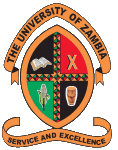
**[](https://www.unza.zm/) The University of Zambia** 

***in association with***

**ZCAS University**

**BIT221/BTS/BBM221: Introduction to databases**

**ASSIGNMENT No 2**

**name: cacious siamunyanga**

**student number: zu19254**

**course: introduction to databases**

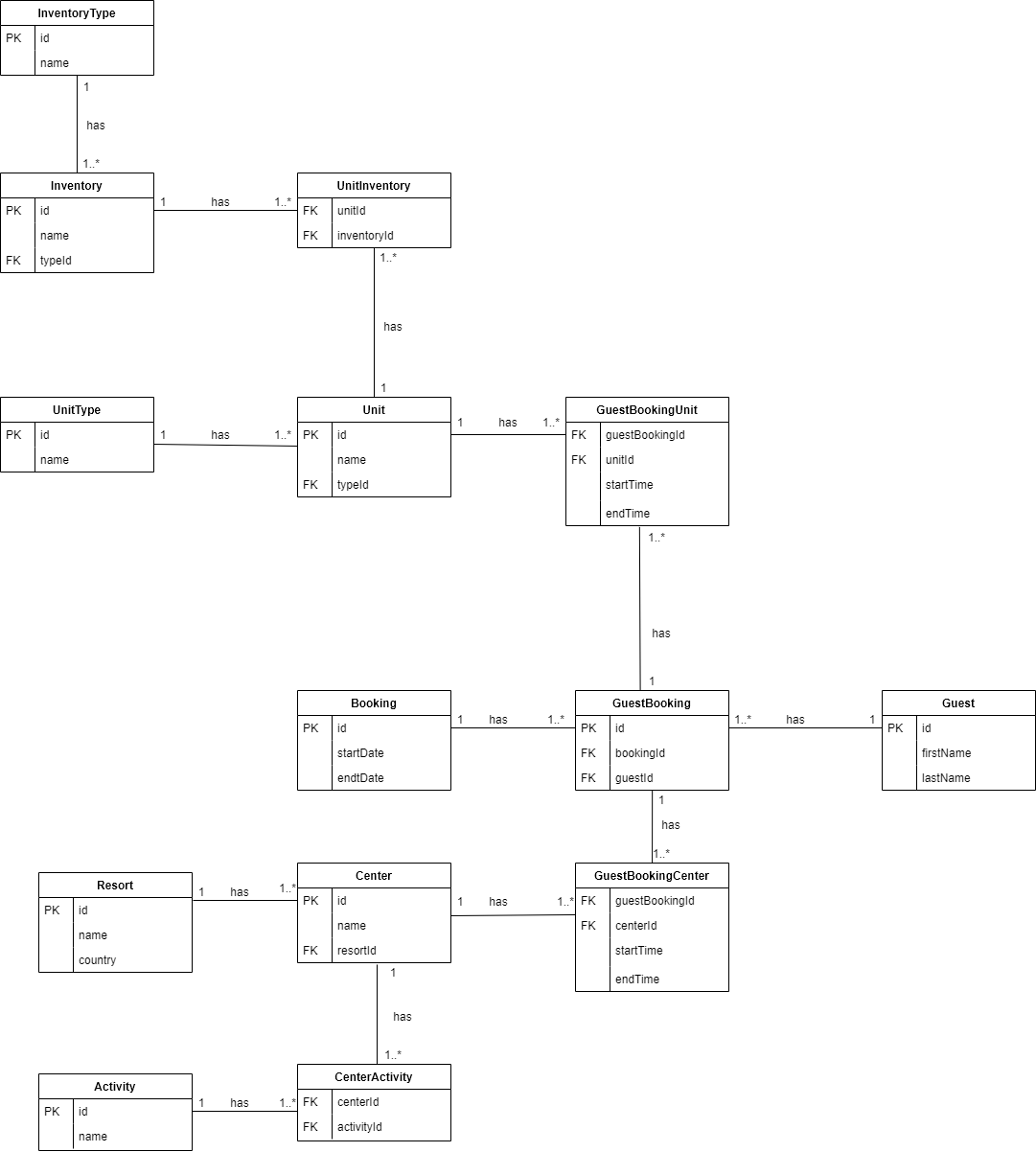
**study program: Bachelor of science in internet technologies and security**

**lecturer name: mr jonathan zulu**

**date: 10th may 2020**

Task 1 – Design

a) Bright Green Holidays ERD



b)

Normalization of the sample data provided resulted in a much more efficient ERD that will lead to less redundancy when updating and deleting values in the database tables. This is so because, some unnecessary fields had to be removed, such as the “Member y/n” field in the “Booking List” table that I found redundant as this can be solved with queries that give much more accurate date for verification if someone is a member or not from the new “Customer” table as seen in the designed ERD.

Other fields were regrouped to create new entities for a much more accurate representation and, I believe, better data entry, analysis and more. An example is the ambiguity that could exists in the “Session and fees” table where two session numbers would have to be entered for the same session if the session happened on both the 1st and 2nd floor on the same day. Extracting the floor and the fee fields to create a new “Floor” table allows for the proper representation of the session and the floor. Therefore, their manipulation becomes easier. The sample data also shows different prices for each floor even if it not specified explicitly in the club’s background. Such normalization would allow changing of that logic to be easy rather than going through the whole table to change every instance if an error was ever made.

c)

1) InventoryType

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fieldName | dataType | size | Constraint | description |
| id | int |  | Primary Key | inventory type unique identifier |
| name | varchar | 50 | not null | the name of the inventory type |

2) Inventory

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fieldName | dataType | size | Constraint | description |
| id | int |  | Primary Key | inventory unique identifier |
| name | varchar | 50 | not null | the name of the inventory item |
| typeId | int |  | Foreign key | inventory type unique identifier and link to the InventoryType table |

3) UnitInventory

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fieldName | dataType | size | Constraint | description |
| unitId | int |  | Foreign Key | the unit unique identifier and link to the Unit table |
| inventoryId | int |  | Foreign Key | the inventory unique identifier and link to the Inventory table |

4) Unit

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fieldName | dataType | size | Constraint | description |
| id | int |  | Primary Key | unit unique identifier |
| name | varchar | 50 | not null | the name of the unit |
| typeid | int |  | Foreign Key | the unit type unique identifier and link to the UnitType table |

5) UnitType

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fieldName | dataType | size | Constraint | description |
| id | int |  | Primary Key | the unit type unique identifier |
| name | varchar | 50 | not null | the name of the unit type |

6) GuestBookingUnit

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fieldName | dataType | size | Constraint | description |
| guestBookingId | int |  | Foreign Key | guest booking unique identifier and link to the GuestBooking table |
| unitId | int |  | Foreign Key | unit unique identifier and link to the Unit table |
| startTime | datetime |  | not null | the date and time at which a guest started staying in a unit during a specific booking |
| endTime | datetime |  |  | the date and time at which a guest stopped staying at a unit during a specific booking |

7) GuestBooking

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fieldName | dataType | size | Constraint | description |
| id | int |  | Primary Key | Guest booking unique identifier. Since multiple guests can have the same booking, this gives each user a unique identifier related to that specific guest |
| bookingId | varchar | 100 | Foreign Key | booking unique identifier and link to the Booking table |
| guestId | int |  | Foreign Key | guest unique identifier and link to the Guest table |

8) Booking

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fieldName | dataType | size | Constraint | description |
| id | varchar | 100 | Primary Key | booking unique identifier |
| startDate | date |  | not null | date the booking was made |
| endDate | date |  |  | date the booking ends |

9) Guest

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fieldName | dataType | size | Constraint | description |
| id | int |  | Primary Key | guest unique identifier |
| firstName | varchar | 50 | not null | guest first name |
| lastName | varchar | 50 | not null | guest last name |

10) GuestBookingCenter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fieldName | dataType | size | Constraint | description |
| guestBookingId | int |  | Foreign Key | guest booking unique identifier and link to the GuestBooking table |
| centerId | int |  | Foreign Key | center unique identifier and link to the Center table |
| startTime | datetime |  | not null | the date and time a guest registered at a center during a specific booking |
| endTime | datetime |  |  | the date and time a guest deregistered at a center during a specific booking |

11) Center

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fieldName | dataType | size | Constraint | description |
| id | int |  | Primary Key | center unique identifier |
| name | varchar | 75 | not null | the name of the center |
| resortId | int |  | Foreign Key | resort unique identifier and link to the Resort table |

12) Resort

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fieldName | dataType | size | Constraint | description |
| id | int |  | Primary Key | resort unique identifier |
| name | varchar | 75 | not null | the resort's name |
| country | varchar | 50 | Not null | Country in which the resort is based |

13) CenterActivity

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fieldName | dataType | size | Constraint | description |
| centerId | int |  | Foreign Key | center unique identifier and link to the Center table |
| activityId | int |  | Foreign Key | activity unique identifier and link to the Activity table |

14) Activity

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fieldName | dataType | size | Constraint | description |
| id | int |  | Primary Key | activity unique identifier |
| name | varchar | 50 | not null | name of the activity |

TASK 2

* Create the tables. Make sure you show the CREATE scripts as running in the programming environment

1. CREATE TABLE InventoryType

(

id int primary key,

name varchar(50) not null

);

1. CREATE TABLE Inventory

(

id int primary key,

name varchar(50) not null,

typeId int not null,

foreign key(typeId) references InventoryType(id)

);

1. CREATE TABLE UnitType

(

id int primary key,

name varchar(50) not null

);

1. CREATE TABLE Unit

(

id int primary key,

name varchar(50) not null,

typeId int not null,

foreign key(typeId) references UnitType(id)

);

1. CREATE TABLE UnitInventory

(

unitId int not null,

inventoryId int not null,

foreign key(unitId) references Unit(id),

foreign key(inventoryId) references Inventory(id)

);

1. CREATE TABLE Guest

(

id int primary key,

firstName varchar(50) not null,

lastName varchar(50) not null

);

1. CREATE TABLE Booking

(

id varchar(100) primary key,

startDate date not null,

endDate date

);

1. CREATE TABLE GuestBooking

(

id int primary key,

bookingId varchar(100) not null,

guestId int not null,

foreign key(bookingId) references Booking(id),

foreign key(guestId) references Guest(id)

);

1. CREATE TABLE GuestBookingUnit

(

guestBookingId int not null,

unitId int not null,

startTime datetime not null,

endTime datetime,

foreign key(guestBookingId) references GuestBooking(id),

foreign key(unitId) references Unit(id)

);

1. CREATE TABLE Resort

(

id int primary key,

name varchar(75) not null,

country varchar(50) not null

);

1. CREATE TABLE Center

(

id int primary key,

name varchar(75) not null,

resortId int not null,

foreign key(resortId) references Resort(id)

);

1. CREATE TABLE GuestBookingCenter

(

guestBookingId int not null,

centerId int not null,

startTime datetime not null,

endTime datetime,

foreign key(guestBookingId) references GuestBooking(id),

foreign key(centerId) references Center(id)

);

1. CREATE TABLE Activity

(

id int primary key,

name varchar(50) not null

);

1. CREATE TABLE CenterActivity

(

centerId int not null,

activityId int not null,

foreign key(centerId) references Center(id),

foreign key(activityId) references Activity(id)

);

DATA ENTRY

INSERT INTO Booking (id, startDate, endDate)

VALUES ('B2001', '01/July/2020', '14/July/2020')

VALUES ('B2003', '07/July/2020', '14/July/2020')

VALUES ('B2009', '07/July/2020', '14/July/2020')

VALUES ('B2010', '07/July/2020', '14/July/2020')

VALUES ('B2013', '15/July/2020', '22/July/2020')

INSERT INTO Guest

(id, firstName, lastName)

VALUES ('3399', 'Martina', 'Bywater')

VALUES ('3400', 'Dan', 'Bywater')

VALUES ('3300', 'David', 'Olusonga')

VALUES ('3301', 'Sonia', 'Chaplow')

VALUES ('4101', 'Maria', 'De Silva')

VALUES ('4102', 'Jermine', 'Easter')

VALUES ('3111', 'Ameer', 'Akhta')

INSERT INTO Resort

(id, name, country)

VALUES ( '1', 'Anjozorobe', 'Madagascar' )

VALUES ( '2', 'Tsingy De Bemaraha', 'Madagascar' )

VALUES ( '3', 'Miliwane', 'Swaziland' )

VALUES ( '4', 'Malolotja', 'Swaziland' )

INSERT INTO Center

(id, name, resortId)

VALUES ( '1', ' Anjozorobe Forest Center', '1' )

VALUES ( '2', 'Explores Center', '1' )

VALUES ( '3', 'Treetop Center', '2' )

VALUES ( '4', 'Miliwane Freedom Center', '3' )

VALUES ( '5', 'Highland Center', '4' )

INSERT INTO Activity

(id, name)

VALUES ( '1', ' Hiking')

VALUES ( '2', ' Conservation')

VALUES ( '3', ' Cycling')

VALUES ( '4', ' Mini-golf')

VALUES ( '5', ' Diving')

VALUES ( '6', ' Village Visit')

INSERT INTO UnitType

(id, name)

VALUES ( '1', 'Standard Tent')

VALUES ( '2', 'Deluxe Tent')

VALUES ( '3', 'Lodge')

VALUES ( '4', 'Yurt')

INSERT INTO Unit

(id, name, typeId)

VALUES ( '9934', 'Verdigis', '1')

VALUES ( '4500', 'Jade', '2')

VALUES ( '5670', 'Celadon', '2')

VALUES ( '2321', 'Aquamarine', '3')

VALUES ( '2500', 'Turquoise', '4')

INSERT INTO InventoryType

(id, name)

VALUES ( '1', 'Bedding')

VALUES ( '2', 'Kitchen Equipment')

VALUES ( '3', 'Outdoor Equipment')

INSERT INTO Inventory

(id, name, typeId)

VALUES ( '1', 'Set of bed sheets', '1')

VALUES ( '2', 'Pillows', '1')

VALUES ( '3', 'Coffee maker', '2')

VALUES ( '4', 'Cutlery set', '2')

VALUES ( '5', 'Crockery set', '2')

VALUES ( '6', 'Pan set', '2')

VALUES ( '7', 'Sun umbrella', '3')

VALUES ( '8', 'Lounger set', '2')

VALUES ( '9', 'Barbeque set', '3')

INSERT INTO CenterActivity

(centerId, activityId)

VALUES ( '1', '1')

VALUES ( '1', '2')

VALUES ( '1', '3')

VALUES ( '1', '4')

VALUES ( '2', '1')

VALUES ( '2', '5')

VALUES ( '4', '1')

VALUES ( '5', '1')

VALUES ( '5', '5')

VALUES ( '5', '6')

INSERT INTO GuestBooking

(id, bookingId, guestId)

VALUES ( '1', 'B2001', '3399')

VALUES ( '2', 'B2001', '3400')

VALUES ( '3', 'B2003', '3300')

VALUES ( '4', 'B2003', '3301')

VALUES ( '5', 'B2009', '4101')

VALUES ( '6', 'B2009', '4102')

VALUES ( '7', 'B2010', '3111')

VALUES ( '8', 'B2013', '3111')

INSERT INTO GuestBookingCenter

(guestBookingId, centerId, startTime, endTime)

VALUES ( '1', '1', '07/01/2020 10:00:00', '07/14/2020 10:00:00')

VALUES ( '2', '1', '07/01/2020 10:00:00', '07/14/2020 10:00:00')

VALUES ( '3', '2', '07/07/2020 10:00:00', '07/14/2020 10:00:00')

VALUES ( '4', '2', '07/07/2020 10:00:00', '07/14/2020 10:00:00')

VALUES ( '5', '3', '07/07/2020 10:00:00', '07/14/2020 10:00:00')

VALUES ( '6', '3', '07/07/2020 10:00:00', '07/14/2020 10:00:00')

VALUES ( '7', '4', '07/07/2020 10:00:00', '07/14/2020 10:00:00')

VALUES ( '8', '5', '07/15/2020 10:00:00', '07/22/2020 10:00:00')

INSERT INTO GuestBookingUnit

(guestBookingId, unitId, startTime, endTime)

VALUES ( '1', '9934', '07/1/2020 10:00:00', '07/14/2020 10:00:00')

VALUES ( '2', '9934', '07/1/2020 10:00:00', '07/14/2020 10:00:00')

VALUES ( '3', '4500', '07/7/2020 10:00:00', '07/14/2020 10:00:00')

VALUES ( '4', '4500', '07/7/2020 10:00:00', '07/14/2020 10:00:00')

VALUES ( '5', '5670', '07/7/2020 10:00:00', '07/14/2020 10:00:00')

VALUES ( '6', '5670', '07/7/2020 10:00:00', '07/14/2020 10:00:00')

VALUES ( '7', '2321', '07/7/2020 10:00:00', '07/14/2020 10:00:00')

VALUES ( '8', '2500', '07/15/2020 10:00:00', '07/22/2020 10:00:00')

INSERT INTO UnitInventory

(unitId, inventoryId)

VALUES ( '9934', '1')

VALUES ( '9934', '2')

VALUES ( '9934', '3')

VALUES ( '9934', '4')

VALUES ( '9934', '5')

VALUES ( '9934', '6')

VALUES ( '4500', '1')

VALUES ( '4500', '2')

VALUES ( '4500', '3')

VALUES ( '4500', '4')

VALUES ( '4500', '5')

VALUES ( '4500', '6')

VALUES ( '4500', '7')

VALUES ( '4500', '8')

VALUES ( '5670', '1')

VALUES ( '5670', '2')

VALUES ( '5670', '9')

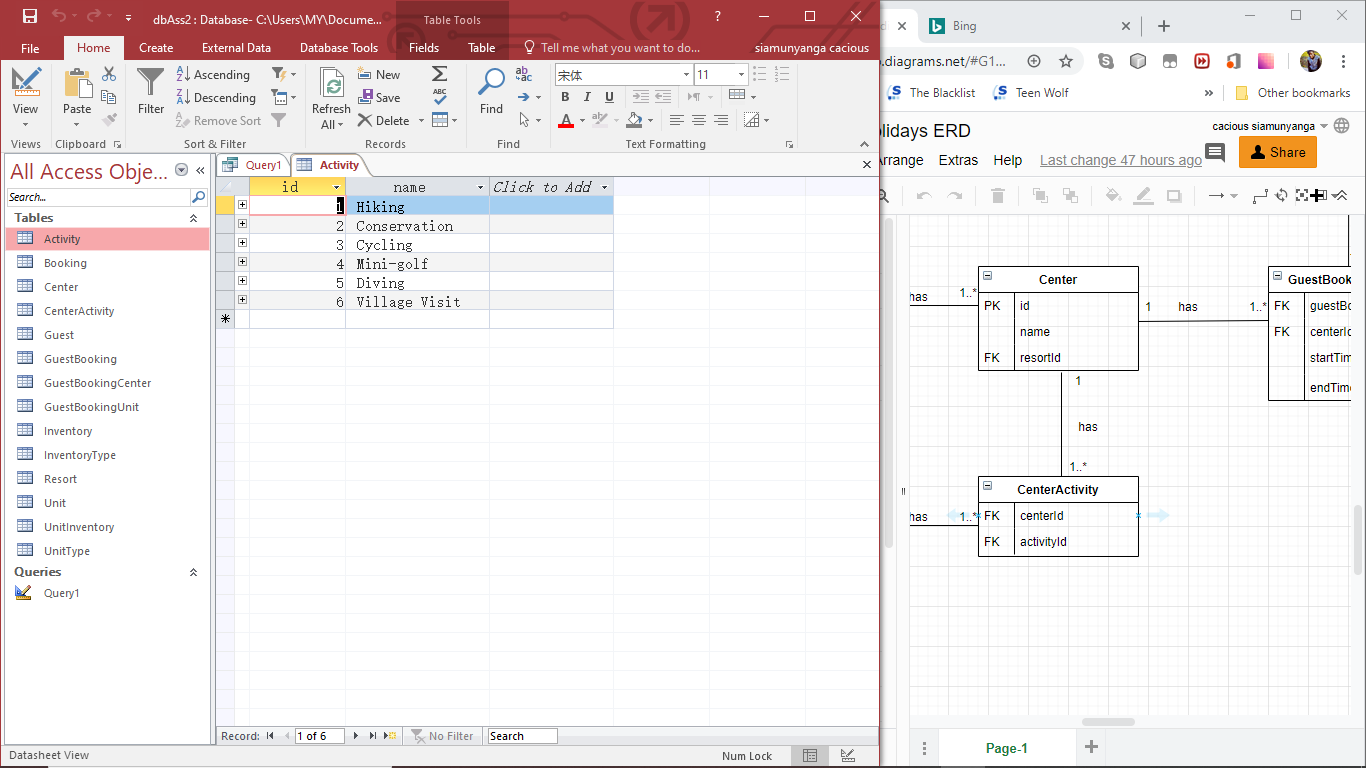
VALUES ( '2321', '2')

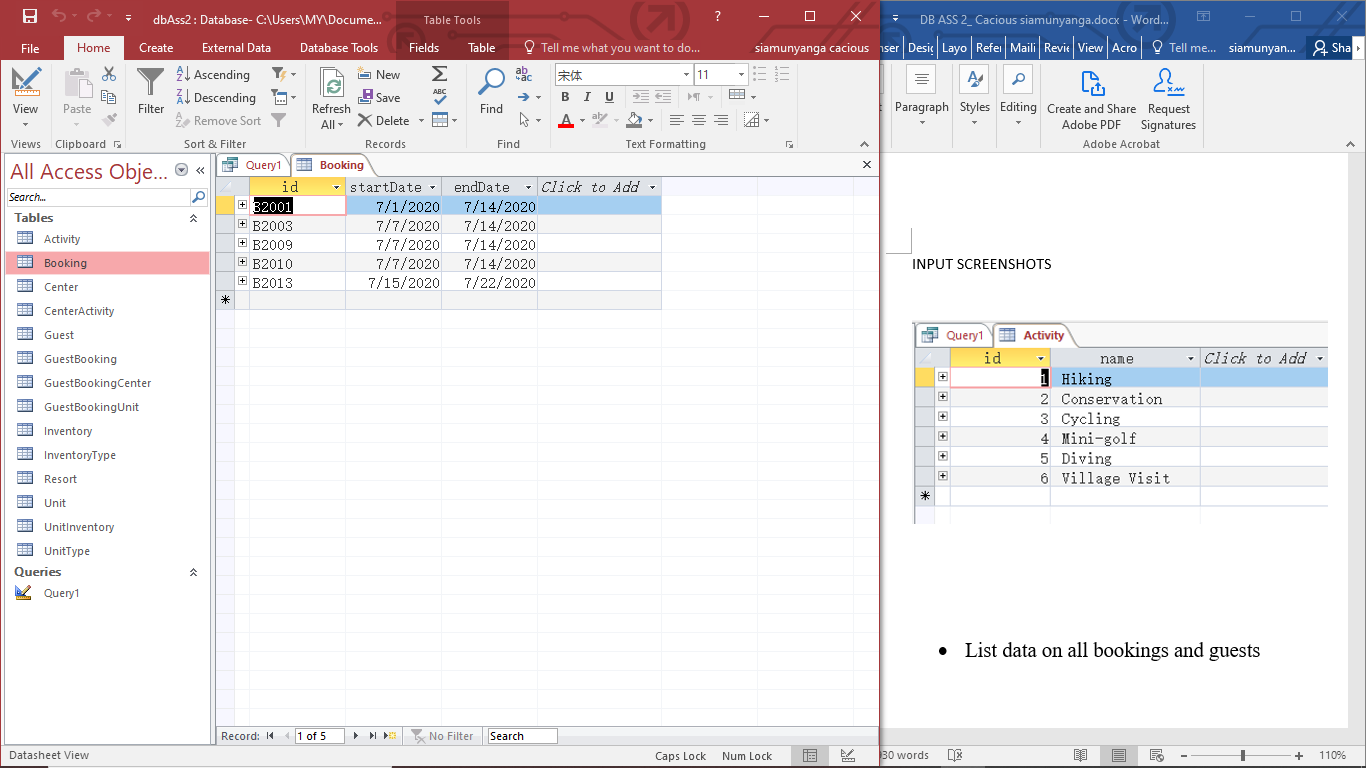
VALUES ( '2321', '1')

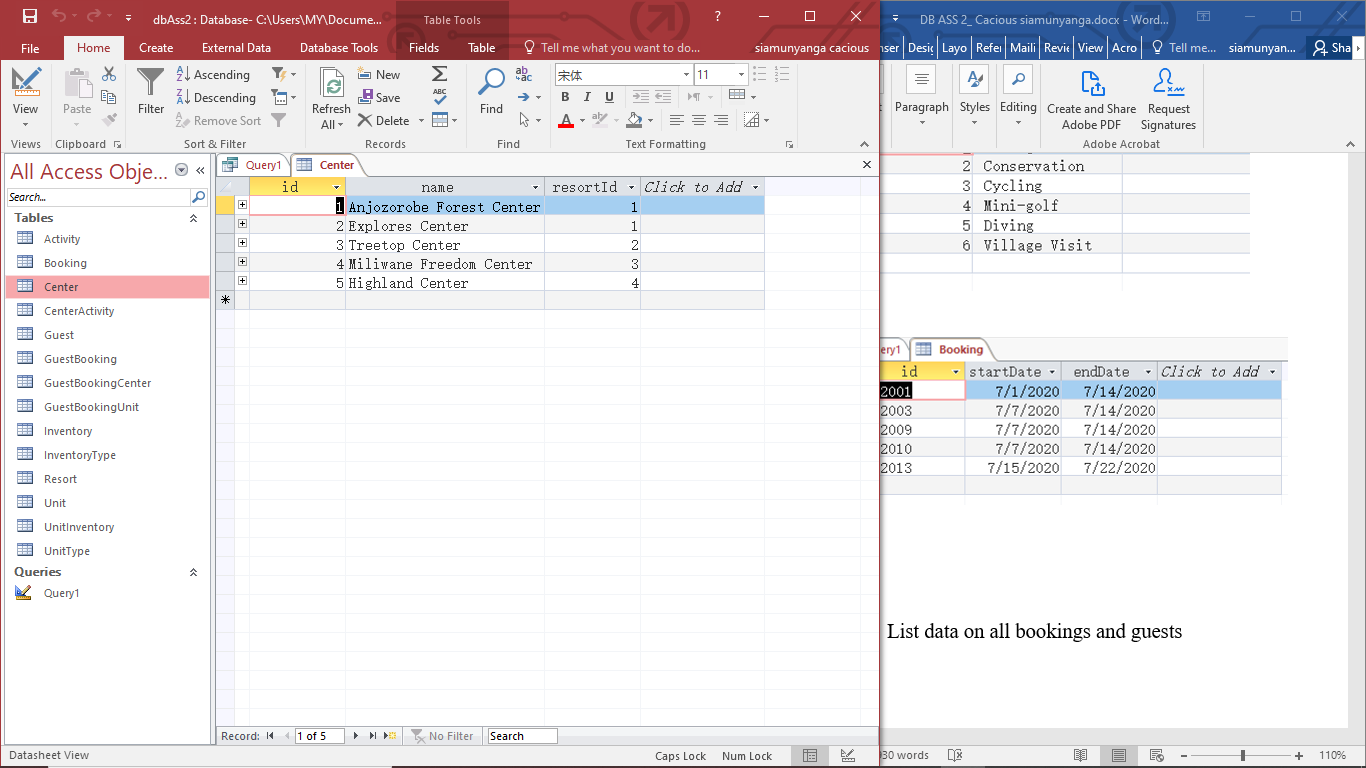
VALUES ( '2321', '9')

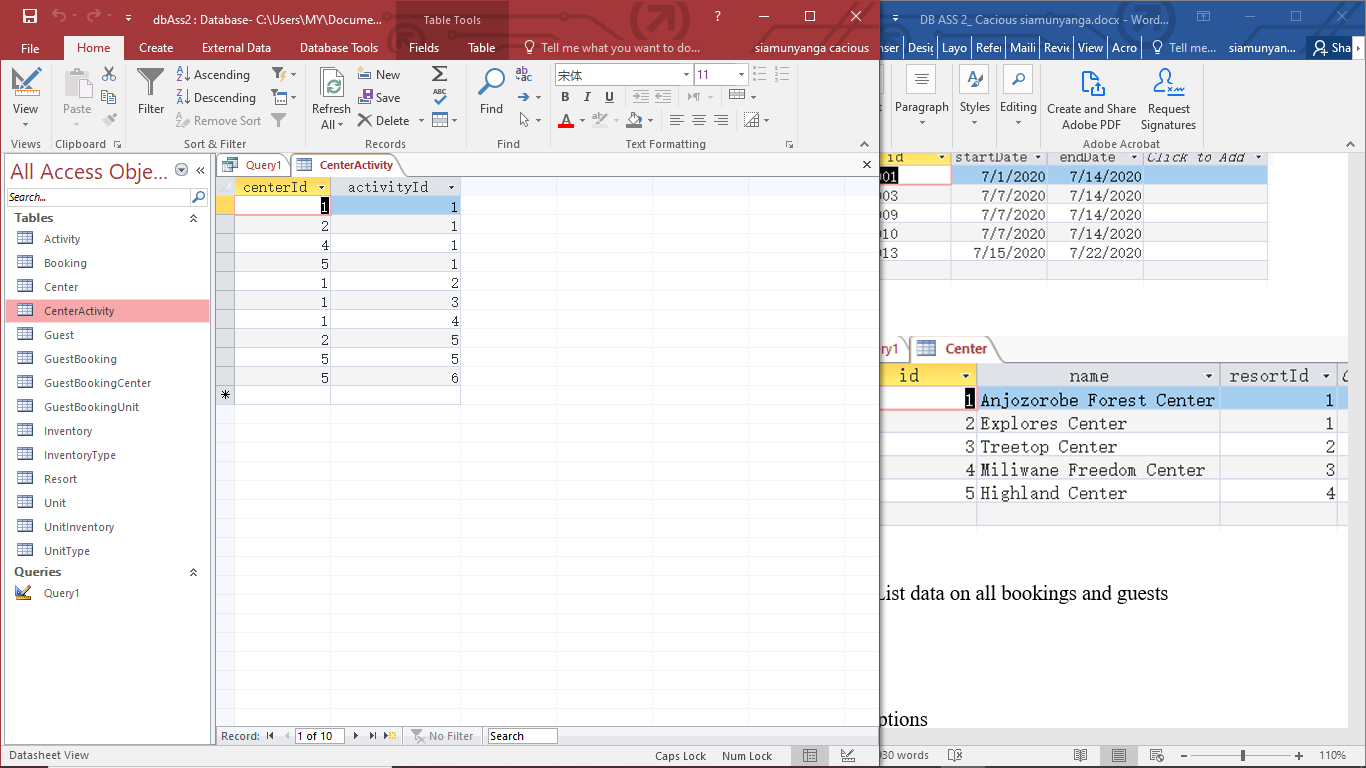
VALUES ( '2500', '9')

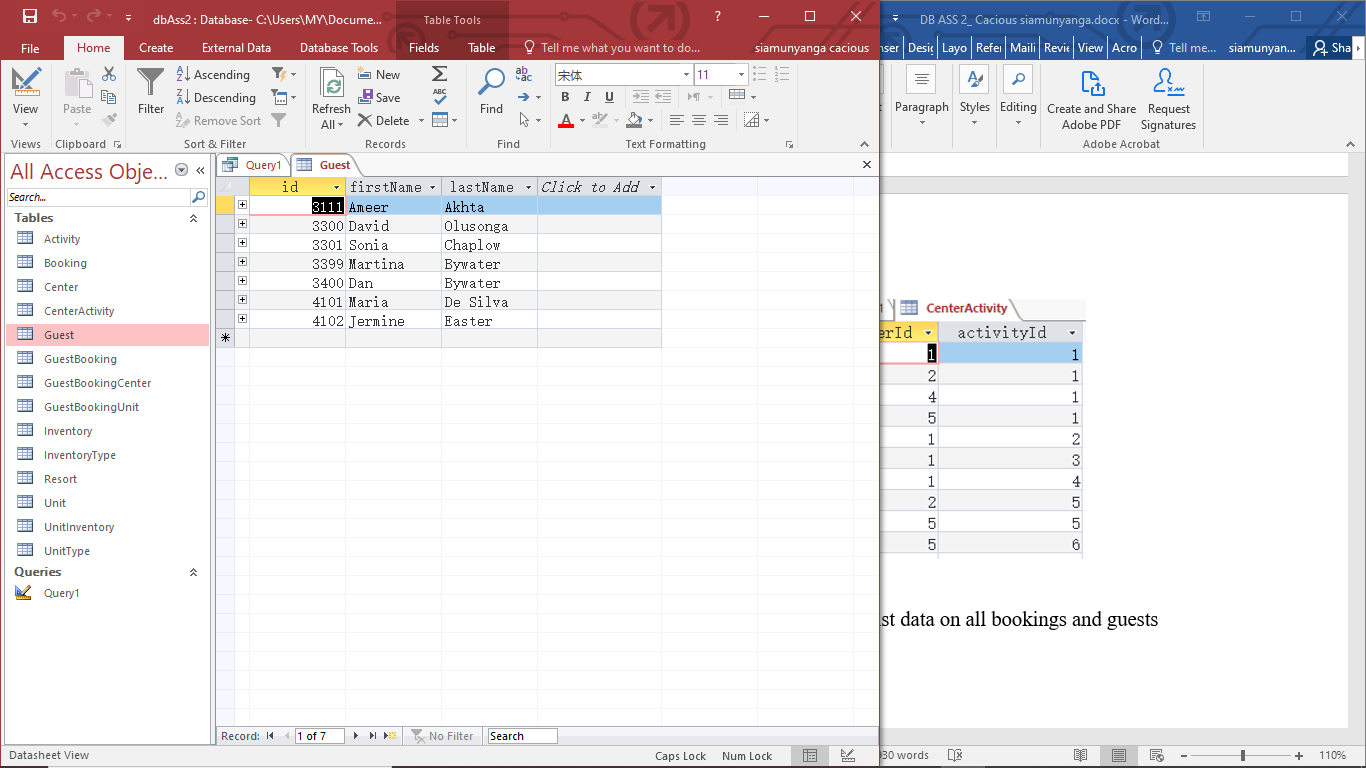
INPUT SCREENSHOTS

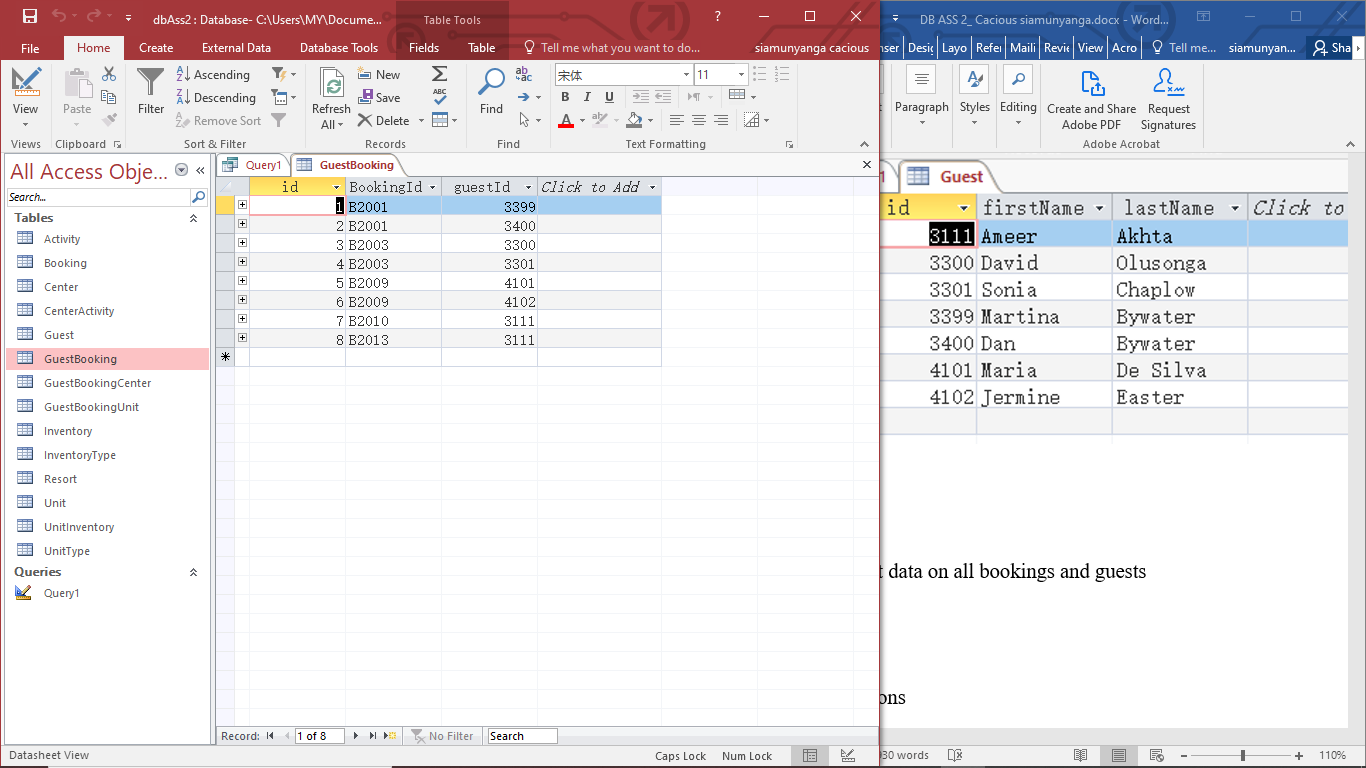


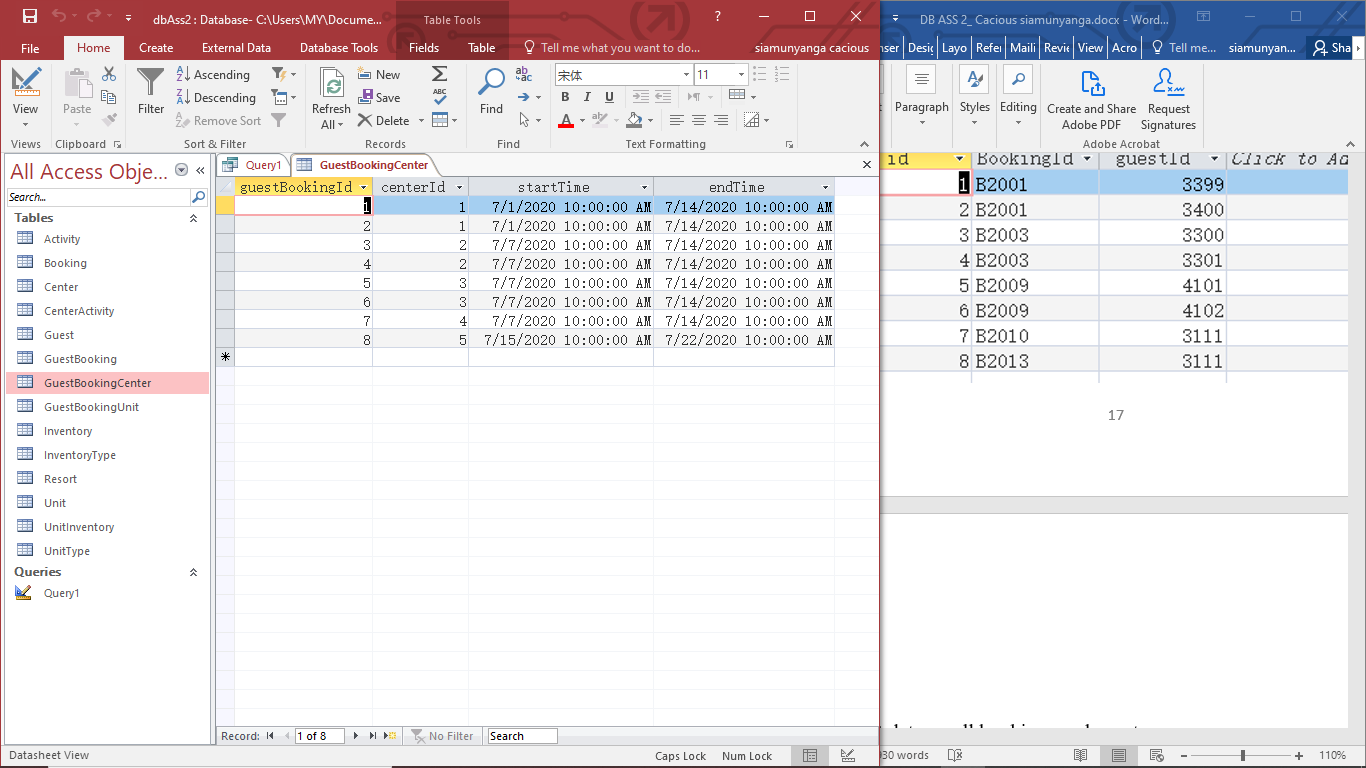


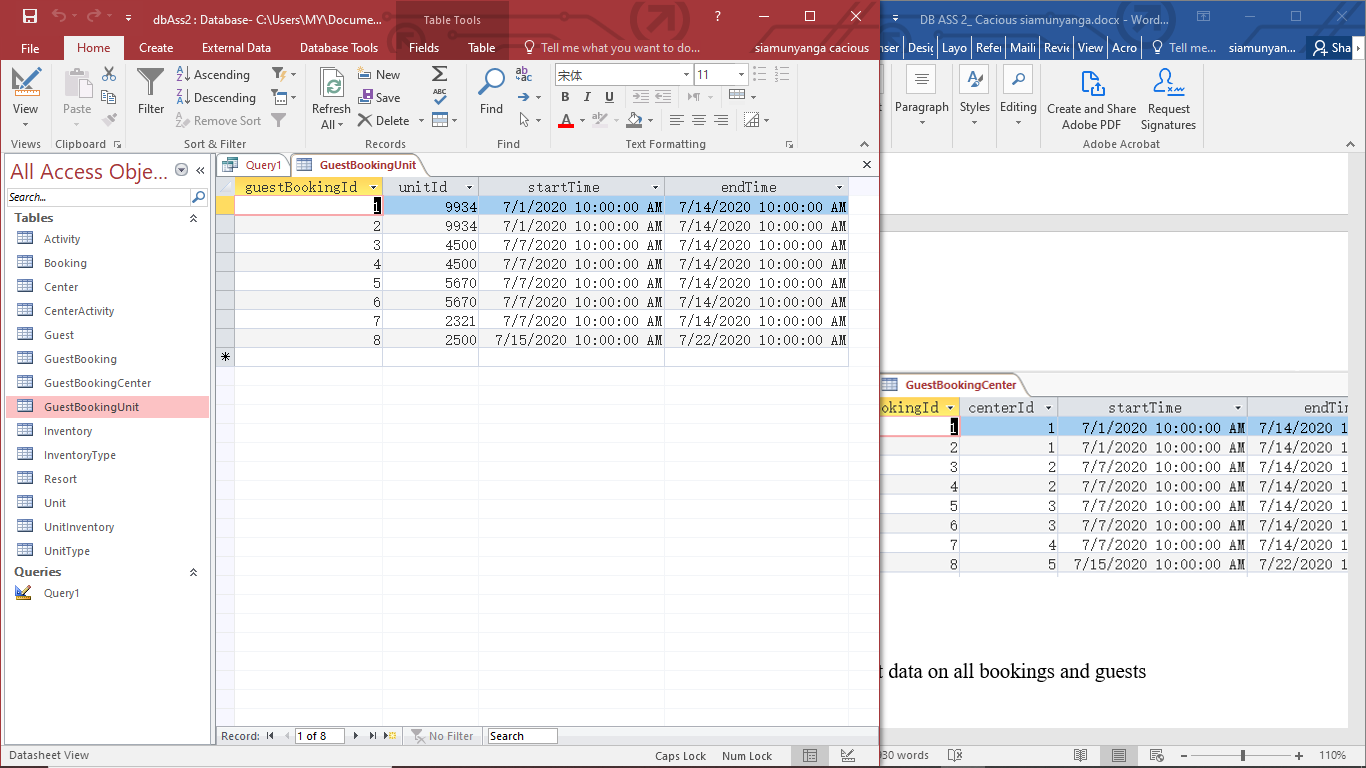


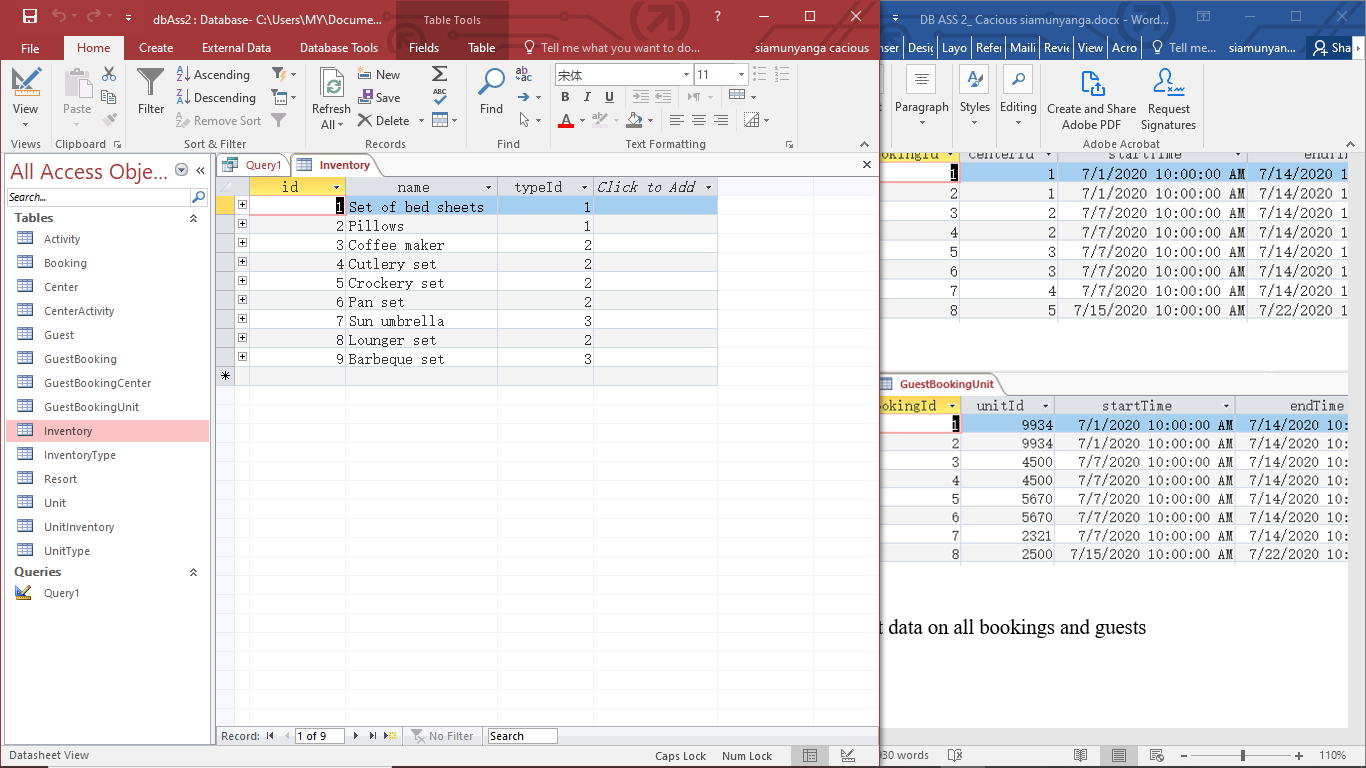


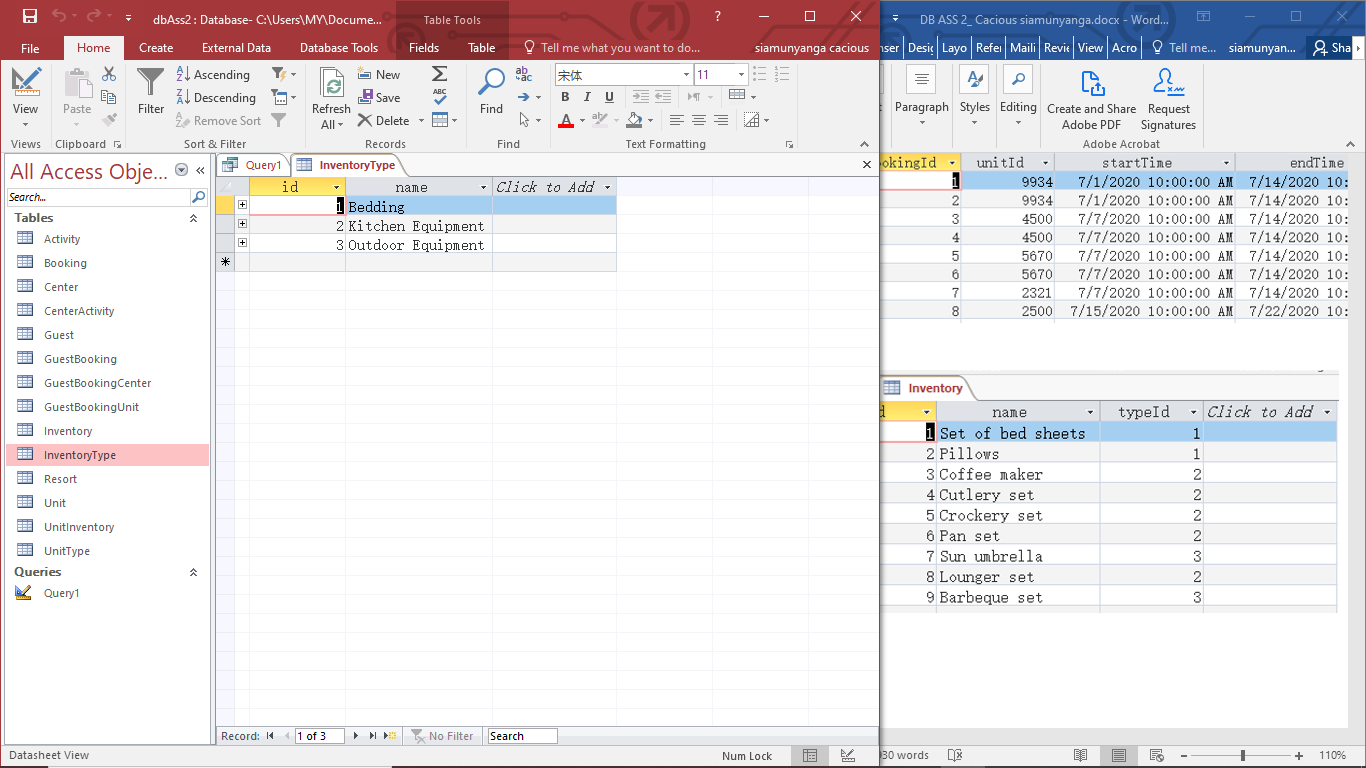


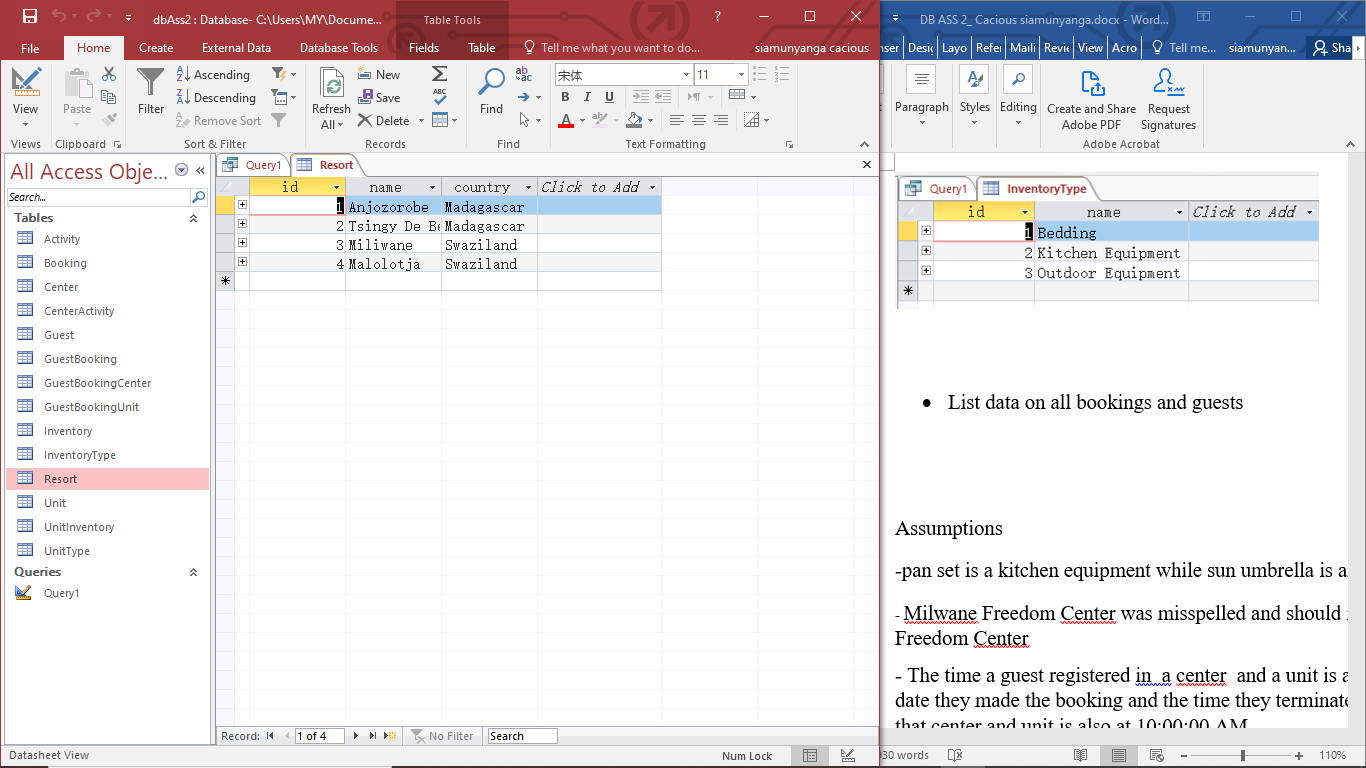


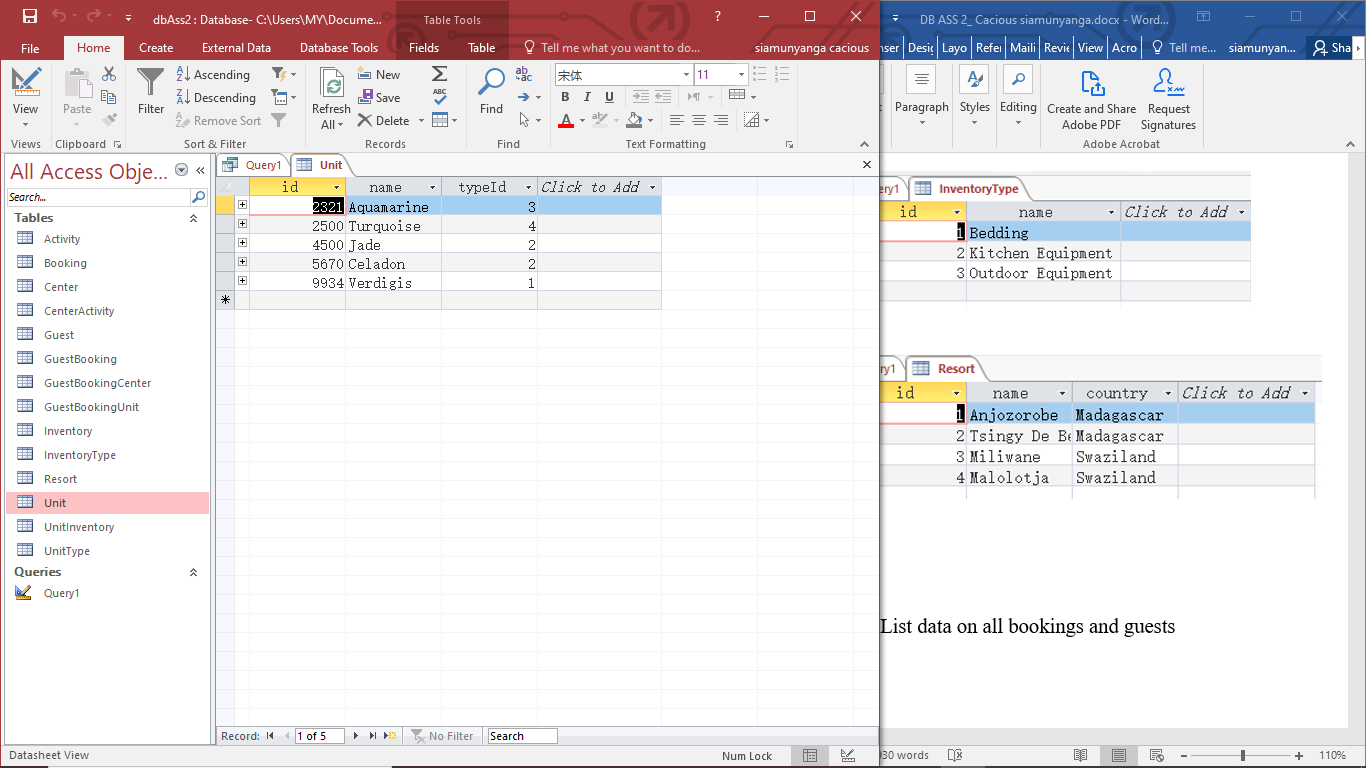


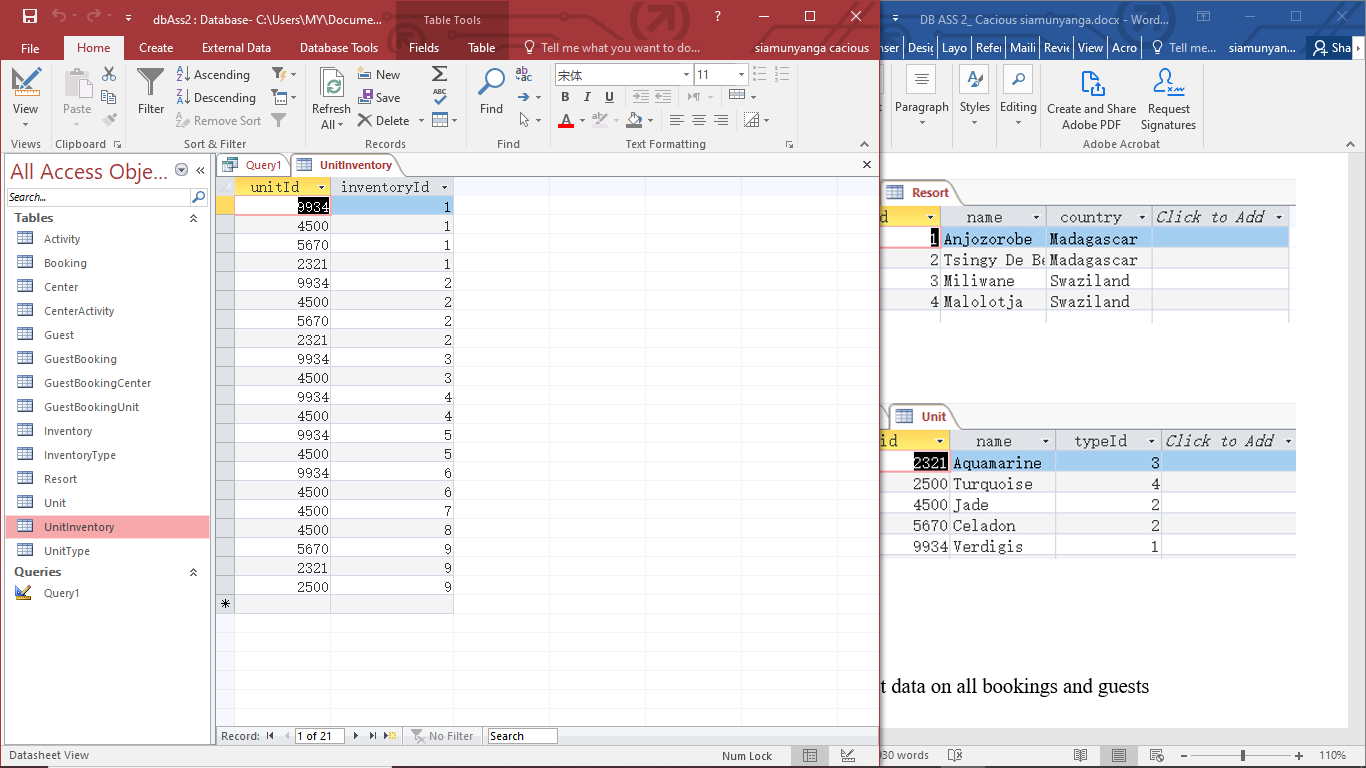


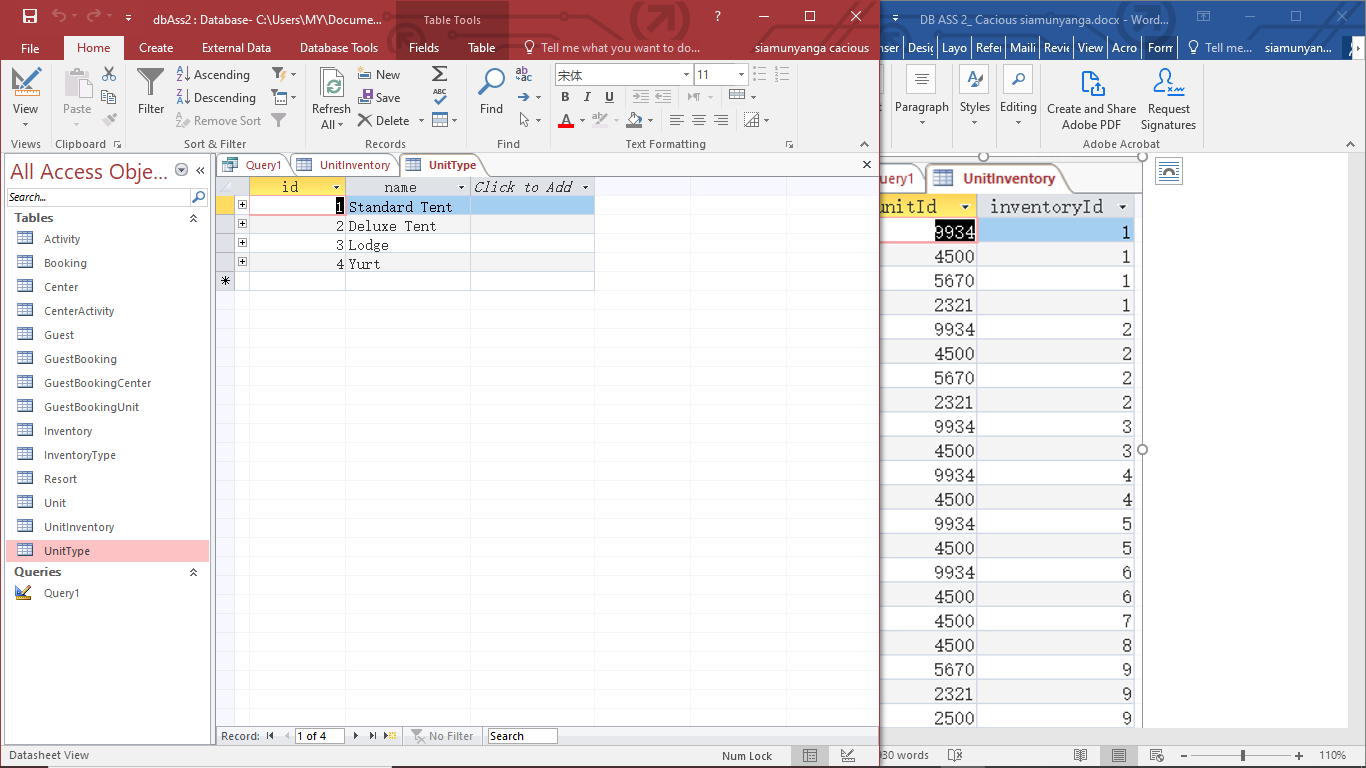












* List data on all bookings and guests

SELECT gb.id, b.id, b.startDate, b.endDate, g.id, g.firstName, g.lastName, c.name, u.name, ut.name

FROM Booking b, Guest g, GuestBooking gb, GuestBookingCenter gbc, Center c, GuestBookingUnit gbu, Unit u, UnitType ut

WHERE gb.bookingId = b.id

AND gb.guestId = g.id

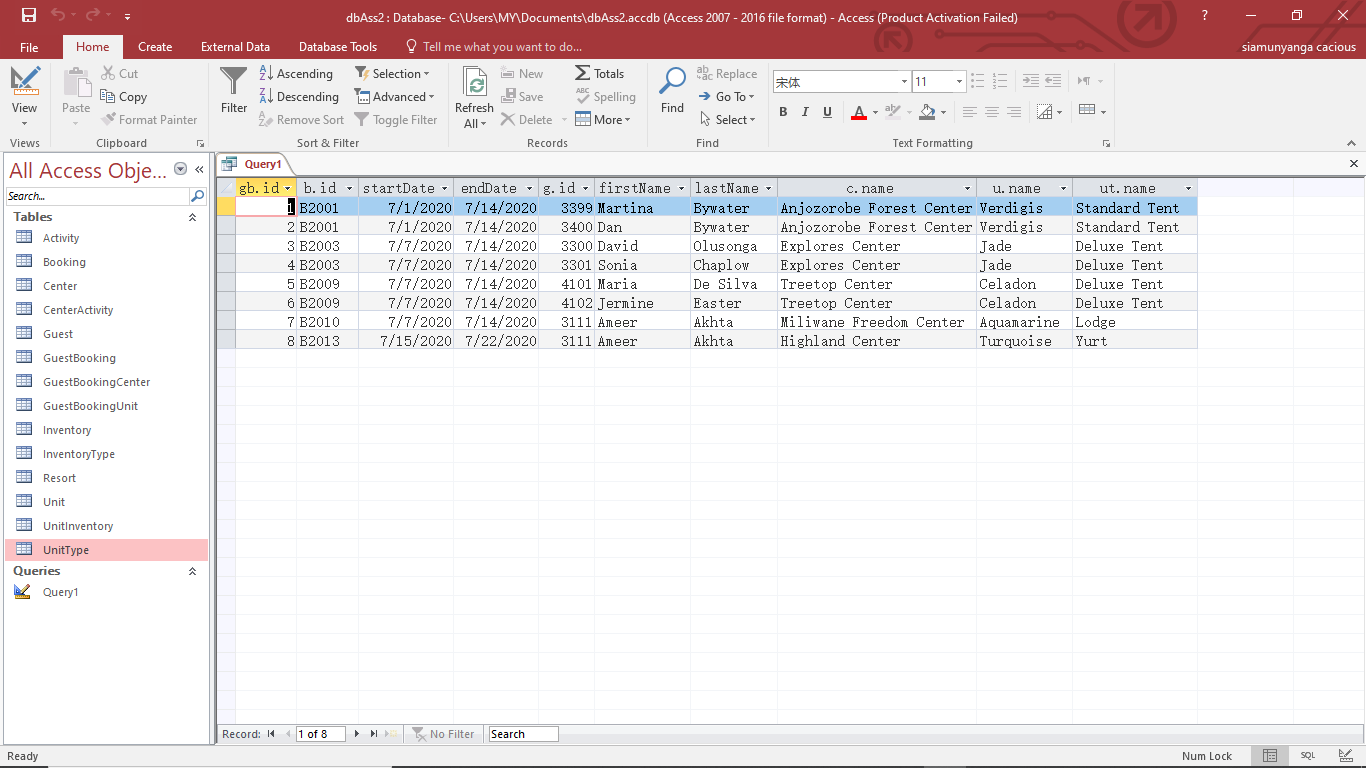
AND gbc.guestBookingId = gb.id

AND gbc.centerId = c.id

AND gbu.guestBookingId = gb.id

AND gbu.UnitId = u.id

AND u.typeId = ut.id

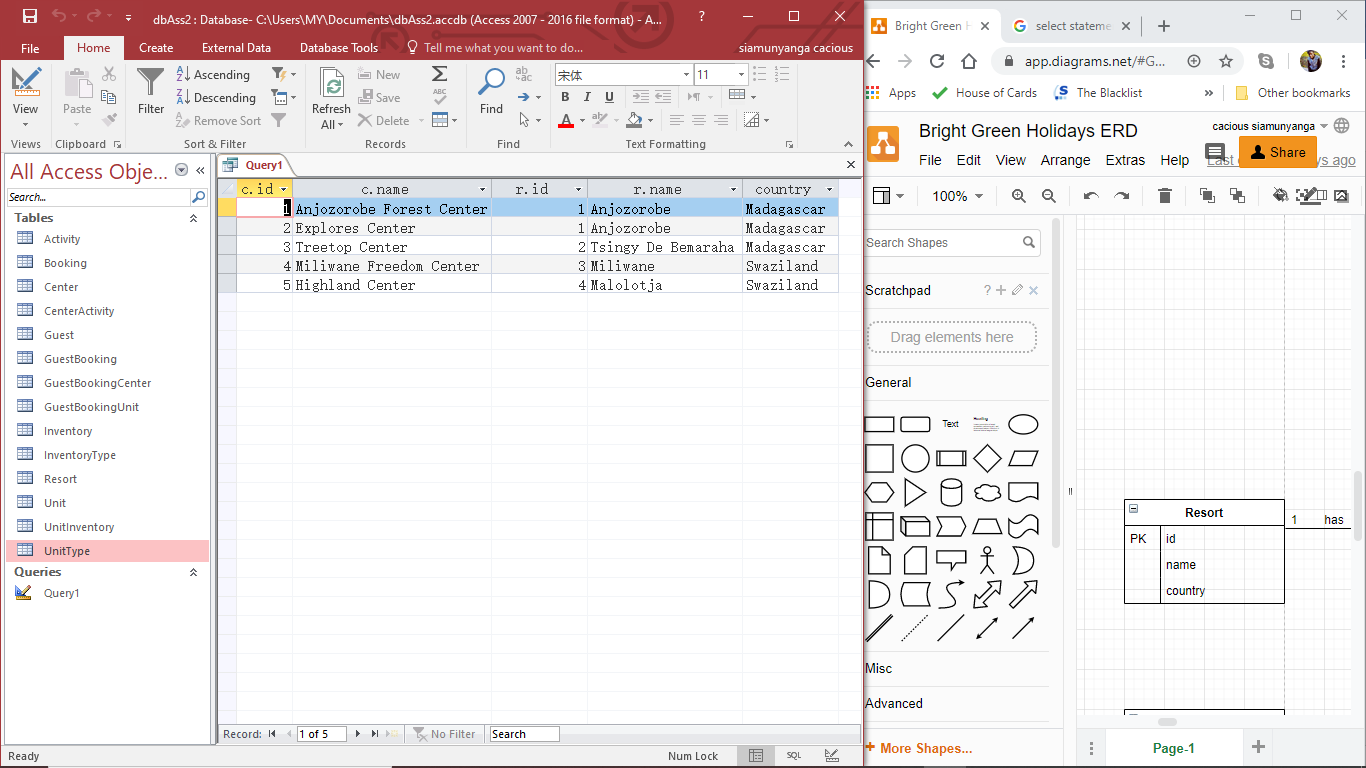


* List data on all resorts, centres, units and unit types

SELECT c.id, c.name, r.id, r.name, r.country

FROM Resort r, Center c

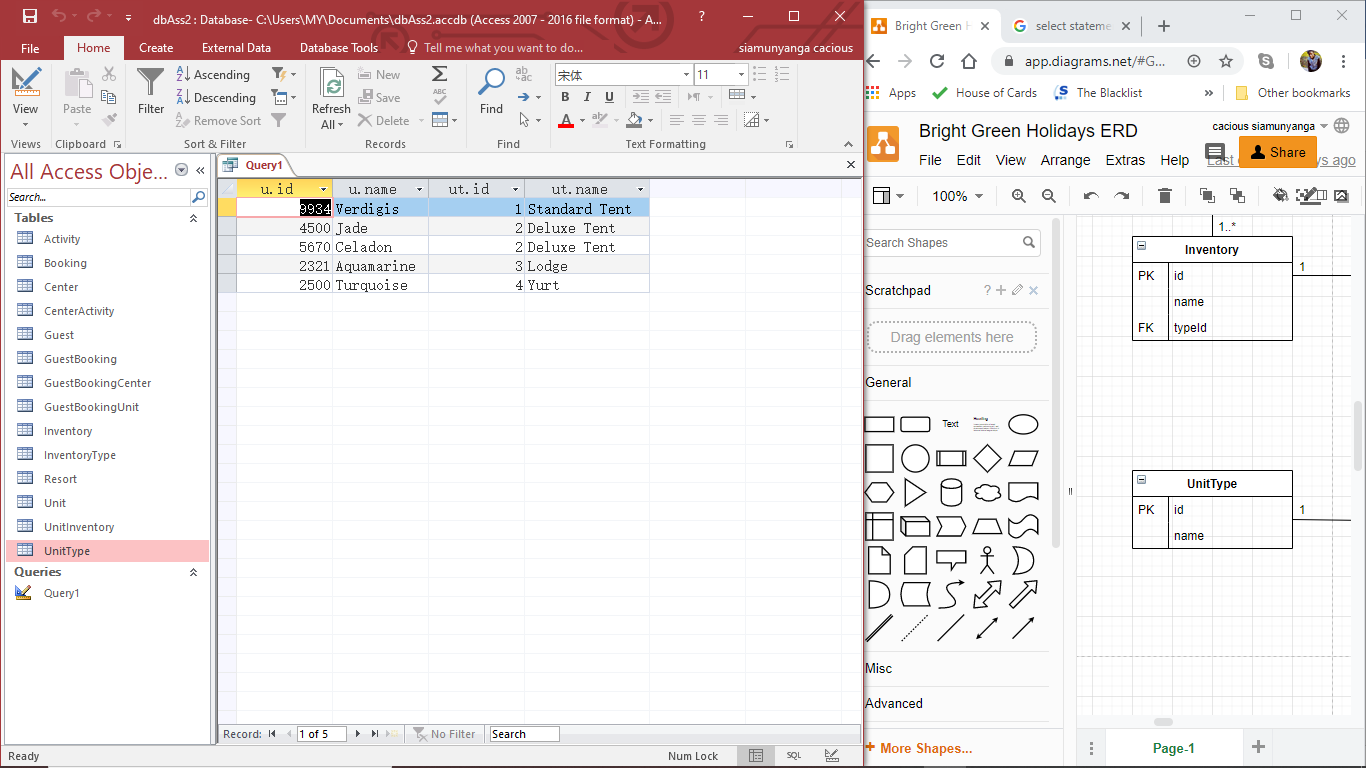
WHERE c.resortId = r.id



SELECT u.id, u.name, ut.id, ut.name

FROM Unit u, UnitType ut

WHERE u.typeId = ut.id



* List data on all inventory items for all units

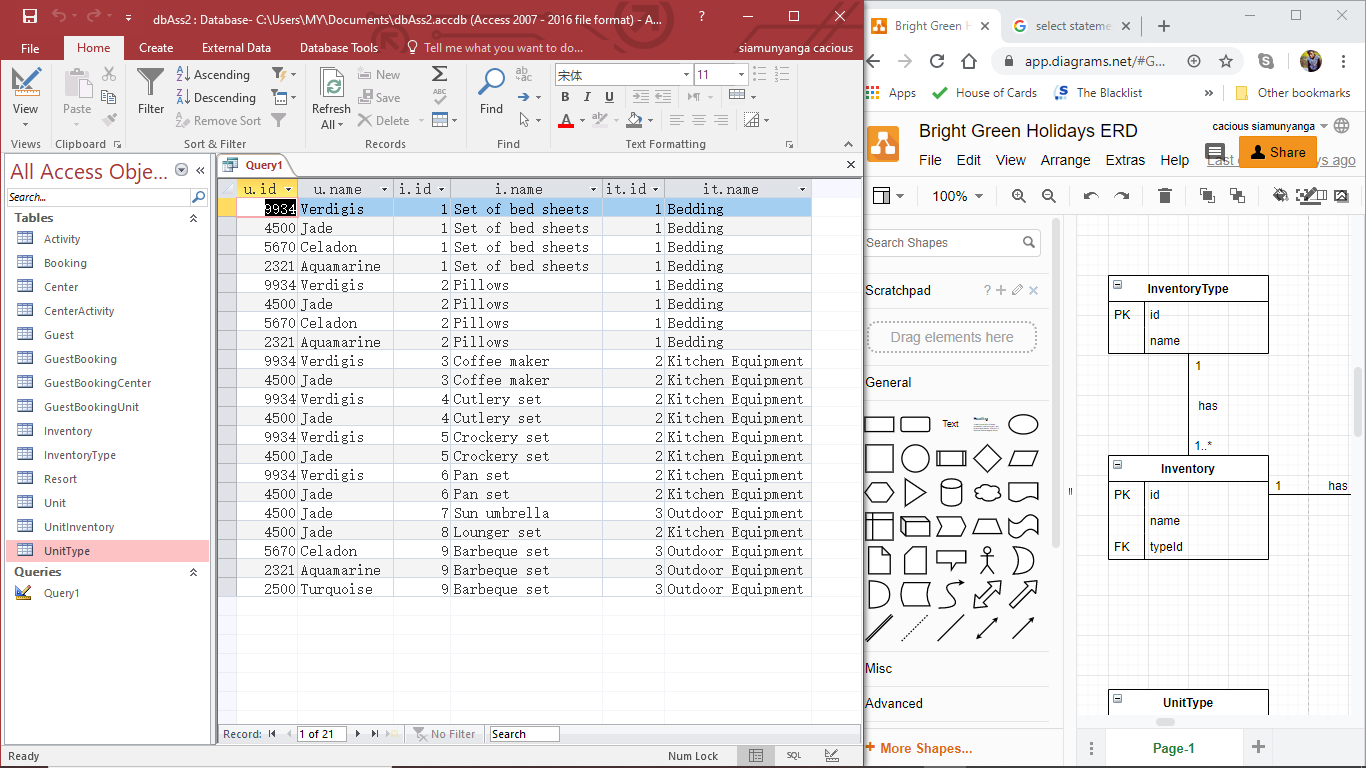
SELECT u.id, u.name, i.id, i.name,it.id, it.name

FROM Unit u, Inventory i, InventoryType it, UnitInventory ui

WHERE ui.unitId = u.id

AND ui.inventoryId = i.id

AND i.typeId = it.id



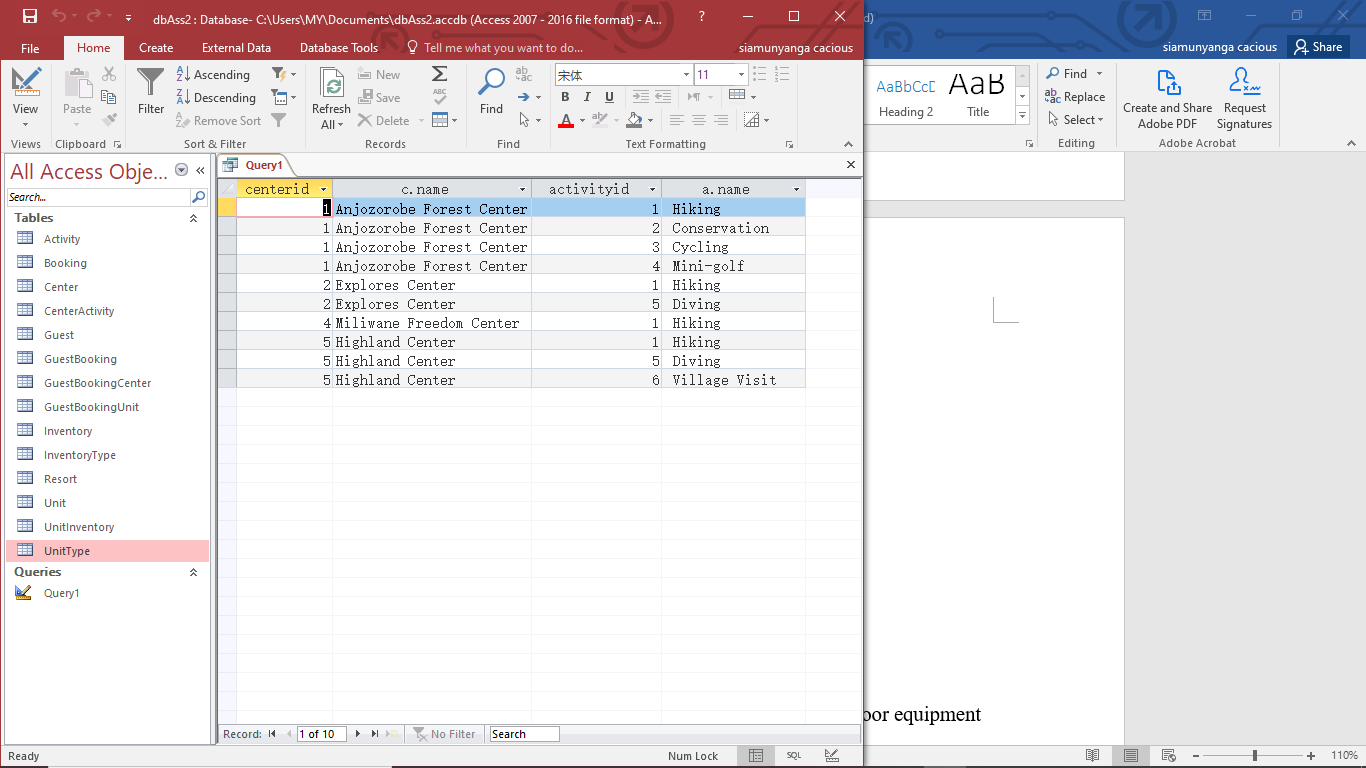
* List data on the activities found at all centres

SELECT ca.centerid, c.name, ca.activityid, a.name

FROM Center c, Activity a, CenterActivity ca

WHERE ca.centerId = c.id

AND ca.activityId = a.id



* Select the booking ID, guest ID and names for guests for booking B2001

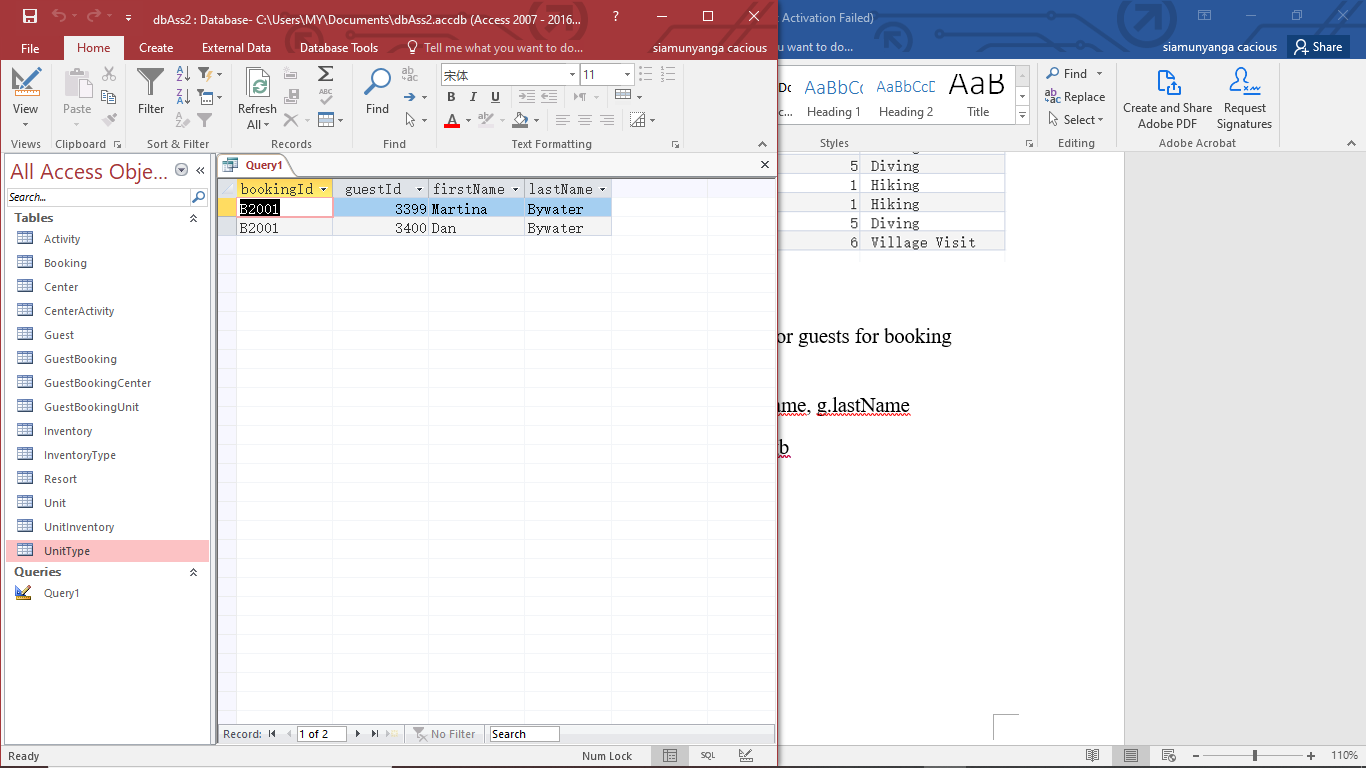
SELECT gb.bookingId, gb.guestId, g.firstName, g.lastName

FROM Booking b, Guest g, GuestBooking gb

WHERE gb.guestId = g.id

AND gb.bookingId = b.id

AND gb.bookingId = 'B2001'

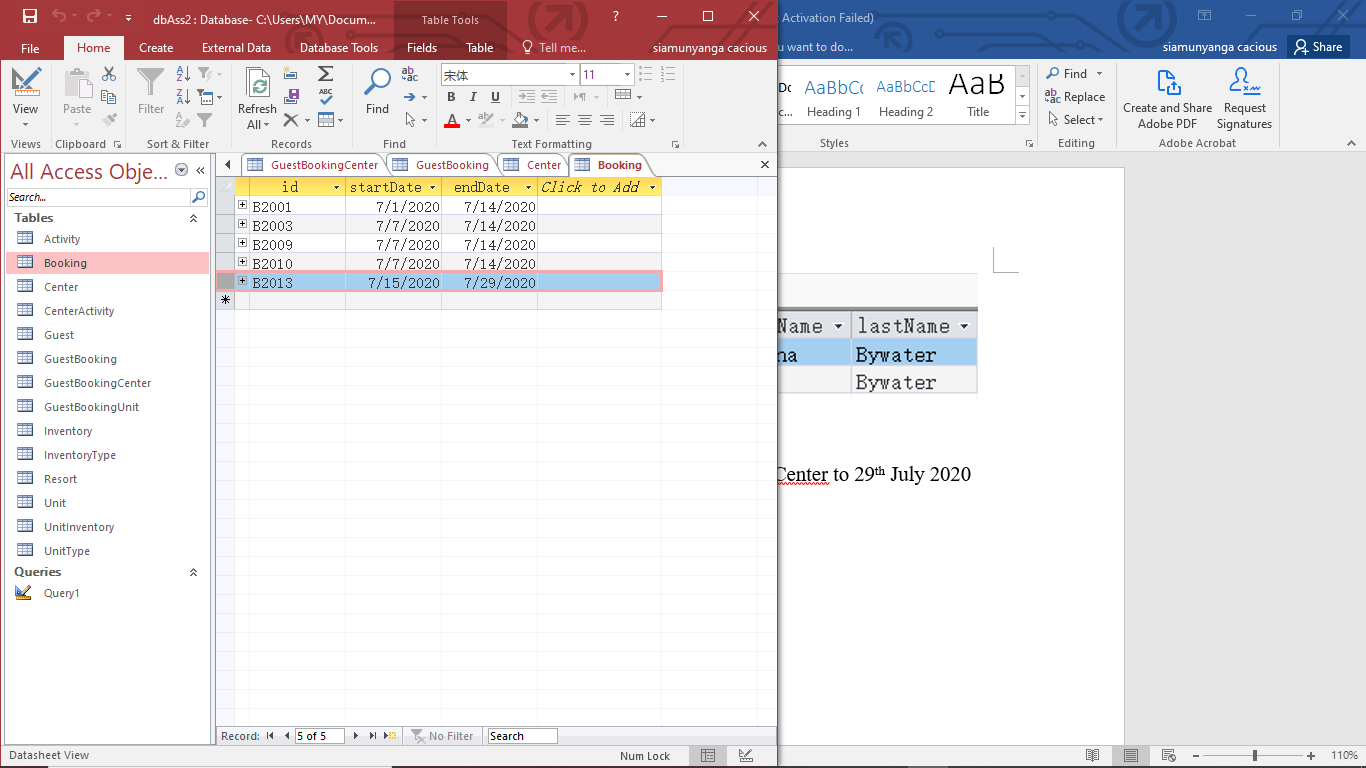


* Change Ameer Akhtar’s end date Highland Center to 29th July 2020

UPDATE Booking b

SET b.endDate = '7/29/2020'

WHERE b.id = 'B2013'



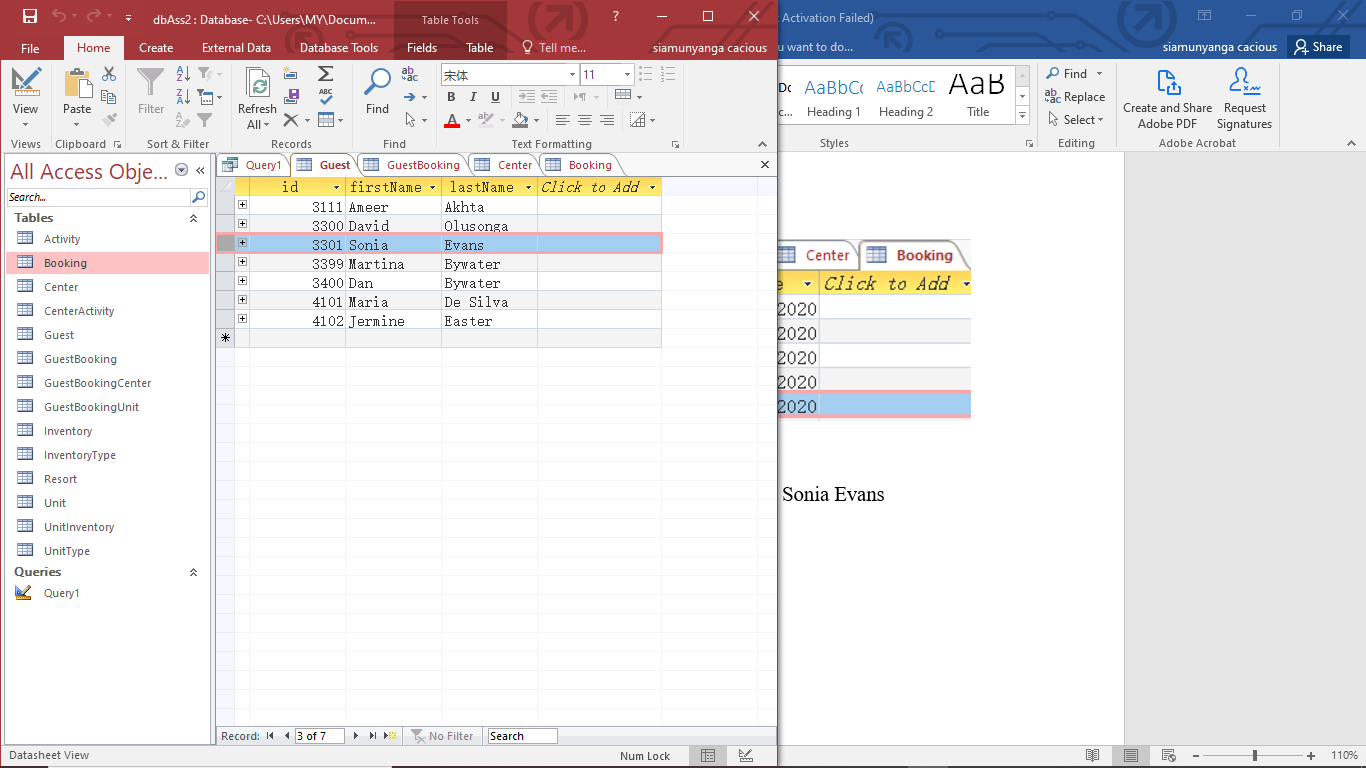
* Change the name of guest Sonia Chaplow to Sonia Evans

UPDATE Guest g

SET lastName = 'Evans'

WHERE g.firstName = "Sonia"

AND g.lastName = "Chaplow"



* List all inventory items for all the deluxe tents

SELECT u.id, u.name, ut.id, ut.name, i.id, i.name

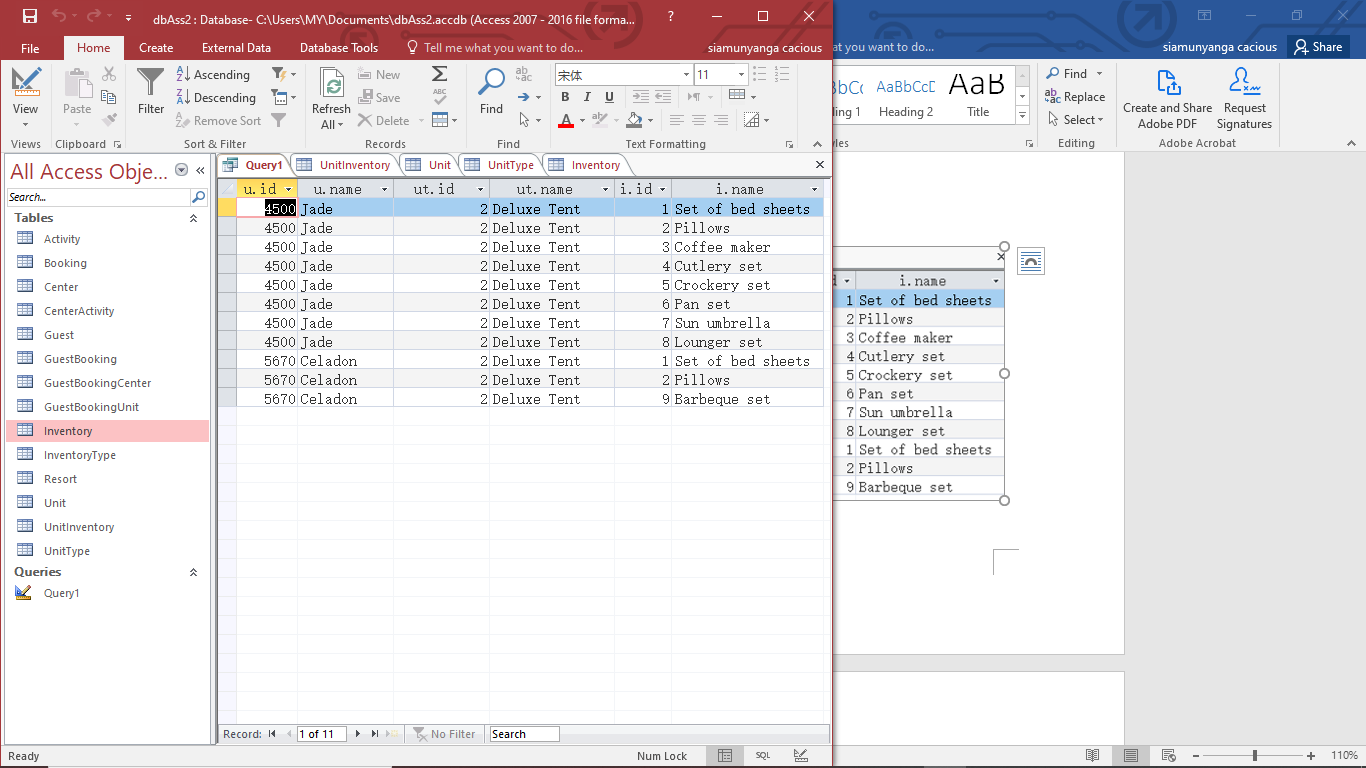
FROM Inventory i, Unit u, UnitType ut, UnitInventory ui

WHERE ui.unitId = u.id

AND ui.inventoryId = i.id

AND u.typeId = ut.id

AND ut.name = "Deluxe Tent"



* List all the guests who are staying at Anjozorobe Forest Centre and which units they are staying at

SELECT g.id, g.firstName, g.lastName, c.name, u.id, u.name

FROM Guest g, GuestBooking gb, GuestBookingCenter gbc, GuestBookingUnit gbu, Unit u, Center c

WHERE gb.guestId = g.id

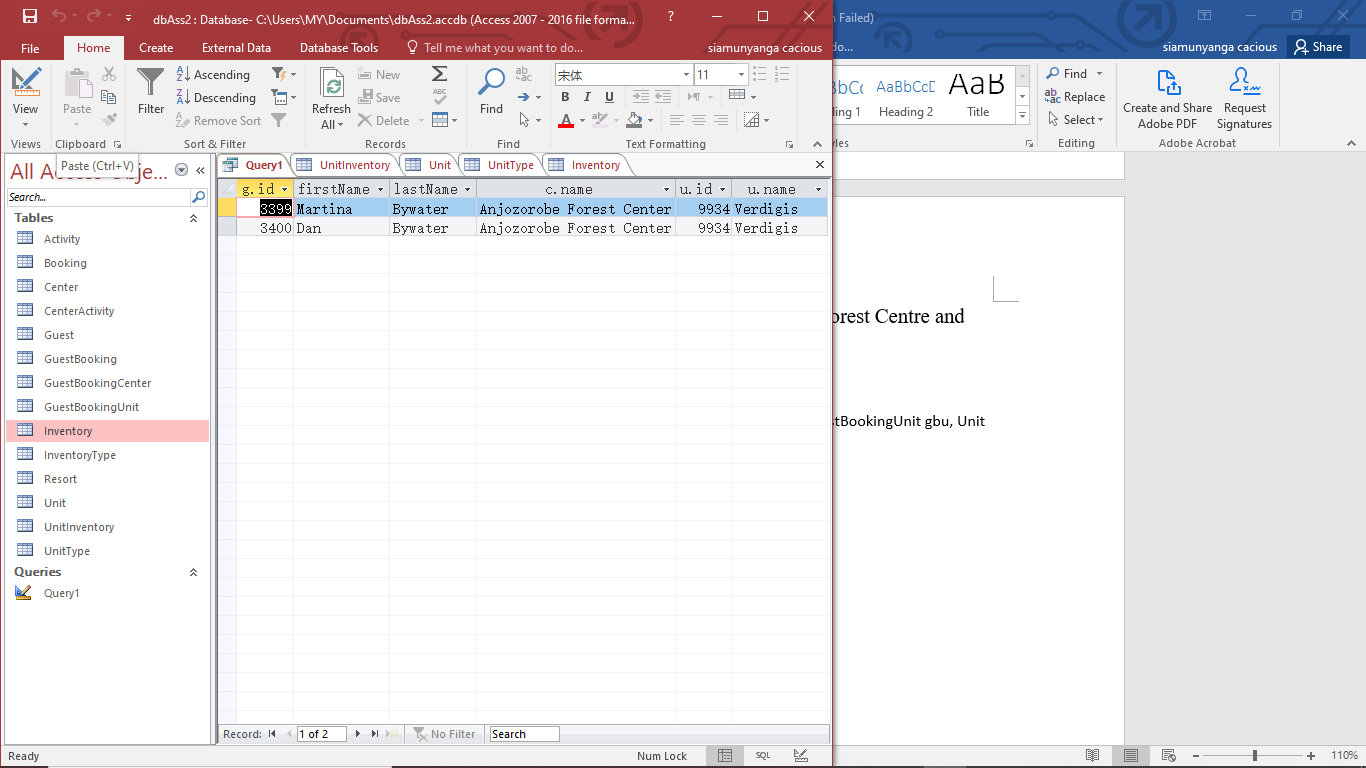
AND gbc.guestBookingId = gb.id

AND gbc.centerId = c.id

AND gbu.guestBookingId = gb.id

AND gbu.unitId = u.id

AND c.name = "Anjozorobe Forest Center"



* List the details of all the bookings at Highland centre including the dates of the bookings

SELECT b.id, b.startDate,b.endDate ,g.id, g.firstName, g.lastName, c.name as CenterName

FROM Guest g, GuestBooking gb, GuestBookingCenter gbc,Center c, Booking b

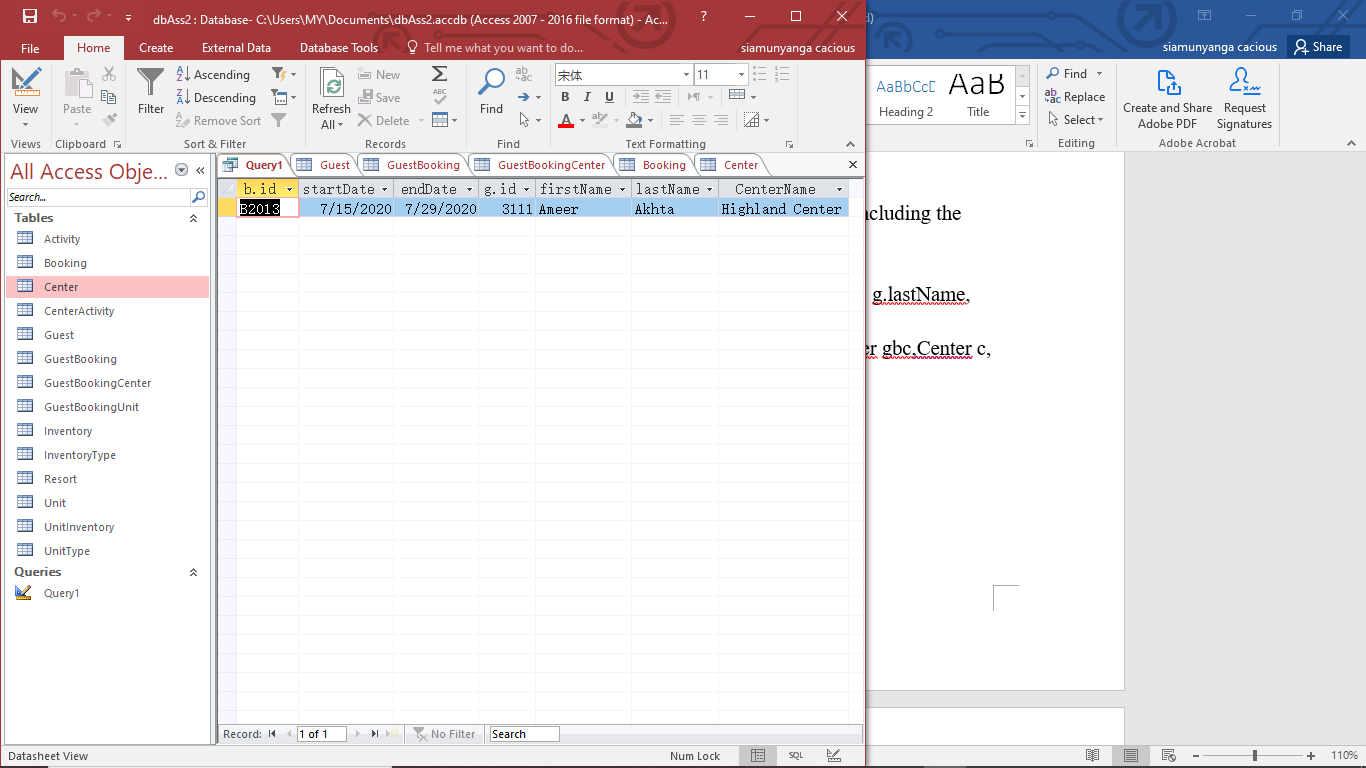
WHERE gb.guestId = g.id

AND gb.bookingId = b.id

AND gbc.guestBookingId = gb.id

AND gbc.centerId = c.id

AND c.name = "Highland Center"



Task 3

* Firstly, the Inventory table should be altered to add a replacement cost field. This would allow the management and staff to know exactly how much each inventory item costs. The table should also be altered in such a manner that it includes more items from the units that could get damaged and need replacement at some time in the future. Such as bulbs and so on. In short, the Inventory table, should account for every element/item in every unit and taking account their replacement costs.
* Add a Staff and Role table to the database. The Staff table should contain the id as primary key, first and last name, role id as foreign key and center id as another foreign key. The Role table on the other hand should have id as primary key, name of the role and a pay per month field. The purpose of the pay per month field is to be able to allocate how much each role is worth on a monthly basis. Considering that the relationship between the Staff and Role table is that, *a staff can have only one role and that a role can belong to many staff,* it means that the pay per month field in the Role table would automatically specify the staff pay rate as long as the role id field in the Staff table has a not null constraint and references the id field in the Role table correctly. It is also important to mention that the relationship between the Staff and Center table is that, *each staff belongs to only one center and a center can have multiple staff.*
* Finally, to calculate how much each work at each unit and center costs, the following instructions would have to be followed.
  + **Work cost per unit:** add the pay per month of all staff with an associated role name of “unit maintenance” at a center and divide the sum by the number of units in the center.
  + **Work cost per center:** add the pay per month of all staff at a center.

TASK 4

* A distributed database is one in which data is stored in databases that are geographically separate. In the scenario given, the companies being taken over already have their own databases. Therefore, to redesign or relocate the databases could prove troublesome. Furthermore, I suggest that the data should remain in the locations that it already is and the company should instead use a Distributed Database Management System to manipulate the databases and keep them synchronized to appear and behave as one.
* The advantages and disadvantages of the distributed database compared to a centralized database are listed below:
  + **Advantages:** 
    - **Modular development:** If a database needs to be expanded, a distributed database only needs addition of new hardware and data at the local site and connecting it to the rest of the database system without interrupting the normal functions of it. However, with a centralized database, much more effort is required to restructure the database and this would definitely interrupt the normal function of it.
    - **Higher reliability:** In case of failure, the whole centralized database will cease operation but a distributed database will continue operating if one of its databases fails. Maybe the performance might reduce, depending on configurations and uses, but it will still remain functional. Hence making it more reliable.
    - **Faster response:** If data is well distributed, then queries from users can be handled from local sites or sites closest to the user, making the responses much faster. In centralized databases, data always has to go to the main and central database computer before it can respond. This could lead to slow response times if it has to respond to customers in geographically distant areas.
  + **Disadvantages:**
    - **Expensive and complex software:** distributed databases management systems require expensive and complex software to maintain coordination and data transparency across the different sites.
    - **Processing Overhead:** there can easily be a processing overhead issue considering that small operations including large ones need a large number of calculations and operations to keep harmonized data across databases.
    - **Data integrity:** updating and altering data at various sites causes a potential risk of data integrity. This is always a worrying item with distributed databases.

TASK 5

The company’s requirements have been met as is evident from the system created. The database tables are well created and designed. The relationships and constraints prevent usual data entry errors from occurring such as the ones noticeable in their previous tables. The tables are designed in such a way as to reduce redundancy and still stay efficient in data storage and management. Due to the normalization of the database, it creates much more tables than was previously in existence. However, the company’s requirement is to be able store data for them to efficiently run their business. The database is built in such a way that queries can be created, like the ones created for this project that will easily put the data together for much easier management without compromising on the quality of the data storage itself. Thereby allowing Bright Green Holidays to run a business with an efficient data storage system for all their needs.

By breaking their data down and creating much more clear entities, I was able to create constraints throughout the database that strive to only keep relevant data in. This in turn standardized the data being stored in the database. In short, for the data to be stored in the designed database, it has to abide by certain standards that are being enforced by the relationships and constraints built into the database.

Assumptions

* Booking can have many guests and guests can belong to many bookings, in case they make more bookings as returning customers/guests.
* Since centres can have many activities, I have also assumed that those activities can belong to multiple centres.
* A guest may stay at a particular centre during a specific booking and they may change centres during that booking for a specific period.
* A guest may change units for a specific period during the same booking. If they change centres, they may or may not change units.
* A guest of the same booking may stay in different units and centres as desired.
* pan set is a kitchen equipment while sun umbrella is an outdoor equipment
* Milwane was misspelled and should instead be Miliwane.
* The time a guest registered in a centre and a unit is at 10:00:00 AM of the date they made the booking and the time they terminated their registration at that centre and unit is also at 10:00:00 AM
* Since Booking Details table given is only showing one boking id, I assume that it is showing the unit they checked in is exactly the same in case there are multiple users. However, due to my assumption that different guests can have the same booking but different units if desired, I will show the unit a guest checked in. But each record entered will be specific to that guest using the GuestBooking table id column I created. I believe this will cut out the ambiguity.