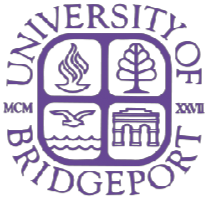
CPSC-551: Advance Database, Fall’2017

**Project Report**  
**UB Room Reservation System**

**Submitted by:**

Tanveen Kaur (1014603)

****

**Department of Computer Science and Engineering**

**University of Bridgeport,**

**Bridgeport, CT-06604**  
**USA**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **DESCRIPTION** | **PAGE NO.** |
| **1** | **Problem Statement** | **3** |
| **2** | **Introduction** | **1** |
| **3** | **Requirements** | **3-4** |
| **4** | **Project Features** | **5-6** |
| **5** | **Assumptions** | **6** |
| **6** | **EER MODEL** | **7** |
| **7** | **Technical Model** | **8** |
| **8** | **Relations** | **9-12** |
| **9** | **Relational Algebra** | **13-15** |
| **10** | **Database Server with Application** | **15** |
| **11** | **Physical Implementation (Scripts Source)** | **16-50** |
| **12** | **Application Development – SYSTEM DIAGRAM** | **51** |
| **13** | **Application Developemnt – SOURCE CODE** | **52-73** |
| **14** | **Conclusion** | **74** |

**Problem Statement**

The UB Room Reservation System will be used to reserve rooms in the different buildings of the University of Bridgeport by present Students or Faculty members based on their needs. It aims to provide an efficient and convenient way of reserving rooms for all the individuals involved. Not only just rooms, the faculty members or the students can also book an additional item like a computer or a projector in the room.

**Introduction**

The proposed system will make the task of reserving a room with different sizes and equipments or the possibility to choose between different date and times, very easy. The basic functionality of the system would be to keep track of the reservations of the rooms in different buildings, along with the additional items reserved with the room. The system will also keep track of the student’s or faculty’s feedback for the reservation. The system will have an admin user who will monitor or approve all the requests for the reservations and the normal user i.e a Student or a Faculty member, who will request the reservation, or further modify or cancel the reservation. The system overall aims to prevent the errors caused by humans as the reservation of the rooms in large buildings or campus becomes tedious, complicated and more error prone task.

**Requirements**

**Managing Users.**

There are two types of users: “**Admin**” and “**Normal**” Users. These two different users has different roles which are briefly explained below.

* + - **Admin User:*There is only one Admin User.***
  1. This user will monitor all the requests for the reservations of the rooms and other equipments or items.
  2. Also, this user can add, modify, delete the privileges of the normal user related to the reservations.
  3. This user can also cancel or modify any reservation of any normal user.
  4. This user will also add or delete the normal user, rooms and equipments in the system.
  5. Admin User can also send and receive messages to and from normal user regarding any issue on the reservation of the room or equipment.
  6. This user can also see any past, current or future status of reservations of the rooms and further, if need be, can purge the data/history or the reservations.
  + **Normal User: *There are more than one normal users.***
  1. This user can book, cancel or modify the reservation of the room or/and any equipment or item according to their needs by providing his/her details. Under this category are two kind of users i.e student and faculty.
  2. This user will have to sign into the system to make the reservation and if not already registered with the system, the user will first register himself with the system.
  3. This user can provide his/her feedback and can send or receive the message to and from the Admin user regarding any issue on the reservation.
  4. This user can view the status of the past, current or future reservations related to his/her account.

**Managing Rooms.**

1. The rooms will be reserved by the normal user upon a request and that request will be processed by the admin user.
2. The rooms will be searched based upon the building, floor, department or/and equipments. Also, the rooms will be available based on the date and time or according to the availability on the specific week or month.
3. The rooms can be added and deleted by the admin user according to the requirements.
4. Also, the part of the reservation or the whole reservation can be modified or cancelled by the normal user, if need be.
5. Rooms will categorized based upon the number of equipments it has and also based on the departments.

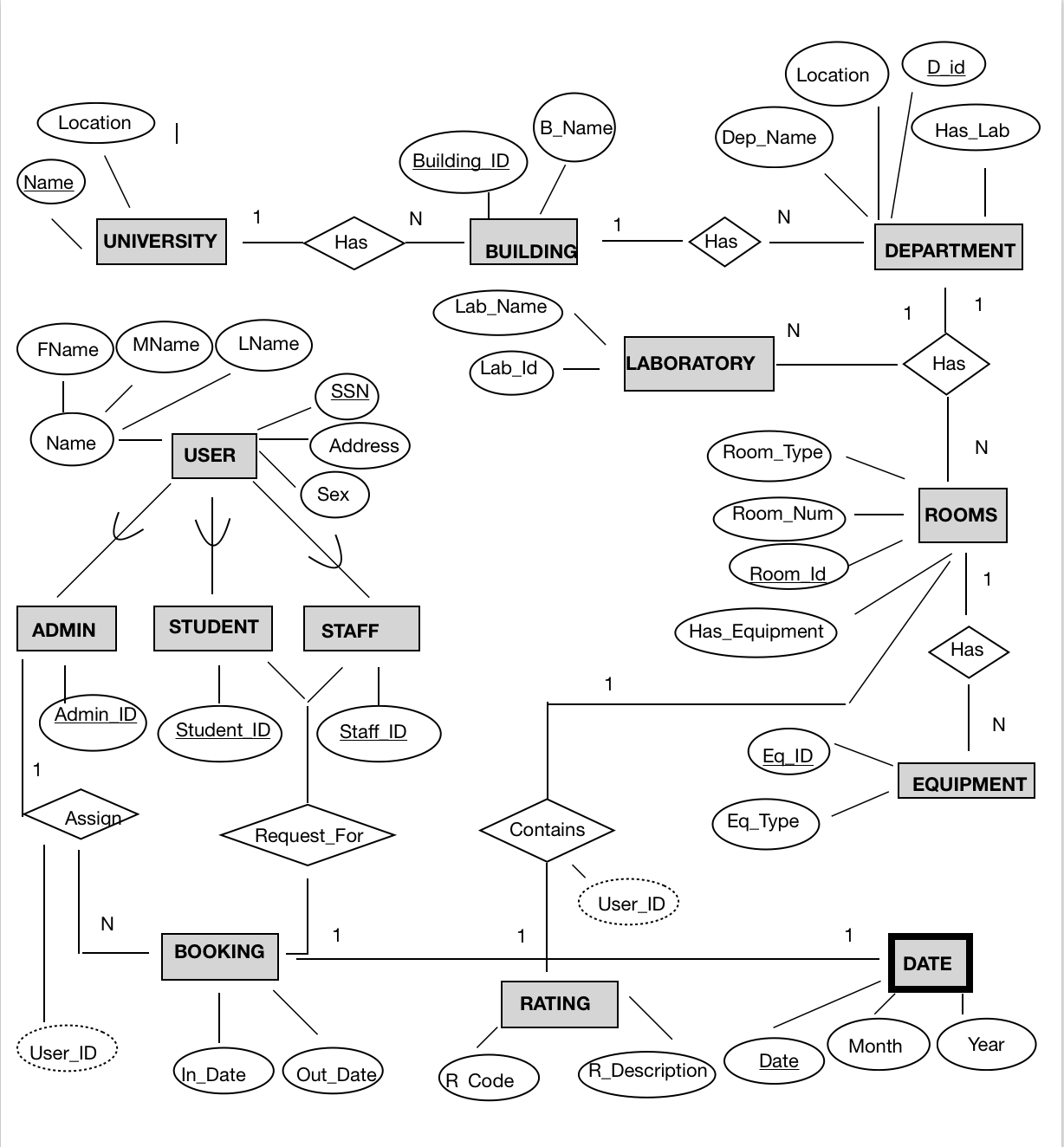
**Features (Realistic Queries)**

**These queries will be useful to somebody who will be using the data from the database.**

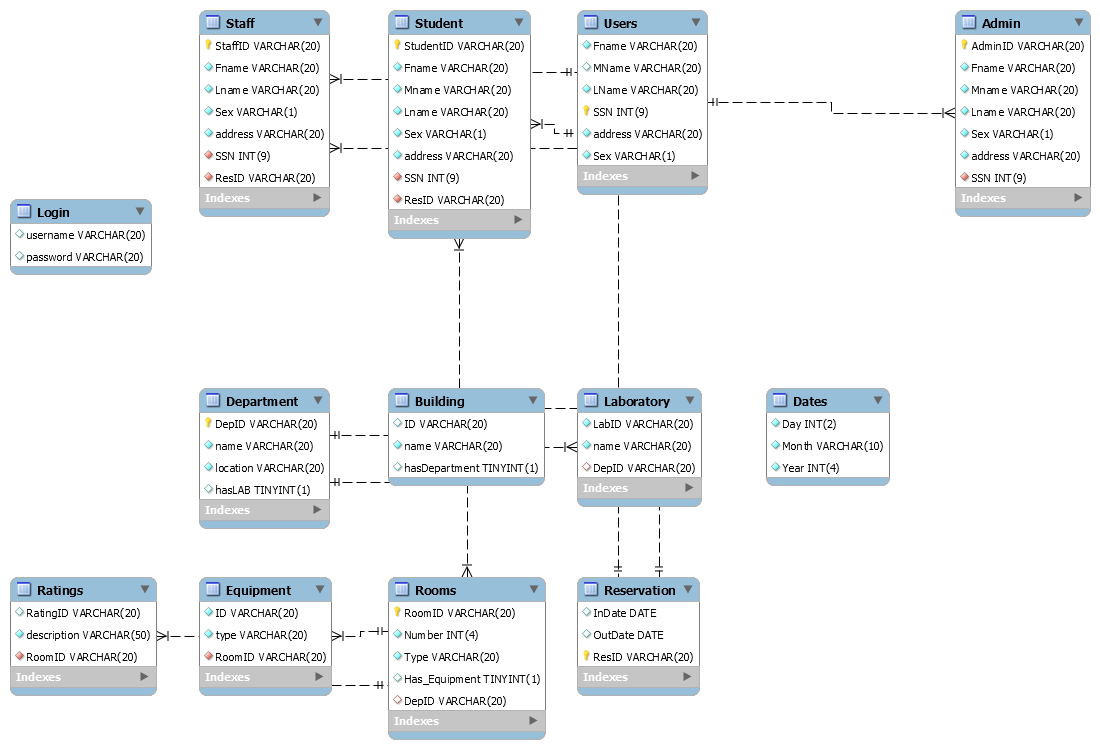
1. Select university name based on its location.
2. Select location of the university based on the building name it has.
3. Select building name based on the building ID or location.
4. Select department name based on the building name or building ID.
5. Select department location based on the department ID or department name.
6. Select room based on the department name or department ID.
7. Select department name if it has laboratory available for booking or not.
8. Select department name if it has rooms available for booking.
9. Select laboratory name based on the ID of the lab and department name.
10. Select rooms based on if it has equipment or not.
11. Select room number based on room type.
12. Select room type based on room number.
13. Select room based on the department name.
14. Select department name based on the room number or room ID.
15. Select equipment type based on Equipment ID.
16. Select equipment based on the room number that has it.
17. Select rating number when given room ID.
18. Select room ID and Room name when given rating ID.
19. Show ratings, room name when given rating ID.
20. Show ratings when given User ID.
21. Select if user is Admin, Student or Staff when given User ID.
22. Select first Name, Middle name and last one when given SSN of the user.
23. Select booking ID when given user ID to see if the user has any bookings.
24. Select In date and out date when given Booking ID.

**Assumptions**

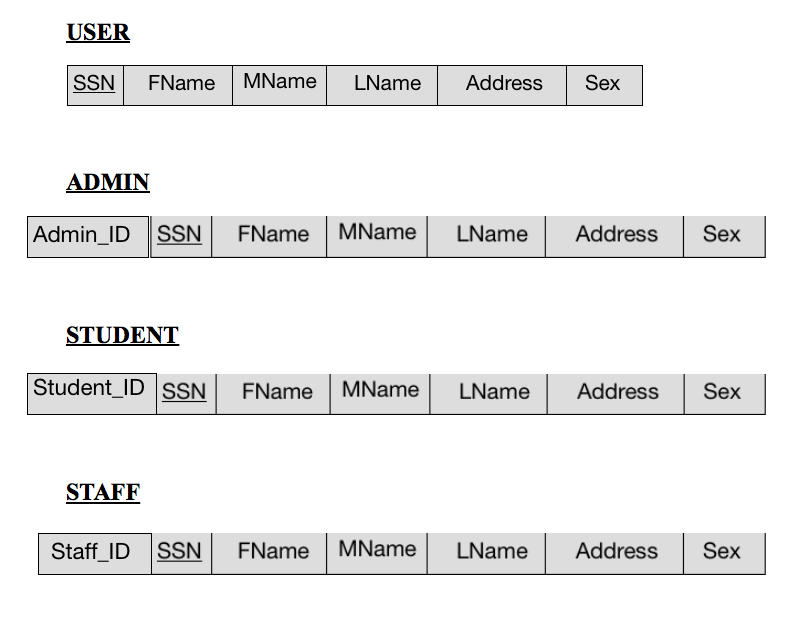
Talking about attributes, keys and the relationship between entities, the focus of the whole reservation system is on the rooms and the normal users who request the booking of the room, which is processed by the admin user who monitors the whole system. The entities may include USERS, DEPARTMENT, MANAGER, BUILDING, ROOMS, and EQUIPMENTS etc. These ENTITIES will have the the respective attributes like USERS will have Name, Student ID (if student), Faculty ID (if faculty) or Admin ID (if admin), Address, SSN. MANAGER will have Name, Address, SSN etc. The relationship between the two entities here i.e ADMIN and NORMAL USER is one-to-many i.e there is only one admin user is associated with multiple normal users. For instance, the only admin user approves various requests for reservations made by various normal users, or the admin user can add, modify or delete various normal users, rooms, equipments or privileges associated with any specific normal user account. There is only one admin user account and multiple normal account users categorized in two subclasses i.e STUDENT and FACULTY. Specific attributes of FACULTY are rank, office, office phone and Faculty ID. And the specific attributes of STUDENT are Student ID, major, minor (if known). The STUDENT or FACULTY user will be able to make the reservation by providing his/her first name and the last name, which is his/her composite attribute along with the single value attribute i.e his/her student ID. And same is the case with faculty members who will provide their faculty ID as their single value attribute. We further can make an assumption about rooms getting categorized based upon the number of equipments it has and also based on the departments.

**EER Modelling**

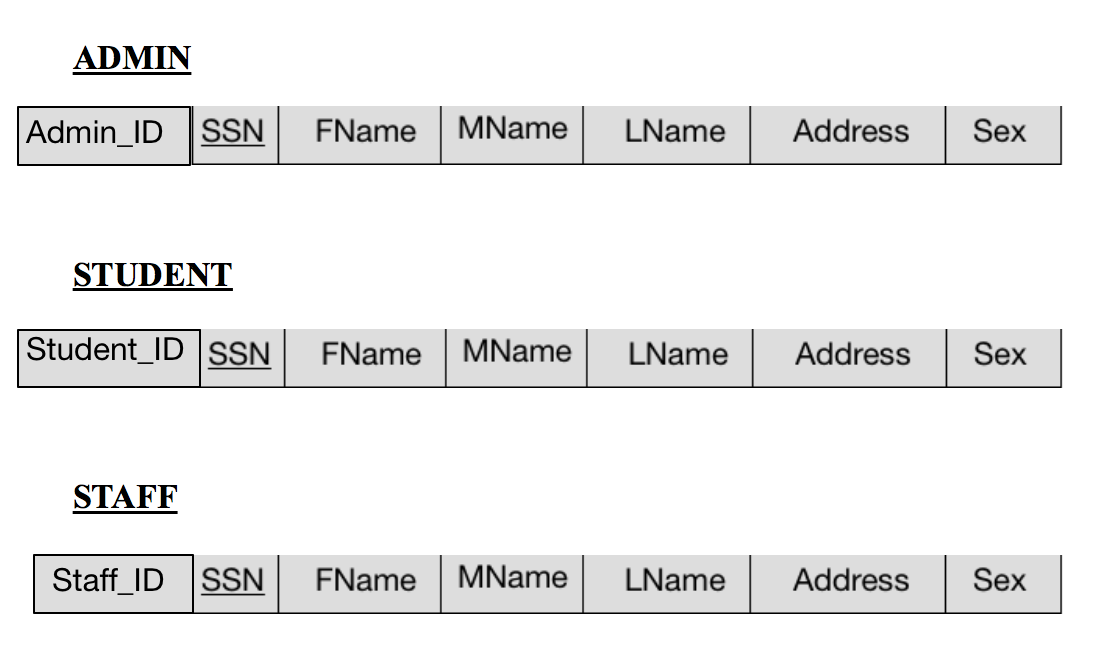
**Technical Document**

****

**Relations**

**Multiple Relations: Superclasses and Subclasses:**

**Multiple Relations: Subclasses Only:**

**Singl****e Relation with One type Attribute:**

**USER**

SSN

Sex

Address

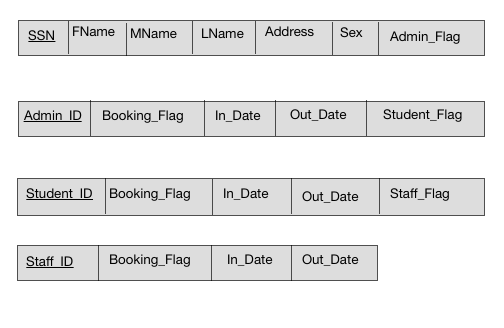
LName

MName

FName

User\_ID

**Single Relation with multiple type Attribute:**

**USER**

* **Identify other keys (candidate keys) in your relations.**

1. User ID in USER entity is the candidate key.
2. University ID in UNIVERSITY entity is the candidate key.
3. EQUIPMENT NAME is the candidate key.

* **Identify functional dependencies in your application and list them.**

1. University ID can uniquely identify University Name.
2. University Name can uniquely identify the Location of the University.
3. Building ID can identify Building Name.
4. Department ID can identify Department Name and Location.
5. Room ID can identify Room Type.
6. Room ID can identify whether the Room is Reserved or not.
7. Room ID can identify the Room Ratings.
8. Equipment ID can identify Equipment Name.
9. User ID can identify User SSN.
10. SSN can identify whether the USER is Student, Staff or the Admin.
11. Student ID can identify Student Name.
12. Student ID can identify sex, address and SSN.
13. Staff ID can uniquely identify Staff Name.

**Relational Algebra**

1. Select university name based on its location.

**RA**: σ Location=Connecticut(UNIVERSITY);

1. Select building name based on the building ID or location.

**RA**: σ BuildingID = B101(BUILDING); OR σ BuildingID = B102(BUILDING);

1. Select department name based on the building name or building ID.

**RA**: σ (BuildingID = B101 AND BuildingName = Engineering) (DEPARTMENT);

1. Select department location based on the department ID.

**RA**: π DepartmentLoc(σ DeptID = D101 (DEPARTMENT));

1. Select room based on the department name or department ID.
2. Select department location based on the department name.

**RA**: π DepartmentLoc(σ DeptName = Business (DEPARTMENT));

1. Select department name if it has laboratory available for booking or not.

**RA**: π DepName(σ LabID != NULL (DEPARTMENT));

1. Select department name if it has rooms available for booking.

**RA**: π DepName(σ RoomID!= “NULL” AND BookingID ==“NULL” (DEPARTMENT));

1. Select laboratory name based on the ID of the lab and department name.

**RA**: π DepName(σ LabID = L101 (DEPARTMENT));

1. Select rooms based on if it has equipment or not.

**RA**: π RoomType, RoomID(σ EquipmentID !=“NULL” (ROOMS));

1. Select room number based on room type.

**RA**: π RoomID, RoomNum (σ Type == “HasEquipments” (ROOMS));

1. Select room type based on room number.

**RA**: π RoomType (σ RoomNum = R103 (ROOMS));

1. Select room based on the department name.

**RA**: π RoomID (σ DepName = “Engineering” (ROOMS));

1. Select department name based on the room number or room ID.

**RA**: π DepName (σ RoomNum = R102 (ROOMS));

1. Select equipment type based on Equipment ID.

**RA**: π EquipmentType (σ EquipmentID = E109 (EQUIPMENT));

1. Select equipment based on the room number that has it.

**RA**: π EquipmentID, EquipmentName (σ RType == “HasEquipments” (ROOMS));

1. Select rating number and Rating Description when given room ID.

**RA**: π RatingNum, RatingDescription (σ RoomID= R103 (RATING));

1. Select room ID and Room number when given rating ID.

**RA**: π RoomID, RoomNum (σ RatingID = RT101 (RATING));

1. Show ratings, room name when given rating ID.

**RA**: π RatingDescription, RoomNum (σ RatingID = RT101 (ROOMS));

1. Show ratings when given User ID.

**RA**: π RatingDecription (σ UserID = A101 || UserID = S101 || UserID = ST101 (RATING));

1. Select if user is Admin, Student or Staff when given User ID.

**RA**: ADMINRESULT ← π AdminName, AdminSSN (σ UserID = A101(USER));

STUDENTRESULT ← π StudentName, StudentMajor (σ UserID = S101(USER));

STAFFRESULT ← π FacultyName, StaffSSN (σ UserID = ST101(USER));

RESULT ← π UserType(σ UserID (ADMINRESULT, STUDENTRESULT, STAFFRESULT));

1. Select first Name, Middle name and last one when given SSN of the user.

**RA**: π FName, MName,LName (σ SSN (USER));

1. Select booking ID when given user ID to see if the user has any bookings.

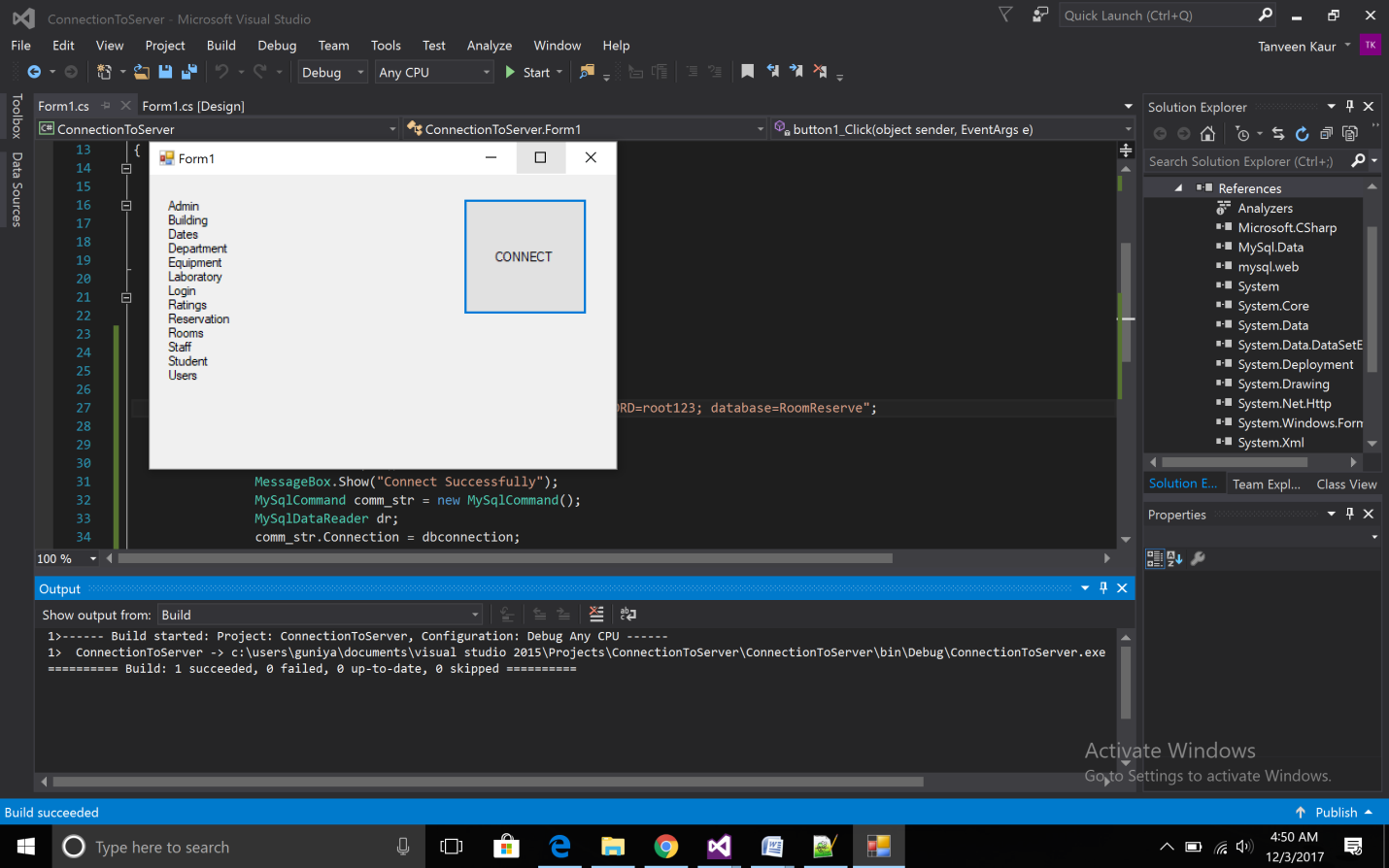
**RA**: π BookingID (σ USERID (USER));

1. Select In date and out date when given Booking ID.

**RA**: π InDate, OutDate (σ BookingID (BOOKING));

**Creating Database Server and Web Application Server**

**C# Visual Studio 2015 Professional**

****

**Physical Database Design/Implementation**

1. **Creating a DDL Script:**

FileName: **dbDDL.sql,**

Created in: **MySQL shell,**

Database schema: **RoomReserve**

Tables: **13 Tables in total.**

Objects: **2 Objects in DDL**

Constraints: **Primary Key, Foreign Key, NOT NULL, DEFAULT, UNIQUE.**

**SOURCE:**

-- mysql Script generated by mysql Workbench

-- Fri Nov 17 18:03:25 2017

-- Model: New Model Version: 1.0

-- mysql Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='TRADITIONAL,ALLOW\_INVALID\_DATES';

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema roomreserve

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema roomreserve

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `roomreserve` DEFAULT CHARACTER SET latin1 ;

USE `roomreserve` ;

-- -----------------------------------------------------

-- Table `roomreserve`.`Users`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `roomreserve`.`Users` (

`Fname` VARCHAR(20) NOT NULL,

`mname` VARCHAR(20) NULL DEFAULT NULL,

`lname` VARCHAR(20) NOT NULL,

`SSN` INT(9) NOT NULL,

`address` VARCHAR(20) NOT NULL,

`Sex` VARCHAR(1) NOT NULL,

PRIMARY KEY (`SSN`),

INDEX `user\_ssn` (`SSN` ASC))

ENGINE = innodb

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `roomreserve`.`Admin`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `roomreserve`.`Admin` (

`adminid` VARCHAR(20) NOT NULL,

`Fname` VARCHAR(20) NOT NULL,

`Mname` VARCHAR(20) NOT NULL,

`Lname` VARCHAR(20) NOT NULL,

`Sex` VARCHAR(1) NOT NULL,

`address` VARCHAR(20) NOT NULL,

`SSN` INT(9) NOT NULL,

PRIMARY KEY (`adminid`),

INDEX `SSN` (`SSN` ASC),

CONSTRAINT `admin\_ibfk\_1`

FOREIGN KEY (`SSN`)

REFERENCES `roomreserve`.`Users` (`SSN`))

ENGINE = innodb

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `roomreserve`.`Building`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `roomreserve`.`Building` (

`Id` int(1) NOT NULL AUTOINCREMENT,

`buildingid` VARCHAR(20) NOT NULL,

`name` VARCHAR(20) NOT NULL,

`hasdepartment` TINYINT(1) NULL DEFAULT NULL,

PRIMARY KEY (`ID`))

ENGINE = innodb

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `roomreserve`.`Date`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `roomreserve`.`Dates` (

`Day` INT(2) NOT NULL,

`Month` VARCHAR(10) NOT NULL,

`Year` INT(4) NOT NULL)

ENGINE = innodb

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `roomreserve`.`Department`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `roomreserve`.`Department` (

`depid` VARCHAR(20) NOT NULL,

`name` VARCHAR(20) NOT NULL,

`location` VARCHAR(20) NOT NULL,

`haslab` TINYINT(1) NULL DEFAULT NULL,

PRIMARY KEY (`depid`))

ENGINE = innodb

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `roomreserve`.`Rooms`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `roomreserve`.`Rooms` (

`roomid` VARCHAR(20) NOT NULL,

`Number` INT(4) NOT NULL,

`Type` VARCHAR(20) NOT NULL,

`Has\_Equipment` TINYINT(1) NULL DEFAULT NULL,

`depid` VARCHAR(20) NULL DEFAULT NULL,

PRIMARY KEY (`roomid`),

INDEX `depid` (`depid` ASC),

CONSTRAINT `rooms\_ibfk\_1`

FOREIGN KEY (`depid`)

REFERENCES `roomreserve`.`Department` (`depid`))

ENGINE = innodb

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `roomreserve`.`Equipment`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `roomreserve`.`Equipment` (

`ID` VARCHAR(20) NOT NULL,

`type` VARCHAR(20) NOT NULL,

`roomid` VARCHAR(20) NOT NULL,

INDEX `roomid` (`roomid` ASC),

CONSTRAINT `equipment\_ibfk\_1`

FOREIGN KEY (`roomid`)

REFERENCES `roomreserve`.`Rooms` (`roomid`))

ENGINE = innodb

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `roomreserve`.`Laboratory`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `roomreserve`.`Laboratory` (

`labid` VARCHAR(20) NOT NULL,

`name` VARCHAR(20) NOT NULL,

`depid` VARCHAR(20) NULL DEFAULT NULL,

INDEX `depid` (`depid` ASC),

CONSTRAINT `laboratory\_ibfk\_1`

FOREIGN KEY (`depid`)

REFERENCES `roomreserve`.`Department` (`depid`))

ENGINE = innodb

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `roomreserve`.`Ratings`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `roomreserve`.`Ratings` (

`ratingid` VARCHAR(20) NULL DEFAULT NULL,

`description` VARCHAR(50) NOT NULL,

`roomid` VARCHAR(20) NOT NULL,

INDEX `roomid` (`roomid` ASC),

CONSTRAINT `ratings\_ibfk\_1`

FOREIGN KEY (`roomid`)

REFERENCES `roomreserve`.`Rooms` (`roomid`))

ENGINE = innodb

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `roomreserve`.`Reservation`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `roomreserve`.`Reservation` (

`indate` DATE NULL DEFAULT NULL,

`outdate` DATE NULL DEFAULT NULL,

`resid` VARCHAR(20) NOT NULL,

PRIMARY KEY (`resid`))

ENGINE = innodb

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `roomreserve`.`Staff`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `roomreserve`.`Staff` (

`staffid` VARCHAR(20) NOT NULL,

`Fname` VARCHAR(20) NOT NULL,

`Lname` VARCHAR(20) NOT NULL,

`Sex` VARCHAR(1) NOT NULL,

`address` VARCHAR(20) NOT NULL,

`SSN` INT(9) NOT NULL,

`resid` VARCHAR(20) NOT NULL,

PRIMARY KEY (`staffid`),

INDEX `SSN` (`SSN` ASC),

INDEX `resid` (`resid` ASC),

CONSTRAINT `staff\_ibfk\_1`

FOREIGN KEY (`SSN`)

REFERENCES `roomreserve`.`Users` (`SSN`),

CONSTRAINT `staff\_ibfk\_2`

FOREIGN KEY (`resid`)

REFERENCES `roomreserve`.`Reservation` (`resid`))

ENGINE = innodb

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `roomreserve`.`Student`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `roomreserve`.`Student` (

`studentid` VARCHAR(20) NOT NULL,

`Fname` VARCHAR(20) NOT NULL,

`Mname` VARCHAR(20) NOT NULL,

`Lname` VARCHAR(20) NOT NULL,

`Sex` VARCHAR(1) NOT NULL,

`address` VARCHAR(20) NOT NULL,

`SSN` INT(9) NOT NULL,

`resid` VARCHAR(20) NOT NULL,

PRIMARY KEY (`studentid`),

INDEX `SSN` (`SSN` ASC),

INDEX `resid` (`resid` ASC),

CONSTRAINT `student\_ibfk\_1`

FOREIGN KEY (`SSN`)

REFERENCES `roomreserve`.`Users` (`SSN`),

CONSTRAINT `student\_ibfk\_2`

FOREIGN KEY (`resid`)

REFERENCES `roomreserve`.`Reservation` (`resid`))

ENGINE = innodb

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `roomreserve`.`dates`

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Placeholder table for view `roomreserve`.`roomratings`

-- -----------------------------------------------------

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

1. **Creating a DML Script:** *folder Attached with script and screenshots*

FileName: **dbDML.sql,**

Created in: **MySQL shell,**

Database schema: **RoomReserve**

Tables: **13 Tables in total.**

Constraints: **Primary Key, Foreign Key, NOT NULL, DEFAULT, UNIQUE.**

**SOURCE**

use RoomReserve;

insert ignore into Building values('B1021', 'Cartenson', '1');

insert ignore into Building values('B1031', 'Engineering', '1');

insert ignore into Building values('B1041', 'Mandeville', '1');

insert ignore into Building values('B1051', 'ABC', '0');

insert ignore into Building values('B1061', 'North Hall', '0');

insert ignore into Building values('B1071', 'Carlson Hall', '1');

insert ignore into Building values('B1081', 'Bodine Hall', '0');

insert ignore into Building values('B1091', 'Library', '1');

commit;

insert ignore into Department values('D1','Bookstore','NorthCampus',0);

insert ignore into Department values('D2','Cafe','SouthCampus',1);

insert ignore into Department values('D3','Biomedical','MainCampus',1);

insert ignore into Department values('D4','Business','BehindCampus',0);

insert ignore into Department values('D5','Activity','EastCampus',0);

insert ignore into Department values('D6','Communication','NorthCampus',1);

insert ignore into Department values('D7','Admissions','MainCampus',1);

commit;

insert ignore into Users values('Ellen','hatie','lewa','211233126','North Side','M');

insert ignore into Users values('Ana','senior','jona','211233444','Linden Ave','M');

insert ignore into Users values('Eva','nuv','kotawa','211236274','65 waldmare','M');

insert ignore into Users values('Marshal','aro','lindsay','212036274','Brooklyn','F');

insert ignore into Users values('Norahnson','mari','hateuwa','231098674','South end','F');

insert ignore into Users values('Taran','lona','singh','232233123','Water street','F');

insert ignore into Admin values('A102','Maya','k','kellog','F','Waldemare st','211232344');

insert ignore into Admin values('A103','George','hows','felma','F','North street','211233123');

insert ignore into Admin values('A104','Angie','hoze','smith','F','9 University Ave','211233444');

insert ignore into Admin values('A105','Neva','loe','Jith','F','7 lafayette','211236274');

insert ignore into Admin values('A106','Paul','nea','Otoma','F','John str','212036274');

insert ignore into Admin values('A106','Guniya','junior','Huya','F','broad street','231098674');

insert ignore into Admin values('A107','Shelja','senior','Johnson','F','95 Ave','232233123');

commit;

insert ignore into Dates values(20,'January',2014);

insert ignore into Dates values(12,'February',2015);

insert ignore into Dates values(21,'March',2017);

insert ignore into Dates values(2,'May',2017);

insert ignore into Dates values(5,'April',2016);

insert ignore into Dates values(9,'July',2016);

insert ignore into Dates values(23,'July',2017);

insert ignore into Dates values(11,'November',2017);

insert ignore into Dates values(8,'October',2017);

insert ignore into Dates values(5,'December',2017);

commit;

insert ignore into Rooms values('R2',101,'single',1,'D102');

insert ignore into Rooms values('R3',102,'double',1,'D102');

insert ignore into Rooms values('R4',103,'single',1,'D103');

insert ignore into Rooms values('R5',104,'double',1,'D104');

insert ignore into Rooms values('R6',105,'single',1,'D105');

insert ignore into Rooms values('R7',106,'double',1,'D101');

insert ignore into Rooms values('R8',107,'single',1,'D106');

insert ignore into Rooms values('R9',108,'double',1,'D107');

insert ignore into Rooms values('R10',109,'single',1,'D108');

commit;

insert ignore into Laboratory values('L103','Lab111','D102');

insert ignore into Laboratory values('L104','Lab112','D103');

insert ignore into Laboratory values('L105','Lab113','D104');

insert ignore into Laboratory values('L106','Lab114','D105');

insert ignore into Laboratory values('L107','Lab115','D106');

insert ignore into Laboratory values('L108','Lab116','D107');

insert ignore into Laboratory values('L109','Lab117','D108');

insert ignore into Laboratory values('L110','Lab118','D109');

commit;

insert ignore into Ratings values('Rating2','Good and Airy','R2');

insert ignore into Ratings values('Rating3','Excellent','R1');

insert ignore into Ratings values('Rating4','No internet','R7');

insert ignore into Ratings values('Rating5','Good but no internet','R8');

insert ignore into Ratings values('Rating6','Good','R9');

insert ignore into Ratings values('Rating7','Comfortable','R2');

insert ignore into Ratings values('Rating8','Excellent and internet','R2');

insert ignore into Ratings values('Rating9','Easy accesible','R3');

insert ignore into Ratings values('Rating10','Good environment','R4');

insert ignore into Ratings values('Rating11','Good and comfortable','R1');

commit;

insert ignore into Reservation values('2017-03-12','2017-04-11','Reserve102');

insert ignore into Reservation values('2017-04-13','2017-05-13','Reserve103');

insert ignore into Reservation values('2017-05-14','2017-06-14','Reserve104');

insert ignore into Reservation values('2017-06-15','2017-07-15','Reserve105');

insert ignore into Reservation values('2017-07-16','2017-08-15','Reserve106');

insert ignore into Reservation values('2017-08-16','2017-09-15','Reserve107');

insert ignore into Reservation values('2017-09-16','2017-10-13','Reserve108');

insert ignore into Reservation values('2017-10-16','2017-11-23','Reserve109');

insert ignore into Reservation values('2017-11-16','2017-12-22','Reserve110');

commit;

insert ignore into Staff values('ST12','Tanveen','Kaur','F','Broad Street','211232344','Reserve102');

insert ignore into Staff values('ST13','Jaag','Josh','M','John St','211233123','Reserve103');

insert ignore into Staff values('ST14','Anam','Menns','M','Linden AVe','211233444','Reserve104');

insert ignore into Staff values('ST15','Harpreet','Kaur','F','Waldemere','211236274','Reserve105');

insert ignore into Staff values('ST16','Joy','Ruya','M','Broad Ave','212036274','Reserve106');

insert ignore into Staff values('ST17','Paul','Seyfert','M','Atlantic ave','212098674','Reserve107');

insert ignore into Staff values('ST18','John','alex','M','Lafayette st','211232314','Reserve107');

insert ignore into Staff values('ST19','Jinny','Taskin','M','Loan ave','231098674','Reserve108');

commit;

insert ignore into Student values('S1011234','Guniya','K','Kaur','F','Broad ave','211233123','Reserve102');

insert ignore into Student values('S1014609','Anita','K','George','F','9 North ave','211233444','Reserve102');

insert ignore into Student values('S1014532','Pauline','K','Seyfert','F','Linden Ave','212036274','Reserve103');

insert ignore into Student values('S1012123','Michael','K','Lolley','F','South Hall','212098674','Reserve105');

insert ignore into Student values('S1014009','Duane','K','Kellog','F','BlackRock','231098674','Reserve106');

insert ignore into Student values('S1015643','Ellen','K','Ave','F','95 University','232233123','Reserve107');

commit;

insert ignore into Equipment values('E2','Simple','R2');

insert ignore into Equipment values('E3','Electronic','R3');

insert ignore into Equipment values('E4','Accessory','R3');

insert ignore into Equipment values('E5','Electronic','R4');

insert ignore into Equipment values('E6','Stationary','R5');

insert ignore into Equipment values('E7','Simple','R6');

insert ignore into Equipment values('E8','Extras','R9');

insert ignore into Equipment values('E9','Projector','R7');

insert ignore into Equipment values('E10','Electronic','R8');

commit;

1. **Creating a Drop Script:** *script file attached along*

FileName: **dbDROP.sql,**

Created in: **MySQL Workbench,**

**SOURCE:**

SET FOREIGN\_KEY\_CHECKS = 0;

drop table if exists Admin;

drop table if exists Building;

drop table if exists dates;

drop table if exists Department;

drop table if exists Equipment;

drop table if exists Laboratory;

drop table if exists Ratings;

drop table if exists Reservation;

drop table if exists Rooms;

drop table if exists Staff;

drop table if exists Student;

drop table if exists University;

drop table if exists Users;

SET FOREIGN\_KEY\_CHECKS = 1;

CREATE SCHEMA IF NOT EXISTS `RoomReserve` DEFAULT CHARACTER SET latin1 ;

USE `RoomReserve` ;

-- -----------------------------------------------------

-- Table `RoomReserve`.`Users`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `RoomReserve`.`Users` (

`Fname` VARCHAR(20) NOT NULL,

`MName` VARCHAR(20) NULL DEFAULT NULL,

`LName` VARCHAR(20) NOT NULL,

`SSN` INT(9) NOT NULL,

`address` VARCHAR(20) NOT NULL,

`Sex` VARCHAR(1) NOT NULL,

PRIMARY KEY (`SSN`),

INDEX `user\_ssn` (`SSN` ASC))

ENGINE = InnoDB

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `RoomReserve`.`Admin`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `RoomReserve`.`Admin` (

`AdminID` VARCHAR(20) NOT NULL,

`Fname` VARCHAR(20) NOT NULL,

`Mname` VARCHAR(20) NOT NULL,

`Lname` VARCHAR(20) NOT NULL,

`Sex` VARCHAR(1) NOT NULL,

`address` VARCHAR(20) NOT NULL,

`SSN` INT(9) NOT NULL,

PRIMARY KEY (`AdminID`),

INDEX `SSN` (`SSN` ASC),

CONSTRAINT `admin\_ibfk\_1`

FOREIGN KEY (`SSN`)

REFERENCES `RoomReserve`.`Users` (`SSN`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `RoomReserve`.`Building`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `RoomReserve`.`Building` (

`ID` VARCHAR(20) NOT NULL,

`name` VARCHAR(20) NOT NULL,

`hasDepartment` TINYINT(1) NULL DEFAULT NULL,

PRIMARY KEY (`ID`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `RoomReserve`.`Date`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `RoomReserve`.`Date` (

`Day` INT(2) NOT NULL,

`Month` VARCHAR(10) NOT NULL,

`Year` INT(4) NOT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `RoomReserve`.`Department`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `RoomReserve`.`Department` (

`DepID` VARCHAR(20) NOT NULL,

`name` VARCHAR(20) NOT NULL,

`location` VARCHAR(20) NOT NULL,

`hasLAB` TINYINT(1) NULL DEFAULT NULL,

PRIMARY KEY (`DepID`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `RoomReserve`.`Rooms`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `RoomReserve`.`Rooms` (

`RoomID` VARCHAR(20) NOT NULL,

`Number` INT(4) NOT NULL,

`Type` VARCHAR(20) NOT NULL,

`Has\_Equipment` TINYINT(1) NULL DEFAULT NULL,

`DepID` VARCHAR(20) NULL DEFAULT NULL,

PRIMARY KEY (`RoomID`),

INDEX `DepID` (`DepID` ASC),

CONSTRAINT `rooms\_ibfk\_1`

FOREIGN KEY (`DepID`)

REFERENCES `RoomReserve`.`Department` (`DepID`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `RoomReserve`.`Equipment`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `RoomReserve`.`Equipment` (

`ID` VARCHAR(20) NOT NULL,

`type` VARCHAR(20) NOT NULL,

`RoomID` VARCHAR(20) NOT NULL,

INDEX `RoomID` (`RoomID` ASC),

CONSTRAINT `equipment\_ibfk\_1`

FOREIGN KEY (`RoomID`)

REFERENCES `RoomReserve`.`Rooms` (`RoomID`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `RoomReserve`.`Laboratory`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `RoomReserve`.`Laboratory` (

`LabID` VARCHAR(20) NOT NULL,

`name` VARCHAR(20) NOT NULL,

`DepID` VARCHAR(20) NULL DEFAULT NULL,

INDEX `DepID` (`DepID` ASC),

CONSTRAINT `laboratory\_ibfk\_1`

FOREIGN KEY (`DepID`)

REFERENCES `RoomReserve`.`Department` (`DepID`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `RoomReserve`.`Ratings`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `RoomReserve`.`Ratings` (

`RatingID` VARCHAR(20) NULL DEFAULT NULL,

`description` VARCHAR(50) NOT NULL,

`RoomID` VARCHAR(20) NOT NULL,

INDEX `RoomID` (`RoomID` ASC),

CONSTRAINT `ratings\_ibfk\_1`

FOREIGN KEY (`RoomID`)

REFERENCES `RoomReserve`.`Rooms` (`RoomID`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `RoomReserve`.`Reservation`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `RoomReserve`.`Reservation` (

`InDate` DATE NULL DEFAULT NULL,

`OutDate` DATE NULL DEFAULT NULL,

`UserID` VARCHAR(20) NOT NULL,

`ResID` VARCHAR(20) NOT NULL,

PRIMARY KEY (`ResID`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `RoomReserve`.`Staff`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `RoomReserve`.`Staff` (

`StaffID` VARCHAR(20) NOT NULL,

`Fname` VARCHAR(20) NOT NULL,

`Mname` VARCHAR(20) NOT NULL,

`Lname` VARCHAR(20) NOT NULL,

`Sex` VARCHAR(1) NOT NULL,

`address` VARCHAR(20) NOT NULL,

`SSN` INT(9) NOT NULL,

`ResID` VARCHAR(20) NOT NULL,

PRIMARY KEY (`StaffID`),

INDEX `SSN` (`SSN` ASC),

INDEX `ResID` (`ResID` ASC),

CONSTRAINT `staff\_ibfk\_1`

FOREIGN KEY (`SSN`)

REFERENCES `RoomReserve`.`Users` (`SSN`),

CONSTRAINT `staff\_ibfk\_2`

FOREIGN KEY (`ResID`)

REFERENCES `RoomReserve`.`Reservation` (`ResID`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `RoomReserve`.`Student`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `RoomReserve`.`Student` (

`StudentID` VARCHAR(20) NOT NULL,

`Fname` VARCHAR(20) NOT NULL,

`Mname` VARCHAR(20) NOT NULL,

`Lname` VARCHAR(20) NOT NULL,

`Sex` VARCHAR(1) NOT NULL,

`address` VARCHAR(20) NOT NULL,

`SSN` INT(9) NOT NULL,

`ResID` VARCHAR(20) NOT NULL,

PRIMARY KEY (`StudentID`),

INDEX `SSN` (`SSN` ASC),

INDEX `ResID` (`ResID` ASC),

CONSTRAINT `student\_ibfk\_1`

FOREIGN KEY (`SSN`)

REFERENCES `RoomReserve`.`Users` (`SSN`),

CONSTRAINT `student\_ibfk\_2`

FOREIGN KEY (`ResID`)

REFERENCES `RoomReserve`.`Reservation` (`ResID`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = latin1;

-- -----------------------------------------------------

-- Table `RoomReserve`.`University`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `RoomReserve`.`University` (

`UnivID` VARCHAR(20) NOT NULL,

`name` VARCHAR(20) NOT NULL,

PRIMARY KEY (`UnivID`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = latin1;

1. **Creating SQL Script:** *script file attached along*

FileName: **dbDROP.sql,**

Created in: **MySQL shell,**

**SOURCE:**

/\* Retrieve all student names.

\*/

select Fname, Lname from Student;

/\* Retrieve names, SSN and Sex of all Users.

\*/

select Fname, SSN, Sex from Users;

/\* Retrieve roomID and Type from Rooms.

\*/

select RoomID, type from Rooms;

/\*

Retrieve ID and name of the Building.

\*/

select ID, name from Building;

/\*

Retrieve Ratings and description by Room ID (JOIN tables Ratings and Rooms).

\*/

Select rt.RatingID, rt.description, rt.RoomID, ro.Type from Ratings rt, Rooms ro where rt.RoomID = ro.RoomID;

/\*

Retrieve Room number and Department Name by Department ID (JOIN tables Rooms and Department).

\*/

select ro.Number, ro.Type, dp.name from Rooms ro, Department dp where ro.DepID = dp.DepID;

/\*Retrieve the Room number, Room ID and Room Type which was never rated (NESTED QUERY)

\*/

select ro.RoomID, ro.Number, ro.Type from Rooms ro where ro.RoomID NOT IN ( select DISTINCT rt.RoomID from Ratings rt);

/\*

Sorting table Rooms by ascending ORDER OF RoomID (ORDER BY)

\*/

select ro.Number, ro.DepID from Rooms ro ORDER BY ro.DepID;

/\* Sorting table Rooms by descending ORDER OF RoomID (ORDER BY)

\*/

select ro.Number, ro.DepID from Rooms ro ORDER BY ro.DepID desc;

/\* Retrieve the count of male and female student.

\*/

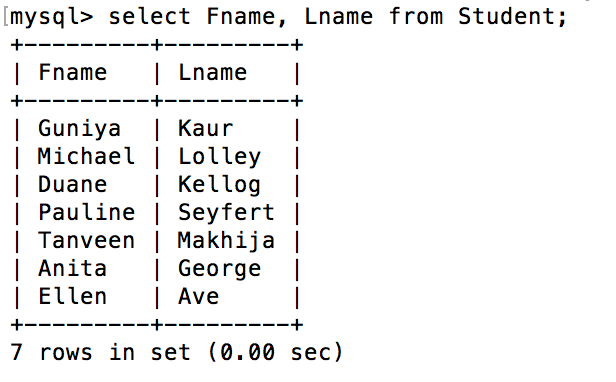
select Sex, count(\*) from Student GROUP BY Sex;

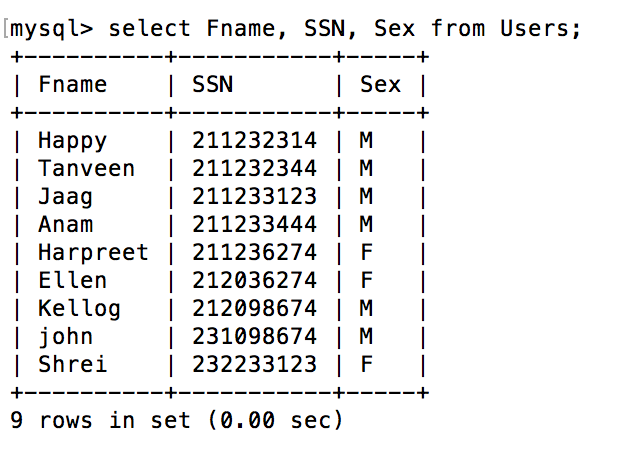
/\*Retrieve the room number and room ID having more than one ratings.

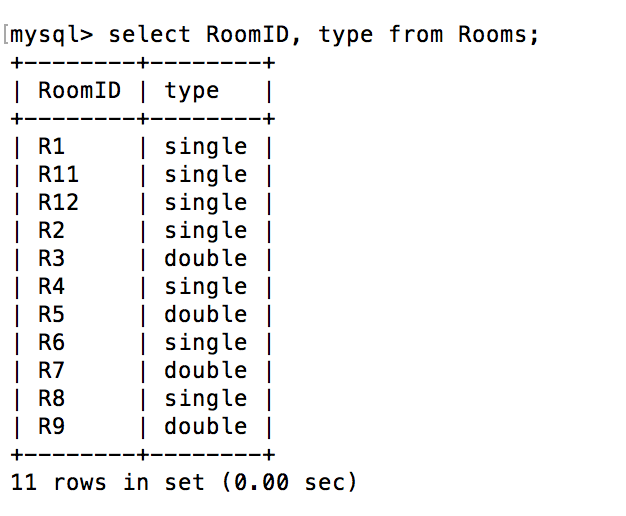
\*/

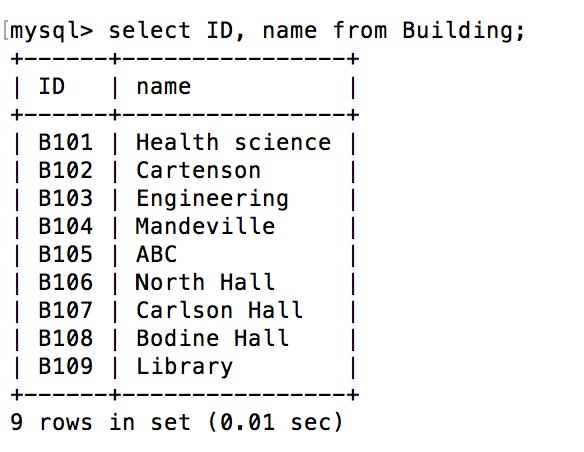
select RoomID from Ratings GROUP BY RoomID HAVING count(RoomID)>1;

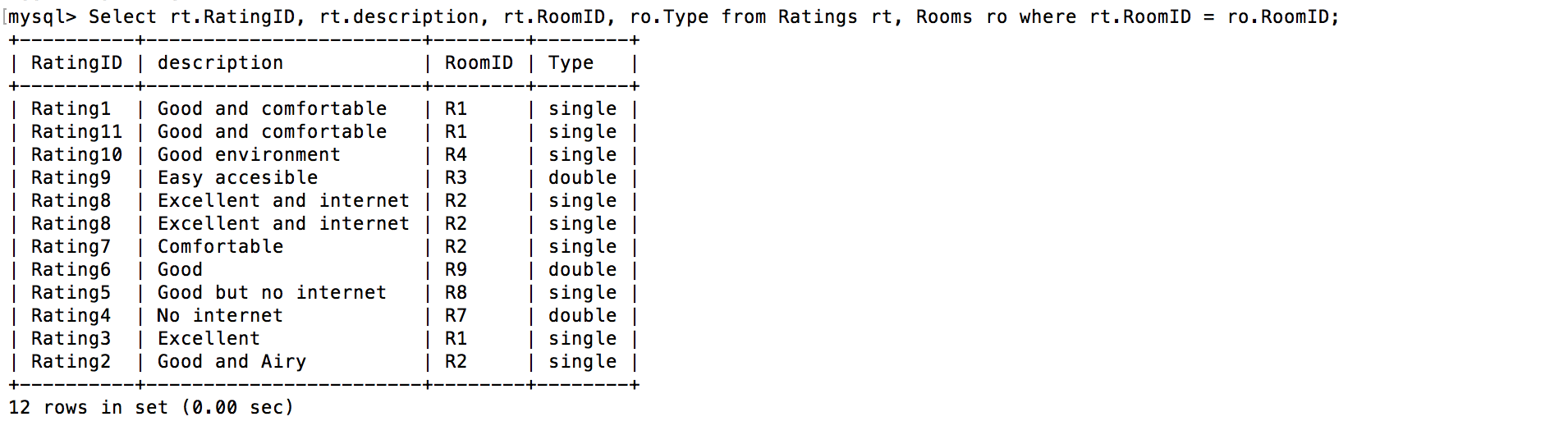
**Below are the few simple queries and their output screenshots:**

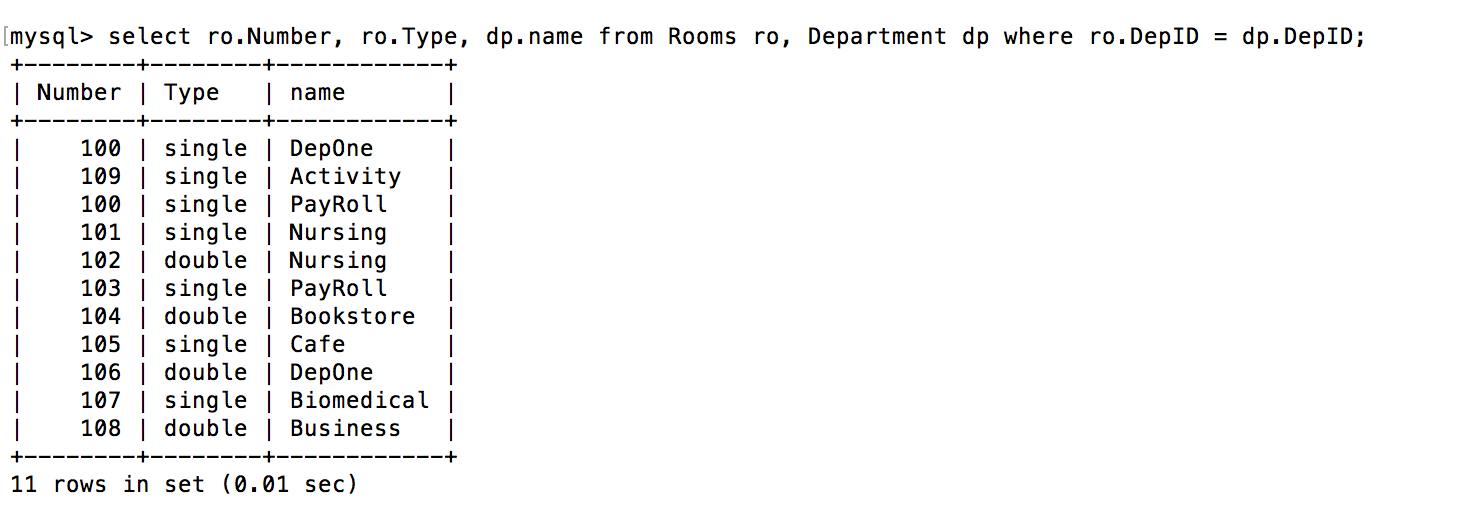
**Query 1. Retrieve all student names.**

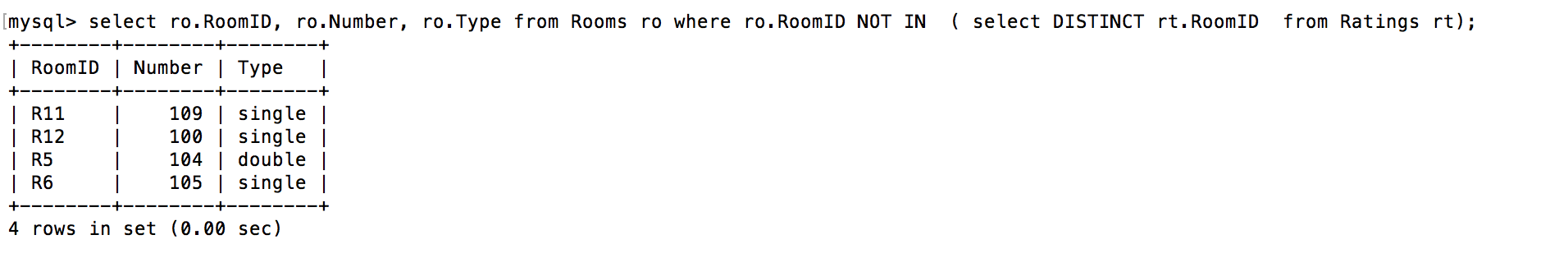
**Query 2. Retrieve names, SSN and Sex of all Users.**

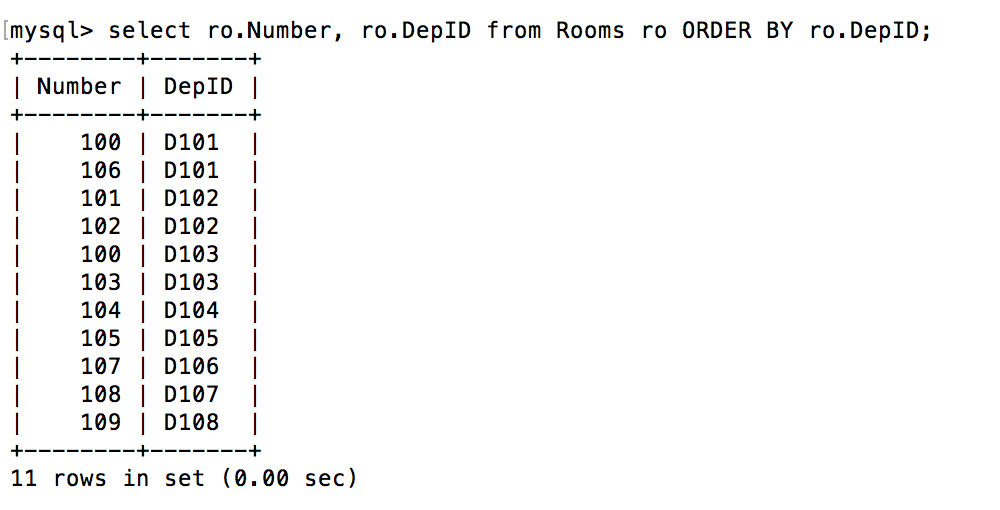
**Query 3. Retrieve roomID and Type from Rooms.**

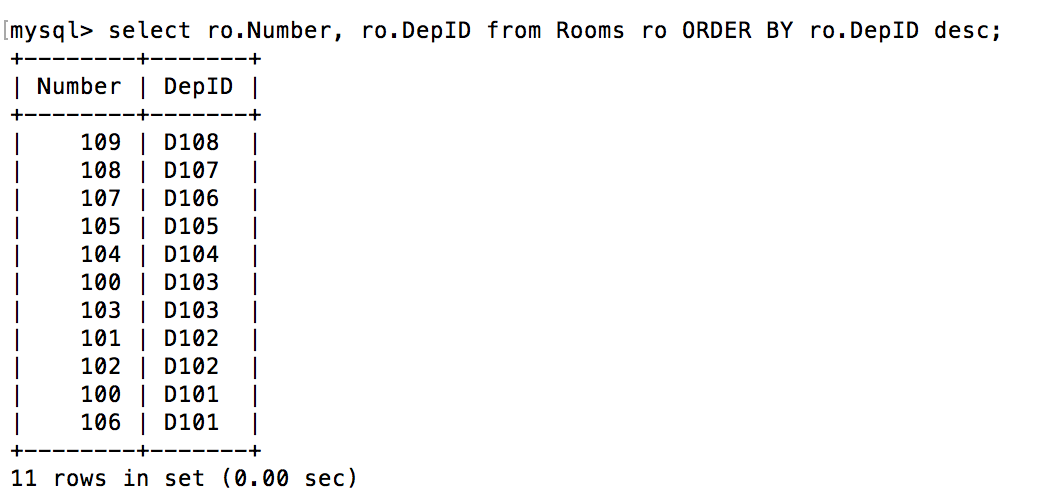
**Query 4. Retrieve ID and name of the Building.**

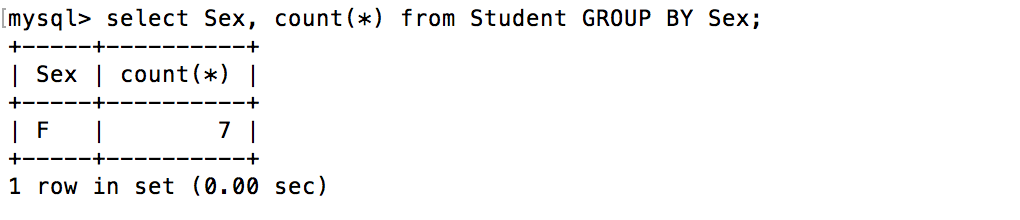
**Query 5. Retrieve Ratings and description by Room ID (JOIN tables Ratings and Rooms).**

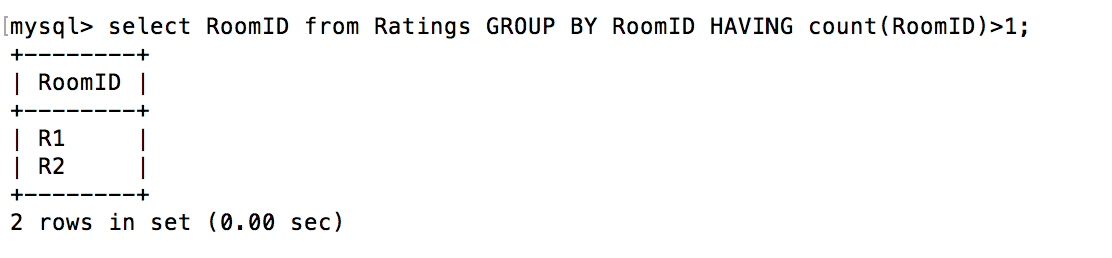
**Query 6. Retrieve Room number and Department Name by Department ID (JOIN tables Rooms and Department).**

**Query 7. Retrieve the Room number, Room ID and Room Type which was never rated (NESTED QUERY)**

**Query 8. Sorting table Rooms by ascending ORDER OF RoomID (ORDER BY)**

**Query 9. Sorting table Rooms by descending ORDER OF RoomID (ORDER BY)**

**Query 10. Retrieve the number of Male and Female Students with Student ID and Reservation ID (GROUP BY).**

**Query 11. Retrieve Room Id and room Number having more than one ratings. (GROUP BY , HAVING)**

**Application Development**

**System Configuration of the project (SYSTEM DIAGRAM)**

**DATABASE SERVER**

GOOGLE CLOUD PLATFORM

**USER**

FILL CANCELLATION FORM

RETRIEVE/CANCEL

INFO

**ROOM**

**RESERVATION SYSTEM**

**USER**

ROOM INFO

**ADMIN**

FILL RESERVATION FORM

FILL RESERVATION FORM

**Sources of Application (C#)**

Start- FORM1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.OleDb;

using MySql.Data.MySqlClient;

namespace adbProject

{

public partial class Form1 : Form

{

private string conn;

private MySqlConnection connect;

public Form1()

{

InitializeComponent();

}

private void db\_connection()

{

try

{

conn = "Server=35.196.198.132;Database=RoomReserve1;Uid=root;Pwd=root123;";

connect = new MySqlConnection(conn);

connect.Open();

}

catch (MySqlException e)

{

throw;

}

}

private bool validate\_login(string user, string pass)

{

db\_connection();

MySqlCommand cmd = new MySqlCommand();

cmd.CommandText = "Select \* from Login where username=@user and password=@pass";

cmd.Parameters.AddWithValue("@user", user);

cmd.Parameters.AddWithValue("@pass", pass);

cmd.Connection = connect;

MySqlDataReader login = cmd.ExecuteReader();

if (login.Read())

{

connect.Close();

return true;

}

else

{

connect.Close();

return false;

}

}

private void btnSubmit\_Click(object sender, EventArgs e)

{

string user = txtUsername.Text;

string pass = txtPassword.Text;

if (user == "" || pass == "")

{

MessageBox.Show("Empty Fields Detected ! Please fill up all the fields");

return;

}

bool r = validate\_login(user, pass);

if (r)

{

MessageBox.Show("Correct Login Credentials");

MessageBox.Show("Connection Successful");

new Form2().Show();

this.Hide();

}

else

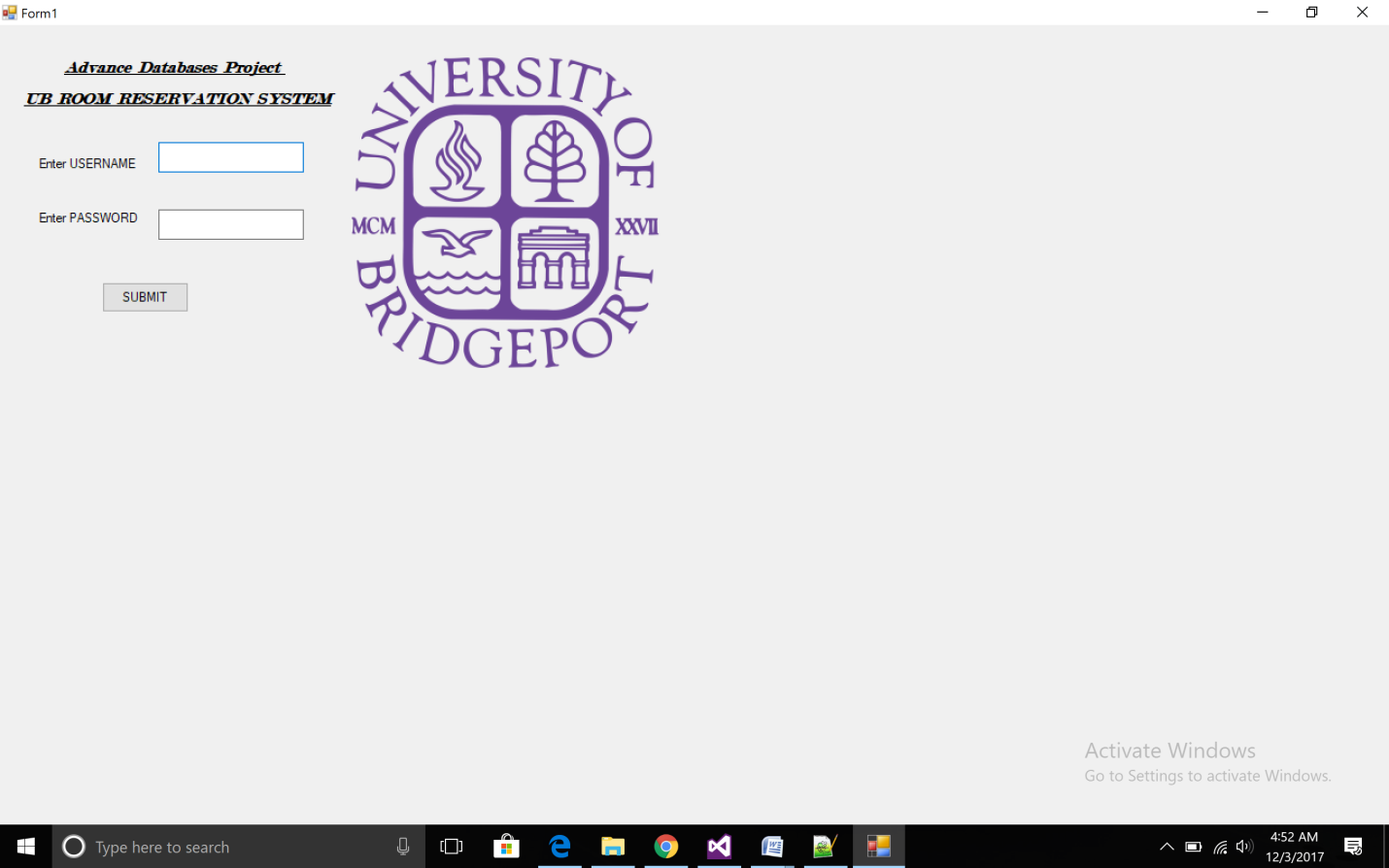
MessageBox.Show("Incorrect Login Credentials");

}

}

}

Output:



FORM2.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace adbProject

{

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

// do nothing

}

private void btnOK\_Click(object sender, EventArgs e)

{

switch(comboBox.SelectedItem.ToString().Trim())

{

case "BUILDING":

new Building().Show();

this.Hide();

break;

case "DEPARTMENT":

new Department().Show();

this.Hide();

break;

case "EQUIPMENT":

new Equipment().Show();

this.Hide();

break;

case "RATINGS":

new Ratings().Show();

this.Hide();

break;

case "LOGIN":

MessageBox.Show("YOU HAVE NO ACCESS TO THIS TABLE: SORRY!!");

break;

case "RESERVATION":

new Reservations().Show();

this.Hide();

break;

case "ROOMS":

new Rooms().Show();

this.Hide();

break;

case "USERS":

MessageBox.Show("You are not allowed to access this table, Sorry! ");

break;

default:

MessageBox.Show("SELECT ANY TABLE TO PROCEED:");

break;

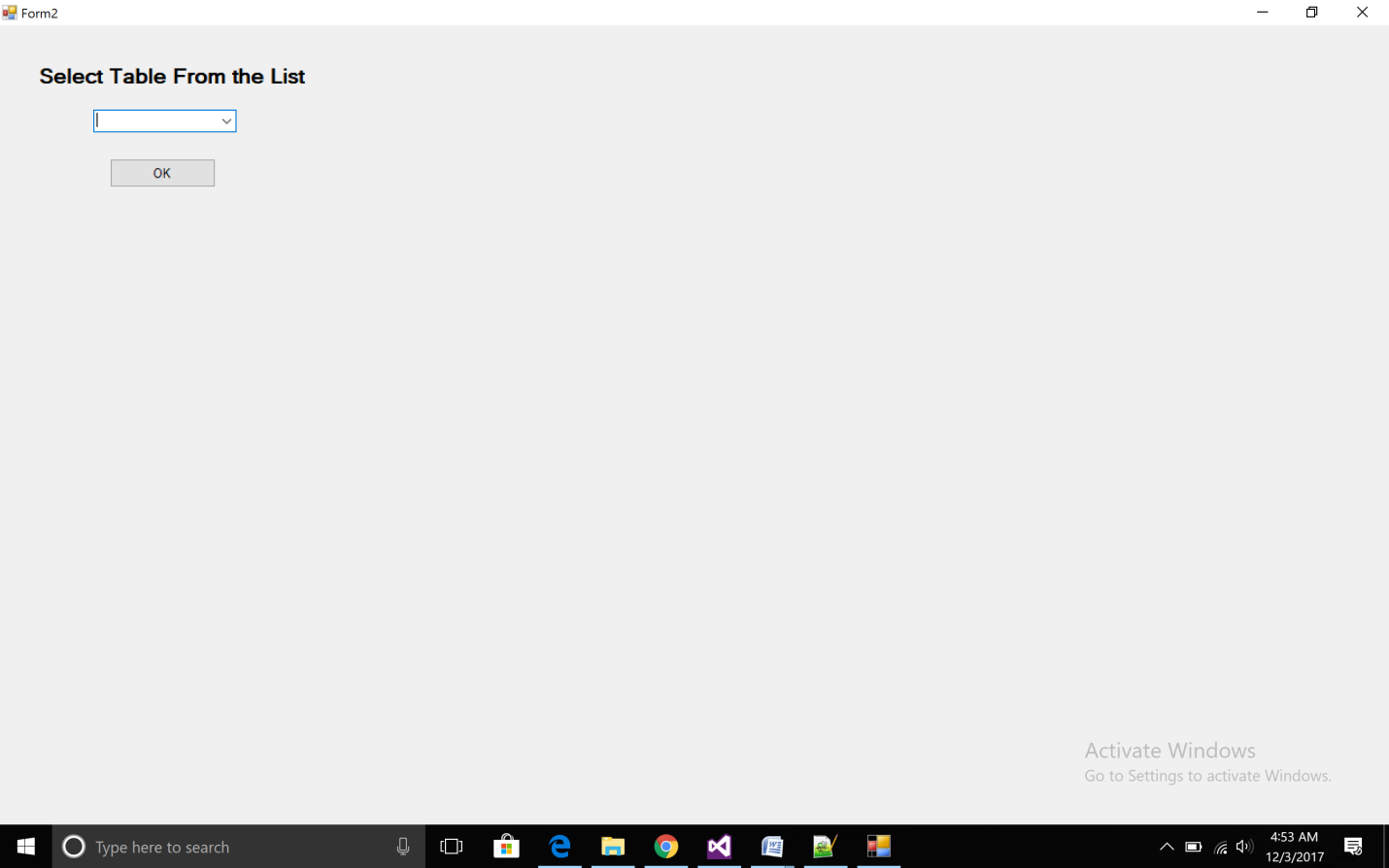
}

}

}

}

Output:



Building.cs

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace adbProject

{

public partial class Building : Form

{

string connectionStr = "Server=35.196.198.132;Database=RoomReserve1;Uid=root;Pwd=root123;";

// int ID = 0;

public Building()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

ClearData();

}

private void ClearData()

{

txtBuildingName.Text = "";

//txtID.Text = "";

// ID = 0;

// txtBuildingID.Text = "";

comboBox.Text = "";

// btnInsert.Text = "Save";

// btnDelete.Enabled = false;

}

private void btnGoBack\_Click(object sender, EventArgs e)

{

new Form2().Show();

this.Hide();

}

private void btnCancel\_Click(object sender, EventArgs e)

{

ClearData();

}

private void btnDelete\_Click(object sender, EventArgs e)

{

}

private void txtBuildingName\_TextChanged(object sender, EventArgs e)

{

}

private void btnAdd\_Click\_1(object sender, EventArgs e)

{

insertData();

}

private void insertData()

{

string query = "insert into Building(name,hasDep) values ('" + txtBuildingName.Text + "',' " + comboBox.SelectedItem.ToString() + " ') ";

try

{

MySqlConnection conn = new MySqlConnection(connectionStr);

conn.Open();

MySqlCommand cmd = new MySqlCommand(query, conn);

cmd.ExecuteNonQuery();

conn.Close();

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void btnGoBack\_Click\_1(object sender, EventArgs e)

{

new Form2().Show();

this.Hide();

}

private void btnDelete\_Click\_1(object sender, EventArgs e)

{

MySqlConnection con = new MySqlConnection(connectionStr);

MySqlCommand com = new MySqlCommand();

if(dgv.Rows.Count > 1 && dgv.SelectedRows[0].Index != dgv.Rows.Count - 1)

{

com.CommandText = "Delete from Building where Id=" + dgv.SelectedRows[0].Cells[0].Value.ToString() + "" ;

con.Open();

com.Connection = con;

com.ExecuteNonQuery();

con.Close();

dgv.Rows.RemoveAt(dgv.SelectedRows[0].Index);

MessageBox.Show("Record Successfully Deleted");

}

loadData();

}

private void btnShow\_Click(object sender, EventArgs e)

{

loadData();

}

private void loadData()

{

string sql = "Select \* from Building;";

MySqlConnection conn = new MySqlConnection("Server = 35.196.198.132; Database = RoomReserve1; Uid = root; Pwd = root123; ");

MySqlDataAdapter adp = new MySqlDataAdapter(sql, conn);

DataSet dt = new DataSet();

conn.Open();

adp.Fill(dt, "Building");

conn.Close();

dgv.DataSource = dt;

dgv.DataMember = "Building";

}

private void btnUpdate\_Click(object sender, EventArgs e)

{

MySqlConnection con = new MySqlConnection(connectionStr);

con.Open();

string str = "Update Building set name = ' " +txtBuildingName.Text + "' , hasDep = ' " +comboBox.SelectedItem.ToString() +

"' where Id = " + dgv.SelectedRows[0].Cells[0].Value.ToString() + "";

MySqlCommand com = new MySqlCommand(str, con);

com.ExecuteNonQuery();

MessageBox.Show("Record Successfully Updated");

con.Close();

loadData();

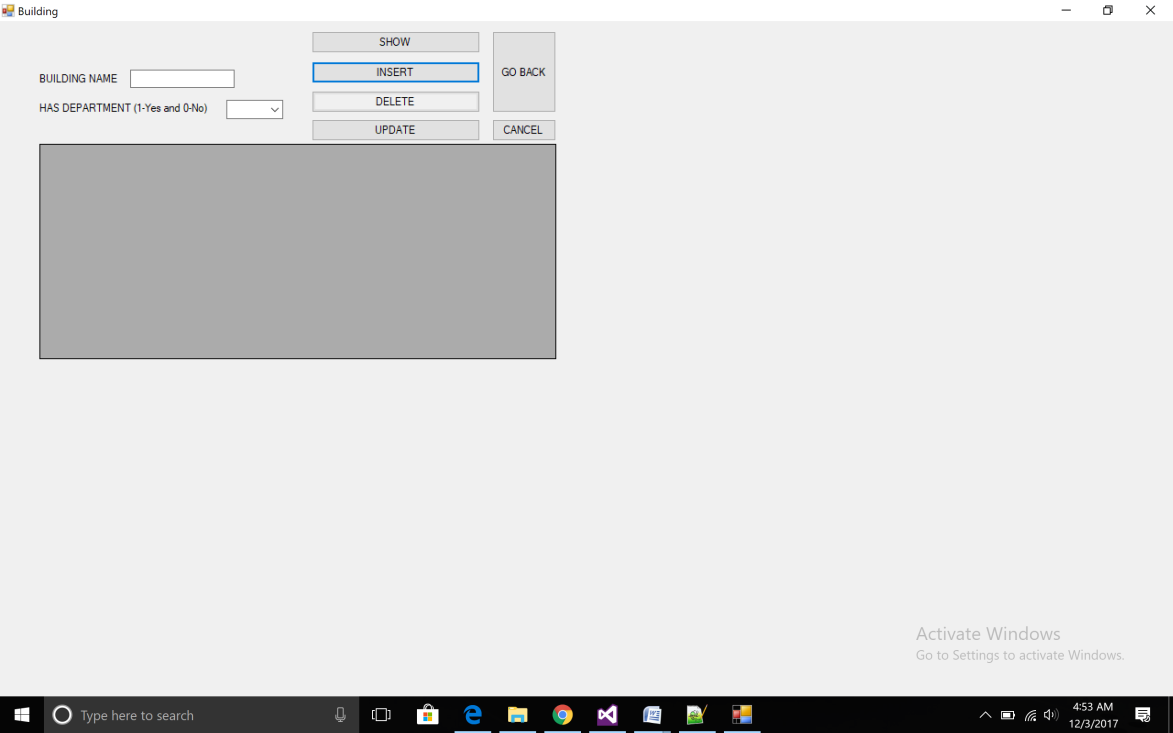
ClearData();

}

}

}

Output:



Department.cs

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace adbProject

{

public partial class Department : Form

{

private string conn = "Server=35.196.198.132;Database=RoomReserve1;Uid=root;Pwd=root123;";

public Department()

{

InitializeComponent();

}

void GridFill()

{

using (MySqlConnection mysqlCon = new MySqlConnection(conn))

{

mysqlCon.Open();

MySqlDataAdapter sqlDa = new MySqlDataAdapter("DepartmentViewAll", mysqlCon);

sqlDa.SelectCommand.CommandType = CommandType.StoredProcedure;

DataTable dtblBook = new DataTable();

sqlDa.Fill(dtblBook);

dgv.DataSource = dtblBook;

dgv.Columns[0].Visible = true;

}

}

void Clear()

{

txtID.Text = txtName.Text = comboBox.Text = txtSearch.Text = txtLocation.Text = "";

// btnAdd.Text = "Save";

btnDelete.Enabled = false;

}

private void Department\_Load(object sender, EventArgs e)

{

Clear();

GridFill();

}

private void btnAdd\_Click(object sender, EventArgs e)

{

using (MySqlConnection mysqlCon = new MySqlConnection(conn))

{

mysqlCon.Open();

MySqlCommand mySqlCmd = new MySqlCommand("DepartmentAddOrEdit", mysqlCon);

mySqlCmd.CommandType = CommandType.StoredProcedure;

mySqlCmd.Parameters.AddWithValue("\_DepartmentID", txtID.Text.Trim());

mySqlCmd.Parameters.AddWithValue("\_Name", txtName.Text.Trim());

mySqlCmd.Parameters.AddWithValue("\_Location", txtLocation.Text.Trim());

mySqlCmd.Parameters.AddWithValue("\_HasLab", comboBox.SelectedItem.ToString().Trim());

mySqlCmd.ExecuteNonQuery();

MessageBox.Show("Submitted Successfully");

Clear();

GridFill();

}

}

private void txtID\_TextChanged(object sender, EventArgs e)

{

}

private void btnDelete\_Click(object sender, EventArgs e)

{

using (MySqlConnection mysqlCon = new MySqlConnection(conn))

{

mysqlCon.Open();

MySqlCommand mySqlCmd = new MySqlCommand("DepartmentDeleteByID", mysqlCon);

mySqlCmd.CommandType = CommandType.StoredProcedure;

mySqlCmd.Parameters.AddWithValue("\_DepartmentID", txtID);

mySqlCmd.ExecuteNonQuery();

MessageBox.Show("Deleted Successfully");

Clear();

GridFill();

}

}

private void btnCancel\_Click(object sender, EventArgs e)

{

Clear();

}

private void btnGoBack\_Click(object sender, EventArgs e)

{

new Form2().Show();

this.Hide();

}

private void dgvBook\_DoubleClick(object sender, EventArgs e)

{

if (dgv.CurrentRow.Index != -1)

{

txtID.Text = dgv.CurrentRow.Cells[0].Value.ToString();

txtName.Text = dgv.CurrentRow.Cells[1].Value.ToString();

comboBox.Text = dgv.CurrentRow.Cells[3].Value.ToString();

txtLocation.Text = dgv.CurrentRow.Cells[2].Value.ToString();

// comboBox.SelectedItem.ToString().Trim() = dgv.CurrentRow.Cells[2].Value.ToString();

// bookID = Convert.ToInt32(dgv.CurrentRow.Cells[0].Value.ToString());

btnAdd.Text = "Update";

btnDelete.Enabled = Enabled;

}

}

private void btnSearch\_Click(object sender, EventArgs e)

{

using (MySqlConnection mysqlCon = new MySqlConnection(conn))

{

mysqlCon.Open();

MySqlDataAdapter sqlDa = new MySqlDataAdapter("DepartmentSearByValue", mysqlCon);

sqlDa.SelectCommand.CommandType = CommandType.StoredProcedure;

sqlDa.SelectCommand.Parameters.AddWithValue("\_SearchValue", txtSearch.Text);

DataTable dtblBook = new DataTable();

sqlDa.Fill(dtblBook);

dgv.DataSource = dtblBook;

dgv.Columns[0].Visible = true;

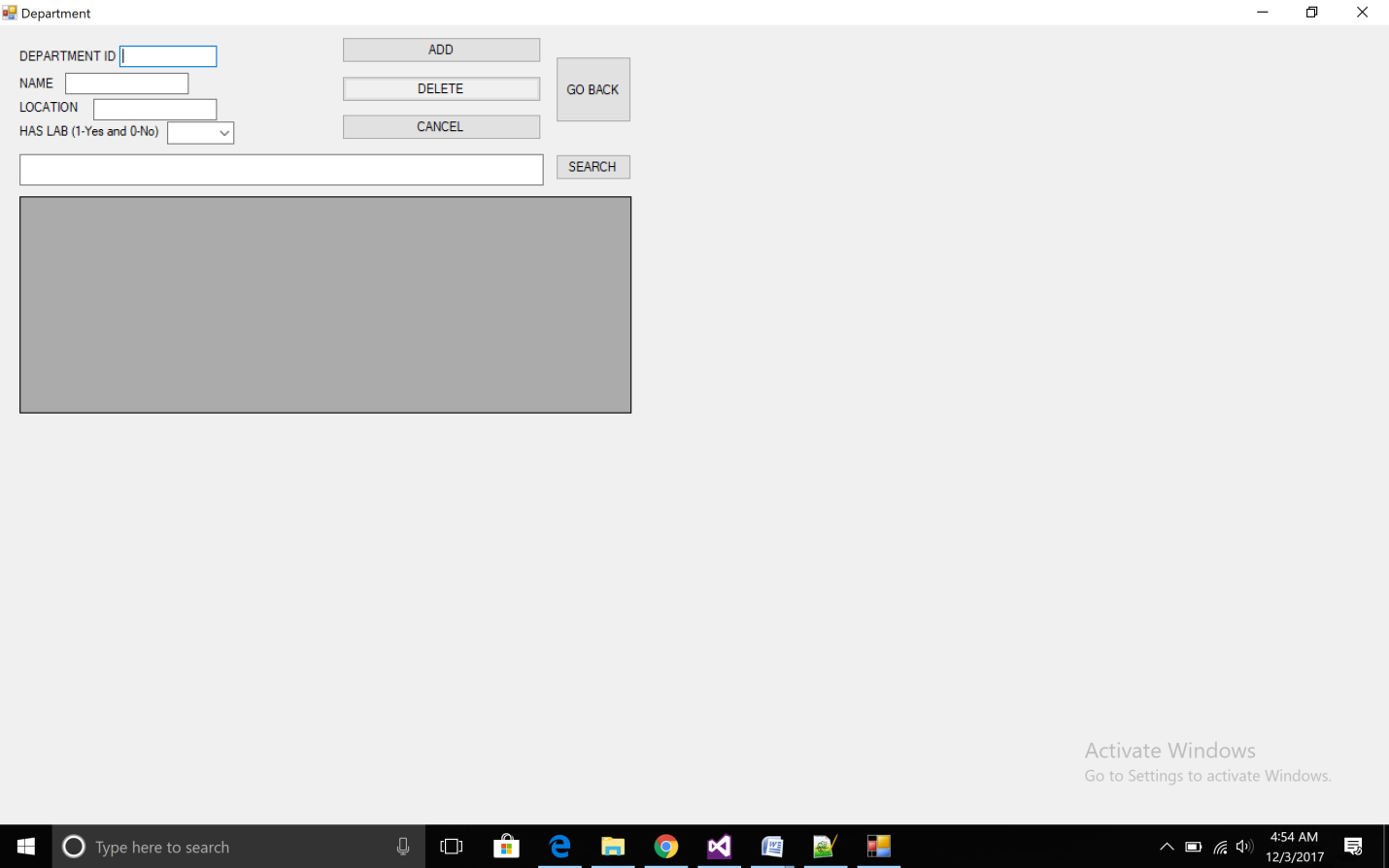
}

}

}

}

Output:



Equipment.cs

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace adbProject

{

public partial class Equipment : Form

{

private string conn = "Server=35.196.198.132;Database=RoomReserve1;Uid=root;Pwd=root123;";

public Equipment()

{

InitializeComponent();

}

void GridFill()

{

using (MySqlConnection mysqlCon = new MySqlConnection(conn))

{

mysqlCon.Open();

MySqlDataAdapter sqlDa = new MySqlDataAdapter("EquipmentViewAll", mysqlCon);

sqlDa.SelectCommand.CommandType = CommandType.StoredProcedure;

DataTable dtblBook = new DataTable();

sqlDa.Fill(dtblBook);

dgv.DataSource = dtblBook;

// dgv.Columns[0].Visible = false;

}

}

void Clear()

{

txtID.Text = txtType.Text = txtSearch.Text = "";

btnDelete.Enabled = false;

}

private void Equipment\_Load(object sender, EventArgs e)

{

Clear();

GridFill();

}

private void btnGoBack\_Click(object sender, EventArgs e)

{

new Form2().Show();

this.Hide();

}

private void btnCancel\_Click(object sender, EventArgs e)

{

Clear();

}

private void dgvBook\_DoubleClick(object sender, EventArgs e)

{

if (dgv.CurrentRow.Index != -1)

{

txtID.Text = dgv.CurrentRow.Cells[0].Value.ToString();

txtType.Text = dgv.CurrentRow.Cells[1].Value.ToString();

// comboBox.Text = dgv.CurrentRow.Cells[3].Value.ToString();

// txtLocation.Text = dgv.CurrentRow.Cells[2].Value.ToString();

// comboBox.SelectedItem.ToString().Trim() = dgv.CurrentRow.Cells[2].Value.ToString();

// bookID = Convert.ToInt32(dgv.CurrentRow.Cells[0].Value.ToString());

btnAdd.Text = "Update";

btnDelete.Enabled = Enabled;

}

}

private void btnSearch\_Click(object sender, EventArgs e)

{

using (MySqlConnection mysqlCon = new MySqlConnection(conn))

{

mysqlCon.Open();

MySqlDataAdapter sqlDa = new MySqlDataAdapter("EquipmentSearch", mysqlCon);

sqlDa.SelectCommand.CommandType = CommandType.StoredProcedure;

sqlDa.SelectCommand.Parameters.AddWithValue("\_SearchValue", txtSearch.Text);

DataTable dtblBook = new DataTable();

sqlDa.Fill(dtblBook);

dgv.DataSource = dtblBook;

// dgv.Columns[0].Visible = false;

}

}

private void btnDelete\_Click(object sender, EventArgs e)

{

using (MySqlConnection mysqlCon = new MySqlConnection(conn))

{

mysqlCon.Open();

MySqlCommand mySqlCmd = new MySqlCommand("EquipmentDeleteByID", mysqlCon);

mySqlCmd.CommandType = CommandType.StoredProcedure;

mySqlCmd.Parameters.AddWithValue("\_ID", txtID);

mySqlCmd.ExecuteNonQuery();

MessageBox.Show("Deleted Successfully");

Clear();

GridFill();

}

}

private void btnAdd\_Click(object sender, EventArgs e)

{

using (MySqlConnection mysqlCon = new MySqlConnection(conn))

{

mysqlCon.Open();

MySqlCommand mySqlCmd = new MySqlCommand("EquipmentAddorEdit", mysqlCon);

mySqlCmd.CommandType = CommandType.StoredProcedure;

mySqlCmd.Parameters.AddWithValue("\_ID", txtID.Text.Trim());

mySqlCmd.Parameters.AddWithValue("\_Type", txtType.Text.Trim());

mySqlCmd.ExecuteNonQuery();

MessageBox.Show("Submitted Successfully");

Clear();

GridFill();

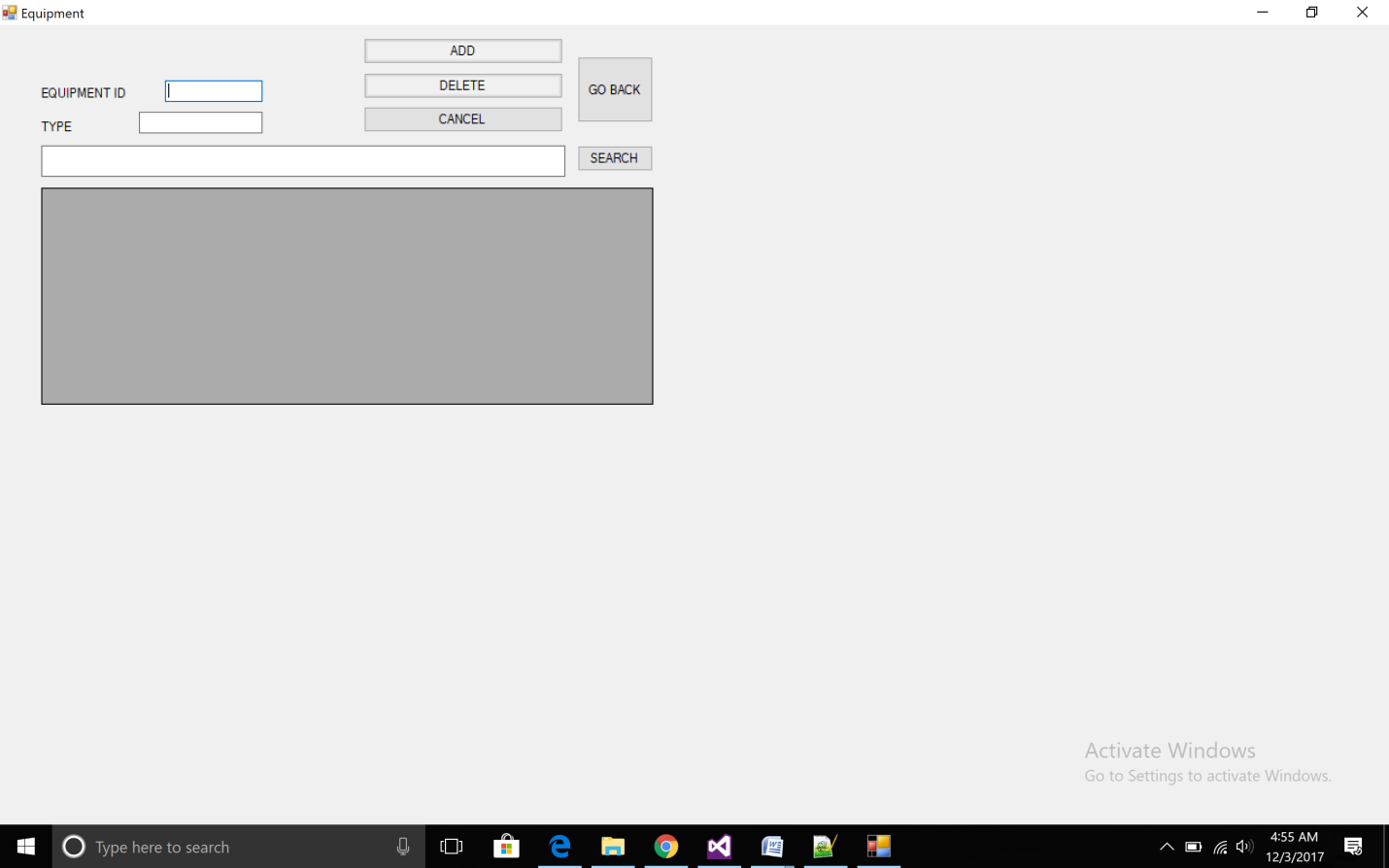
}

}

}

}

Output:



Ratings.cs

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace adbProject

{

public partial class Ratings : Form

{

string conn = "Server=35.196.198.132;Database=RoomReserve1;Uid=root;Pwd=root123;";

public Ratings()

{

InitializeComponent();

}

void GridFill()

{

using (MySqlConnection mysqlCon = new MySqlConnection(conn))

{

mysqlCon.Open();

MySqlDataAdapter sqlDa = new MySqlDataAdapter("RatingViewAll", mysqlCon);

sqlDa.SelectCommand.CommandType = CommandType.StoredProcedure;

DataTable dtblBook = new DataTable();

sqlDa.Fill(dtblBook);

dgv.DataSource = dtblBook;

// dgv.Columns[0].Visible = false;

}

}

void Clear()

{

txtID.Text = txtDesc.Text = txtSearch.Text = "";

btnDelete.Enabled = false;

}

private void Ratings\_Load(object sender, EventArgs e)

{

Clear();

GridFill();

}

private void btnDelete\_Click(object sender, EventArgs e)

{

using (MySqlConnection mysqlCon = new MySqlConnection(conn))

{

mysqlCon.Open();

MySqlCommand mySqlCmd = new MySqlCommand("RatingDeleteByID", mysqlCon);

mySqlCmd.CommandType = CommandType.StoredProcedure;

mySqlCmd.Parameters.AddWithValue("\_RatingID", txtID);

mySqlCmd.ExecuteNonQuery();

MessageBox.Show("Deleted Successfully");

Clear();

GridFill();

}

}

private void btnCancel\_Click(object sender, EventArgs e)

{

Clear();

}

private void btnGoBack\_Click(object sender, EventArgs e)

{

new Form2().Show();

this.Hide();

}

private void dgvBook\_DoubleClick(object sender, EventArgs e)

{

if (dgv.CurrentRow.Index != -1)

{

txtID.Text = dgv.CurrentRow.Cells[0].Value.ToString();

txtDesc.Text = dgv.CurrentRow.Cells[1].Value.ToString();

// comboBox.Text = dgv.CurrentRow.Cells[3].Value.ToString();

// txtLocation.Text = dgv.CurrentRow.Cells[2].Value.ToString();

// comboBox.SelectedItem.ToString().Trim() = dgv.CurrentRow.Cells[2].Value.ToString();

// bookID = Convert.ToInt32(dgv.CurrentRow.Cells[0].Value.ToString());

btnAdd.Text = "Update";

btnDelete.Enabled = Enabled;

}

}

private void btnSearch\_Click(object sender, EventArgs e)

{

using (MySqlConnection mysqlCon = new MySqlConnection(conn))

{

mysqlCon.Open();

MySqlDataAdapter sqlDa = new MySqlDataAdapter("RatingSearch", mysqlCon);

sqlDa.SelectCommand.CommandType = CommandType.StoredProcedure;

sqlDa.SelectCommand.Parameters.AddWithValue("\_SearchValue", txtSearch.Text);

DataTable dtblBook = new DataTable();

sqlDa.Fill(dtblBook);

dgv.DataSource = dtblBook;

// dgv.Columns[0].Visible = false;

}

}

private void btnAdd\_Click(object sender, EventArgs e)

{

using (MySqlConnection mysqlCon = new MySqlConnection(conn))

{

mysqlCon.Open();

MySqlCommand mySqlCmd = new MySqlCommand("RatingAddOrEdit", mysqlCon);

mySqlCmd.CommandType = CommandType.StoredProcedure;

mySqlCmd.Parameters.AddWithValue("\_RatingID", txtID.Text.Trim());

mySqlCmd.Parameters.AddWithValue("\_Description", txtDesc.Text.Trim());

mySqlCmd.ExecuteNonQuery();

MessageBox.Show("Submitted Successfully");

Clear();

GridFill();

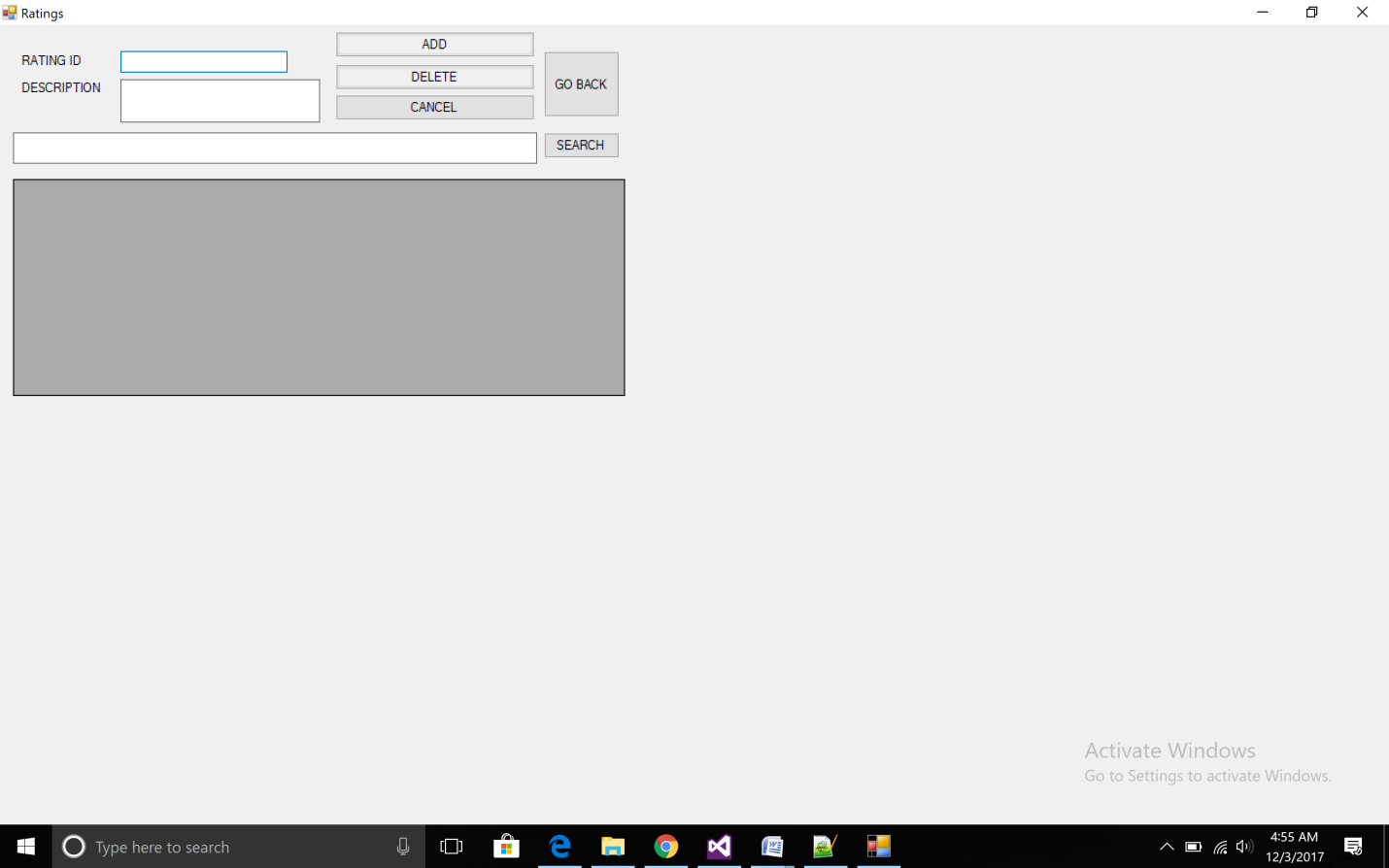
}

}

}

}

Output:



Rooms.cs

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace adbProject

{

public partial class Rooms : Form

{

string conn = "Server=35.196.198.132;Database=RoomReserve1;Uid=root;Pwd=root123;";

public Rooms()

{

InitializeComponent();

}

void GridFill()

{

using (MySqlConnection mysqlCon = new MySqlConnection(conn))

{

mysqlCon.Open();

MySqlDataAdapter sqlDa = new MySqlDataAdapter("RoomViewAll", mysqlCon);

sqlDa.SelectCommand.CommandType = CommandType.StoredProcedure;

DataTable dtblBook = new DataTable();

sqlDa.Fill(dtblBook);

dgv.DataSource = dtblBook;

// dgv.Columns[0].Visible = false;

}

}

void Clear()

{

txtID.Text = txtRNumber.Text = txtSearch.Text = "";

btnDelete.Enabled = false;

}

private void Rooms\_Load(object sender, EventArgs e)

{

Clear();

GridFill();

}

private void btnDelete\_Click(object sender, EventArgs e)

{

using (MySqlConnection mysqlCon = new MySqlConnection(conn))

{

mysqlCon.Open();

MySqlCommand mySqlCmd = new MySqlCommand("RoomDeleteByID", mysqlCon);

mySqlCmd.CommandType = CommandType.StoredProcedure;

mySqlCmd.Parameters.AddWithValue("\_RoomID", txtID);

mySqlCmd.ExecuteNonQuery();

MessageBox.Show("Deleted Successfully");

Clear();

GridFill();

}

}

private void btnGoBack\_Click(object sender, EventArgs e)

{

new Form2().Show();

this.Hide();

}

private void btnSearch\_Click(object sender, EventArgs e)

{

using (MySqlConnection mysqlCon = new MySqlConnection(conn))

{

mysqlCon.Open();

MySqlDataAdapter sqlDa = new MySqlDataAdapter("RoomSearch", mysqlCon);

sqlDa.SelectCommand.CommandType = CommandType.StoredProcedure;

sqlDa.SelectCommand.Parameters.AddWithValue("\_SearchValue", txtSearch.Text);

DataTable dtblBook = new DataTable();

sqlDa.Fill(dtblBook);

dgv.DataSource = dtblBook;

// dgv.Columns[0].Visible = false;

}

}

private void btnCancel\_Click(object sender, EventArgs e)

{

Clear();

}

private void dgvBook\_DoubleClick(object sender, EventArgs e)

{

if (dgv.CurrentRow.Index != -1)

{

txtID.Text = dgv.CurrentRow.Cells[0].Value.ToString();

txtRNumber.Text = dgv.CurrentRow.Cells[1].Value.ToString();

txtRoomType.Text = dgv.CurrentRow.Cells[2].Value.ToString();

comboBox2.Text = dgv.CurrentRow.Cells[3].Value.ToString();

// txtRNumber.Text = dgv.CurrentRow.Cells[2].Value.ToString();

// comboBox.SelectedItem.ToString().Trim() = dgv.CurrentRow.Cells[2].Value.ToString();

// bookID = Convert.ToInt32(dgv.CurrentRow.Cells[0].Value.ToString());

btnAdd.Text = "Update";

btnDelete.Enabled = Enabled;

}

}

private void btnAdd\_Click(object sender, EventArgs e)

{

using (MySqlConnection mysqlCon = new MySqlConnection(conn))

{

mysqlCon.Open();

MySqlCommand mySqlCmd = new MySqlCommand("RoomAddOrEdit", mysqlCon);

mySqlCmd.CommandType = CommandType.StoredProcedure;

mySqlCmd.Parameters.AddWithValue("\_RoomID", txtID.Text.Trim());

mySqlCmd.Parameters.AddWithValue("\_Number", txtRNumber.Text.Trim());

mySqlCmd.Parameters.AddWithValue("\_Type", txtRoomType.Text.Trim());

mySqlCmd.Parameters.AddWithValue("\_DepID", txtRNumber.Text.Trim());

mySqlCmd.Parameters.AddWithValue("\_HasEquipment", comboBox2.SelectedItem.ToString().Trim());

mySqlCmd.ExecuteNonQuery();

MessageBox.Show("Submitted Successfully");

Clear();

GridFill();

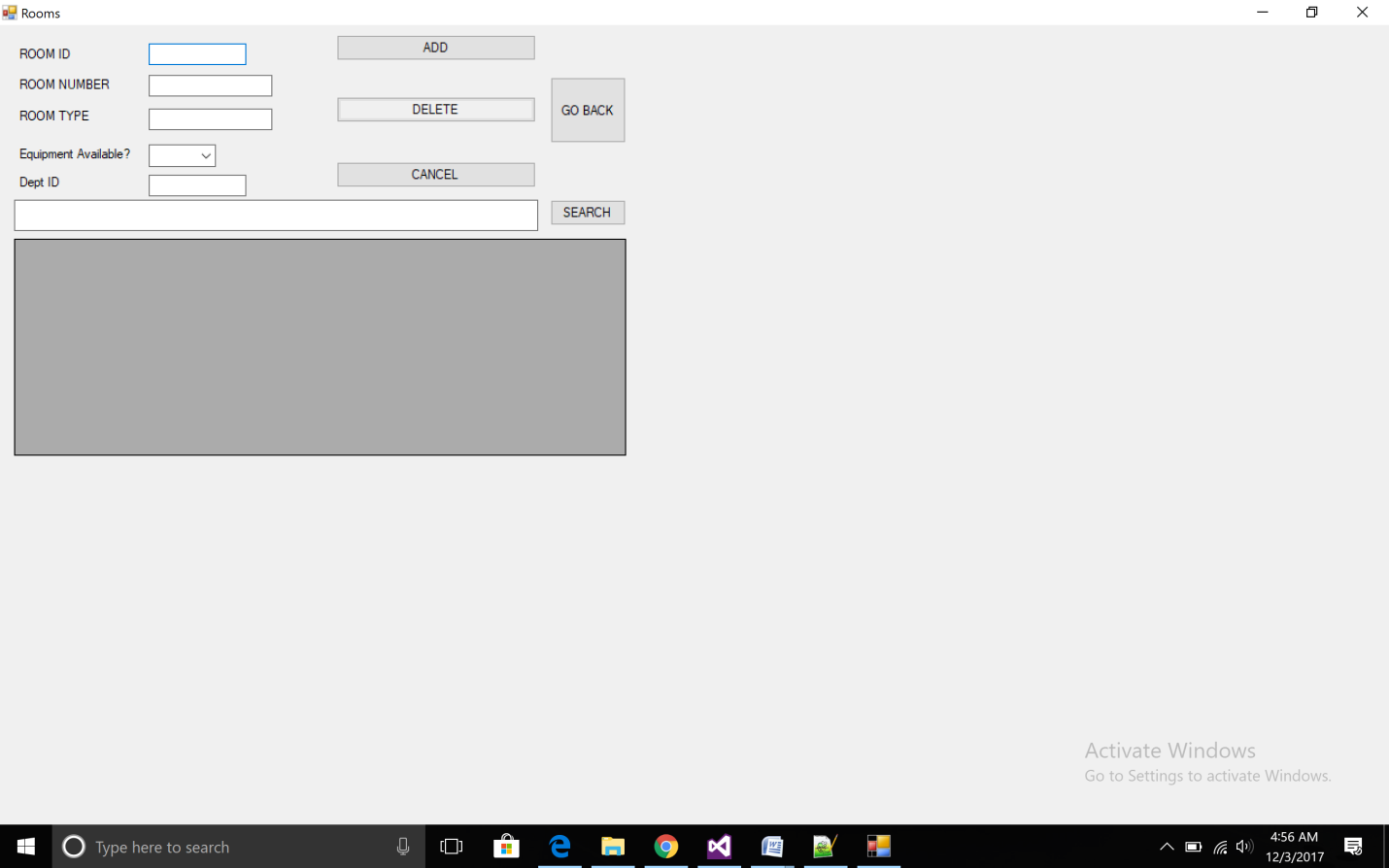
}

}

}

}

Output:



Reservations.cs

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace adbProject

{

public partial class Reservations : Form

{

string conn = "Server=35.196.198.132;Database=RoomReserve1;Uid=root;Pwd=root123;";

public Reservations()

{

InitializeComponent();

}

void GridFill()

{

using (MySqlConnection mysqlCon = new MySqlConnection(conn))

{

mysqlCon.Open();

MySqlDataAdapter sqlDa = new MySqlDataAdapter("ReservationViewAll", mysqlCon);

sqlDa.SelectCommand.CommandType = CommandType.StoredProcedure;

DataTable dtblBook = new DataTable();

sqlDa.Fill(dtblBook);

dgv.DataSource = dtblBook;

// dgv.Columns[0].Visible = false;

}

}

void Clear()

{

txtID.Text = "";

btnDelete.Enabled = false;

}

private void Reservations\_Load(object sender, EventArgs e)

{

Clear();

GridFill();

}

private void btnDelete\_Click(object sender, EventArgs e)

{

using (MySqlConnection mysqlCon = new MySqlConnection(conn))

{

mysqlCon.Open();

MySqlCommand mySqlCmd = new MySqlCommand("ReservationDeleteByID", mysqlCon);

mySqlCmd.CommandType = CommandType.StoredProcedure;

mySqlCmd.Parameters.AddWithValue("\_ReservationID", txtID);

mySqlCmd.ExecuteNonQuery();

MessageBox.Show("Deleted Successfully");

Clear();

GridFill();

}

}

private void dgvBook\_DoubleClick(object sender, EventArgs e)

{

if (dgv.CurrentRow.Index != -1)

{

txtID.Text = dgv.CurrentRow.Cells[0].Value.ToString();

// txtBuildingName.Text = dgv.CurrentRow.Cells[1].Value.ToString();

// comboBox.SelectedItem.ToString().Trim() = dgv.CurrentRow.Cells[2].Value.ToString();

// bookID = Convert.ToInt32(dgv.CurrentRow.Cells[0].Value.ToString());

// btnAdd.Text = "Update";

btnDelete.Enabled = Enabled;

}

}

private void btnCancel\_Click(object sender, EventArgs e)

{

Clear();

}

private void btnGoBack\_Click(object sender, EventArgs e)

{

new Form2().Show();

this.Hide();

}

private void btnSearch\_Click(object sender, EventArgs e)

{

using (MySqlConnection mysqlCon = new MySqlConnection(conn))

{

mysqlCon.Open();

MySqlDataAdapter sqlDa = new MySqlDataAdapter("ReservationSearchByValue", mysqlCon);

sqlDa.SelectCommand.CommandType = CommandType.StoredProcedure;

sqlDa.SelectCommand.Parameters.AddWithValue("\_SearchValue", txtSearch.Text);

DataTable dtblBook = new DataTable();

sqlDa.Fill(dtblBook);

dgv.DataSource = dtblBook;

// dgv.Columns[0].Visible = false;

}

}

private void btnAdd\_Click(object sender, EventArgs e)

{

using (MySqlConnection mysqlCon = new MySqlConnection(conn))

{

mysqlCon.Open();

MySqlCommand mySqlCmd = new MySqlCommand("ReservationAddOrEdit", mysqlCon);

mySqlCmd.CommandType = CommandType.StoredProcedure;

mySqlCmd.Parameters.AddWithValue("\_ResID", txtID.Text.Trim());

// mySqlCmd.Parameters.AddWithValue("\_InDate", txtID.Text.Trim());

//mySqlCmd.Parameters.AddWithValue("\_OutDate", txtID.Text.Trim());

mySqlCmd.ExecuteNonQuery();

MessageBox.Show("Submitted Successfully");

Clear();

GridFill();

}

}

private void dgv\_CellContentClick(object sender, DataGridViewCellEventArgs e)

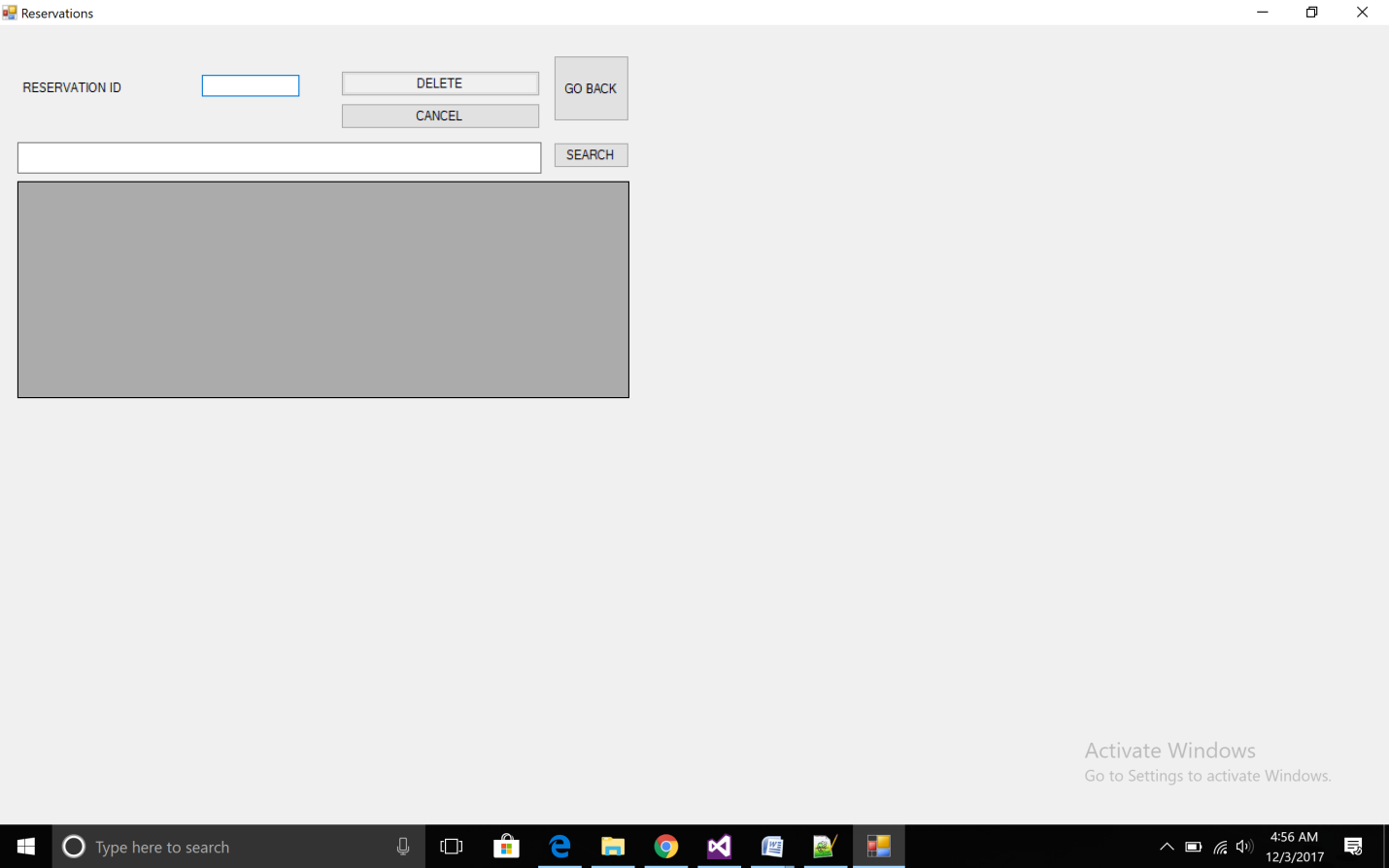
{

}

}

}

Output:



**Conclusion**

There are no information systems that are currently being implemented and the RRS is a project will be replacing the reservation system of the DCS, which is currently being implemented manually by people, building admin. The Room Reservation System will aim to be self-contained, meaning that it will have no dependencies to other existing information system. It is expected that more functionalities will be included, or that this system might be extended. The RRS will be implemented and tested manually however, if successful, it can be done be deployed in the entire college.

There are over ten entities and relationships in the database for reserving or booking the room in the university. Designing the database and assigning the relationship to entities is really a unique and important task which is needed in order to have the clear conceptual view of the Database. There is also aggregation involved and also few attributes which are derived from the relations. In order to improve this, I may introduce few other relationship between entities in future which would help in the structure of the database and conceptual as well as physical view. Relational Algebra will also be introduced to the model and assumptions made will be implemented. It is expected that more functionalities will be included, or that this application might be extended. This application will be implemented and tested and then further can be deployed in the entire university.

In this phase, I setup database server and web server using GCP. I created Google Compute Engine for the database server where RDBMS has been selected and installed. On the other hand, Google App Engine has been created for the web application server for the selected web application server. Physical Database Design and Implementation leads to proper understanding of structured Query Language with the proper hands on knowledge on Constraints, Objects and other Functional capabilities of SQL.