ECE 656 – Winter 2015 Assignment #2

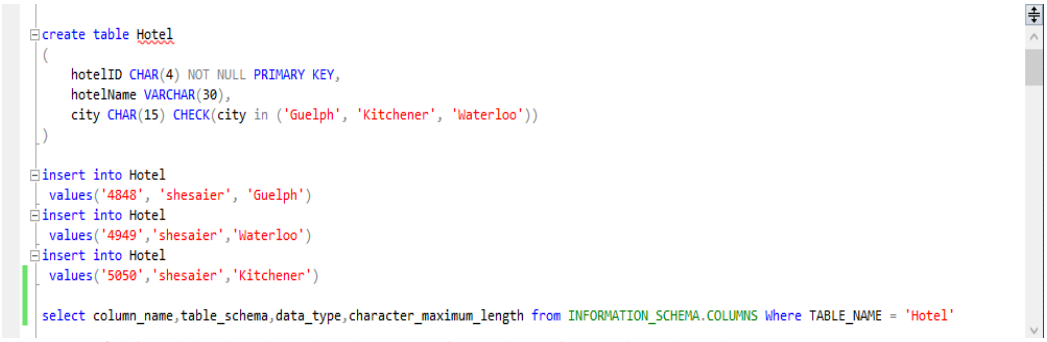
Zeya Liu(20551072),Anqi Yao (20557257), Hanning Zhu (20564432),Yuantao Ji (20511320)

Part 1. SQL DDL and Advanced SQL Applications

In this part, we used Microsoft SQL Server as implementation of the DBMS.
Firstly, we created a database named HotelDB.

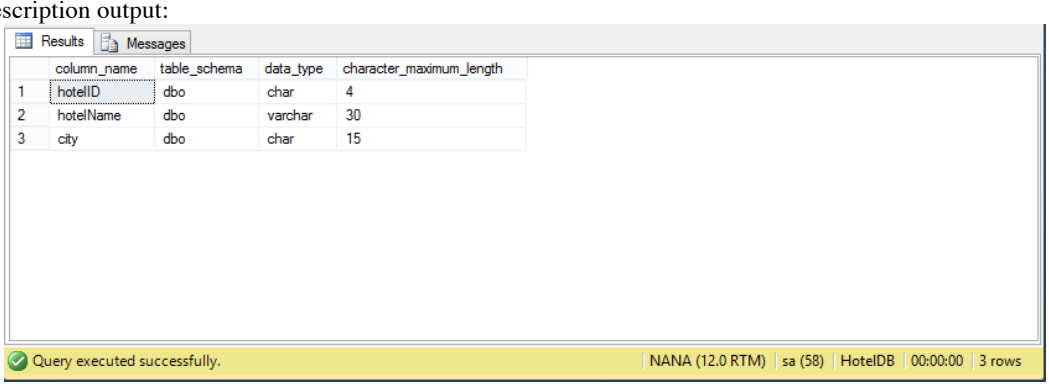
The tables we created are as below:

1) Table Hotel



```
create table Hotel
(
    hotelID CHAR(4) NOT NULL PRIMARY KEY,
    hotelName VARCHAR(30),
    city CHAR(15) CHECK(city in ('Guelph', 'Kitchener', 'Waterloo'))
)

insert into Hotel
 values('4848', 'shesaier', 'Guelph')
insert into Hotel
 values('4949','shesaier','Waterloo')
insert into Hotel
 values('5050','shesaier','Kitchener')

select column_name,table_schema,data_type,character_maximum_length from INFORMATION_SCHEMA.COLUMNS Where TABLE_NAME = 'Hotel'
```

The parameters of this table: hotelID, hotelName, city. The primary key: hotelID.
The Description output:



Results | Messages

	column_name	table_schema	data_type	character_maximum_length
1	hotelID	dbo	char	4
2	hotelName	dbo	varchar	30
3	city	dbo	char	15

Query executed successfully. | NANA (12.0 RTM) | sa (58) | HotelDB | 00:00:00 | 3 rows

2) Table Room

a2 creat table.sql -...NA.HotelDB (sa (51))

```
create table Room
(
    hotelID CHAR(4) NOT NULL,
    roomNo CHAR(4) NOT NULL,
    price DECIMAL(10,2) CHECK(price>=50.00 and price<=250.00),
    type CHAR(6) CHECK(type in ('Single','Double', 'Queen', 'King')),

    PRIMARY KEY (hotelID, roomNo),
    FOREIGN KEY (hotelID)
        REFERENCES Hotel(hotelID)
)

insert into Room values('4848','1001','50.00','Single')
insert into Room values('4848','1002','50.00','Single')
insert into Room values('4848','1003','50.00','Single')
insert into Room values('4848','2001','80.00','Double')
insert into Room values('4848','2002','80.00','Double')
insert into Room values('4848','2003','80.00','Double')
insert into Room values('4848','3001','150.00','Queen')
insert into Room values('4848','3002','150.00','Queen')
insert into Room values('4848','3003','150.00','Queen')
insert into Room values('4848','4001','250.00','King')
insert into Room values('4848','4002','250.00','King')
insert into Room values('4848','4003','250.00','King')
insert into Room values('4949','1001','50.00','Single')
insert into Room values('4949','1002','50.00','Single')
insert into Room values('4949','1003','50.00','Single')
insert into Room values('4949','2001','80.00','Double')
insert into Room values('4949','2002','80.00','Double')
insert into Room values('4949','2003','80.00','Double')
insert into Room values('4949','3001','150.00','Queen')
insert into Room values('4949','3002','150.00','Queen')
insert into Room values('4949','3003','150.00','Queen')
insert into Room values('4949','4001','250.00','King')
insert into Room values('4949','4002','250.00','King')
insert into Room values('4949','4003','250.00','King')
insert into Room values('5050','1001','50.00','Single')
insert into Room values('5050','1002','50.00','Single')
insert into Room values('5050','1003','50.00','Single')
insert into Room values('5050','2001','80.00','Double')
insert into Room values('5050','2002','80.00','Double')
insert into Room values('5050','2003','80.00','Double')
insert into Room values('5050','3001','150.00','Queen')
insert into Room values('5050','3002','150.00','Queen')
insert into Room values('5050','3003','150.00','Queen')
insert into Room values('5050','4001','250.00','King')
insert into Room values('5050','4002','250.00','King')
insert into Room values('5050','4003','250.00','King')

--drop table Room
select column_name,table_schema,data_type,character_maximum_length from INFORMATION_SCHEMA.COLUMNS Where TABLE_NAME = 'Room'
```

The parameters of this table: hotelID, roomNo, type, price. The primary key: hotelID, roomNo.
The description output:

Results | Messages

	column_name	table_schema	data_type	character_maximum_length
1	hotelID	dbo	char	4
2	roomNo	dbo	char	4
3	price	dbo	decimal	NULL
4	type	dbo	char	6

Query executed successfully. | NANA (12.0 RTM) | sa (58) | HotelDB | 00:00:00 | 4 rows

3) Table Guest

```
create table Guest
(
    guestID CHAR(4) NOT NULL PRIMARY KEY,
    guestName VARCHAR(30),
    guestAddress VARCHAR(50),
    guestAffiliation VARCHAR(30)
)

insert into Guest values('0001','Kassey','10 King Street','WTO')
insert into Guest values('0002','Hannah','5 Union Street','IMF')
insert into Guest values('0003','Zoe','18 Yonge Street','IEEE')

select column_name,table_schema,data_type,character_maximum_length from INFORMATION_SCHEMA.COLUMNS Where TABLE_NAME = 'Guest'
```

The parameters of this table: guestID, guestName, guestAddress, guestAffiliation. The primary key: guestID.
The description output:

Results | Messages

	column_name	table_schema	data_type	character_maximum_length
1	guestID	dbo	char	4
2	guestName	dbo	varchar	30
3	guestAddress	dbo	varchar	50
4	guestAffiliation	dbo	varchar	30

Query executed successfully. NANA (12.0 RTM) | sa (58) | HotelDB | 00:00:00 | 4 rows

4) Table Booking

```
create table Booking
(
    hotelID CHAR(4) NOT NULL,
    roomNo CHAR(4) NOT NULL,
    guestID CHAR(4) NOT NULL,
    startDate DATE NOT NULL,
    endDate DATE,

    PRIMARY KEY (hotelID, roomNo, guestID, startDate),
    FOREIGN KEY (hotelID)
        REFERENCES Hotel(hotelID),
    FOREIGN KEY (hotelID, roomNo)
        REFERENCES Room(hotelID, roomNo),
    FOREIGN KEY (guestID)
        REFERENCES Guest(guestID)
)
insert into Booking values('4848','1003','0001','2015-03-15','2015-03-18')
insert into Booking values('4949','2001','0002','2015-03-10','2015-03-12')
insert into Booking values('5050','3001','0003','2015-03-16','2015-04-16')

select column_name,table_schema,data_type,character_maximum_length from INFORMATION_SCHEMA.COLUMNS Where TABLE_NAME = 'Booking'
```

The parameters of this table: hotelID, roomNo, guestID, startDate, endDate. The primary key: hotelID, roomNo, guestID, startDate.
The description output:

Results | Messages

	column_name	table_schema	data_type	character_maximum_length
1	hotelID	dbo	char	4
2	roomNo	dbo	char	4
3	guestID	dbo	char	4
4	startDate	dbo	date	NULL
5	endDate	dbo	date	NULL

Query executed successfully. NANA (12.0 RTM) | sa (58) | HotelDB | 00:00:00 | 5 rows

Step 3:

1) PSM

```
--populate the database with the information for the individual rooms in the area hotels
create procedure PopulateHotelRooms
(@hid CHAR(4), @noFloors INT, @noFRooms INT )
as
    declare @Floor INT;
    declare @Room INT;
    declare @RoomNo CHAR(4);

begin
        if @noFloors not between 1 and 99 return;
        if @noFRooms not between 1 and 99 return;

        set @Floor = 1;

        while @Floor <= @noFloors
        begin
            set @Room = 1;
            while @Room <= @noFRooms
            begin
                if @Floor<10
                    set @RoomNo = concat('0',cast(@Floor*100+@Room as CHAR(3)));
                else
                    set @RoomNo = cast(@Floor*100+@Room as CHAR(4));
            insert into Room values(@hid,@RoomNo,'50.00','Double');
            set @Room = @Room +1
            end

        set @Floor = @Floor +1;
        end
end;
exec PopulateHotelRooms '4848',2,4;
```

In this PSM PROCEDURE, we can insert the information for any individual rooms in certain hotel.
Using the input '4848', 2, 4 as an example (where '4848' is hotelID, 2 is number of floors, 4 is number of rooms per each floor), the output showed as below:

Results | Messages

	hotelID	roomNo	price	type
1	4848	0101	50.00	Double
2	4848	0102	50.00	Double
3	4848	0103	50.00	Double
4	4848	0104	50.00	Double
5	4848	0201	50.00	Double
6	4848	0202	50.00	Double
7	4848	0203	50.00	Double
8	4848	0204	50.00	Double

Query executed successfully. NANA (12.0 RTM) | sa (54) | master | 00:00:00 | 44 rows

According to this output, after executed the PSM PROCEDURE, all expected rooms were added to database with the specified format.

2) TRIGGERS

The **first trigger** we created is using for figure out the conflicts in booking function, which means when we want to book an unavailable room in this system, the system will produces an error message prompt user cannot do such operation as below:

a2 creat table.sql -...ANA.master (sa (54))* | psm & triggers.sql...ANA.master (sa (52))

```
create trigger BookingConfliction_Insert on Booking instead of insert
as
    declare @hotelID CHAR(4);
    declare @roomNo CHAR(4);
    declare @guestID CHAR(4);
    declare @startDate DATE;
    declare @endDate DATE;
    declare @error VARCHAR(10);

begin
    set @error='You cannot execute this operation!';
    select @hotelID=hotelID, @roomNo=roomNo, @guestID=guestID, @startDate=startDate,@endDate=endDate
    from inserted

    if  ( exists(
                select hotelID,roomNo
                from Booking
                where Booking.hotelID = @hotelID
                    and Booking.roomNo = @roomNo
                    and (
                            hotelID in (
                                    select hotelID
                                    from Booking
                                    where (startDate < @startDate and endDate > @endDate)
                                        or(startDate >= @startDate and startDate < @endDate)
                                        or (endDate > @startDate and endDate <= @endDate)
                                        )
                        and
                            roomNo in (
                                    select roomNo
                                    from Booking
                                    where (startDate < @startDate and endDate > @endDate)
                                        or(startDate >= @startDate and startDate < @endDate)
                                        or (endDate > @startDate and endDate <= @endDate)
                                     )
                        )
                )
        )
    begin
        raiserror ('Your booking is conflicting with our database!',8,1,@error);
        rollback transaction;
        return
    end;

    else
    begin
        insert into Booking(hotelID, roomNo, guestID, startDate, endDate) values (@hotelID,@roomNo,@guestID,@startDate,@endDate)
    end;
end;


--DROP TRIGGER BookingConfliction_Insert
```

This trigger is aimed to avoid that when new booking is conflicting with exist one. The prompt message when we try to insert an existing booking message is as below:

```
insert into Booking values('4848','0101','0001','20150101','20150102')
```

100 %

Messages

```
Your booking is conflicting with our database!
Msg 50000, Level 5, State 1
```

The **second trigger** we make aims to change booing information in the database system. When user want to change information for instance change the start booking date, this part will execute such operation as below:

```
create trigger BookingConfliction_Update on Booking instead of update
as
    declare @hotelID CHAR(4);
    declare @roomNo CHAR(4);
    declare @guestID CHAR(4);
    declare @startDate DATE;
    declare @endDate DATE;
    declare @oldstartDate DATE;
    declare @oldendDate DATE;
    declare @error VARCHAR(10);

begin
    set @error='You cannot execute this operation!';
    select @hotelID=hotelID, @roomNo=roomNo, @guestID=guestID, @startDate=startDate,@endDate=endDate
    from inserted
    select @oldstartDate=startDate,@oldendDate=endDate
    from DELETED

    delete from Booking where hotelID=@hotelID and roomNo=@roomNo and guestID =@guestID and startDate=@oldstartDate and endDate=@oldendDate
    if  (
            (
            exists(
                select hotelID,roomNo
                from Booking
                where Booking.hotelID = @hotelID
                    and Booking.roomNo = @roomNo
                    and (
                            hotelID in (
                                    select hotelID
                                    from Booking
                                    where (startDate < @startDate and endDate > @endDate)
                                        or (startDate >= @startDate and startDate < @endDate)
                                        or (endDate > @startDate and endDate <= @endDate)
                                            )
                        and
                            roomNo in (
                                    select roomNo
                                    from Booking
                                    where (startDate < @startDate and endDate > @endDate)
                                        or (startDate >= @startDate and startDate < @endDate)
                                        or (endDate > @startDate and endDate <= @endDate)
                                        )
                        )
                    )
            )
        )
    begin
        raiserror ('Your booking is conflicting with our database!',8,1,@error);
        rollback transaction;
        return
    end;

    else
    begin
        insert into Booking(hotelID, roomNo,guestID, startDate,endDate) values (@hotelID,@roomNo,@guestID,@startDate,@endDate)
    end;
end;
--DROP TRIGGER BookingConfliction_Update
```

For instance, in our existed database, we already have a booking in the period: 2015-01-01 to 2015-01-02, then we update our table using the command code:" UPDATE Booking set endDate='20150104' where hotelID='4848' and roomNo='0101'", which change the first record, the changing output is as below:

The original record is:

	hotelID	roomNo	guestID	startDate	endDate
1	4848	0101	0001	2015-01-01	2015-01-02
2	4848	1001	0006	2015-02-03	2015-02-04
3	4848	1003	0001	2015-03-15	2015-03-18
4	4949	2001	0002	2015-03-10	2015-03-12
5	5050	3001	0003	2015-03-16	2015-04-16

The update record is:

	hotelID	roomNo	guestID	startDate	endDate
1	4848	0101	0001	2015-01-01	2015-01-04
2	4848	1001	0006	2015-02-03	2015-02-04
3	4848	1003	0001	2015-03-15	2015-03-18
4	4949	2001	0002	2015-03-10	2015-03-12
5	5050	3001	0003	2015-03-16	2015-04-16

If we update a new record is conflict with rest several data records, there will produces an error message prompt user the booking operation is conflict with the database system, the output is as below.
The command code is" UPDATE Booking set endDate='20150107' where hotelID='4848' and roomNo='0101' and startDate='20150101'", the exist table is

	hotelID	roomNo	guestID	startDate	endDate
1	4848	0101	0001	2015-01-01	2015-01-04
2	4848	0101	0001	2015-01-05	2015-01-08
3	4848	1001	0006	2015-02-03	2015-02-04
4	4848	1003	0001	2015-03-15	2015-03-18
5	4949	2001	0002	2015-03-10	2015-03-12
6	5050	3001	0003	2015-03-16	2015-04-16

The system output is:

```
(1 row(s) affected)
Your booking is conflicting with our database!
Msg 50000, Level 5, State 1
```

Which shows the changing information operation cannot be executed.

3) Maintenance table

This table aims to record every change added to the table Booking, the source code is as follow:

```
create table Bookinglog
(
    hotelID CHAR(4) NOT NULL,
    roomNo CHAR(4) NOT NULL,
    guestID CHAR(4) NOT NULL,
    startDate DATE NOT NULL,
    endDate DATE,
    userID VARCHAR(30),
    changeTime DATETIME,

    PRIMARY KEY (hotelID, roomNo, guestID, startDate),
    FOREIGN KEY (hotelID)
        REFERENCES Hotel(hotelID),
    FOREIGN KEY (hotelID, roomNo)
        REFERENCES Room(hotelID, roomNo),
    FOREIGN KEY (guestID)
        REFERENCES Guest(guestID)
)
```

The trigger source code is:

```
CREATE TRIGGER Booking_Log ON Booking AFTER INSERT
AS
    DECLARE @hotelID CHAR(4);
    DECLARE @roomNo CHAR(4);
    DECLARE @guestID CHAR(4);
    DECLARE @startDate DATE;
    DECLARE @endDate DATE;

BEGIN
    SELECT @hotelID=hotelID, @roomNo=roomNo, @guestID=guestID, @startDate=startDate,@endDate=endDate
    FROM INSERTED

    INSERT INTO Bookinglog(hotelID, roomNo,guestID, startDate,endDate,userID,changeTime) VALUES
    (@hotelID,@roomNo,@guestID,@startDate,@endDate,CURRENT_USER,CURRENT_TIMESTAMP)
END
```

The description output is:

	COLUMN_NAME	TABLE_SCHEMA	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH
1	hotelID	dbo	char	4
2	roomNo	dbo	char	4
3	guestID	dbo	char	4
4	startDate	dbo	date	NULL
5	endDate	dbo	date	NULL
6	userID	dbo	varchar	30
7	changeTime	dbo	datetime	NULL

Based on the previous operation, the changed process were record as follow:

	hotelID	roomNo	guestID	startDate	endDate	userID	changeTime
1	4848	0101	0001	2015-01-01	2015-01-04	dbo	2015-03-23 19:19:54.390
2	4848	0101	0001	2015-01-05	2015-01-08	dbo	2015-03-23 19:25:53.437
3	4848	1001	0006	2015-02-03	2015-02-04	dbo	2015-03-23 18:00:29.047
4	4949	2001	0002	2015-03-10	2015-03-12	dbo	2015-03-23 16:51:04.637

Part 2. SQL API Applications

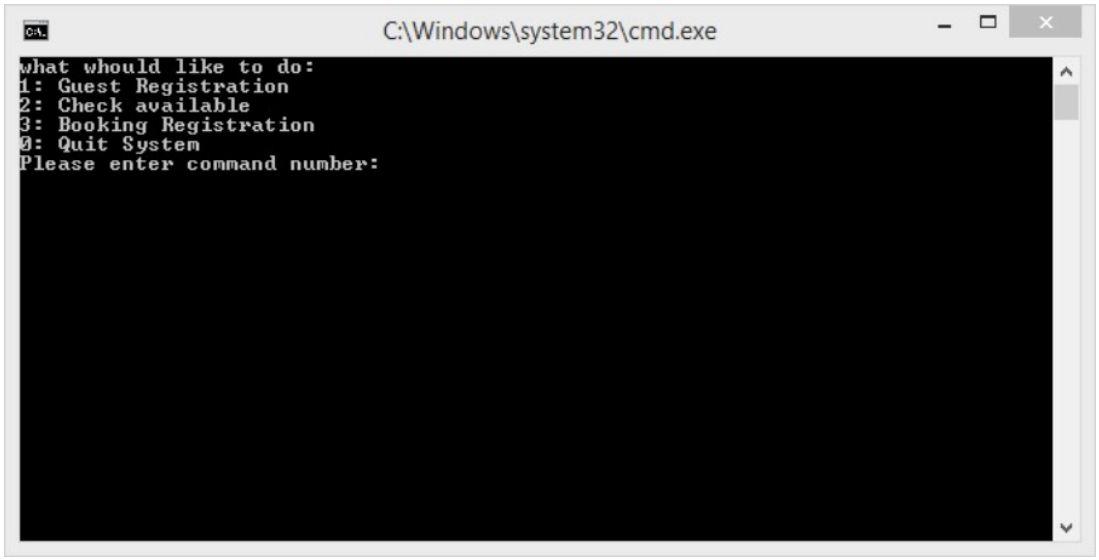
In this SQL API Applications we have three main functions to achieve the requirement. We have one main class named hotel system and four functionality classes, which are: the database_connection class, guest class, check_available class and booking_registration class.

The source codes and their operation windows are as below:

1. The main class hotel system:

```
12  namespace database_a2
13  {
14      class hotel_system
15      {
16          public static void Main(string[] args)
17          {
18              while (true)
19              {
20                  System.Console.WriteLine("what whould like to do:");
21                  System.Console.WriteLine("1: Guest Registration");
22                  System.Console.WriteLine("2: Check available ");
23                  System.Console.WriteLine("3: Booking Registration");
24                  System.Console.WriteLine("0: Quit System");
25                  System.Console.Write("Please enter command number: ");
26                  int i = Convert.ToInt32(System.Console.ReadLine());
27
28                  switch (i)
29                  {
30                      case 0: Console.Clear(); System.Console.WriteLine("Thank you for using our system."); break;
31                      case 1: guest.Registration_Guest(); break;
32                      case 2: check_available.Query_Booking(); break;
33                      case 3: booking_registration.Registration_Booking(); break;
34                      default: Console.Clear(); MessageBox.Show("Error: Wrong Command!"); break;
35                  }
36              }
37          }
38      }
39  }
```

The operation window:



2. database_connection class:

```
database_a2.database_connection                                  con
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Text;
 5  using System.Threading.Tasks;
 6  using System.Data.SqlClient;
 7  using System.IO;
 8  using System.Windows.Forms;
 9  using System.Threading;
10  using System.Data;
11
12  namespace database_a2
13  {
14      class database_connection
15      {
16          static SqlConnection con;
17          public static SqlConnection createCon()
18          {
19              con = new SqlConnection("server=kassey;uid=sa;pwd=sql2014;database=HotelDB");
20              return con;
21          }
22
23          public static SqlDataReader getRow(string sql)
24          {
25              SqlConnection con = createCon();
26              con.Open();
27              SqlCommand com = new SqlCommand(sql, con);
28              return com.ExecuteReader();
29              con.Close();
30          }
31
32
```

In this static we connect our c# project to our SQL Sever database management system, and we can also get the information from the database system.

```
database_a2.database_connection
31
32
33
34          public static bool execSQL(string sql)
35          {
36              SqlConnection con = createCon();
37              con.Open();
38              SqlCommand com = new SqlCommand(sql, con);
39              try
40              {
41                  com.ExecuteNonQuery();
42                  con.Close();
43              }
44              catch (Exception e)
45              {
46                  con.Close();
47                  return false;
48              }
49              return true;
50          }
51      }
52  }
53
```

In the second static, we execute the SQL in our program.

3. guest _registration class:

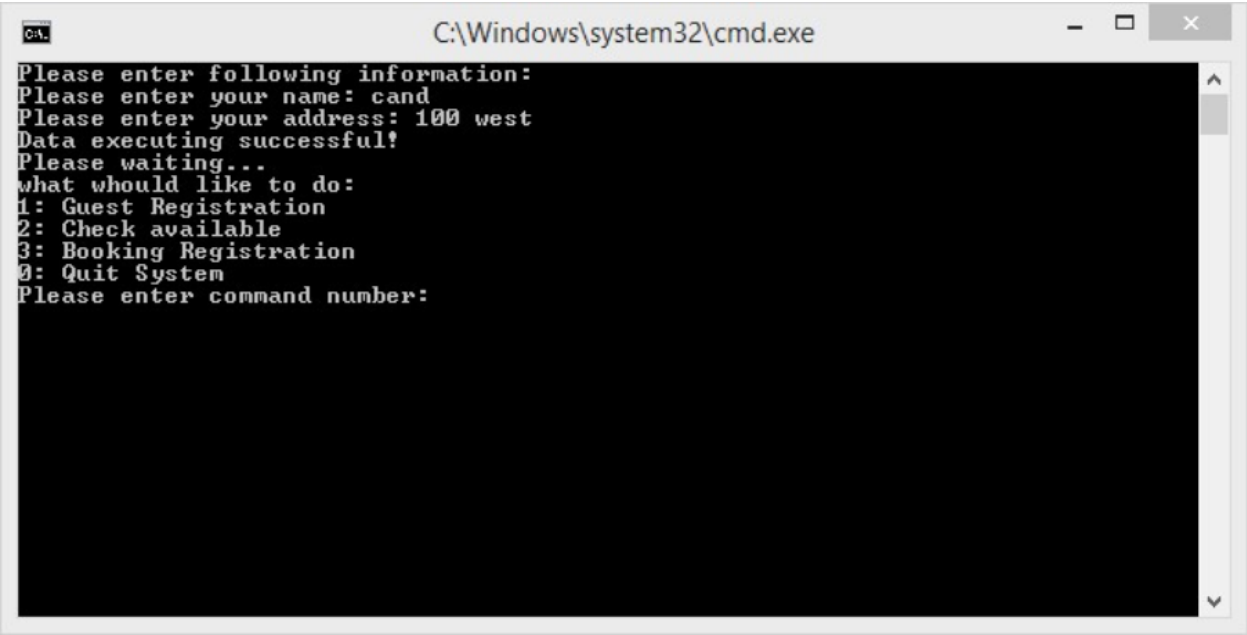
```
namespace database_a2
{
    class guest
    {
        public static string GenerateID()
        {
            string stmt = "SELECT COUNT(*) as sum FROM Guest";
            int count = 0;
            int ID;
            string guestID;

            SqlDataReader myreader = database_connection.getRow(stmt
            myreader.Read();
            count =Convert.ToInt32( myreader["sum"]);

            ID= count + 1;
            if(ID<10){
                guestID = "000" + Convert.ToString(ID);
            }
            else if(ID>=10 && ID<100){
                guestID="00"+Convert.ToString(ID);
            }
            else if (ID >= 100 && ID < 1000) {
                guestID = "0" + Convert.ToString(ID);
            }
            else {
                guestID =Convert.ToString(ID);
            }
            System.Console.WriteLine(guestID);
           return guestID;
        }
```

In this static we generate the guest ID by get the amount of original data form database, we use the amount of the record plus 1 to generate the new guest ID, then make the ID to the standard format like “0001” to “9999” with a fixed-length string of 4 characters.

```
        public static void Registration_Guest()
        {
            string GuestName;
            string GuestAddress;

            Console.Clear();
            System.Console.WriteLine("Please enter following information:");
            System.Console.Write("Please enter your name: ");
            GuestName = System.Console.ReadLine();
            System.Console.Write("Please enter your address: ");
            GuestAddress = System.Console.ReadLine();
            string GuestID = GenerateID();
            string GuestInsert = "insert into Guest values('" + GuestID + "','" + GuestName + "','" + GuestAddress + "','')";
            if (database_connection.execSQL(GuestInsert))
            {
                System.Console.WriteLine("Data executing successful!");
                System.Console.WriteLine("Please waiting...");
            }
            else
            {
                System.Console.WriteLine("Data executing failure!");
            }
        }
```

After generate the new guest ID for the customer, we insert the information include guest ID, guest name, and guest address to our database system for the table guest.

The operation window of this function is as follow:



The records changing in the database is as below, we add a new guest called cand:

	guestID	guestName	guestAddress	guestAffiliation
1	0001	Kassey	10 King Street	WTO
2	0002	Hannah	5 Union Street	IMF
3	0003	Zoe	18 Yonge Street	IEEE
4	0004	fiona	248,shagbark ct	
5	0005	cat	110,shagbark ct	
6	0006	candy	600 weststreet	
7	0007	cand	100 west	

4. check_available class:

```
namespace database_a2
{
    class check_available
    {
        public static void ExecuteData(string sql, string hn)
        {
            SqlDataReader myreader = database_connection.getRow(sql);
            int pdinfo = 0;
            while (myreader.Read())
            {
                pdinfo++;
                System.Console.WriteLine("\n{0}\n{1}\n{2}\n{3}\n{4}\n{5}\n{6}",
                    pdinfo, myreader["hotelID"], myreader["hotelName"], myreader["city"], myreader["roomNo"],
                    myreader["price"], myreader["type"]);
            }
            if (pdinfo == 0)
            {
                Console.Clear();
                MessageBox.Show("There is no available room in hotel:\t" + hn, "Notice");

            }
        }
```

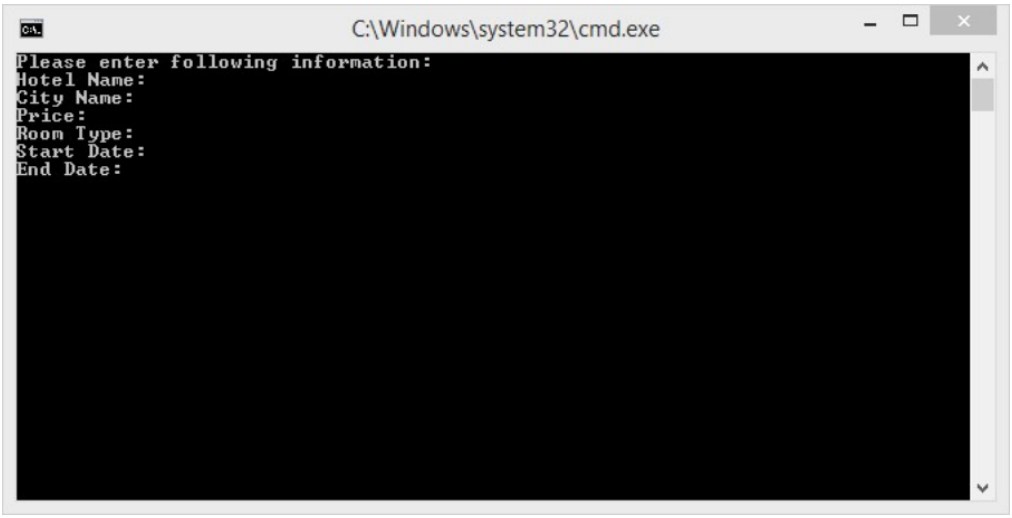
In this static, we list the records in database system, which satisfied the conditions that the user entered to the system. The conditions need to be entered are in the next static.

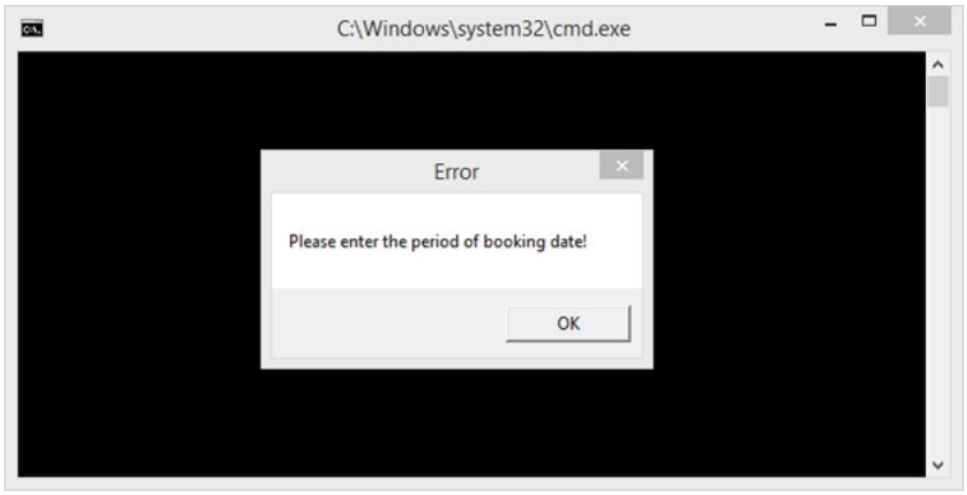
```
public static void Query_Booking()
{
    string startDate = "";
    string endDate = "";
    string hotelName = "";
    string city = "";
    string price = "";
    string type = "";

    Console.Clear();
    System.Console.WriteLine("Please enter following information:");
    System.Console.Write("Hotel Name: ");
    hotelName = System.Console.ReadLine();
    System.Console.Write("City Name: ");
    city = System.Console.ReadLine();
    System.Console.Write("Price: ");
    price = System.Console.ReadLine();
    System.Console.Write("Room Type: ");
    type = System.Console.ReadLine();
    System.Console.Write("Start Date: ");
    startDate = System.Console.ReadLine();
    System.Console.Write("End Date: ");
    endDate = System.Console.ReadLine();
```

```
database_a2.check_available                                    Query_Booking()
58          try
59          {
60              StringBuilder sqlall = new StringBuilder();
61              sqlall.Append("select h.hotelID,hotelName,city,roomNo,price,type from Hotel h,Room r where h.hotelID=r.hotelID ");
62
63              if (startDate.CompareTo("") != 0 && endDate.CompareTo("") != 0)
64              {
65                  if (hotelName.CompareTo("") != 0)
66                  {
67                      string sql1 = "and h.hotelName='" + hotelName + "'";
68                      sqlall.Append(sql1);
69                  }
70                  if (city.CompareTo("") != 0)
71                  {
72                      string sql1 = "and h.city='" + city + "'";
73                      sqlall.Append(sql1);
74                  }
75                  if (type.CompareTo("") != 0)
76                  {
77                      string sql1 = "and r.type='" + type + "'";
78                      sqlall.Append(sql1);
79                  }
80                  if (price.CompareTo("") != 0)
81                  {
82                      string sql1 = "and r.price='" + Convert.ToDouble(price) + "'";
83                      sqlall.Append(sql1);
84                  }
85                      string sql2 = "and concat(r.hotelID,r.roomNo) not in (select concat(hotelID,roomNo) from Booking where (startDate between
86                      sqlall.Append(sql2);
87                      ExecuteData(Convert.ToString(sqlall), hotelName);
88              }
89              else
90              {
91                  Console.Clear();
92                  MessageBox.Show("Please enter the period of booking date!","Error");
93                  return;
94              }
95          }
96          catch (Exception e)
97          {
98              Console.Clear();
99              MessageBox.Show("Error: Wrong!", "Notice");
100         }
101     }
102 }
103 }
104
```

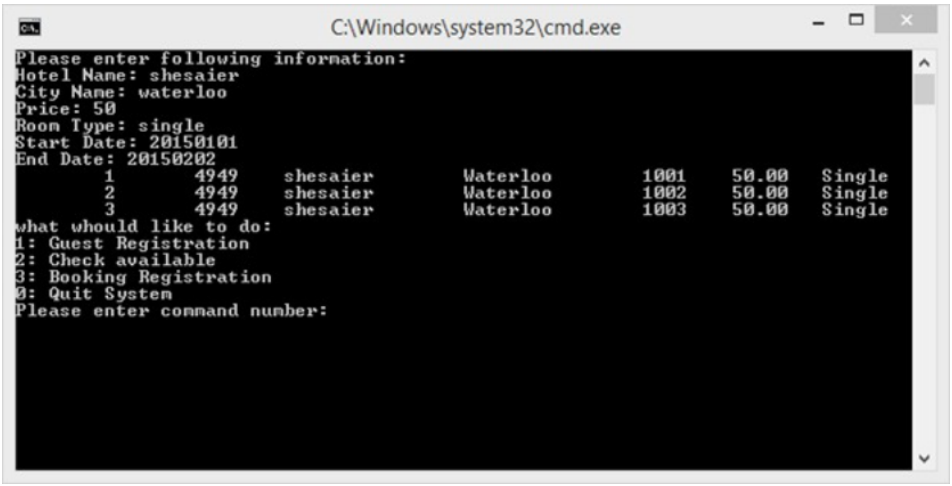
In this static, we search the records by using the restricted parameter such as the hotel name, the city name, the price of the room, the room type, and the start date and the end date. Then we use the entered parameters to search records in the database system. In this static, we use a logistic as the start date and end date is the restricted conditions, then when user enter one parameter, the system automatically add one parameter to the query language which connecting the system to database system, until the last parameter been added.

If the user does not enter the start date and end date, the system will produce a prompt message as below:

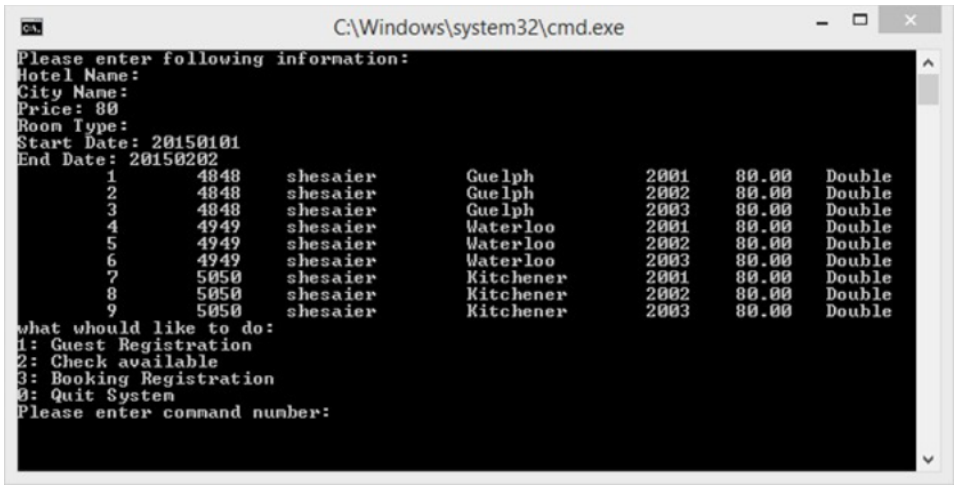




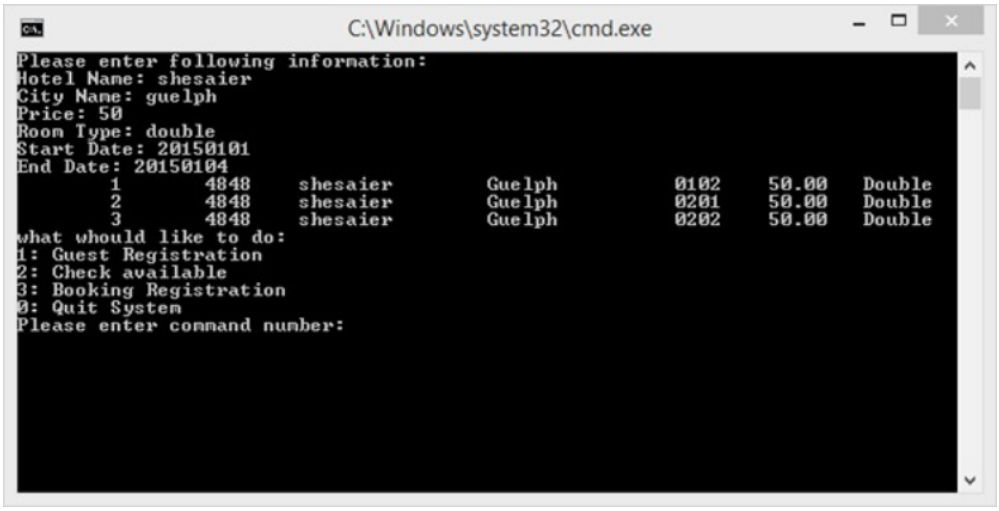
If the user enters all the information that system asked, the operation window is:



If the user only enters part of the required information, the system can still work well, the operation window is,



If the user enters conflict information, the system will only shows the available room but not conclude the conflict room:



In this example, we enter a conflict time period between “2015-01-01” to “2015-01-04” but there exist in the period from “2015-01-01” to “2015-01-02” for room “0101” in the database, so it not shown in our output.

	hotelID	roomNo	guestID	startDate	endDate
1	4848	0101	0001	2015-01-01	2015-01-02
2	4848	1001	0006	2015-02-03	2015-02-04
3	4848	1003	0001	2015-03-15	2015-03-18
4	4949	2001	0002	2015-03-10	2015-03-12
5	5050	3001	0003	2015-03-16	2015-04-16

	hotelID	roomNo	price	type
1	4848	0101	50.00	Double
2	4848	0102	50.00	Double
3	4848	0201	50.00	Double
4	4848	0202	50.00	Double

In the database system we have 4 rooms that satisfied the entered conditions, but room “0101” has been booked, so it did not shown in the result.

5. booking_registration class:

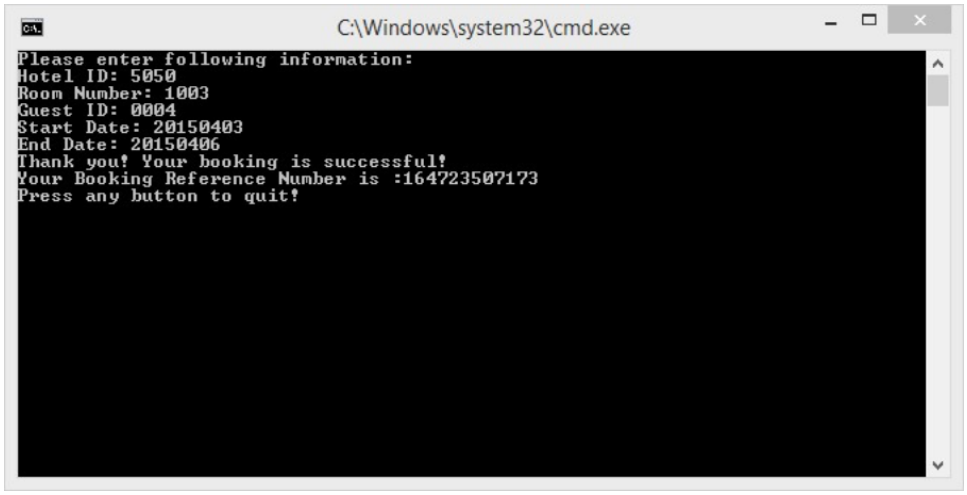
```
15      class booking_registration
16      {
17          public static string GenerateBookingID()
18          {
19              Random ro = new Random();
20              StringBuilder bookingID = new StringBuilder();
21              for (int i = 0; i <12 ; i++)
22              {
23                  bookingID.Append(ro.Next(0, 9));
24              }
25              return Convert.ToString(bookingID);
26          }
```

In this static, we create an 12-digits random number as booking ID which give to the customer for tracking the booking status.

```
27  public static void Registration_Booking()
28  {
29      string startDate = "";
30      string endDate = "";
31      string hotelID = "";
32      string guestID = "";
33      string roomNo = "";
34
35      Console.Clear();
36      System.Console.WriteLine("Please enter following information:");
37      System.Console.Write("Hotel ID: ");
38      hotelID = System.Console.ReadLine();
39      System.Console.Write("Room Number: ");
40      roomNo = System.Console.ReadLine();
41      System.Console.Write("Guest ID: ");
42      guestID = System.Console.ReadLine();
43      System.Console.Write("Start Date: ");
44      startDate = System.Console.ReadLine();
45      System.Console.Write("End Date: ");
46      endDate = System.Console.ReadLine();
47
48      if (startDate.CompareTo("") == 0 && endDate.CompareTo("") == 0 && hotelID.CompareTo("") == 0 && roomNo.CompareTo("") == 0 && guestID.CompareTo
49      {
50          Console.Clear();
51          MessageBox.Show("Please enter all the information!", "Error");
52          return;
53      }
54      try
55      {
56          if (startDate.CompareTo("") != 0 && endDate.CompareTo("") != 0 && hotelID.CompareTo("") != 0 && roomNo.CompareTo("") != 0 && guestID.Compa
57          {
58              string sql = "insert into Booking values('" + hotelID + "','" + roomNo + "','" + guestID + "','" + startDate + "','" + endDate + "')";
59              string booking_reference_num = GenerateBookingID();
60              if (database_connection.execSQL(sql))
61              {
62                  System.Console.WriteLine("Thank you! Your booking is successful!");
63                  System.Console.WriteLine("Your Booking Reference Number is :" + booking_reference_num);
64                  System.Console.WriteLine("Press any button to quit!");
65                  System.Console.ReadKey();
66                  Console.Clear();
67              }
68              else
69              {
70                  MessageBox.Show("There is no available room in hotel:\t" + hotelID + "\n time between:\t" + startDate + "\t and \t" + endDate, "No
71                  Console.Clear();
72              }
73              return;
74          }
75          else
76          {
77              Console.Clear();
78              MessageBox.Show("Error: All Input Cannot Empty!", "Error");
79              return;
80          }
81      }
82      catch (Exception e)
83      {
84          Console.Clear();
85          MessageBox.Show("Error: Wrong!", "Notice");
86      }
87  }
```

In this static, we will register a new room if the booking information does not conflict with the records in database system.

The operation window is:

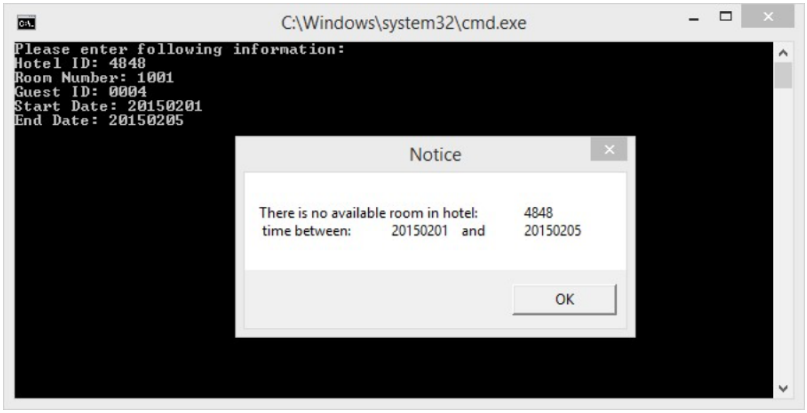


This window shows we add an new booking for guest 0004, the tracking ID is 164723507173,

	hotelID	roomNo	guestID	startDate	endDate
1	4848	0101	0001	2015-01-01	2015-01-04
2	4848	0101	0001	2015-01-05	2015-01-08
3	4848	1001	0006	2015-02-03	2015-02-04
4	4848	1003	0001	2015-03-15	2015-03-18
5	4949	2001	0002	2015-03-10	2015-03-12
6	5050	1003	0004	2015-04-03	2015-04-06
7	5050	3001	0003	2015-03-16	2015-04-16

This is the new booking record that adding to the database system.

If the booking information is conflict with the records in database system, the operation window will shows the prompt message as below:



If the customer did not enter all the information that the system required, the booking will failed, the system will produce a error message shows all input need to be not empty.