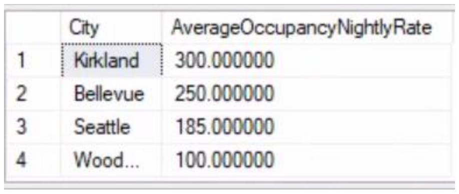




	LocationID	cityLocation	CountofOccupancyRegulations
1	NULL	Kirkland	0
2	1	Bellevue	1

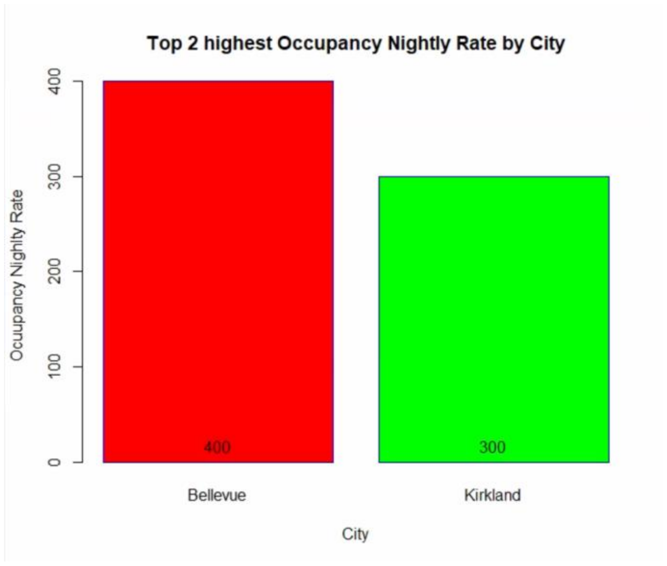
```
-- Question 3
-- What is the  average occupancy nightly rate for each unique city?

SELECT Locations.City, AVG(Property.Occupancy_Nightly_Rate)as   AverageOccupa
ncyNightlyRate
FROM   Property
INNER  JOIN  Locations  ON  Locations.LocationID  =  Property.LocationID

GROUP          BY  Locations.City
ORDER          BY  AverageOccupancyNightlyRate  DESC
```



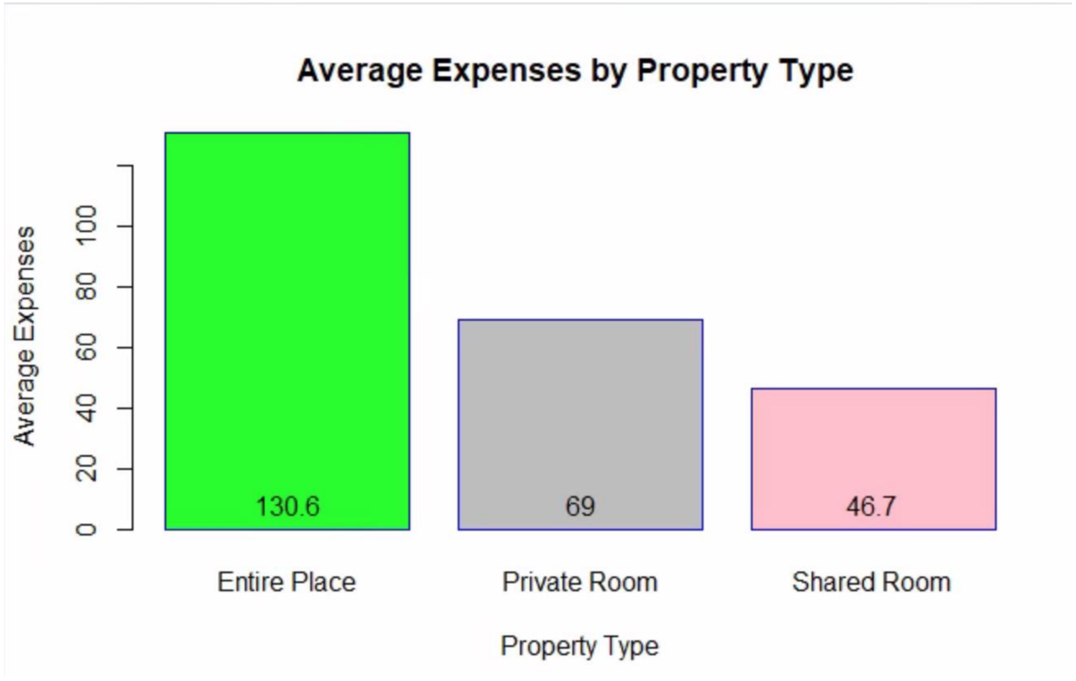


	City	AverageOccupancyNightlyRate
1	Kirkland	300.000000
2	Bellevue	250.000000
3	Seattle	185.000000
4	Wood...	100.000000

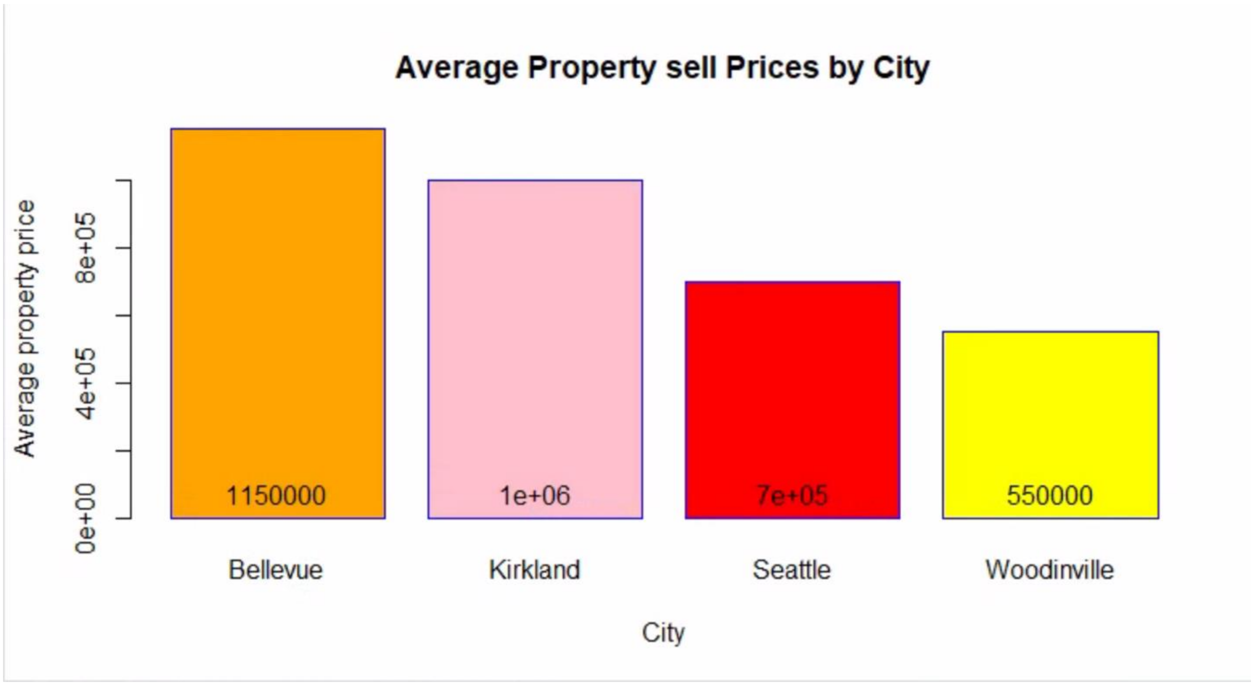
```
--  The top 2 highest Occupancy nightly rate by city
--Although average nightly rate in Kirkland city is higher but highest
--nightly rate in the listings belongs to Bellevue
      SELECT
      TOP 2 Property.Occupancy_Nightly_Rate
            , Locations.City
            ,AVG(Property.Occupancy_Nightly_Rate)as
AverageOccupancyNightlyRate
      FROM Property
      INNER JOIN Locations ON Locations.LocationID = Property.LocationID
      GROUP BY Property.Occupancy_Nightly_Rate, Locations.City
      ORDER BY AverageOccupancyNightlyRate DESC
```



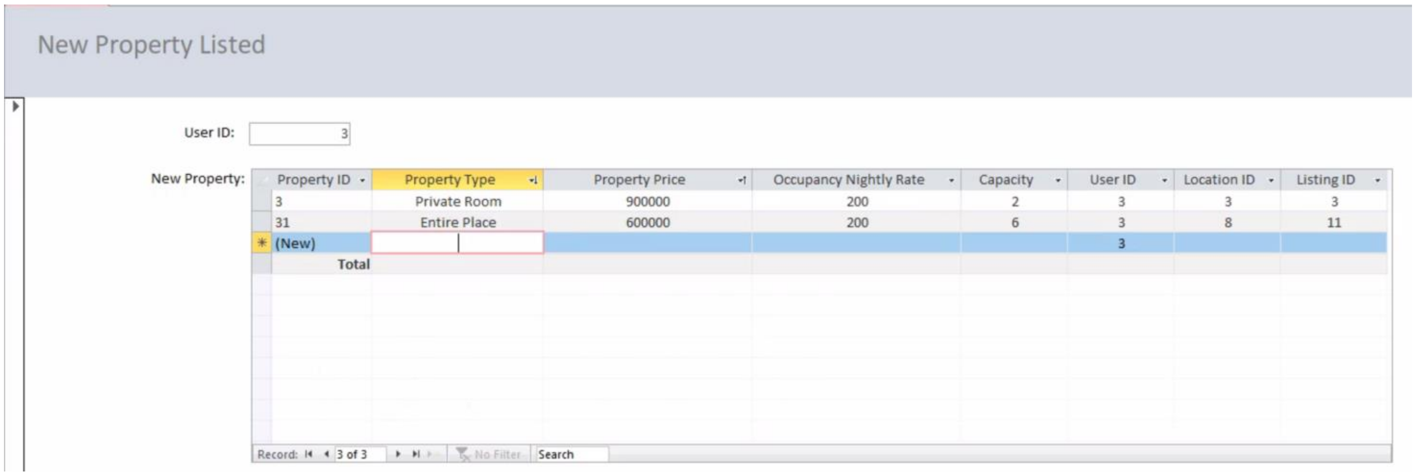
```
--Question 4
--what is the average rental expenses in each property with Certain property
type ?
SELECT
              Property.Property_Type as PropertyType
             ,AVG(Rental_Expenses.Expense_Amount) as AverageRentalExpenses
FROM Rental_Expenses

INNER JOIN Property ON Rental_Expenses.PropertyID = Property.PropertyID

group by
      Property.Property_Type
```



```
--Question 5
--What is the average property prices in each unique City?

SELECT
              locations.City as City
             ,AVG(Property.Property_Price) as AveragePropertPrice
FROM Property
INNER JOIN Locations ON Locations.LocationID = Property.LocationID

group by
      locations.City
```



```
-- Use a form to update new properties added to the listings
-- We added 3 more properties which is updated in our Property table
```



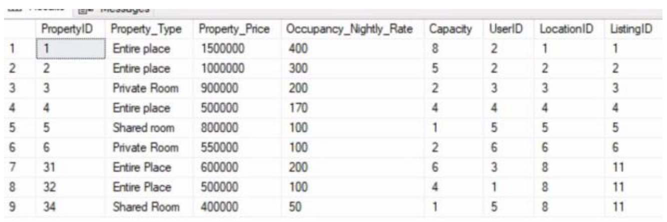
New Property Listed

User ID: 3

New Property:

Property ID	Property Type	Property Price	Occupancy Nightly Rate	Capacity	User ID	Location ID	Listing ID
3	Private Room	900000	200	2	3	3	3
31	Entire Place	600000	200	6	3	8	11
(New)					3		
Total							

Record: 3 of 3 No Filter Search



	PropertyID	Property_Type	Property_Price	Occupancy_Nightly_Rate	Capacity	UserID	LocationID	ListingID
1	1	Entire place	1500000	400	8	2	1	1
2	2	Entire place	1000000	300	5	2	2	2
3	3	Private Room	900000	200	2	3	3	3
4	4	Entire place	500000	170	4	4	4	4
5	5	Shared room	800000	100	1	5	5	5
6	6	Private Room	550000	100	2	6	6	6
7	31	Entire Place	600000	200	6	3	8	11
8	32	Entire Place	500000	100	4	1	8	11
9	34	Shared Room	400000	50	1	5	8	11

REFLECTION

In the beginning of the project, I had this idea that this project is about selecting a dataset and analyzing it with SQL. At first, I did not know at all that we have to make a database from scratch in SQL to be able to work on it. For example, I though it would be something like a programming language that we read the data into the platform and then we can start from there. After I started making the data relation diagrams, I realized that we are modeling a pattern and a map for creating our database. I was super excited that I was able to make the database piece by piece by myself and even better I was working on a database in my area of interest.

I believe that it was very helpful that we leraned the process step by step without knowing what comes next and then as we were preceeding with the project, the previous steps became more clear for me. Also Doing homeworks alongside with the project was super helpful. If I did not have to do an independent project, I would not deeply learn the materials in this class. However, without homeworks I could not do the project at all.

The next time that I do such a project, I would definitely work more closly on the logical relations diagram and make them verey closly in relation to my data questions. My attributes and my entities relations are all very important on how I can answer my data questions in the last step. However, in this project, It was harder to answer questions the way we desired because we did not insert enough data values and our results are sometimes strange and off. Also learning about Access I realized at the end of the project that I could use access to add user names and visits instead of using a procedure in SQL which cpould save me a lot of time.

In overall, This learning was extremely helpful to me to understand databases specially the part that we connected the data to R and Access was facimnating for me as I did not know we can ever do that.

After this class, I love working with SQL because it is super fun and useful. Also, our professor teaching strategy and encouragement made this class super interesting to me. I deffinitly would love to keep working in this project and make it more professional so I can present it to future companies. Who knows? That company could be Airbnb; because that's one of my favarite places to work as a data scientist!

SUMMARY

I followed my logical relation diagram and my business rules closely during the second half of the project. I answered the most important data questions by my select statements while I was going back and forward in my data inserts as I would realize that I actually need more data to answer my questions. I did use Access very late in my project otherwise it could be such a big help in my data entry updates and having better and more reasonable results for my data questions.

I added my from which updated the property table and added some new property to it at the very end and the list of property will come up as a different one that I posted in my question 5 and this is the reason because I understood too late what is the point of using Access at all.

In overall, comibination of using SQL, R and Access is a a great tool to make a good database and provide analysis and visualizations for our data.