

- Build a Booking & Scheduling System -

Project Report

MIS 531 - 002

LIGHT^HOUSE Team:

A.H., Yazan

Du, Po-Yi

Li, Xiao

Luo, Xi

Shentu, Nan

Wang, Mingda

Table of Contents

| | |
|-----------------|----|
| Chapter 1 | 2 |
| Chapter 2 | 4 |
| Chapter 3 | 10 |
| Chapter 4 | 34 |
| Chapter 5 | 60 |
| Chapter 6 | 74 |
| Chapter 7 | 88 |

Chapter 1.

Our client is the Arizona Wildcat Academic Center (abbreviate that by calling it C.A.T.S. Academics in the rest of report). The normal goal of C.A.T.S Academics is help student-athletes graduate and there are more than 500 student-athletes belong C.A.T.S Academics. Therefore, the department provides these students a good place to study and offer all possible academic supports and resources, such as tutoring services, to student athletes. This October, the department moved to a bigger building with more various purpose rooms and then want to hire more amounts of high-quality workers. They intend to specify the correct purpose of each room in this new building. In addition, they want to track **Tutors**, for examples, keeping records whether the tutor attends the tutoring appointments or not and the total number of working hours. Considered these new demands, they want to setup a new online booking & scheduling system. The mission of our team is to help C.A.T.S. Academics create a database system and front-end booking & scheduling procedure.

The main **Users** of system include: staffs (Administrators, Counselors, Front Desks, and Learning Specialists) and students. Based on their roles, these users will be able to access different features of the system. And we are achieving to offer different level of login access based on user authority. The users that access an unpermitted section of the site must be bounced back to the front page or redirected somewhere else. The department wants to record every staff's First Name, Last Name, Password, Email, Phone Number, User Type, Street Address, City, State, and Zip Code. Otherwise, every staff has a unique User ID. Users are composited to **User Type**. Each User Type records the unique Usertype_ID and Number of users.

Each student has a student ID. C.A.T.S wants to track the **courses** that students are taking, and the **professors** who are teaching the courses. There is a course schedule recorded the corresponding section Number. Course has a course ID. C.A.T.S wants to know the title of courses that students are taking as well as the description of the course. Further, C.A.T.S would like to know students' attendance to courses. The *attendance record* the record_ID and details. Professor has a university employee ID. C.A.T.S would like to track to know the professor's full name, the department that a given professor is in. Student's Sport Type and Academic Standing should be recorded in the database.

In addition, C.A.T.S. Academics require make a variety of **Bookings**. Hence, there will be four types of Room Booking in the Booking Type: room reservation for self-studying, tutoring appointments to improve students learning skills, training sessions among the learning specialists and tutors, and special events that could block any rooms anytime. Also, they want to record the utilization of each room, and keep tracks of the users who reserve and check-in to each room in details. Student alone can make reservations for self-study purpose, however, student must talk to his/her counselor to make tutor appointments. The Bookings are composited to **Booking Type**. Each booking Type records the unique BookingType_ID and number of bookings. Booking has a booking ID. C.A.T.S wants to track booking types, duration, start time, end time, semester, booking date and status (e.g: on time, canceled, no show and updated). For all kinds of booking, users must check-in at the front desk to confirm their attendance for their booking. However, students can self-check in their self-studying reservations. C.A.T.S. Academics staffs (Administrators, Counselors, Front Desks, and Learning Specialists) will also be able to make room reservations based on the availability of rooms for students. A reservation created by a

student can have a maximum duration of two hours, but can be as short as 30 minutes. Students can only make one reservation at the same period. However, Staffs except front desks can make several reservations at a time with no limit on duration, adjacent reservations, or time in advance. Each booking must correspond only one room. The period of reservation can be if two hours. One shorter than two-hour reservation can be extended two hours based on rooms' availability. A reservation must be booked for at least 30 minutes. Students must check-in at the front desk to start their reservation session. Students can also early check-out for their reservations. If C.A.T.S staff decided to book rooms based on **events**, existing reservations on the room at that time will be cancelled, and students will be notified via email.

C.A.T.S has a group of administrative staff. Administrators can access all features of the system, e.g: administrators can create reservations without restrictions. Administrator can create events in the system. Events will occupy rooms.

Counselors guide the **sports**. Sports expertise students. C.A.T.S wants to record the sports name and coach of sports. Counselors will be assisting students in finding a tutor. Counselors will help student in making appointments with tutors based on student's requests or if the Learning specialist determined the student is struggling with his/her coursework. Counselors will collect students' availabilities, and pass that along to Learning specialist. Learning specialist are responsible for communicating with tutors, and reserve rooms for appointments. Learning specialist charge the subjects to find the related tutors. Counselors can also see their corresponding students of current reservations and appointments.

Tutor has a tutor ID. C.A.T.S will track tutors' first names, last names, and academic standings. Tutors need to participate both training sections and appointments. Tutors will provide their available tutoring schedules and competent courses at the beginning of semester. Based on students' and tutors' schedule and subject, learning specialists match students and tutors, which result in tutor appointments. Based on the university's policy, the maximum working hours for each tutor are 20 hours per week. Once a given tutor has worked 20 hours in a week, tutor coordinators will not match that tutor with other students. Each tutor may have one **subject** or subjects that he/she may be comfortable tutoring. C.A.T.S wants to also track what are the level of tutor skill in the *skill level*. Subject has a subject name and description. Subjects belong to courses.

Learning specialists are responsible for matching the needs of students with tutors' availabilities. When tutor coordinators receive tutoring requests from academic counselors, they will check tutors' availabilities through their existing system, GradesFirst. The learning specialists will then arrange tutor appointments for both students and tutors. A tutoring appointment results in a reservation for a room. C.A.T.S will also track the tutor ID and subjects for appointments. If the students or tutors request to change scheduled appointments, they need to contact a learning specialist to adjust appointments.

Tutor coordinators can also create reservations for staff meetings, or cancel any reservations they created. Further, coordinator can create the **training sessions** for the entry-level tutors. Training session has a training title. We want to record the contents in *training record*.

The employee at the front desk can also create reservations or cancel reservations per students' requests. For front desks, C.A.T.S would like to know their *daily work records*. In the daily work records, the system will journalize the date and contents for the corresponding front desk staffs. The **rooms** in the C.A.T.S learning center varies in sizes, room availability and room type. There are no rooms share the same room number. C.A.T.S can track task, schedule date, start time and end time in the **Room Schedule**. The system provides them a real-time room calendar. **Room classification** has a unique room type name. For each room type, C.A.T.S wants to track total number of rooms in each room type.

The system should also track the equipment in rooms, such as, electronics, desks, chairs, etc. Administrators are responsible for keeping the room utility information up to date. The equipment is also instantiated from equipment type. **Equipment type** has a type name, description, and total number of equipment in each equipment type. Equipment id will track the **Equipment**. C.A.T.S would also like to track the equipment type and equipment condition for an equipment instance.

Chapter 2.

A revised version of your conceptual schema (ER diagram) along with a data dictionary describing the semantics of each entity class. The data dictionary is the place to clarify units (such as "salary refers to salary per month") and abbreviations used. Cardinality (beyond diagram) and other integrity constraints should also be listed in the dictionary.

Our ERD has 13 strong entities, 10 subclasses and 6 weak entities. All entities are listed below:

- Strong entities:
- Users
- Professors
- Courses
- Sports
- Bookings
- EquipmentType
- RoomEquipment
- Rooms
- Tutors
- Subjects
- UserType
- BookingType
- RoomClassification

Subclasses:

- Students
- Staff
- Counselors
- Administrators
- FrontDesks

- LearningSpecialists
- Events
- Reservations
- Training-Sessions
- Appointments

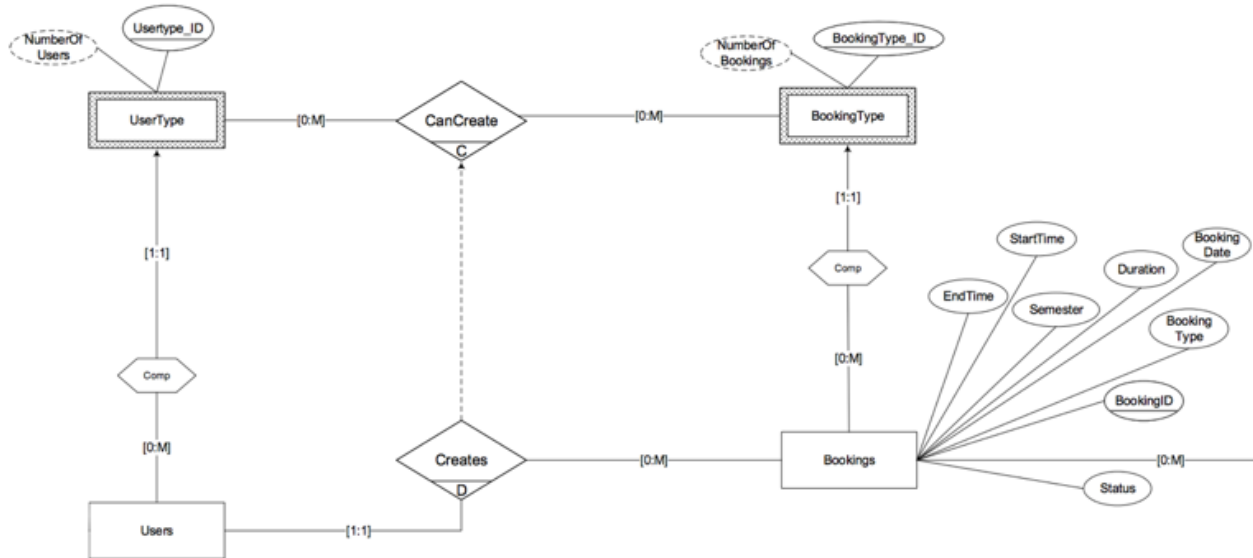
Weak entities:

- RoomSchedule
- TrainingRecord
- AttendanceRecord
- CourseSchedule
- DailyWorkRecord
- SkillLevel

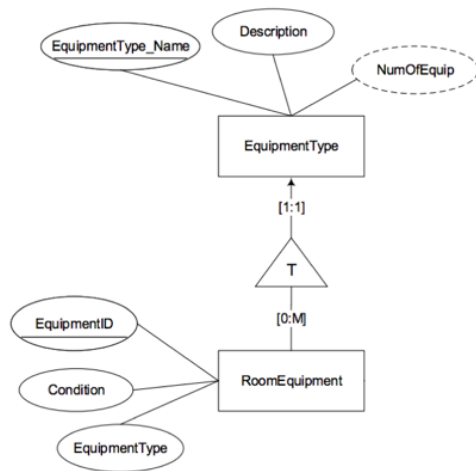
Some key parts of our ERD are broken down and listed below:



The previous image describes our design of the two-level subclass for all users. For each user, no matter what user type they are, the system will record their userID, usertype, FirstName, LastName, Email, Password, PhoneNumber, State, Zipcode, City, StreetAddress. Users can be classified in two categories, Student and Staff. There are four types of staff, Counselors, Administrators, FrontDesks and LearningSpecialists. In our design, students and staff have complete different power when entering the system. Students should not be able to modify the system at all. They can only make reservation for their study session. Four types of staffs have different power when making bookings. For example, only Administrators can create Events.



This image shows the design of the Create and CanCreate constrained entities. Some bookings can only be created by certain user types. The system will make sure that each booking is finished only by authorized users. For each booking, the system will record the BookingID, semester, duration, booking type, status, bookingDate, bookingType, startTime and endTime.



The image above is a typing class in our ERD. The system is recording the RoomEquipment located in each room. Equipment can be classified into different types, for example TV, chairs, cable, tables, etc. This will help the department track the condition of equipment in each room.

Conceptual Data Dictionary

| Schema Construct | Construct Description | Other Information |
|--------------------|---|---|
| ATTENDANCERECORD | Weak entity class, to model attendance record | |
| Student | Identifying user ID | Foreign key, references UserID, in STUDENTS |
| Course | Identifying course ID for the record | Foreign key, references CourseID, in COURSES |
| AttendanceDetails | Record the attendance details, e.g., 'Attended', 'Didn't attend' | Should not be NULL |
| Record_ID | Identifying the ID number of the record report | |
| ASSOCIATEDWITH | Relation representing the relationship ASSOCIATEDWITH | |
| BELONG | Relation representing the relationship BELONG | |
| BOOKINGTYPE | Composite entity class, to model booking type | |
| BookingType_ID | Identifying BookingType ID | Identifying attribute |
| NumOfBookings | Show the number of booking for the booking type | default 0, a derived attribute |
| BOOKINGS | Entity class, to model booking information | |
| BookingID | Identifying the booking ID | Identifying Attribute |
| BookingType_ID | List booking type of an booking, e.g., 'Event', 'Reservation', and more | Should not be NULL |
| Status | Record the status of Booking entity | Should not be NULL |
| Room | List the roomnumber for the booking | Foreign key references ROOMS |
| Duration | The duration of the booking | Range from 0 - 2 hours exclusively |
| Semester | Determin the semester of the Booking | Should not be NULL |
| Start_Time | Start time of the booking | Should not be NULL |
| End_Time | End time of the booking | Should not be NULL |
| Booking_Date | List name of the users that attend the booking | Should not be NULL |
| UserID | Identifying user ID of the Booking | Foreign key references USERS |
| RESERVATION | Entity class, to model reservation | |
| BookingID | Identifying the BookingID for the reservation | Foreign key, references BOOKING |
| ReservationDetails | Record the reservation details | Should not be NULL |
| APPOINTMENTS | Entity class, to model appointments information | |
| BookingID | Identifying the appointment ID | Foreign key references BOOKING |
| TutorID | Recording the name of tutor in the appointment | Foreign key references USERS |
| CONTAIN | Relationship representing the relationship CONTAIN | |
| EVENTS | Entity class, to model events information | |
| BookingID | Identify the ID number of the event | Identifying Attribute |
| Topic | List the topic of the event | Should not be NULL |
| EventsName | Record the name of the event | Should not be NULL |
| EXPERTIES | Relationship representing the relationship EXPERTIES | |
| TRAININGSESSION | Entity class, to model training session information | |
| BookingID | Identifying BookingID number | Identifying Attribute |
| Training_Title | Record the title of an training session | Should not be NULL |
| CANCREATE | Relationship representing the relationship cancreate | |
| CREATES | Relationship representing the relationship create | |
| COURSES | Entity class, to model course | |
| Course_ID | Identifying course ID | Identifying Attribute |
| Title | List the course title | Should not be NULL |
| Description | Describe the course material | |
| Subject_Name | Describe the name of the subject | Foreign key, references SUBJECTS |
| COURSESCCHEDULE | Weak entity class, to model courseschedule record | |
| Professor | Identifying ID of the professor | Foreign key references PROFESSORS |
| Course | Identifying ID of the course | Foreign key references COURSES |
| Section | Identifying section number | |
| DAILYWORKRECORD | Entity class, to model dailyworkrecord information | |
| FrontDesk_ID | Identifying the UserID of the work record | Identifying Attribute, Foreign key references USERS |
| Record_Date | Record the date of the work record | Should not be NULL |
| Contents | Record the hour of work | Range from 0 to 1 |
| EQUIPMENTTYPE | Entity class, to model room equipment type | |
| EquipmentType_Name | Identifying equipment type name | Identifying Attribute |
| Description | list the equipment type name, e.g., chair, TV, table | Should not be NULL |
| NumOfEquip | List the total number of each equipment type | range 1 - 99 |
| FORM | Relation representing the relationship FORM | |
| HOLD | Relation representing the relationship HOLD | |
| INCHARGE | Relation representing the relationship INCHARGE | |
| INTERACT | Relation representing the relationship INTERACT | |
| JOURNALIZE | Relation representing the relationship JOURNALIZE | |
| GUIDE | Relation representing the relationship GUIDE | |
| MASTER | Relation representing the relationship MASTER | |
| PARTICIPATE | Relation representing the relationship PARTICIPATE | |
| PROFESSORS | Entity class, to model professor | |
| EmployeeID | Identifying the Employee ID for the professor | Identifying Attribute |
| FName | List the first name of the professor | Should not be NULL |
| LName | List the last name of the professor | Should not be NULL |
| Department | List the course title the professor teach | Should not be NULL |
| ROOMS | Entity class, to model room | |
| RoomNumber | Identify the number of the room | Identifying Attribute |
| Room_Availability | List availability of the room. E.g., 'Y', 'N', 'L' | Should not be NULL |
| RoomSize | Record the size of the room | range 0 - 50 |

| | | |
|---------------------|---|---|
| ROOMCLASSIFICATION | Entity class, to model room classification | |
| RoomType_Name | List the name of the equipments | Should not be NULL |
| NumOfRoom | Record the status of each equipment in room | range 0 - 50 |
| ROOMEQUIPMENT | Entity class, to model room equipments | |
| EquipmentID | Identifying the equipment ID | Identifying Attribute |
| Condition | Record the condition of the equipments | Should not be NULL |
| Equipment_Type | Record the equipement type ID | Foreign key references EQUIPMENTTYPE(EquipmentType_Name) |
| Room | Identifying Room Number | Foreign key references ROOMS(RoomNumber) |
| ROOMSCHEDULE | Weak entity class, to model room schedules | |
| RoomNumber | Identifying the number of the room | Foreign key references ROOMS(RoomNumber) |
| Schedule_Date | Record the date scheduled | Partial Primary Key |
| Start_Time | Record the time the schedule starts | Should not be NULL |
| End_Time | Record the time the schedule ends | Should not be NULL |
| Task | Record the tasks for the schedule time | |
| SUBJECTS | Entity class, to model subject information | |
| SubjectName | List the subject name | Identifying Attribute |
| Description | Describe the subject | |
| SKILLLEVEL | Weak entity class, to model skill level | |
| Subject | Identifying the subject number with the skill level | Foreign key, references SubjectID, in SUBJECTS |
| Tutor | Identifying the tutor ID with the skill level | Foreign key, references TutorID, in TUTORS |
| TutorSkillLevel | Record the skill level of tutor, e.g., 'Good', 'Bad', 'Fine' | Should not be NULL |
| SPORTS | Entity class, to model sports information | |
| SportName | List the name of the sports | Identifying Attribute |
| Counselor | List the counselor ID of the sport | Foreign key references USERS --> STAFF --> COUNSELORS(USERID) |
| Coach | Identifying the coach of the sport | Should not be NULL |
| TRAININGRECORD | Weak Entity, to model training record information | |
| T_Session | Identifying the training session number | Foreign key, references Trainingnumber, in TRAININGSESSION |
| Tutor | Identifying the tutor ID | Foreign key, references TutorID, in TUTORS |
| Contents | Record Training Contents | Should not be NULL |
| TUTORS | Entity class, to model tutors information | |
| TutorID | Identifying tutor ID for the tutor | Identifying Attribute |
| Tutor_FName | List the first name of the tutor | Should not be NULL |
| Tutor_LName | List the last name of the tutor | Should not be NULL |
| AcademicStanding | Show the academic standing of the student, e.g., 'Freshman', 'Sophomore', 'Junior', 'Senior', 'Graduate', 'PHD' | Should not be NULL |
| USERTYPE | Composite entity, to model user type | |
| UserType_ID | Identifying ID of a user type | Identifying Attribute |
| NumOfUsers | Show the number of users for a certain type | Derived attribute, range 1 - 100 |
| USERS | Entity class, to model users information | |
| UserID | List the ID of the users | Identifying Attribute |
| FirstName | Record the first name of users | Should not be NULL |
| LastName | Record the last name of users | Should not be NULL |
| Email | Record the email address of users | Should not be NULL |
| PhoneNO | Record the phone number of users | Should not be NULL |
| Password | Each user can select his or her own password | Should not be NULL |
| StreetAddress | Record the street address of users | Should not be NULL |
| City | Record the city of users | |
| State | Record the state of the users | |
| ZipCode | Record the zipcode of the users | |
| UniversityID | Record the university id of each user | Should not be NULL |
| UserType_ID | Record the type of users | Foreign key references CANCREATE(UserType_ID) |
| STUDENTS | Entity class, to model students information | |
| UserID | List the user ID of the student | Foreign key references USERS |
| SportType | List the type of sport the users play | Foreign key references SPORTS(SportName) |
| Counselor_ID | Record the ID of the counselor | Foreign key references USERS --> STAFF --> COUNSELORS(USERID) |
| Num_Bad_Marks | Record the total number bad booking records | |
| Blocked | Record the blocking status of each student based on his/her Num_Bad_Marks | |
| Academic_Standing | Record the academic standing of the user | Should not be NULL |
| STAFF | Entity class, to model staff information | |
| UserID | List the user ID of the staff | Foreign key references USERS |
| StaffType | List the staff type | Foreign key references CANCREATE(UserType_ID) |
| ADMINISTRATORS | Entity class, to model administrators information | |
| UserID | List the user ID of the administrators | Foreign key references USERS --> STAFF(UserID) |
| Managed_Departments | List the department the administrator manages | Should not be NULL |
| COUNSELORS | Entity class, to model counselors information | |
| UserID | List the user ID of the counselors | Foreign key references USERS --> STAFF(UserID) |
| LEARNINGSPECIALISTS | Entity class, to model learning specialist information | |
| UserID | List the user ID of the learning specialists | Foreign key references USERS --> STAFF(UserID) |
| FRONTDESKS | Entity class, to model front desk information | |
| UserID | List the user ID of the frontdesk worker | Foreign key references USERS --> STAFF(UserID) |
| UPDATE | Relation representing the relationship UPDATE | |
| TAKE | Relation representing the relationship TAKE | |
| TEACH | Relation representing the relationship TEACH | |

Chapter 3.

In our system, we have a constraining relationship: only authorized user(s) can make corresponding booking(s) in our system. This relationship was also discussed in Chapter 2. Different attributes in entities has different datatypes, based on length of the attributes. There are some value constraints in our system. For example, the system accepts email address up to 50 characters, in case of extremely long email addresses. Users can have userID up to 16 characters, Phone numbers have 12 characters, no decimal points. Duration of a booking can range from 0 to 4 hours.

Some defaulted values are set on our system. For example, the default **numofuser** displayed in usertype is zero. Each time a user is classified into a user type, the **numofuser** will go up by 1. There are also some CHECK constraints existing in our system. For instance, the **AcademicStanding** of **Tutors** only can be one of the Freshman, Sophomore, Junior, Senior, Graduate or PHD. The Tutors' skill level is evaluated and recorded in our system as well. The system will record them as 'Good', 'Bad' or 'Fine'. The Null-value check will be realized through our system as well.

Below, we offer the detailed normalization results, following with our logical data dictionary.

Translated Relations (Normalization)

| Translated Relations |
|--|
| ATTENDANCERECORD (<u>Student</u> , <u>Course</u> , <u>Record_ID</u>) |
| BOOKINGTYPE (<u>BookingType_ID</u> , NumOfBookings) |
| BOOKINGS (<u>BookingID</u> , <u>BookingType_ID</u> , Status, Room, Duration, Semester, Start_Time, End_Time, Booking_Date, User_ID) |
| RESERVATIONS (<u>BookingID</u> , Reservation_Detail) |
| APPOINTMENTS (<u>BookingID</u> , TutorID) |
| EVENTS (<u>BookingID</u> , Topic, Event_Name) |
| TRAINING_SESSIONS (<u>BookingID</u> , Training_Title) |
| CANCREATE (<u>BookingType_ID</u> , <u>UserType_ID</u>) |
| COURSES (<u>Course_ID</u> , Title, Description, SubjectName) |
| COURSESCHEDULE (<u>Professor</u> , <u>Course</u> , <u>Section</u>) |
| DAILYWORKRECORD (<u>FrontDesk_ID</u> , <u>Record_Date</u> , Contents) |
| EQUIPMENTTYPE (<u>EquipmentType_Name</u> , Description, NumOfEquip) |
| FORM (<u>Room</u> , <u>Room_Type</u>) |
| INCHARGE (<u>Subject</u> , <u>LearningSpecialist</u>) |
| PROFESSORS (<u>EmployeeID</u> , FName, LName, Department) |
| ROOMS (<u>RoomNumber</u> , <u>Room_Availability</u> , RoomSize) |
| ROOMCLASSIFICATION (<u>RoomType_Name</u> , NumOfRoom) |
| ROOMEQUIPMENT (<u>EquipmentID</u> , Condition, <u>Equipment_Type</u> , Room) |
| ROOMSCHEDULE (<u>RoomNumber</u> , <u>Schedule_Date</u> , Start_Time, End_Time, Task) |
| SUBJECTS (<u>SubjectName</u> , Description) |
| SKILLLEVEL (<u>Subject</u> , <u>Tutor</u> , TutorSkillLevel) |
| SPORTS (<u>SportName</u> , Counselor, Coach) |
| TRAININGRECORD (<u>T_Session</u> , <u>Tutor</u> , Contents) |
| TUTORS (<u>TutorID</u> , Tutor_FName, Tutor_LName, AcademicStanding) |
| USERTYPE (<u>UserType_ID</u> , NumOfUsers) |
| USERS (<u>UserID</u> , FirstName, LastName, Email, PhoneNO, Password, StreetAddress, City, State, ZipCode, UniversityID, UserType_ID) |
| STUDENTS (<u>UserID</u> , SportType, Counselor_ID, Num_Bad_Marks, Blocked, Academic_Standing) |
| STAFF (<u>UserID</u> , StaffType) |
| ADMINISTRATORS (<u>UserID</u> , Managed_Departments) |
| COUNSELORS (<u>UserID</u>) |
| LEARNINGSPECIALISTS (<u>UserID</u>) |
| FRONTDESKS (<u>UserID</u>) |

Relational Data Dictionary (*Display all constraints in detail)

| Schema Construct | Data Type | Constraint |
|--|---|--|
| ATTENDANCERECORD | Relation representing the weak entity class ATTENDANCERECORD | |
| Student | varchar2 (16) | Foreign key references USERS --> STUDENTS(UserID) |
| Course | varchar2 (10) | Foreign key references COURSES(Course_ID) |
| Record_ID | varchar2 (10) | Should not be NULL |
| Details | varchar2 (100) | |
| Primary Key Constraint: Student, Course, Record_ID | | |
| FD: Student, Course, Record_ID --> Details | | |
| BOOKINGTYPE | Relation representing the entity class BOOKINGTYPE | |
| BookingType_ID | varchar2 (20) | Primary Key |
| NumOfBookings | num(7,0) | default 0, a derived attribute |
| FD: BookingType_ID --> NumOfBookings | | |
| BOOKINGS | Relation representing the entity class BOOKINGS | |
| BookingID | varchar2 (16) | Primary Key |
| BookingType_ID | varchar2 (20) | Foreign key references CanCreate relationship class |
| Status | varchar2 (9) | CHECK (Status IN ('OnTime', 'Cancelled', 'NoShow', 'Updated')) |
| Room | varchar2 (6) | Foreign key references ROOMS(RoomNumber) |
| Duration | number (1,1) | CHECK (Duration BETWEEN 0 AND 4) |
| Semester | varchar (12) | Should not be NULL |
| Start_Time | DATE | Should not be NULL |
| End_Time | DATE | Should not be NULL |
| Booking_Date | DATE | |
| UserID | varchar2 (16) | Foreign key references USERS |
| Primary Key Constraint: BookingID | | |
| FD: BookingID --> BookingType_ID, Status, Room, Duration, Semester, Start_Time, End_Time, Booking_Date, UserID | | |
| RESERVATIONS | Relation representing the entity subclass RESERVATION | |
| BookingID | varchar2 (16) | Foreign key references BOOKINGS |
| Reservation_Detail | varchar2 (50) | |
| Primary Key Constraint: BookingID | | |
| FD: BookingID --> Reservation_Detail | | |
| APPOINTMENTS | Relation representing the entity subclass APPOINTMENTS | |
| BookingID | varchar2 (16) | Foreign key references BOOKINGS |
| TutorID | varchar2 (8) | Foreign key references TUTORS |
| Primary Key Constraint: BookingID | | |
| FD: BookingID --> TutorID | | |
| EVENTS | Relation representing the entity subclass EVENTS | |
| BookingID | varchar2 (16) | Foreign key references BOOKINGS |
| Topic | varchar2 (20) | |
| Event_Name | varchar2 (20) | Should not be NULL |
| Primary Key Constraint: BookingID | | |
| FD: BookingID --> Event_Name, Topic | | |
| TRAINING_SESSIONS | Relation representing the entity subclass TRAINING_SESSIONS | |
| BookingID | varchar2 (16) | Foreign key references BOOKINGS |
| Training_Title | varchar2 (20) | |
| Primary Key Constraint: BookingID | | |
| FD: BookingID --> Training_Title | | |
| CANCREATE | Relation representing the constraining relationship CANCREATE | |
| BookingType_ID | varchar2 (20) | Foreign Key references BOOKINGTYPE |
| UserType_ID | varchar2 (20) | Foreign Key references USERTYPE |
| Primary Key Constraint: BookingType_ID, UserType_ID | | |
| FD: BookingType_ID, UserType_ID --> BookingType_ID, UserType_ID | | |
| COURSES | Relation representing the entity class COURSES | |
| Course_ID | varchar2 (10) | Primary Key |
| Title | varchar2 (20) | unique |
| Description | varchar2 (50) | Should not be NULL |
| SubjectName | varchar2 (20) | Foreign key references SUBJECTS |
| FD: Course_ID --> Title, Description, SubjectName | | |
| COURSESCCHEDULE | Relation representing the weak entity class COURSESCCHEDULE | |
| Professor | varchar2 (16) | Foreign key references PROFESSORS |
| Course | varchar2 (10) | Foreign key references COURSES |
| Section | varchar2 (3) | Partial Primary Key |
| Primary Key Constraint: Professor, Course, Section | | |
| FD: Professor, Course, SECNO --> Professor, Course, Section | | |
| DAILYWORKRECORD | Relation representing the weak entity class DAILYWORKRECORD | |
| FrontDesk_ID | varchar2 (16) | Foreign key references USERS --> STAFF --> FRONTDESKS |
| Record_Date | DATE | Partial Primary Key |
| Contents | varchar2 (50) | Should not be NULL |
| Primary Key Constraint: FrontDesk_ID, Record_Date | | |
| FD: FrontDesk_ID, Record_Date --> Contents | | |

| | | |
|--|--|--|
| EQUIPMENTTYPE | Relation representing the entity class EQUIPMENTTYPE | |
| EquipmentType_Name | varchar2 (16) | Primary Key |
| Description | varchar2 (20) | Should not be NULL |
| NumOfEquip | number (3,0) | Default 0, a derived attribute |
| FD: EquipmentType_Name --> | Description, NumOfEquip | |
| FORM | Relation representing the relationship FORM | |
| Room | varchar2 (6) | Foreign Key references ROOMS(RoomNumber) |
| Room_Type | varchar2 (16) | Foreign Key references ROOMCLASSIFICATION(RoomType_Name) |
| Primary Key Constraint: Room, Room_Type | | |
| FD: Room, Room_Type --> | Room, Room_Type | |
| INCHARGE | Relation representing the relationship INCHARGE | |
| Subject | varchar2 (20) | Foreign Key references SUBJECTS |
| LearningSpecialist | varchar2 (16) | Foreign Key references USERS --> STAFF --> LEARNINGSPECIALISTS |
| Primary Key Constraint: Subject, LearningSpecialists | | |
| FD: Subject, LearningSpecialists --> | Subject, LearningSpecialists | |
| PROFESSORS | Relation representing the entity class PROFESSORS | |
| EmployeeID | varchar2 (16) | Primary Key |
| FName | varchar2 (10) | Should not be NULL |
| LName | varchar2 (10) | Should not be NULL |
| Department | varchar2 (20) | |
| FD: EmployeeID --> | FName, LName, Department | |
| ROOMS | Relation representing the entity class ROOMS | |
| RoomNumber | varchar2 (6) | Primary Key |
| * Room number won't be changed. | usually kept same in each semester. | |
| Room_Availability | varchar2 (1) | CHECK (Room_Availability IN ('Y', 'N', 'L')) |
| RoomSize | number (3) | CHECK (RoomSize > 0) |
| FD: RoomNumber --> | Room_Availability, RoomSize | |
| ROOMCLASSIFICATION | Relation representing the entity class ROOMCLASSIFICATION | |
| RoomType_Name | varchar2 (16) | Primary Key |
| NumOfRoom | number (2,0) | Default 0, a derived attribute |
| FD: RoomType_Name --> | NumOfRoom | |
| ROOMEQUIPMENT | Relation representing the entity class ROOMEQUIPMENT | |
| EquipmentID | varchar2 (16) | Primary Key |
| Condition | varchar2 (50) | Should not be NULL |
| Equipment_Type | varchar2 (16) | Foreign key references EQUIPMENTTYPE(EquipmentType_Name) |
| Room | varchar2 (6) | Foreign key references ROOMS(RoomNumber) |
| FD: EquipmentID --> | Condition, Equipment_Type, Room | |
| ROOMSCHEDULE | Relation representing the weak entity class SCHEDULE | |
| RoomNumber | varchar2 (6) | Foreign key references ROOMS(RoomNumber) |
| Schedule_Date | DATE | Partial Primary Key |
| Start_Time | DATE | Should not be NULL |
| End_Time | DATE | Should not be NULL |
| Task | varchar2 (60) | |
| Primary Key Constraint: RoomNumber, Schedule_Date | | |
| FD: RoomNumber, Schedule_Date --> | Start_Time, End_Time, Task | |
| SUBJECTS | Relation representing the entity class SUBJECTS | |
| SubjectName | varchar2 (20) | Primary Key |
| Description | varchar2 (50) | |
| FD: SubjectName --> | Description | |
| SKILLEVEL | Relation representing the weak entity class SKILLEVEL | |
| Subject | varchar2 (20) | Foreign key references SUBJECTS(SubjectName) |
| Tutor | varchar2 (8) | Foreign key references TUTORS(TutorID) |
| TutorSkillLevel | varchar2 (50) | CHECK (TutorSkillLevel IN ('Good', 'Bad', 'Fine')) |
| Primary Key Constraint: Subject, Tutor | | |
| FD: Subject, Tutor --> | TutorSkillLevel | |
| SPORTS | Relation representing the entity class SPORTS | |
| SportName | varchar2 (25) | Primary Key |
| Counselor | varchar2 (16) | Foreign key references USERS --> STAFF --> COUNSELORS(USERID) |
| Coach | varchar2 (30) | Should not be NULL |
| FD: SportName --> | Counselor, Coach | |
| TRAININGRECORD | Relation representing the weak entity class TRAININGRECORD | |
| T_Session | varchar2 (16) | Foreign key references TRAINING_SESSIONS(BookingID) |
| Tutor | varchar2 (8) | Foreign key references TUTORS(TutorID) |
| Contents | varchar2 (100) | |
| Primary Key Constraint: T_Session, Tutor | | |
| FD: T_Session, Tutor --> | Contents | |
| TUTORS | Relation representing the entity class TUTORS | |
| TutorID | varchar2 (8) | Primary Key |
| Tutor_FName | varchar2 (10) | Should not be NULL |
| Tutor_LName | varchar2 (10) | Should not be NULL |
| AcademicStanding | varchar2 (10) | CHECK (AcademicStanding IN ('Freshman', 'Sophomore', 'Junior', 'Senior', 'Graduate', 'PHD')) |
| FD: TutorID --> | Tutor_FName, Tutor_LName, AcademicStanding | |

| | | |
|--|---|---|
| USERTYPE | Relation representing the entity class USERTYPE | |
| UserType_ID | varchar2 (20) | Primary Key |
| NumOfUsers | num(4,0) | default 0, a derived attribute |
| FD: UserType_ID --> NumOfUsers | | |
| USERS | Relation representing the entity class USERS | |
| UserID | varchar2 (16) | Primary Key |
| FirstName | varchar2 (20) | Should not be NULL |
| LastName | varchar2 (20) | Should not be NULL |
| Email | varchar2 (50) | Should not be NULL |
| PhoneNO | number (12, 0) | Should not be NULL |
| Password | varchar2 (12) | Should not be NULL |
| StreetAddress | varchar2 (50) | |
| City | varchar2 (20) | |
| State | varchar2 (20) | |
| ZipCode | varchar2 (20) | |
| UniversityID | int | Unique, and Should not be NULL |
| UserType_ID | varchar2 (20) | Foreign key references CANCREATE(UserType_ID) |
| FD: UserID --> FirstName, LastName, Email, PhoneNO, Password, StreetAddress, City, State, ZipCode, UniversityID, UserType_ID | | |
| STUDENTS | Relation representing the entity subclass STUDENTS | |
| UserID | varchar2 (16) | Foreign key references USERS |
| SportType | varchar2 (25) | Foreign key references SPORTS(SportName) |
| Counselor_ID | varchar2 (16) | Foreign key references USERS --> STAFF --> COUNSELORS(USERID) |
| Num_Bad_Marks | number (7) | Default value as 0 |
| Blocked | char (1) | Default value as 'N', and CHECK (Blocked IN ('Y', 'N')) |
| Academic_Standing | char (10) | Should not be NULL |
| Primary Key Constraint: UserID | | |
| FD: UserID --> SportType, Counselor_ID, Num_Bad_Marks, Blocked, Academic_Standing | | |
| STAFF | Relation representing the entity subclass STAFF | |
| UserID | varchar2 (16) | Foreign key references USERS |
| StaffType | varchar2 (20) | Foreign key references CANCREATE(UserType_ID) |
| Primary Key Constraint: UserID | | |
| FD: UserID --> StaffType | | |
| ADMINISTRATORS | Relation representing the entity subclass ADMINISTRATORS | |
| UserID | varchar2 (16) | Foreign key references USERS --> STAFF(UserID) |
| Managed_Departments | varchar2 (20) | |
| Primary Key Constraint: UserID | | |
| FD: UserID --> Managed_Departments | | |
| COUNSELORS | Relation representing the entity subclass COUNSELORS | |
| UserID | varchar2 (16) | Foreign key references USERS --> STAFF(UserID) |
| Primary Key Constraint: UserID | | |
| FD: UserID --> UserID | | |
| LEARNINGSPECIALISTS | Relation representing the entity subclass LEARNINGSPECIALISTS | |
| UserID | varchar2 (16) | Foreign key references USERS --> STAFF(UserID) |
| Primary Key Constraint: UserID | | |
| FD: UserID --> UserID | | |
| FRONTDESKS | Relation representing the entity subclass FRONTDESKS | |
| UserID | varchar2 (16) | Foreign key references USERS --> STAFF(UserID) |
| Primary Key Constraint: UserID | | |
| FD: UserID --> UserID | | |

After finishing our logical design, we started our SQL work. Below, we attached our database table script, including all constraints, together with our triggers (code only).

SQL statements to create tables and define constraints

```
--sequences
```

```
-- Drop sequences code:
```

```
drop sequence pk_attendancerecord;
```

```
-- PK sequence.
```

```
create sequence pk_attendancerecord start with 1000001 maxvalue 1999999 increment by 1;
```

```
-- Drop sequences code:
```

```
drop sequence pk_users;
```

```
-- PK sequence.
```

```

create sequence pk_users start with 10000001 maxvalue 1999999 increment by 1;

-- Drop sequences code:
drop sequence pk_bookings;
-- PK sequence.
create sequence pk_bookings start with 100000000001 maxvalue 19999999999 increment by 1;

-- Drop sequences code:
drop sequence pk_roomequipment;
-- PK sequence.
create sequence pk_roomequipment start with 100000000001 maxvalue 19999999999
increment by 1;

-- USERTYPE
-- No PK trigger Needed.
drop table USERTYPE cascade constraints;
CREATE TABLE USERTYPE
    (USERTYPE_ID VARCHAR2(20) constraint USERTYPE_pk primary key,
    NUMOFUSERS number(4, 0) DEFAULT 0 -- JUST give default ZERO, use trigger to
automatically update after insert/update later
    );

-- BOOKINGTYPE
-- No PK Trigger Needed.
drop table BOOKINGTYPE cascade constraints;
CREATE TABLE BOOKINGTYPE
    (BOOKINGTYPE_ID VARCHAR2(20) constraint BOOKINGTYPE_pk primary key,
    NUMOFBOOKINGS number(7, 0) DEFAULT 0 -- JUST give default ZERO, use
trigger to automatically update after insert/update later
    );

-- CANCREATE: constraining relationship
-- No PK Trigger Needed.
drop table CANCREATE cascade constraints;
CREATE TABLE CANCREATE

```

```

        (BOOKINGTYPE_ID VARCHAR2(20) references
BOOKINGTYPE(BOOKINGTYPE_ID) ON DELETE CASCADE,
        USERTYPE_ID VARCHAR2(20) references USERTYPE(USERTYPE_ID) ON
DELETE CASCADE,
        constraint CANCREATE_pk primary key(BOOKINGTYPE_ID, USERTYPE_ID)
);

```

```

-- USERS
drop table USERS cascade constraints;
CREATE TABLE USERS
(USERID varchar2(16) constraint USERS_pk primary key,
universityid INT unique not null,
        PASSWORD VARCHAR2(255) not null,
        FIRSTNAME VARCHAR2(20) not null,
        LASTNAME VARCHAR2(20) not null,
        STREETADDRESS VARCHAR2(50),
        CITY VARCHAR2(20),
        STATE VARCHAR2(20),
        ZIPCODE VARCHAR2(20),
        EMAIL VARCHAR2(50) not null,
        PHONENO NUMBER(10,0) not null,
        USERTYPE_ID VARCHAR2(20) references USERTYPE(USERTYPE_ID) ON
DELETE cascade
);

```

```

-- Trigger to assign primary keys
create or replace trigger assign_users_pk
BEFORE INSERT
ON USERS
FOR EACH ROW
DECLARE
begin
:new.userid := pk_users.nextval;
end;
/

```

```

CREATE OR REPLACE TRIGGER update_user_type_counts
--- runs only once for any of these query types, not once for each row.

```



```

AFTER INSERT OR UPDATE OR DELETE
on USERS
DECLARE
---- cursor to find usertypes and counts from their table
CURSOR c1 IS
select * from usertype
for update;
BEGIN
---- go through each usertype and update the count
for x in c1
loop
update usertype
---- for the specific usertype, count how many records are now there.
set numofusers = (select count(*) from users where usertype_id = x.usertype_id)
---- update the row that corresponds to the current one in the cursor/loop.
where current of c1;
end loop;
END;
/

drop table REFKEYMAPPING cascade constraints;

CREATE TABLE REFKEYMAPPING(
UNIVERSITYID INT references users(universityid) ON DELETE cascade,
usertype_id varchar(20) references usertype(usertype_id) on delete cascade,
refkey varchar(255),
primary key (universityid, usertype_id, refkey)
);

-- ROOMS
-- No PK Trigger Needed.
drop table ROOMS cascade constraints;
CREATE TABLE ROOMS
    (ROOMNUMBER VARCHAR2(6) constraint ROOMNUMBER_pk primary key,
    ROOMSIZE number(3),
    ROOM_AVAILABILITY VARCHAR2(1),
    -- 'L' means 'Locked'
    constraint rooms_chk CHECK (ROOMSIZE > 0 AND ROOM_AVAILABILITY IN
('Y', 'N', 'L'))
    );

```

```

-- BOOKINGS
drop table BOOKINGS cascade constraints;
CREATE TABLE BOOKINGS
  (BOOKINGID varchar2(16) constraint BOOKINGS_pk primary key,
   BOOKINGTYPE_ID VARCHAR2(20) references BOOKINGTYPE(BOOKINGTYPE_ID)
ON DELETE cascade,
   STATUS VARCHAR2(9),
   constraint STATUS_chk CHECK (STATUS IN ('OnTime','Cancelled', 'NoShow',
'Updated')),
   ROOM VARCHAR2(6) REFERENCES ROOMS(ROOMNUMBER) ON DELETE cascade,
   DURATION NUMBER(1) CHECK (DURATION BETWEEN 0 AND 4),
   SEMESTER VARCHAR2(12) not null,
   START_TIME DATE not null,
   END_TIME DATE not null,
   BOOKING_DATE DATE,
   USERID varchar2(16) references USERS(USERID) ON DELETE cascade
);

```

```

-- Trigger to assign primary keys
create or replace trigger assign_bookings_pk
BEFORE INSERT
ON BOOKINGS
FOR EACH ROW
DECLARE
begin
:new.BOOKINGID := pk_bookings.nextval;
end;
/

```

```

CREATE OR REPLACE TRIGGER update_booking_type_counts
--- runs only once for any of these query types, not once for each row.
AFTER INSERT OR UPDATE OR DELETE
on bookings
DECLARE
---- cursor to find bookingtypes and counts from their table
CURSOR c1 IS
select * from bookingtype
for update;

```

```

BEGIN
---- go through each bookingtype and update the count
for x in c1
loop
update bookingtype
---- for the specific bookingtype, count how many records are now there.
set numofbookings = (select count(*) from bookings where BOOKINGTYPE_ID =
x.BOOKINGTYPE_ID)
---- update the row that corresponds to the current one in the cursor/loop.
where current of c1;
end loop;
END;
/

-- STAFF
-- No PK Trigger needed.
drop table STAFF cascade constraints;
CREATE TABLE STAFF
  (USERID varchar2(16) references USERS(USERID) ON DELETE cascade,
  STAFFTYPE VARCHAR2(20) references USERTYPE(USERTYPE_ID) ON DELETE
cascade,
  PRIMARY KEY (USERID)
);

-- COUNSELORS
-- No PK Trigger needed.
drop table COUNSELORS cascade constraints;
CREATE TABLE COUNSELORS
  (USERID varchar2(16) references STAFF(USERID) ON DELETE cascade,
  PRIMARY KEY (USERID)
);

-- SPORTS
-- No PK Trigger needed.
drop table SPORTS cascade constraints;
CREATE TABLE SPORTS
  (SPORTNAME VARCHAR2(25) constraint SPORTS_pk primary key,
  COUNSELOR varchar2(16) references COUNSELORS(USERID) ON DELETE cascade,
  COACH VARCHAR2(30) not null
);

```

```

-- STUDENT
-- No PK Trigger needed.
drop table STUDENTS cascade constraints;
CREATE TABLE STUDENTS
  (USERID varchar2(16) references USERS(USERID) ON DELETE cascade,
  ACADEMIC_STANDING char(10) not null,
  SPORTTYPE VARCHAR2(25) references SPORTS(SPORTNAME) ON DELETE
cascade,
  COUNSELOR_ID varchar2(16) references COUNSELORS(USERID) ON DELETE
cascade,
  BLOCKED varchar(1) default 'N',
  NUM_BAD_MARKS INT default 0,
  constraint num_bad_check CHECK (num_bad_marks >= 0 ),
  constraint block_check CHECK (BLOCKED IN ('Y', 'N')),
  PRIMARY KEY (USERID)
);

-- LEARNINGSPECIALISTS
-- No PK Trigger needed.
drop table LEARNINGSPECIALISTS cascade constraints;
CREATE TABLE LEARNINGSPECIALISTS
  (USERID varchar2(16) references STAFF(USERID) ON DELETE cascade,
  PRIMARY KEY (USERID)
);

-- ADMINISTRATORS
-- No PK Trigger needed.
drop table ADMINISTRATORS cascade constraints;
CREATE TABLE ADMINISTRATORS
  (USERID VARCHAR(15) references STAFF(USERID) ON DELETE cascade,
  MANAGED_DEPARTMENTS VARCHAR2(20),
  PRIMARY KEY (USERID)
);

-- FRONTDESKS
-- No PK Trigger needed.
drop table FRONTDESKS cascade constraints;
CREATE TABLE FRONTDESKS

```

```

(USERID varchar2(16) references STAFF(USERID) ON DELETE cascade,
PRIMARY KEY (USERID)
);

-- DAILYWORKRECORD
-- No PK Trigger needed.
drop table DAILYWORKRECORD cascade constraints;
CREATE TABLE DAILYWORKRECORD
(FRONTDESK_ID varchar2(16) references FRONTDESKS(USERID) ON DELETE cascade,
RECORD_DATE DATE,
CONTENTS VARCHAR2(50),
PRIMARY KEY (FRONTDESK_ID, RECORD_DATE)
);

-- RESERVATIONS
-- No PK Trigger needed.
drop table RESERVATIONS cascade constraints;
CREATE TABLE RESERVATIONS
(BOOKINGID varchar2(16) references BOOKINGS(BOOKINGID) ON DELETE cascade,
PRIMARY KEY (BOOKINGID),
RESERVATION_DETAIL VARCHAR2(50)
);

-- TUTORS
-- No PK Trigger needed.
drop table TUTORS cascade constraints;
CREATE TABLE TUTORS
(TUTORID VARCHAR2(8) constraint TUTORID_pk primary key,
TUTOR_FNAME VARCHAR2(10) not null,
TUTOR_LNAME VARCHAR2(10) not null,
SUBJECT VARCHAR(20) references SUBJECTS(SUBJECTNAME),
ACADEMICSTANDING VARCHAR2(10),
constraint ACADEMICSTANDING_chk CHECK (ACADEMICSTANDING IN
('Freshman', 'Sophomore', 'Junior', 'Senior', 'Graduate', 'PHD'))
);

```

```

-- APPOINTMENTS
-- No PK Trigger needed.
drop table APPOINTMENTS cascade constraints;
CREATE TABLE APPOINTMENTS
  (BOOKINGID varchar2(16) references BOOKINGS(BOOKINGID) ON DELETE cascade,
   PRIMARY KEY (BOOKINGID),
   TUTORID VARCHAR2(8) references TUTORS(TUTORID) ON DELETE cascade
  );

-- EVENTS
-- No PK Trigger needed.
drop table EVENTS cascade constraints;
CREATE TABLE EVENTS
  (BOOKINGID varchar2(16) references BOOKINGS(BOOKINGID) ON DELETE cascade,
   PRIMARY KEY (BOOKINGID),
   EVENT_NAME VARCHAR2(20) not null,
   TOPIC VARCHAR2(20)
  );

-- TRAINING_SESSIONS
-- No PK Trigger needed.
drop table TRAINING_SESSIONS cascade constraints;
CREATE TABLE TRAINING_SESSIONS
  (BOOKINGID varchar2(16) references BOOKINGS(BOOKINGID) ON DELETE cascade,
   PRIMARY KEY (BOOKINGID),
   TRAINING_TITLE VARCHAR2(20)
  );

-- PROFESSORS
-- No PK Trigger needed.
drop table PROFESSORS cascade constraints;
CREATE TABLE PROFESSORS
  (EMPLOYEEID varchar2(16) constraint PROFESSORS_pk primary key,
   FNAME VARCHAR2(10) not null,
   LNAME VARCHAR2(10) not null,
   DEPARTMENT VARCHAR2(20)
  );

```

```

-- SUBJECTS
-- No PK Trigger needed.
drop table SUBJECTS cascade constraints;
CREATE TABLE SUBJECTS
    (SUBJECTNAME VARCHAR2(20) constraint SUBJECTS_pk primary key,
    DESCRIPTION VARCHAR2(50)
    );

-- COURSES
-- No PK Trigger needed.
drop table COURSES cascade constraints;
CREATE TABLE COURSES
    (COURSE_ID VARCHAR2(10) constraint COURSES_pk primary key,
    TITLE VARCHAR2(20),
    DESCRIPTION VARCHAR2(50),
    SUBJECT VARCHAR2(20) references SUBJECTS(SUBJECTNAME) ON DELETE
cascade
    );

-- COURSESCHEDULE
-- No PK Trigger needed.
drop table COURSESCHEDULE cascade constraints;
CREATE TABLE COURSESCHEDULE
    (PROFESSOR varchar2(16) references PROFESSORS(EMPLOYEEID) ON DELETE
cascade,
    COURSE VARCHAR2(10) references COURSES(COURSE_ID) ON DELETE
cascade,
    SECTION VARCHAR2(3) not null,
    PRIMARY KEY (PROFESSOR, COURSE, SECTION)
    );

-- ATTENDANCERECORD
drop table ATTENDANCERECORD cascade constraints;
CREATE TABLE ATTENDANCERECORD
    (STUDENT varchar2(16) references STUDENTS(USERID) ON DELETE cascade,
    COURSE VARCHAR2(10) references COURSES(COURSE_ID) ON DELETE
cascade,
    RECORD_ID VARCHAR2(10) not null,
    DETAILS VARCHAR2(100),

```

```
PRIMARY KEY (RECORD_ID, STUDENT, COURSE)
);
```

```
-- Trigger to assign primary keys
create or replace trigger assign_attendancerecord_pk
BEFORE INSERT
ON ATTENDANCERECORD
FOR EACH ROW
DECLARE
begin
:new.record_id := pk_attendancerecord.nextval;
end;
/
```

```
-- EQUIPMENTTYPE
-- No PK Trigger needed.
drop table EQUIPMENTTYPE cascade constraints;
CREATE TABLE EQUIPMENTTYPE
    (EQUIPMENTTYPE_NAME varchar2(16) constraint EQUIPMENTTYPE_pk primary
key,
    NUMOFEQUIP number(3, 0) DEFAULT 0 -- JUST give default ZERO, use trigger to
automatically update after insert/update later
    );
```

```
-- ROOMEQUIPMENT
drop table ROOMEQUIPMENT cascade constraints;
CREATE TABLE ROOMEQUIPMENT
    (EQUIPMENT_ID varchar2(16) constraint ROOMEQUIPMENT_pk primary key,
    EQUIPMENTCONDITION VARCHAR2(50) not null,
    EQUIPMENT_TYPE varchar2(16) references
EQUIPMENTTYPE(EQUIPMENTTYPE_NAME),
    ROOM VARCHAR2(6) REFERENCES ROOMS(ROOMNUMBER) ON DELETE cascade
    );
```



```

-- Trigger to assign primary keys
create or replace trigger assign_roomequipment_pk
BEFORE INSERT
ON ROOMEQUIPMENT
FOR EACH ROW
DECLARE
begin
:new.EQUIPMENT_ID := pk_roomequipment.nextval;
end;
/

```

```

-- ROOMCLASSIFICATION
-- No PK Trigger Needed
drop table ROOMCLASSIFICATION cascade constraints;
CREATE TABLE ROOMCLASSIFICATION
    (ROOMTYPE_NAME varchar2(16) constraint ROOMCLASSIFICATION_pk primary
key,
    NUMOFROOM number(2,0) DEFAULT 0 -- JUST give default ZERO, use trigger to
automatically update after insert/update later
    );

```

```

-- FORM: relationship table
-- No PK Trigger Needed
drop table FORM cascade constraints;
CREATE TABLE FORM
    (ROOM VARCHAR2(6) REFERENCES ROOMS(ROOMNUMBER) ON DELETE
cascade,
    ROOM_TYPE varchar2(16) references ROOMCLASSIFICATION(ROOMTYPE_NAME)
ON DELETE cascade,
    PRIMARY KEY (ROOM, ROOM_TYPE)
    );

```

```

-- ROOMSCHEDULE
-- No PK Trigger Needed
drop table ROOMSCHEDULE cascade constraints;
CREATE TABLE ROOMSCHEDULE

```

```
(ROOMNUMBER VARCHAR2(6) references ROOMS(ROOMNUMBER) ON
DELETE cascade,
  SCHEDULE_DATE DATE not null,
  START_TIME DATE not null,
  END_TIME DATE not null,
  TASK VARCHAR2(60),
  PRIMARY KEY (ROOMNUMBER, SCHEDULE_DATE)
);
```

```
-- TRAININGRECORD
-- No PK Trigger Needed
drop table TRAININGRECORD cascade constraints;
CREATE TABLE TRAININGRECORD
  (T_SESSION varchar2(16) references TRAINING_SESSIONS(BOOKINGID) ON
DELETE cascade,
  TUTOR VARCHAR2(8) references TUTORS(TUTORID) ON DELETE cascade,
  CONTENTS VARCHAR2(100),
  PRIMARY KEY (T_SESSION, TUTOR)
);
```

```
--SKILLLEVEL
-- No PK Trigger Needed
drop table SKILLLEVEL cascade constraints;
CREATE TABLE SKILLLEVEL
  (TUTOR VARCHAR2(8) references TUTORS(TUTORID) ON DELETE cascade,
  SUBJECT VARCHAR2(20) references SUBJECTS(SUBJECTNAME) ON DELETE
cascade,
  TutorSkillLevel VARCHAR2(4),
  constraint TutorSkillLevel_chk CHECK (TutorSkillLevel IN ('Good', 'Bad', 'Fine'))
);
```

```
--INCHARGE: relationship table between SUBJECTS and LEARNINGSPECIALISTS
-- No PK Trigger Needed
drop table INCHARGE cascade constraints;
CREATE TABLE INCHARGE
  (SUBJECT VARCHAR2(20) references SUBJECTS(SUBJECTNAME) ON DELETE
cascade,
  LEARNINGSPECIALIST varchar2(16) references STAFF(USERID) ON DELETE cascade,
```

```
primary key(SUBJECT, LEARNINGSPECIALIST)
);
```

```
commit;
```

Triggers and Procedures (code only) related to the tables

```
CREATE OR REPLACE TRIGGER update_user_type_counts
AFTER INSERT OR UPDATE OR DELETE
```

```
on USERS
```

```
DECLARE
```

```
CURSOR c1 IS
```

```
select * from usertype
```

```
where usertype_id = 'STAFF' OR usertype_id = 'STUDENT'
```

```
for update;
```

```
CURSOR c2 IS
```

```
select * from usertype
```

```
where usertype_id <> 'STAFF' AND usertype_id <> 'STUDENT'
```

```
for update;
```

```
BEGIN
```

```
for x in c1
```

```
loop
```

```
update usertype
```

```
set numofusers = (select count(*) from users where usertype_id = x.usertype_id)
```

```
where current of c1;
```

```
end loop;
```

```
for x in c2
```

```
loop
```

```
update usertype
```

```
set numofusers = (select count(*) from staff where stafftype = x.usertype_id)
```

```
where current of c2;
```

```
end loop;
```

```
END;
```

```
/
```

```

CREATE OR REPLACE TRIGGER delete_refkey_new_user
AFTER INSERT
on USERS
for each row
BEGIN
delete from refkeymapping where
refkeymapping.universityID = :new.universityID;
END;
/

/* 5 updateable view for users
These work with a trigger to allow us to simplify adding users (who are split across multiple
tables). */
/* STUDENTS */
drop view view_students_updateable;
create view view_students_updateable
AS select * from users natural join STUDENTS;

create or replace trigger insert_stu_view
instead of insert
on view_students_updateable
for each row
declare
x users.userid%type;
begin

insert into users(universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS,
CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID)
values(:new.universityid, :new.PASSWORD, :new.FIRSTNAME, :new.LASTNAME, :new.ST
REETADDRESS, :new.CITY,
:new.STATE, :new.ZIPCODE, :new.EMAIL, :new.PHONENO, :new.USERTYPE_ID)
RETURNING USERID INTO x;

insert into students(USERID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
values(x,
:new.ACADEMIC_STANDING, :new.SPORTTYPE, :new.COUNSELOR_ID);

end;
/

```

```

/* END STUDENTS */

/* LEARNING SPECIALISTS */
create or replace view view_learningspecs_updateable
AS select * from users natural join LEARNINGSPECIALISTS natural join staff;

create or replace trigger insert_learningspec_view
instead of insert
on view_learningspecs_updateable
for each row
declare
x users.userid%type;
begin

insert into users(universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS,
CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID)
values(:new.universityid, :new.PASSWORD, :new.FIRSTNAME, :new.LASTNAME, :new.ST
REETADDRESS, :new.CITY,
:new.STATE, :new.ZIPCODE, :new.EMAIL, :new.PHONENO, :new.USERTYPE_ID)
RETURNING USERID INTO x;

insert into staff(userid,STAFFTYPE) values(x,:new.STAFFTYPE)
RETURNING USERID INTO x;

insert into LEARNINGSPECIALISTS(userid) values(x);

end;
/
/* END LEARNING SPECIALISTS */

/* ADMINISTRATORS */
create or replace view view_administrators_updateable
AS select * from users natural join staff natural join ADMINISTRATORS;

create or replace trigger insert_admins_view
instead of insert
on view_administrators_updateable
for each row

```

```

declare
x users.userid%type;
begin

insert into users(universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS,
CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID)
values(:new.universityid, :new.PASSWORD, :new.FIRSTNAME, :new.LASTNAME, :new.ST
REETADDRESS, :new.CITY,
:new.STATE, :new.ZIPCODE,      :new.EMAIL, :new.PHONENO, :new.USERTYPE_ID)
RETURNING USERID INTO x;

insert into staff(userid,STAFFTYPE) values(x,:new.STAFFTYPE)
RETURNING USERID INTO x;

insert into ADMINISTRATORS(userid,MANAGED_DEPARTMENTS)
values(x,:new.MANAGED_DEPARTMENTS);

end;
/
/* END ADMINISTARTORS */

/* FRONTDESKS */
create or replace view view_frontdesks_updateable
AS select * from users natural join staff natural join FRONTDESKS;

create or replace trigger insert_frontdesks_view
instead of insert
on view_frontdesks_updateable
for each row
declare
x users.userid%type;
begin

insert into users(universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS,
CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID)
values(:new.universityid, :new.PASSWORD, :new.FIRSTNAME, :new.LASTNAME, :new.ST
REETADDRESS, :new.CITY,
:new.STATE, :new.ZIPCODE,      :new.EMAIL, :new.PHONENO, :new.USERTYPE_ID)

```

```

RETURNING USERID INTO x;

insert into staff(userid,STAFFTYPE) values(x,:new.STAFFTYPE)
RETURNING USERID INTO x;

insert into FRONTDESKS(userid) values(x);

end;
/
/* END FRONTDESKS */

/* COUNSELORS */
create or replace view view_counselors_updateable
AS select * from users natural join staff natural join COUNSELORS ;

select * from view_counselors_updateable;

create or replace trigger insert_counselors_view
instead of insert
on view_counselors_updateable
for each row
declare
x users.userid%type;
begin

insert into users(universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS,
CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID)
values(:new.universityid, :new.PASSWORD, :new.FIRSTNAME, :new.LASTNAME, :new.ST
REETADDRESS, :new.CITY,
:new.STATE, :new.ZIPCODE, :new.EMAIL, :new.PHONENO, :new.USERTYPE_ID)
RETURNING USERID INTO x;

insert into staff(userid,STAFFTYPE) values(x,:new.STAFFTYPE)
RETURNING USERID INTO x;

insert into COUNSELORS(userid) values(x);

end;
/

```

```
/* END COUNSELORS*/
```

```
*****
```

```
END OF UPDATEABLE USER VIEWS
```

```
*****
```

```
/* 4 updateable view for bookings
```

```
These work with a trigger to allow us to simplify adding bookings, which are spread across 2 tables. */
```

```
/* Appointments */
```

```
drop view view_appointments_updateable;
```

```
create view view_appointments_updateable
```

```
AS select * from bookings natural join appointments;
```

```
create or replace trigger insert_appointment_view
```

```
instead of insert
```

```
on view_appointments_updateable
```

```
for each row
```

```
declare
```

```
x APPOINTMENTS.BOOKINGID%type;
```

```
begin
```

```
insert into BOOKINGS(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER,  
START_TIME, END_TIME, BOOKING_DATE, USERID)
```

```
values(:new.BOOKINGTYPE_ID, :new.STATUS, :new.ROOM, :new.DURATION, :new.SEM  
ESTER, :new.START_TIME,
```

```
:new.END_TIME, :new.BOOKING_DATE, :new.USERID)
```

```
RETURNING bookingid INTO x;
```

```
insert into appointments(bookingid, tutorid) values(x,:new.tutorid);
```

```
end;
```

```
/
```

```
/* END Appointments */
```

```
/* EVENTS */
```

```
drop view view_events_updateable;
```

```
create view view_events_updateable
```

```
AS select * from bookings natural join events;
```



```

create or replace trigger insert_events_view
instead of insert
on view_events_updateable
for each row
declare
x events.BOOKINGID%type;
begin

insert into BOOKINGS(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER,
START_TIME, END_TIME, BOOKING_DATE, USERID)
values(:new.BOOKINGTYPE_ID, :new.STATUS, :new.ROOM, :new.DURATION, :new.SEM
ESTER, :new.START_TIME,
:new.END_TIME, :new.BOOKING_DATE, :new.USERID)
RETURNING bookingid INTO x;

insert into events(bookingid, event_name, topic) values(x,:new.event_name, :new.topic);

end;
/
/* END EVENTS */

/* TRAINING_SESSIONS */
drop view view_trainingsess_updateable;
create view view_trainingsess_updateable
AS select * from bookings natural join training_sessions;

create or replace trigger insert_trainingsess_view
instead of insert
on view_trainingsess_updateable
for each row
declare
x training_sessions.BOOKINGID%type;
begin

insert into BOOKINGS(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER,
START_TIME, END_TIME, BOOKING_DATE, USERID)
values(:new.BOOKINGTYPE_ID, :new.STATUS, :new.ROOM, :new.DURATION, :new.SEM
ESTER, :new.START_TIME,

```

```

:new.END_TIME, :new.BOOKING_DATE, :new.USERID)
RETURNING bookingid INTO x;

insert into TRAINING_SESSIONS(BOOKINGID, TRAINING_TITLE)
values(x, :new.TRAINING_TITLE);

end;
/
/* END TRAINING_SESSIONS */

/* RESERVATIONS */
drop view view_reservation_updateable;
create view view_reservation_updateable
AS select * from bookings natural join reservations;

create or replace trigger insert_reservations_view
instead of insert
on view_reservation_updateable
for each row
declare
x reservations.BOOKINGID%type;
begin

insert into BOOKINGS(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER,
START_TIME, END_TIME, BOOKING_DATE, USERID)
values(:new.BOOKINGTYPE_ID, :new.STATUS, :new.ROOM, :new.DURATION, :new.SEM
ESTER, :new.START_TIME,
:new.END_TIME, :new.BOOKING_DATE, :new.USERID)
RETURNING bookingid INTO x;

insert into RESERVATIONS(BOOKINGID, RESERVATION_DETAIL)
values(x, :new.RESERVATION_DETAIL);

end;
/
/* END RESERVATIONS */

/*****
END OF UPDATEABLE BOOKINGS VIEWS
*****/

```

Chapter 4.

First, our team provide our insertion code below:

```
-- Drop sequences code:
drop sequence pk_attendancerecord;
-- PK sequence.
create sequence pk_attendancerecord start with 1000001 maxvalue 1999999 increment by 1;

-- Drop sequences code:
drop sequence pk_users;
-- PK sequence.
create sequence pk_users start with 10000001 maxvalue 19999999 increment by 1;

-- Drop sequences code:
drop sequence pk_bookings;
-- PK sequence.
create sequence pk_bookings start with 1000000000001 maxvalue 199999999999 increment by 1;

-- Drop sequences code:
drop sequence pk_roomequipment;
-- PK sequence.
create sequence pk_roomequipment start with 1000000000001 maxvalue 199999999999
increment by 1;

-- cleaning script

truncate table ROOMCLASSIFICATION cascade;
truncate table FORM cascade;
truncate table ROOMS cascade;
truncate table CanCreate cascade;
truncate table USERTYPE cascade;
truncate table BOOKINGTYPE cascade;
truncate table users cascade;
truncate table students cascade;
truncate table staff cascade;
truncate table counselors cascade;
truncate table learningspecialists cascade;
truncate table administrators cascade;
```

```
truncate table frontdesks cascade;
truncate table DAILYWORKRECORD cascade;
truncate table TUTORS cascade;
truncate table PROFESSORS cascade;
truncate table SUBJECTS cascade;
truncate table COURSES cascade;
truncate table COURSESCCHEDULE cascade;
truncate table ATTENDANCERECORD cascade;
truncate table ROOMEQUIPMENT cascade;
truncate table EQUIPMENTTYPE cascade;
truncate table ROOMSCHEDULE cascade;
truncate table RESERVATIONS cascade;
truncate table TRAINING_SESSIONS cascade;
truncate table events cascade;
truncate table appointments cascade;
truncate table TRAININGRECORD cascade;
truncate table SKILLLEVEL cascade;
truncate table INCHARGE cascade;
```

--ROOMCLASSIFICATION

```
INSERT INTO ROOMCLASSIFICATION VALUES('Computer Lab', 0);
INSERT INTO ROOMCLASSIFICATION VALUES('Small Study Room', 0);
INSERT INTO ROOMCLASSIFICATION VALUES('Large Study Room', 0);
INSERT INTO ROOMCLASSIFICATION VALUES('Meetgin Room', 0);
INSERT INTO ROOMCLASSIFICATION VALUES('Auditorium', 0);
```

--ROOM INSERT(20)

```
INSERT INTO ROOMS VALUES('L201', 58, 'L');
INSERT INTO ROOMS VALUES('L202', 28, 'L');
INSERT INTO ROOMS VALUES('L203', 21, 'Y');
INSERT INTO ROOMS VALUES('L204', 42, 'L');
INSERT INTO ROOMS VALUES('L205', 33, 'L');
INSERT INTO ROOMS VALUES('L206', 30, 'Y');
INSERT INTO ROOMS VALUES('L207', 21, 'Y');
INSERT INTO ROOMS VALUES('L208', 32, 'Y');
INSERT INTO ROOMS VALUES('L209', 20, 'N');
```

```
INSERT INTO ROOMS VALUES('L210', 15, 'N');
INSERT INTO ROOMS VALUES('L211', 28, 'Y');
INSERT INTO ROOMS VALUES('L212', 47, 'L');
INSERT INTO ROOMS VALUES('L213', 21, 'N');
INSERT INTO ROOMS VALUES('L214', 39, 'L');
INSERT INTO ROOMS VALUES('L215', 27, 'L');
INSERT INTO ROOMS VALUES('L216', 40, 'N');
INSERT INTO ROOMS VALUES('L217', 18, 'L');
INSERT INTO ROOMS VALUES('L218', 51, 'N');
INSERT INTO ROOMS VALUES('L219', 11, 'L');
INSERT INTO ROOMS VALUES('L220', 17, 'Y');
```

```
--FORM
```

```
INSERT INTO FORM VALUES('L201', 'Computer Lab');
INSERT INTO FORM VALUES('L202', 'Computer Lab');
INSERT INTO FORM VALUES('L203', 'Small Study Room');
INSERT INTO FORM VALUES('L204', 'Computer Lab');
INSERT INTO FORM VALUES('L205', 'Computer Lab');
INSERT INTO FORM VALUES('L206', 'Small Study Room');
INSERT INTO FORM VALUES('L207', 'Small Study Room');
INSERT INTO FORM VALUES('L208', 'Computer Lab');
INSERT INTO FORM VALUES('L209', 'Small Study Room');
INSERT INTO FORM VALUES('L210', 'Large Study Room');
INSERT INTO FORM VALUES('L211', 'Computer Lab');
INSERT INTO FORM VALUES('L212', 'Large Study Room');
INSERT INTO FORM VALUES('L213', 'Auditorium');
INSERT INTO FORM VALUES('L214', 'Computer Lab');
INSERT INTO FORM VALUES('L215', 'Large Study Room');
INSERT INTO FORM VALUES('L216', 'Auditorium');
INSERT INTO FORM VALUES('L217', 'Small Study Room');
INSERT INTO FORM VALUES('L218', 'Computer Lab');
INSERT INTO FORM VALUES('L218', 'Large Study Room');
INSERT INTO FORM VALUES('L219', 'Computer Lab');
INSERT INTO FORM VALUES('L219', 'Large Study Room');
INSERT INTO FORM VALUES('L220', 'Computer Lab');
INSERT INTO FORM VALUES('L220', 'Large Study Room');
```

```
-- usertype
```

```
INSERT INTO USERTYPE VALUES('STUDENT', 0);
INSERT INTO USERTYPE VALUES('STAFF', 0);
```

```
INSERT INTO USERTYPE VALUES('COUNSELOR', 0);
INSERT INTO USERTYPE VALUES('LEARNINGSPECIALIST', 0);
INSERT INTO USERTYPE VALUES('FRONTDESK', 0);
INSERT INTO USERTYPE VALUES('ADMINISTRATOR', 0);
```

```
--bookingtype
```

```
INSERT INTO BOOKINGTYPE VALUES('APPOINTMENT', 0);
INSERT INTO BOOKINGTYPE VALUES('RESERVATION', 0);
INSERT INTO BOOKINGTYPE VALUES('EVENT', 0);
INSERT INTO BOOKINGTYPE VALUES('TRAINING_SESSION', 0);
```

```
-- cancreate
```

```
INSERT INTO CanCreate VALUES('APPOINTMENT', 'STUDENT');
INSERT INTO CanCreate VALUES('RESERVATION', 'STUDENT');
INSERT INTO CanCreate VALUES('EVENT', 'STUDENT');
INSERT INTO CanCreate VALUES('TRAINING_SESSION', 'STUDENT');
INSERT INTO CanCreate VALUES('APPOINTMENT', 'STAFF');
INSERT INTO CanCreate VALUES('RESERVATION', 'STAFF');
INSERT INTO CanCreate VALUES('EVENT', 'STAFF');
INSERT INTO CanCreate VALUES('TRAINING_SESSION', 'STAFF');
```

```
-----USER-----
```

```
--counselor(10)
```

```
INSERT INTO view_counselors_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID, STAFFTYPE) VALUES(1,'pw21', 'counselor21','lastname21','213 W Park
Ave','Tucson','AZ','85719','counselor21@catmail.arizona.edu','5094703818','STAFF','COUNSEL
OR');
```

```
INSERT INTO view_counselors_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID, STAFFTYPE) VALUES(2,'pw22', 'counselor22','lastname22','213 W Park
Ave','Tucson','AZ','85719','counselor22@catmail.arizona.edu','8728849017','STAFF','COUNSEL
OR');
```

```
INSERT INTO view_counselors_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
```

```

USERTYPE_ID, STAFFTYPE) VALUES(3,'pw23', 'counselor23','lastname23','213 W Park
Ave','Tucson','AZ','85719','counselor23@catmail.arizona.edu','7620967375','STAFF','COUNSEL
OR');
INSERT INTO view_counselors_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID, STAFFTYPE) VALUES(4,'pw24', 'counselor24','lastname24','213 W Park
Ave','Tucson','AZ','85719','counselor24@catmail.arizona.edu','2078434693','STAFF','COUNSEL
OR');
INSERT INTO view_counselors_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID, STAFFTYPE) VALUES(6,'pw26', 'counselor26','lastname26','213 W Park
Ave','Tucson','AZ','85719','counselor26@catmail.arizona.edu','3612892249','STAFF','COUNSEL
OR');
INSERT INTO view_counselors_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID, STAFFTYPE) VALUES(5,'pw25', 'counselor25','lastname25','213 W Park
Ave','Tucson','AZ','85719','counselor25@catmail.arizona.edu','1598620115','STAFF','COUNSEL
OR');
INSERT INTO view_counselors_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID, STAFFTYPE) VALUES(7,'pw27', 'counselor27','lastname27','213 W Park
Ave','Tucson','AZ','85719','counselor27@catmail.arizona.edu','9199381458','STAFF','COUNSEL
OR');
INSERT INTO view_counselors_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID, STAFFTYPE) VALUES(8,'pw28', 'counselor28','lastname28','213 W Park
Ave','Tucson','AZ','85719','counselor28@catmail.arizona.edu','3455475531','STAFF','COUNSEL
OR');
INSERT INTO view_counselors_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID, STAFFTYPE) VALUES(9,'pw29', 'counselor29','lastname29','213 W Park
Ave','Tucson','AZ','85719','counselor29@catmail.arizona.edu','5143670419','STAFF','COUNSEL
OR');
INSERT INTO view_counselors_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID, STAFFTYPE) VALUES(10,'pw30', 'counselor30','lastname30','213 W Park
Ave','Tucson','AZ','85719','counselor30@catmail.arizona.edu','5460641381','STAFF','COUNSEL
OR');

```

--SPORTS

```
INSERT INTO SPORTS VALUES('swimming', '10000010', 'COACH1');
INSERT INTO SPORTS VALUES('baseball', '10000004', 'COACH2');
INSERT INTO SPORTS VALUES('basketball', '10000001', 'COACH3');
INSERT INTO SPORTS VALUES('wrestling', '10000008', 'COACH4');
INSERT INTO SPORTS VALUES('volleyball', '10000009', 'COACH5');
```

```
--STUDENT INSERT (20)
```

```
INSERT INTO view_students_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
VALUES(11,'pw1', 'student1','lastname1','213 W Park
Ave','Tucson','AZ','85719','student1@catmail.arizona.edu','5233836939','STUDENT','Junior','swi
mming','10000010');
```

```
INSERT INTO view_students_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
VALUES(12,'pw2', 'student2','lastname2','213 W Park
Ave','Tucson','AZ','85719','student2@catmail.arizona.edu','5639032930','STUDENT','Junior','bas
eball','10000004');
```

```
INSERT INTO view_students_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
VALUES(13,'pw3', 'student3','lastname3','213 W Park
Ave','Tucson','AZ','85719','student3@catmail.arizona.edu','9324475721','STUDENT','Junior','wr
estling','10000008');
```

```
INSERT INTO view_students_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
VALUES(14,'pw4', 'student4','lastname4','213 W Park
Ave','Tucson','AZ','85719','student4@catmail.arizona.edu','6632803514','STUDENT','Senior','bas
eball','10000004');
```

```
INSERT INTO view_students_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
VALUES(15,'pw5', 'student5','lastname5','213 W Park
Ave','Tucson','AZ','85719','student5@catmail.arizona.edu','7538439841','STUDENT','Sophomor
e','volleyball','10000009');
```

```
INSERT INTO view_students_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
```



```

USERTYPE_ID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
VALUES(16,'pw6', 'student6','lastname6','213 W Park
Ave','Tucson','AZ','85719','student6@catmail.arizona.edu','9302357872','STUDENT','Freshman','
basketball','10000001');
INSERT INTO view_students_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
VALUES(17,'pw7', 'student7','lastname7','213 W Park
Ave','Tucson','AZ','85719','student7@catmail.arizona.edu','3891263244','STUDENT','Senior','bas
eball','10000004');
INSERT INTO view_students_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
VALUES(18,'pw8', 'student8','lastname8','213 W Park
Ave','Tucson','AZ','85719','student8@catmail.arizona.edu','4710071173','STUDENT','Sophomor
e','wrestling','10000008');
INSERT INTO view_students_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
VALUES(19,'pw9', 'student9','lastname9','213 W Park
Ave','Tucson','AZ','85719','student9@catmail.arizona.edu','5271071145','STUDENT','Freshman','
swimming','10000010');
INSERT INTO view_students_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
VALUES(20,'pw10', 'student10','lastname10','213 W Park
Ave','Tucson','AZ','85719','student10@catmail.arizona.edu','1692607624','STUDENT','Junior','s
wimming','10000010');
INSERT INTO view_students_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
VALUES(21,'pw11', 'student11','lastname11','213 W Park
Ave','Tucson','AZ','85719','student11@catmail.arizona.edu','7050795021','STUDENT','Sophomo
re','volleyball','10000009');
INSERT INTO view_students_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
VALUES(22,'pw12', 'student12','lastname12','213 W Park
Ave','Tucson','AZ','85719','student12@catmail.arizona.edu','5203485070','STUDENT','Senior','v
olleyball','10000009');

```

```

INSERT INTO view_students_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
VALUES(23,'pw13', 'student13','lastname13','213 W Park
Ave','Tucson','AZ','85719','student13@catmail.arizona.edu','9472235572','STUDENT','Freshman
','wrestling','10000008');
INSERT INTO view_students_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
VALUES(24,'pw14', 'student14','lastname14','213 W Park
Ave','Tucson','AZ','85719','student14@catmail.arizona.edu','7279902100','STUDENT','Freshman
','basketball','10000001');
INSERT INTO view_students_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
VALUES(25,'pw15', 'student15','lastname15','213 W Park
Ave','Tucson','AZ','85719','student15@catmail.arizona.edu','1129825091','STUDENT','Senior','s
wimming','10000010');
INSERT INTO view_students_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
VALUES(26,'pw16', 'student16','lastname16','213 W Park
Ave','Tucson','AZ','85719','student16@catmail.arizona.edu','2151533116','STUDENT','Junior','ba
seball','10000004');
INSERT INTO view_students_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
VALUES(27,'pw17', 'student17','lastname17','213 W Park
Ave','Tucson','AZ','85719','student17@catmail.arizona.edu','3590203166','STUDENT','Sophomo
re','volleyball','10000009');
INSERT INTO view_students_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
VALUES(28,'pw18', 'student18','lastname18','213 W Park
Ave','Tucson','AZ','85719','student18@catmail.arizona.edu','1932575108','STUDENT','Freshman
','wrestling','10000008');
INSERT INTO view_students_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
VALUES(29,'pw19', 'student19','lastname19','213 W Park

```

```
Ave','Tucson','AZ','85719','student19@catmail.arizona.edu','9444980923','STUDENT','Senior','basketball','10000001');
```

```
INSERT INTO view_students_updateable (universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO, USERTYPE_ID, ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID) VALUES(30,'pw20', 'student20','lastname20','213 W Park Ave','Tucson','AZ','85719','student20@catmail.arizona.edu','4981906887','STUDENT','Sophomore','basketball','10000004');
```

--Learningspecialist (10)

```
INSERT INTO view_learningspecs_updateable (universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO, USERTYPE_ID, STAFFTYPE) VALUES(31,'pw31', 'specialist31','lastname31','213 W Park Ave','Tucson','AZ','85719','specialist31@catmail.arizona.edu','4819828255','STAFF','LEARNINGSPECIALIST');
```

```
INSERT INTO view_learningspecs_updateable (universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO, USERTYPE_ID, STAFFTYPE) VALUES(32,'pw32', 'specialist32','lastname32','213 W Park Ave','Tucson','AZ','85719','specialist32@catmail.arizona.edu','9500004262','STAFF','LEARNINGSPECIALIST');
```

```
INSERT INTO view_learningspecs_updateable (universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO, USERTYPE_ID, STAFFTYPE) VALUES(33,'pw33', 'specialist33','lastname33','213 W Park Ave','Tucson','AZ','85719','specialist33@catmail.arizona.edu','6069056120','STAFF','LEARNINGSPECIALIST');
```

```
INSERT INTO view_learningspecs_updateable (universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO, USERTYPE_ID, STAFFTYPE) VALUES(34,'pw34', 'specialist34','lastname34','213 W Park Ave','Tucson','AZ','85719','specialist34@catmail.arizona.edu','9202263184','STAFF','LEARNINGSPECIALIST');
```

```
INSERT INTO view_learningspecs_updateable (universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO, USERTYPE_ID, STAFFTYPE) VALUES(35,'pw35', 'specialist35','lastname35','213 W Park Ave','Tucson','AZ','85719','specialist35@catmail.arizona.edu','6725752580','STAFF','LEARNINGSPECIALIST');
```

```
INSERT INTO view_learningspecs_updateable (universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
```

```
USERTYPE_ID, STAFFTYPE) VALUES(36,'pw36', 'specialist36','lastname36','213 W Park Ave','Tucson','AZ','85719','specialist36@catmail.arizona.edu','1805317580','STAFF','LEARNIN GSPECIALIST');
```

```
INSERT INTO view_learningspecs_updateable (universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO, USERTYPE_ID, STAFFTYPE) VALUES(37,'pw37', 'specialist37','lastname37','213 W Park Ave','Tucson','AZ','85719','specialist37@catmail.arizona.edu','4298780619','STAFF','LEARNIN GSPECIALIST');
```

```
INSERT INTO view_learningspecs_updateable (universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO, USERTYPE_ID, STAFFTYPE) VALUES(38,'pw38', 'specialist38','lastname38','213 W Park Ave','Tucson','AZ','85719','specialist38@catmail.arizona.edu','4517654616','STAFF','LEARNIN GSPECIALIST');
```

```
INSERT INTO view_learningspecs_updateable (universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO, USERTYPE_ID, STAFFTYPE) VALUES(39,'pw39', 'specialist39','lastname39','213 W Park Ave','Tucson','AZ','85719','specialist39@catmail.arizona.edu','6631048093','STAFF','LEARNIN GSPECIALIST');
```

```
INSERT INTO view_learningspecs_updateable (universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO, USERTYPE_ID, STAFFTYPE) VALUES(40,'pw40', 'specialist40','lastname40','213 W Park Ave','Tucson','AZ','85719','specialist40@catmail.arizona.edu','5558395891','STAFF','LEARNIN GSPECIALIST');
```

--Admin(5)

```
INSERT INTO view_administrators_updateable (universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO, USERTYPE_ID, STAFFTYPE, MANAGED_DEPARTMENTS) VALUES(41,'pw41', 'admin41','lastname41','213 W Park Ave','Tucson','AZ','85719','admin41@catmail.arizona.edu','3172114281','STAFF','ADMINISTRATOR','Finance');
```

```
INSERT INTO view_administrators_updateable (universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO, USERTYPE_ID, STAFFTYPE, MANAGED_DEPARTMENTS) VALUES(42,'pw42', 'admin42','lastname42','213 W Park Ave','Tucson','AZ','85719','admin42@catmail.arizona.edu','3588005701','STAFF','ADMINISTRATOR','MIS');
```

```
INSERT INTO view_administrators_updateable (universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
```

```

USERTYPE_ID, STAFFTYPE, MANAGED_DEPARTMENTS) VALUES(43,'pw43',
'admin43','lastname43','213 W Park
Ave','Tucson','AZ','85719','admin43@catmail.arizona.edu','5713767536','STAFF','ADMINISTR
ATOR','MIS');
INSERT INTO view_administrators_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID, STAFFTYPE, MANAGED_DEPARTMENTS) VALUES(44,'pw44',
'admin44','lastname44','213 W Park
Ave','Tucson','AZ','85719','admin44@catmail.arizona.edu','7216007744','STAFF','ADMINISTR
ATOR','Accounting');
INSERT INTO view_administrators_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID, STAFFTYPE, MANAGED_DEPARTMENTS) VALUES(45,'pw45',
'admin45','lastname45','213 W Park
Ave','Tucson','AZ','85719','admin45@catmail.arizona.edu','2238948834','STAFF','ADMINISTR
ATOR','Economy');

--frontdesk(5)
INSERT INTO view_frontdesks_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID, STAFFTYPE) VALUES(46,'pw46', 'front46','lastname46','213 W Park
Ave','Tucson','AZ','85719','front46@catmail.arizona.edu','5128938474','STAFF','FRONTDESK')
;
INSERT INTO view_frontdesks_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID, STAFFTYPE) VALUES(47,'pw47', 'front47','lastname47','213 W Park
Ave','Tucson','AZ','85719','front47@catmail.arizona.edu','4426130680','STAFF','FRONTDESK')
;
INSERT INTO view_frontdesks_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID, STAFFTYPE) VALUES(48,'pw48', 'front48','lastname48','213 W Park
Ave','Tucson','AZ','85719','front48@catmail.arizona.edu','2484325888','STAFF','FRONTDESK')
;
INSERT INTO view_frontdesks_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID, STAFFTYPE) VALUES(49,'pw49', 'front49','lastname49','213 W Park
Ave','Tucson','AZ','85719','front49@catmail.arizona.edu','2481133150','STAFF','FRONTDESK')
;
INSERT INTO view_frontdesks_updateable (universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY, STATE, ZIPCODE, EMAIL, PHONENO,

```

```
USERTYPE_ID, STAFFTYPE) VALUES(50,'pw50', 'front50','lastname50','213 W Park
Ave','Tucson','AZ','85719','front50@catmail.arizona.edu','6936881956','STAFF','FRONTDESK')
;
```

```
--DAILYWORKRECORD
```

```
INSERT INTO DAILYWORKRECORD VALUES('10000046', to_date('17-MAR-2016','DD-
MON-YY'), 'Three Attendee totally');
```

```
INSERT INTO DAILYWORKRECORD VALUES('10000047', to_date('18-SEP-2016','DD-
MON-YY'), 'Five Attendee totally');
```

```
INSERT INTO DAILYWORKRECORD VALUES('10000046', to_date('01-JUN-2016','DD-
MON-YY'), 'One Attendee totally');
```

```
INSERT INTO DAILYWORKRECORD VALUES('10000050', to_date('07-MAR-2016','DD-
MON-YY'), 'Two Attendee totally');
```

```
INSERT INTO DAILYWORKRECORD VALUES('10000049', to_date('27-OCT-2016','DD-
MON-YY'), 'Three Attendee totally');
```

```
--subject
```

```
INSERT INTO SUBJECTS VALUES('Englsih', 'A modern language');
```

```
INSERT INTO SUBJECTS VALUES('MIS', 'Management Information System');
```

```
INSERT INTO SUBJECTS VALUES('Statistics', 'Stat is not boring');
```

```
INSERT INTO SUBJECTS VALUES('Finance', 'Earning money');
```

```
INSERT INTO SUBJECTS VALUES('Physics', 'Thanks to Newton');
```

```
INSERT INTO SUBJECTS VALUES('Chemistry', 'All the explosion starts from here');
```

```
INSERT INTO SUBJECTS VALUES('History', 'Humanities stories');
```

```
INSERT INTO SUBJECTS VALUES('Geography', 'Know the Earth');
```

```
INSERT INTO SUBJECTS VALUES('Law', 'Better obey it');
```

```
INSERT INTO SUBJECTS VALUES('Economy', 'Econ');
```

```
--course
```

```
INSERT INTO COURSES VALUES('MIS543', 'Networking', 'Networking Knowledge', 'MIS');
```

```
INSERT INTO COURSES VALUES('FIN123', 'Earning Money', 'basic knowledge about
earning money', 'Finance');
```

```
INSERT INTO COURSES VALUES('MIS531', 'EDM', 'Enterprise Database Management',
'MIS');
```

```
INSERT INTO COURSES VALUES('CHE681', 'Explosion', 'BOOM', 'Chemistry');
```

```
INSERT INTO COURSES VALUES('PHY101', 'The Apple', 'falling off the tree', 'Physics');
```

```
INSERT INTO COURSES VALUES('MIS509', 'Bis Com', 'Communication', 'MIS');
```

```
INSERT INTO COURSES VALUES('MIS541', 'ISAD', 'Info System Analysis and Design',
'MIS');
INSERT INTO COURSES VALUES('MIS513', 'Web Mining', 'Mining', 'MIS');
INSERT INTO COURSES VALUES('MIS688', 'Case Study', 'Counsulting Case Study', 'MIS');
INSERT INTO COURSES VALUES('FIN576', 'Stock', 'Earning money in a different way',
'Finance');
```

--Tutor

```
INSERT INTO TUTORS VALUES('21103942', 'Tutor1', 'Lee', 'Englsih', 'Junior');
INSERT INTO TUTORS VALUES('21103950', 'Tutor2', 'A', 'MIS', 'PHD');
INSERT INTO TUTORS VALUES('21102912', 'Tutor3', 'B', 'Statistics', 'Senior');
INSERT INTO TUTORS VALUES('21123942', 'Tutor4', 'C', 'Finance', 'Sophomore');
INSERT INTO TUTORS VALUES('21114242', 'Tutor5', 'D', 'Physics', 'Graduate');
INSERT INTO TUTORS VALUES('22314223', 'Tutor6', 'E', 'Chemistry', 'Graduate');
INSERT INTO TUTORS VALUES('22314210', 'Tutor7', 'F', 'History', 'PHD');
INSERT INTO TUTORS VALUES('22314211', 'Tutor8', 'G', 'Geography', 'Sophomore');
INSERT INTO TUTORS VALUES('22314212', 'Tutor9', 'H', 'Law', 'Senior');
INSERT INTO TUTORS VALUES('22314213', 'Tutor10', 'I', 'Economy', 'Graduate');
```

--professors

```
INSERT INTO PROFESSORS VALUES('21231231', 'Prof1', 'A', 'English');
INSERT INTO PROFESSORS VALUES('21231232', 'Prof2', 'B', 'MIS');
INSERT INTO PROFESSORS VALUES('21231233', 'Prof3', 'C', 'Finance');
INSERT INTO PROFESSORS VALUES('21231234', 'Prof4', 'D', 'Chemistry');
INSERT INTO PROFESSORS VALUES('21231235', 'Prof5', 'E', 'Economy');
INSERT INTO PROFESSORS VALUES('21231236', 'Prof6', 'F', 'Physics');
INSERT INTO PROFESSORS VALUES('21231237', 'Prof7', 'G', 'Statistics');
INSERT INTO PROFESSORS VALUES('21231238', 'Prof8', 'H', 'History');
INSERT INTO PROFESSORS VALUES('21231239', 'Prof9', 'I', 'Geography');
INSERT INTO PROFESSORS VALUES('21231240', 'Prof10', 'J', 'Law');
```

-- COURSESCHEDULE

```
INSERT INTO COURSESCHEDULE VALUES('21231232', 'MIS543', '001');
INSERT INTO COURSESCHEDULE VALUES('21231232', 'MIS543', '002');
INSERT INTO COURSESCHEDULE VALUES('21231232', 'MIS531', '001');
INSERT INTO COURSESCHEDULE VALUES('21231232', 'MIS531', '002');
INSERT INTO COURSESCHEDULE VALUES('21231233', 'FIN123', '001');
```

```
INSERT INTO COURSESCHEDULE VALUES('21231234', 'CHE681', '001');
INSERT INTO COURSESCHEDULE VALUES('21231236', 'PHY101', '001');
INSERT INTO COURSESCHEDULE VALUES('21231232', 'MIS688', '001');
INSERT INTO COURSESCHEDULE VALUES('21231233', 'FIN576', '001');
INSERT INTO COURSESCHEDULE VALUES('21231232', 'MIS513', '001');
```

```
--ATTENDANCERECORD
```

```
INSERT INTO ATTENDANCERECORD VALUES('10000021', 'MIS543', '1000001', 'missed 3
times');
INSERT INTO ATTENDANCERECORD VALUES('10000021', 'MIS531', '1000002', 'missed 1
times');
INSERT INTO ATTENDANCERECORD VALUES('10000021', 'PHY101', '1000003', 'missed
14 times');
INSERT INTO ATTENDANCERECORD VALUES('10000021', 'FIN123', '1000004', '');
INSERT INTO ATTENDANCERECORD VALUES('10000015', 'CHE681', '1000005', '');
INSERT INTO ATTENDANCERECORD VALUES('10000018', 'MIS531', '1000006', '');
INSERT INTO ATTENDANCERECORD VALUES('10000018', 'MIS543', '1000007', '');
INSERT INTO ATTENDANCERECORD VALUES('10000018', 'FIN123', '1000008', '');
INSERT INTO ATTENDANCERECORD VALUES('10000011', 'CHE681', '1000009', '');
INSERT INTO ATTENDANCERECORD VALUES('10000020', 'FIN123', '1000010', '');
```

```
--EQUIPMENTTYPE
```

```
INSERT INTO EQUIPMENTTYPE VALUES('Computer', 0);
INSERT INTO EQUIPMENTTYPE VALUES('Projector', 0);
INSERT INTO EQUIPMENTTYPE VALUES('Table', 0);
INSERT INTO EQUIPMENTTYPE VALUES('Printer', 0);
INSERT INTO EQUIPMENTTYPE VALUES('Chair', 0);
INSERT INTO EQUIPMENTTYPE VALUES('Screen', 0);
INSERT INTO EQUIPMENTTYPE VALUES('Blackboard', 0);
INSERT INTO EQUIPMENTTYPE VALUES('Whiteboard', 0);
INSERT INTO EQUIPMENTTYPE VALUES('Clicker', 0);
INSERT INTO EQUIPMENTTYPE VALUES('Laptop', 0);
```

```
-- ROOMEQUIPMENT
```

```
INSERT INTO ROOMEQUIPMENT VALUES('100000000001', 'Normal, purchased in 2014',
'Computer', 'L201');
INSERT INTO ROOMEQUIPMENT VALUES('100000000002', 'Normal, purchased in 2015',
'Projector', 'L201');
```



```

INSERT INTO ROOMEQUIPMENT VALUES('100000000003', 'Normal, purchased in 2013',
'Computer', 'L202');
INSERT INTO ROOMEQUIPMENT VALUES('100000000004', 'Normal, purchased in 2010',
'Projector', 'L202');
INSERT INTO ROOMEQUIPMENT VALUES('100000000005', 'Normal, purchased in 2014',
'Computer', 'L203');
INSERT INTO ROOMEQUIPMENT VALUES('100000000006', 'Normal, purchased in 2014',
'Computer', 'L204');
INSERT INTO ROOMEQUIPMENT VALUES('100000000007', 'Normal, purchased in 2015',
'Table', 'L205');
INSERT INTO ROOMEQUIPMENT VALUES('100000000008', 'Normal, purchased in 2013',
'Table', 'L212');
INSERT INTO ROOMEQUIPMENT VALUES('100000000009', 'Normal, purchased in 2013',
'Table', 'L213');
INSERT INTO ROOMEQUIPMENT VALUES('100000000010', 'Normal, purchased in 2014',
'Table', 'L219');

```

-- ROOMSCHEDULE

```

INSERT INTO ROOMSCHEDULE VALUES('L201', to_date('10-OCT-2016','DD-MON-YY'),
to_date('11:30:00','hh24:mi:ss'), to_date('14:30:00','hh24:mi:ss'), 'Study');
INSERT INTO ROOMSCHEDULE VALUES('L203', to_date('10-OCT-2016','DD-MON-YY'),
to_date('11:30:00','hh24:mi:ss'), to_date('14:30:00','hh24:mi:ss'), 'Study');
INSERT INTO ROOMSCHEDULE VALUES('L203', to_date('11-OCT-2016','DD-MON-YY'),
to_date('11:30:00','hh24:mi:ss'), to_date('12:30:00','hh24:mi:ss'), 'Study');
INSERT INTO ROOMSCHEDULE VALUES('L204', to_date('14-OCT-2016','DD-MON-YY'),
to_date('13:30:00','hh24:mi:ss'), to_date('14:30:00','hh24:mi:ss'), 'Study');
INSERT INTO ROOMSCHEDULE VALUES('L205', to_date('10-OCT-2016','DD-MON-YY'),
to_date('12:00:00','hh24:mi:ss'), to_date('14:00:00','hh24:mi:ss'), 'Study');

```

```

INSERT INTO ROOMSCHEDULE VALUES('L207', to_date('12-OCT-2016','DD-MON-YY'),
to_date('11:30:00','hh24:mi:ss'), to_date('12:30:00','hh24:mi:ss'), 'Training Session');
INSERT INTO ROOMSCHEDULE VALUES('L208', to_date('12-OCT-2016','DD-MON-YY'),
to_date('11:30:00','hh24:mi:ss'), to_date('13:30:00','hh24:mi:ss'), 'Training Session');
INSERT INTO ROOMSCHEDULE VALUES('L210', to_date('18-OCT-2016','DD-MON-YY'),
to_date('12:30:00','hh24:mi:ss'), to_date('14:30:00','hh24:mi:ss'), 'Training Session');
INSERT INTO ROOMSCHEDULE VALUES('L211', to_date('09-OCT-2016','DD-MON-YY'),
to_date('13:30:00','hh24:mi:ss'), to_date('14:30:00','hh24:mi:ss'), 'Training Session');
INSERT INTO ROOMSCHEDULE VALUES('L212', to_date('08-OCT-2016','DD-MON-YY'),
to_date('13:30:00','hh24:mi:ss'), to_date('14:30:00','hh24:mi:ss'), 'Training Session');

```

```

INSERT INTO ROOMSCHEDULE VALUES('L212', to_date('04-OCT-2016','DD-MON-YY'),
to_date('09:00:00','hh24:mi:ss'), to_date('11:00:00','hh24:mi:ss'), 'Dept Meeting');
INSERT INTO ROOMSCHEDULE VALUES('L217', to_date('10-OCT-2016','DD-MON-YY'),
to_date('10:30:00','hh24:mi:ss'), to_date('11:30:00','hh24:mi:ss'), 'Dept Meeting');
INSERT INTO ROOMSCHEDULE VALUES('L218', to_date('11-OCT-2016','DD-MON-YY'),
to_date('10:30:00','hh24:mi:ss'), to_date('12:30:00','hh24:mi:ss'), 'Dept Meeting');
INSERT INTO ROOMSCHEDULE VALUES('L219', to_date('17-OCT-2016','DD-MON-YY'),
to_date('12:30:00','hh24:mi:ss'), to_date('14:30:00','hh24:mi:ss'), 'Dept Meeting');
INSERT INTO ROOMSCHEDULE VALUES('L220', to_date('15-OCT-2016','DD-MON-YY'),
to_date('11:30:00','hh24:mi:ss'), to_date('14:30:00','hh24:mi:ss'), 'Dept Meeting');

```

```

INSERT INTO ROOMSCHEDULE VALUES('L211', to_date('03-OCT-2016','DD-MON-YY'),
to_date('09:00:00','hh24:mi:ss'), to_date('11:00:00','hh24:mi:ss'), 'Math Tutor Session');
INSERT INTO ROOMSCHEDULE VALUES('L213', to_date('13-OCT-2016','DD-MON-YY'),
to_date('10:30:00','hh24:mi:ss'), to_date('11:30:00','hh24:mi:ss'), 'MIS Tutor Session');
INSERT INTO ROOMSCHEDULE VALUES('L214', to_date('13-OCT-2016','DD-MON-YY'),
to_date('10:30:00','hh24:mi:ss'), to_date('12:30:00','hh24:mi:ss'), 'MIS Tutor Session');
INSERT INTO ROOMSCHEDULE VALUES('L215', to_date('23-OCT-2016','DD-MON-YY'),
to_date('12:30:00','hh24:mi:ss'), to_date('14:30:00','hh24:mi:ss'), 'Math Tutor Session');
INSERT INTO ROOMSCHEDULE VALUES('L216', to_date('13-OCT-2016','DD-MON-YY'),
to_date('11:30:00','hh24:mi:ss'), to_date('14:30:00','hh24:mi:ss'), 'Stat Tutor Session');

```

--BOOKINGS

--reservation

```

insert into view_reservation_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION,
SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID,
RESERVATION_DETAIL) values('RESERVATION', 'OnTime', 'L201', 3, 'Fall2016', '10-OCT-
16', '10-OCT-16', '10-MAR-16', '10000013', 'Test Reservation Detail A');
insert into view_reservation_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION,
SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID,
RESERVATION_DETAIL) values('RESERVATION', 'OnTime', 'L203', 1, 'Fall2016', '11-OCT-
16', '11-OCT-16', '10-MAR-16', '10000015', 'Test Reservation Detail B');
insert into view_reservation_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION,
SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID,
RESERVATION_DETAIL) values('RESERVATION', 'NoShow', 'L203', 3, 'Fall2016', '10-OCT-
16', '10-OCT-16', '10-MAR-16', '10000019', 'Test Reservation Detail C');
insert into view_reservation_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION,
SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID,

```

```
RESERVATION_DETAIL) values('RESERVATION', 'Cancelled','L204', 1, 'Fall2016','14-OCT-16','14-OCT-16','10-MAR-16', '10000012','Test Reservation Detail D');
insert into view_reservation_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID, RESERVATION_DETAIL) values('RESERVATION', 'OnTime','L205', 2, 'Fall2016','10-OCT-16','10-OCT-16','10-MAR-16', '10000014','Test Reservation Detail E');
```

--trainingsessions

```
insert into view_trainingsess_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID, TRAINING_TITLE) values ('TRAINING_SESSION', 'OnTime','L207', 1, 'Fall2016','12-OCT-16','12-OCT-16','10-MAR-16', '10000031','Test Title1');
insert into view_trainingsess_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID, TRAINING_TITLE) values ('TRAINING_SESSION', 'OnTime','L208', 2, 'Fall2016','12-OCT-16','12-OCT-16','10-MAR-16', '10000034','Test Title2');
insert into view_trainingsess_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID, TRAINING_TITLE) values ('TRAINING_SESSION', 'OnTime','L210', 2, 'Fall2016','18-OCT-16','18-OCT-16','10-MAR-16', '10000035','Test Title3');
insert into view_trainingsess_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID, TRAINING_TITLE) values ('TRAINING_SESSION', 'OnTime','L211', 1, 'Fall2016','09-OCT-16','09-OCT-16','10-MAR-16', '10000037','Test Title4');
insert into view_trainingsess_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID, TRAINING_TITLE) values ('TRAINING_SESSION', 'Cancelled','L212', 1, 'Fall2016','08-OCT-16','08-OCT-16','10-MAR-16', '10000038','Test Title5');
```

--events

```
insert into view_events_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID, EVENT_NAME, TOPIC) values ('EVENT', 'OnTime','L212', 2, 'Fall2016','04-OCT-16', '04-OCT-16','10-MAR-16', '10000041','Test Name', 'Test Topic1');
insert into view_events_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID, EVENT_NAME,
```

```

TOPIC) values ('EVENT', 'OnTime','L217', 1, 'Fall2016','10-OCT-16', '10-OCT-16','10-MAR-16', '10000041','Test Name', 'Test Topic2');
insert into view_events_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID, EVENT_NAME, TOPIC) values ('EVENT', 'OnTime','L218', 2, 'Fall2016','11-OCT-16', '11-OCT-16','10-MAR-16', '10000042','Test Name', 'Test Topic3');
insert into view_events_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID, EVENT_NAME, TOPIC) values ('EVENT', 'OnTime','L219', 2, 'Fall2016','17-OCT-16', '17-OCT-16','10-MAR-16', '10000044','Test Name', 'Test Topic4');
insert into view_events_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID, EVENT_NAME, TOPIC) values ('EVENT', 'OnTime','L220', 3, 'Fall2016','15-OCT-16', '15-OCT-16','10-MAR-16', '10000044','Test Name', 'Test Topic5');

```

--appointment

```

--insert into view_appointments_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID, TUTORID)
insert into view_appointments_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID, TUTORID) values ('APPOINTMENT', 'OnTime','L211', 2, 'Fall2016','03-OCT-16', '03-OCT-16','10-MAR-16', '10000021','22314223');
insert into view_appointments_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID, TUTORID) values ('APPOINTMENT', 'OnTime','L213', 1, 'Fall2016','13-OCT-16', '13-OCT-16','10-MAR-16', '10000016','22314210');
insert into view_appointments_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID, TUTORID) values ('APPOINTMENT', 'Cancelled','L214', 2, 'Fall2016','13-OCT-16', '13-OCT-16','10-MAR-16', '10000017','22314211');
insert into view_appointments_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID, TUTORID) values ('APPOINTMENT', 'Cancelled','L215', 2, 'Fall2016','23-OCT-16', '23-OCT-16','10-MAR-16', '10000021','22314212');
insert into view_appointments_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME, END_TIME, BOOKING_DATE, USERID,

```

```
TUTORID) values ('APPOINTMENT', 'NoShow','L216', 3, 'Fall2016','13-OCT-16', '13-OCT-16','10-MAR-16', '10000020','22314213');
--TRAININGRECORD
```

```
INSERT INTO TRAININGRECORD VALUES('100000000006', '22314223', 'testing content 1');
INSERT INTO TRAININGRECORD VALUES('100000000007', '22314210', 'testing content 2');
INSERT INTO TRAININGRECORD VALUES('100000000008', '22314211', 'testing content 3');
INSERT INTO TRAININGRECORD VALUES('100000000009', '22314212', 'testing content 4');
INSERT INTO TRAININGRECORD VALUES('100000000010', '22314213', 'testing content 5');
```

```
--SKILLLEVEL
```

```
INSERT INTO SKILLLEVEL VALUES('21103942', 'Englsih', 'Good');
INSERT INTO SKILLLEVEL VALUES('21103950', 'MIS', 'Good');
INSERT INTO SKILLLEVEL VALUES('21102912', 'Statistics', 'Fine');
INSERT INTO SKILLLEVEL VALUES('21123942', 'Finance', 'Good');
INSERT INTO SKILLLEVEL VALUES('21114242', 'Physics', 'Bad');
INSERT INTO SKILLLEVEL VALUES('22314223', 'Chemistry', 'Good');
INSERT INTO SKILLLEVEL VALUES('22314210', 'History', 'Good');
INSERT INTO SKILLLEVEL VALUES('22314211', 'Geography', 'Good');
INSERT INTO SKILLLEVEL VALUES('22314212', 'Law', 'Good');
INSERT INTO SKILLLEVEL VALUES('22314213', 'Economy', 'Good');
```

```
--INCHARGE
```

```
INSERT INTO INCHARGE VALUES('Englsih', '10000031');
INSERT INTO INCHARGE VALUES('MIS', '10000032');
INSERT INTO INCHARGE VALUES('Statistics', '10000033');
INSERT INTO INCHARGE VALUES('Finance', '10000034');
INSERT INTO INCHARGE VALUES('Physics', '10000035');
INSERT INTO INCHARGE VALUES('Chemistry', '10000036');
INSERT INTO INCHARGE VALUES('History', '10000037');
INSERT INTO INCHARGE VALUES('Geography', '10000038');
INSERT INTO INCHARGE VALUES('Law', '10000039');
INSERT INTO INCHARGE VALUES('Economy', '10000040');
```

commit;

Then, we offer our SQL queries together with the detailed inline comments that clearly explain how did we make these queries and what these SQL queries can realize.

-- 1

```
-- Find all students who play volleyball that are taking more than 3 courses,  
-- display the tutor sessions for the past 12 months in the current semester.  
select students.userid "Student ID", firstname || ' ' || lastname AS "Student Name",  
count(distinct bookingid) AS "Number of Appointments"  
from students, bookings, users
```

```
where bookingtype_ID = 'APPOINTMENT'  
and sysdate-booking_date<365  
---- check that students have more than 3 courses.  
and students.userid in  
(select student  
from attendancerecord  
group by student  
having count(course) >=3)  
---- joining the tables  
and students.userid = bookings.userid  
and students.userid = users.userid  
---- plays basketball  
and sporttype = 'volleyball'  
---- groupby clause  
group by students.userid, firstname, lastname;
```

-- 2

```
-- Find all students that had canceled at least 1 tutor sessions  
-- in the past year, display their counselor  
-- and the number of course they are taking.  
select students.userid AS "Student ID", users.firstname AS "Student First Name",  
users.lastname AS "Student Last Name",  
counselors1.couns_fn || ' ' || counselors1.couns_ln AS "Counselor Name", course1.num_courses  
"Number of Courses"  
from students, users, bookings,  
---- table with names for each student's counselor
```

```

(select students.userid AS s_id, u2.firstname AS couns_fn,
u2.lastname AS couns_ln from
users u1, users u2, counselors, students
where u1.userid = students.userid
and u2.userid = counselors.userid
and students.counselor_id = counselors.userid) counselors1,
---- table with counts of each student's courses
(select student AS course_stu, count(course) AS num_courses
from attendancerecord
group by student) course1

--- only count appointments that are cancelled in the last 365 days.
where bookingtype_id = 'APPOINTMENT'
and status = 'Cancelled'
and sysdate - booking_date < 365
--- join all the tables
and users.userid = students.userid
and users.userid = bookings.userid
and users.userid = counselors1.s_id
and course1.course_stu = users.userid
--- group by
group by students.userid, users.firstname, users.lastname, counselors1.couns_fn,
counselors1.couns_ln, course1.num_courses
having count(bookingid)>= 1;

-- 3
/*
Give the count of students per counselor and sports type.
*/
select couns1.FIRSTNAME || ' ' || couns1.LASTNAME "Counselor Name", SPORTTYPE
"Sport",
count(distinct stu1.userid) AS "Number of Students"
from counselors, staff, users couns1, students, users stu1
-- join students and counselors
where students.COUNSELOR_ID = couns1.USERID
-- get the full counselor row
AND counselors.USERID = staff.USERID
AND staff.USERID = couns1.userid
-- get the student row
AND students.USERID = stu1.USERID

```

```

group by couns1.FIRSTNAME || ' ' || couns1.LASTNAME, SPORTTYPE;

-- 4
/*
For each student, count the number of reservations in the last year, and most common size of
room they
reserve.*/
select userid "User ID", count (distinct bookingid) "Number of Bookings",
STATS_MODE(roomsize) AS "Common Room Size"
from students natural join users natural join bookings natural join reservations join rooms on
room = roomnumber
where sysdate - booking_date < 365
group by userid;

```

```

-- 5
/*
Display all students that didn't show up to at least 1 tutor sessions without cancellation;
display the number of course they are taking and the name of their counselor.
*/
select stu1.firstname || stu1.lastname "Student Name", couns1.firstname || couns1.lastname
"Counselor Name",
coursenum AS "Number of Courses"
from counselors, staff, users couns1, students, users stu1,
(select student as stu_id, count(distinct course) as coursenum
from attendancerecord
group by student ) stucourse
-- join students and counselors
where students.COUNSELOR_ID = couns1.USERID
-- get the full counselor row
AND counselors.USERID = staff.USERID
AND staff.USERID = couns1.userid
-- get the student row
AND students.USERID = stu1.USERID
-- course numbers
AND students.userid = stucourse.stu_id
-- only students that no showed more than twice in the last 365 days.
and students.userid IN
(select userid from bookings where bookingtype_id = 'APPOINTMENT'
and status = 'NoShow'

```



```

and sysdate - booking_date < 365
group by userid
having count(bookingid) >= 1);

```

```
-- 6
```

```
/*
```

For each subject, count the number of learning specialists and tutors.
Order by the sum of both numbers.

```
*/
```

```

select subjects.subjectname AS "Subject Name",
lscount "Learning Specialist #",
tutscout "Tutors #"
from
(select subjectname, count (distinct LEARNINGSPECIALIST) as lscount
from incharge join subjects on subjectname = subject
group by subjectname) ls1,
(select subjectname, count (distinct tutor) as tutscout
from subjects join skilllevel on subjectname = subject
group by subjectname) tuts1,
subjects
where subjects.subjectname = ls1.subjectname
AND subjects.subjectname = tuts1.subjectname
order by lscout + tutscout DESC;

```

```
-- 7
```

```
/*
```

For each learning specialist in this Booking & Scheduling system who is responsible for at least 1 subject, display the user_id (Heading: ◆User ID◆), full name (Heading: ◆Learning Specialist Name◆), the number of subjects each coordinator is charged of (Heading: ◆Subject Number◆) and also the number of students who enrolled course(s) that is in each subject. Order by the sum of both.

```
*/
```

```

select ls1.userid "User ID",
ls1.FIRSTNAME || ' ' || ls1.LASTNAME AS "Learning Specialist Name",
lssubject.subj_num "Subject Number", ls_students.stu_num "Student Number"
from
-- get LS information
(select * from LEARNINGSPECIALISTS natural join staff natural join users) ls1,
-- get number of subjects per LS

```

```

(select LEARNINGSPECIALIST, count(subject) subj_num from incharge group by
LEARNINGSPECIALIST) lsubject,
-- get number of students per LS
(select LEARNINGSPECIALIST, count(student) stu_num from incharge natural join courses
natural join ATTENDANCERECORD
group by LEARNINGSPECIALIST) ls_students
-- only LS with more than 2 subjects
where ls1.userid in (select LEARNINGSPECIALIST from incharge group by
LEARNINGSPECIALIST having count(subject)>= 1)
-- join tables for counts of subjects
and ls1.userid = lsubject.LEARNINGSPECIALIST
-- join tables for count of students
and ls1.userid = ls_students.LEARNINGSPECIALIST
order by ls_students.stu_num + lsubject.subj_num DESC
;

```

-- 8

/*

For each room size, count the number of reservations in the last month, the number of rooms with reservations in that type. Sort by number of reservations. Also provide the userID of the user who makes the most reservations, and the most common type of booking. */

```

select roomsize AS "Roomsize", count (distinct roomnumber) AS "Number of Rooms",
count(bookingid) AS "Number of Reservations", STATS_MODE(userid) AS "Most
Reservations UserID"
, STATS_MODE (BOOKINGTYPE_ID) AS "Most Common Booking Type"
from rooms join bookings on rooms.roomnumber = bookings.room
where bookingtype_ID = 'RESERVATION'
group by roomsize
order by count (distinct roomnumber);

```

-- 9

/* for each subject, provide the ID's of the professors that teach courses in that subject */

```

select subjectname, LISTAGG(PROFESSOR, ',') WITHIN GROUP (ORDER BY
courses.COURSE_ID) "Professors"
from subjects join courses on subjects.subjectname = courses.subject
join courseschedule on courseschedule.COURSE = courses.COURSE_ID
GROUP BY (subjectname);

```

-- 10

/* For basketball and baseball students, find the tutors they meet with the most. */

```
select tutid AS "Tutor ID", numstudents AS "Number of Students" from (  
select tutors.TUTORID tutid, count (*) AS numstudents  
from appointments join tutors on appointments.tutorid = tutors.tutorid  
join bookings on bookings.BOOKINGid = appointments.BOOKINGID  
join students on bookings.userid = students.userid  
where (SPORTTYPE = 'basketball' OR SPORTTYPE= 'baseball')  
group by tutors.tutorid  
order by numstudents)  
where rownum = 1;
```

-- 11

/* Sports with most cancellations*/

```
SELECT * FROM (  
select students.SPORTTYPE AS "Sports Type", count (*) AS "Num cancellations"  
from bookings join students on bookings.userid = students.userid  
where status = 'Cancelled'  
group by students.SPORTTYPE  
order by count(*) DESC  
)  
where rownum = 1;
```

-- 12

/* Sports with most reservations*/

```
SELECT * FROM (  
select students.SPORTTYPE AS "Sports Type", count (*) AS "Num reservations"  
from bookings join students on bookings.userid = students.userid  
where bookingtype_id = 'RESERVATION'  
group by students.SPORTTYPE  
order by count(*) DESC  
)  
where rownum = 1;
```

-- 13

/* Find the rooms that contain more than 1 equipment and contain at least 1 or more bookings,
where the bookings happened in the last year. */

```
select ROOMNUMBER from rooms
```

where roomnumber IN (select room from roomequipment group by room having count (distinct equipment_ID) > 1)

and roomnumber IN (select room from BOOKINGS where sysdate - booking_date < 365 group by room having count (distinct bookingID) >= 1);

At last, we also provide the following SQL code for future maintenance:

```
drop view VIEW_ADMINISTRATORS_UPDATEABLE;
drop view VIEW_FRONTDESKS_UPDATEABLE;
drop view VIEW_LEARNINGSPECS_UPDATEABLE;
drop view VIEW_EVENTS_UPDATEABLE;
drop view VIEW_TRAININGSESS_UPDATEABLE;
drop view VIEW_RESERVATION_UPDATEABLE;
```

```
drop trigger INSERT_STU_VIEW;
drop trigger INSERT_ADMINS_VIEW;
drop trigger INSERT_FRONTDESKS_VIEW;
drop trigger INSERT_COUNSELORS_VIEW;
drop trigger INSERT_LEARNINGSPEC_VIEW;
drop trigger INSERT_APPOINTMENT_VIEW;
drop trigger INSERT_EVENTS_VIEW;
drop trigger INSERT_TRAININGSESS_VIEW;
drop trigger INSERT_RESERVATIONS_VIEW;
```

```
drop table ATTENDANCERECORD cascade constraints;
drop table BOOKINGS cascade constraints;
drop table BOOKINGTYPE cascade constraints;
drop table CANCREATE cascade constraints;
drop table CANSUPPLY cascade constraints;
drop table COMPONENTS cascade constraints;
drop table COUNSELORS cascade constraints;
drop table COURSES cascade constraints;
drop table COURSESCCHEDULE cascade constraints;
drop table CUSTOMERS cascade constraints;
drop table DAILYWORKRECORD cascade constraints;
drop table EQUIPMENTTYPE cascade constraints;
drop table EVENTS cascade constraints;
drop table FORM cascade constraints;
drop table FRONTDESKS cascade constraints;
drop table INCHARGE cascade constraints;
```

```

drop table LEARNINGSPECIALISTS cascade constraints;
drop table LOANDETAILS cascade constraints;
drop table LOANDOCSSUBMITTED cascade constraints;
drop table LOANDOCUMENTTOSTATUSMAPPING cascade constraints;
drop table LOANTYPEREPORT cascade constraints;
drop table LOANTYPES cascade constraints;
drop table PRODUCTS cascade constraints;
drop table PROFESSORS cascade constraints;
drop table PROPERTIES cascade constraints;
drop table REFKEYMAPPING cascade constraints;
drop table REGISTRATIONS cascade constraints;
drop table RESERVATIONS cascade constraints;
drop table ROOMCLASSIFICATION cascade constraints;
drop table ROOMEQUIPMENT cascade constraints;
drop table ROOMS cascade constraints;
drop table ROOMSCHEDULE cascade constraints;
drop table SCORECONVERSIONCHART cascade constraints;
drop table SKILLLEVEL cascade constraints;
drop table SPORTS cascade constraints;
drop table STAFF cascade constraints;
drop table STUDENTS cascade constraints;
drop table SUBJECTS cascade constraints;
drop table TRAININGRECORD cascade constraints;
drop table TRAINING_SESSIONS cascade constraints;
drop table TUTORS cascade constraints;
drop table USERS cascade constraints;
drop table USERTYPE cascade constraints;
drop table VENDORS cascade constraints;
drop table administrators cascade constraints;

```

Chapter 5.

In the Chapter 5, we show both triggers and views in detail, together with the detailed inline comments and some query tests.

```

/*****
TRIGGERS AND VIEWS
*****/

```

```

-- upon update or insert of a user record with field, usertype,

```

```

-- must update count of user types
CREATE OR REPLACE TRIGGER update_user_type_counts
--- runs only once for any of these query types, not once for each row.
AFTER INSERT OR UPDATE OR DELETE
on USERS
DECLARE
---- cursor to find usertypes and counts from their table
CURSOR c1 IS
select * from usertype
where usertype_id = 'STAFF' OR usertype_id = 'STUDENT'
for update;

CURSOR c2 IS
select * from usertype
where usertype_id <> 'STAFF' AND usertype_id <> 'STUDENT'
for update;
BEGIN
---- go through each usertype and update the count
for x in c1
loop
update usertype
---- for the specific usertype, count how many records are now there.
set numofusers = (select count(*) from users where usertype_id = x.usertype_id)

---- update the row that corresponds to the current one in the cursor/loop.
where current of c1;
end loop;

for x in c2
loop
update usertype
---- for the specific stafftype, count how many records are now there.
set numofusers = (select count(*) from staff where stafftype = x.usertype_id)

---- update the row that corresponds to the current one in the cursor/loop.
where current of c2;
end loop;

END;
/

```

```

-- upon insertion of a new user, delete the matching refkey.
CREATE OR REPLACE TRIGGER delete_refkey_new_user
--- runs only once for any of these query types, not once for each row.
AFTER INSERT
on USERS
for each row
BEGIN
delete from refkeymapping where
refkeymapping.universityID = :new.universityID;
END;
/

-- following trigger can sometimes fail compilation.
/*
-- upon update or insert of an equipment record
-- must update count of equipment by type.
CREATE OR REPLACE TRIGGER update_equipment_counts
--- runs only once for any of these types, not once for each row.
AFTER INSERT OR UPDATE OR DELETE
on roomequipment
DECLARE
CURSOR c1 IS
select * from equipmenttype
for update;
BEGIN
---- go through each equipmenttype and update the count
for x in c1
loop
update equipmenttype
---- for the specific usertype, count how many records are now there.
set numofequip = (select count(*) from roomequipment where equipment_type =
x.EQUIPMENTTYPE_NAME)

---- update the row that corresponds to the current one in the cursor/loop.
where current of c1;
end loop;

END;
/

```

```
*/
```

```
/* Views: Stored Queries */
```

```
/* 5 updateable view for users
```

```
These work with a trigger to allow us to simplify adding users (who are split across multiple tables). */
```

```
/* STUDENTS */
```

```
drop view view_students_updateable;
```

```
create view view_students_updateable
```

```
AS select * from users natural join STUDENTS;
```

```
/*
```

```
create view view_students_updateable
```

```
AS select universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS, CITY,  
STATE, ZIPCODE, EMAIL, PHONENO,
```

```
USERTYPE_ID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID from users  
natural join STUDENTS;
```

```
*/
```

```
create or replace trigger insert_stu_view
```

```
instead of insert
```

```
on view_students_updateable
```

```
for each row
```

```
declare
```

```
x users.userid%type;
```

```
begin
```

```
insert into users(universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS,  
CITY, STATE, ZIPCODE, EMAIL, PHONENO,  
USERTYPE_ID)
```

```
values(:new.universityid, :new.PASSWORD, :new.FIRSTNAME, :new.LASTNAME, :new.ST  
REETADDRESS, :new.CITY,  
:new.STATE, :new.ZIPCODE, :new.EMAIL, :new.PHONENO, :new.USERTYPE_ID)  
RETURNING USERID INTO x;
```

```
insert into students(USERID,ACADEMIC_STANDING, SPORTTYPE, COUNSELOR_ID)
```

```
values(x,
```

```
:new.ACADEMIC_STANDING, :new.SPORTTYPE, :new.COUNSELOR_ID);
```



```

end;
/

/*
select * from view_students_updateable;

insert into view_students_updateable(universityid, PASSWORD, FIRSTNAME, LASTNAME,
STREETADDRESS, CITY,
STATE, ZIPCODE, EMAIL, PHONENO, USERTYPE_ID, ACADEMIC_STANDING,
SPORTTYPE, COUNSELOR_ID)
values (6,'1','Yazan','Fake','street','city','state','zipcode', 'yna@email.com','12312321'
,'STUDENT','Sophomore', 'Basketball', '10000030'); */
/* END STUDENTS */

/* LEARNING SPECIALISTS */
create or replace view view_learningspecs_updateable
AS select * from users natural join LEARNINGSPECIALISTS natural join staff;

/*
create or replace view view_learningspecs_updateable
AS select universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS, CITY,
STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID, STAFFTYPE from users natural join LEARNINGSPECIALISTS natural join
staff; */

create or replace trigger insert_learningspec_view
instead of insert
on view_learningspecs_updateable
for each row
declare
x users.userid%type;
begin

insert into users(universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS,
CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID)
values(:new.universityid, :new.PASSWORD, :new.FIRSTNAME, :new.LASTNAME, :new.ST
REETADDRESS, :new.CITY,
:new.STATE, :new.ZIPCODE, :new.EMAIL, :new.PHONENO, :new.USERTYPE_ID)
RETURNING USERID INTO x;

```

```
insert into staff(userid,STAFFTYPE) values(x,:new.STAFFTYPE)
RETURNING USERID INTO x;
```

```
insert into LEARNINGSPECIALISTS(userid) values(x);
```

```
end;
```

```
/
```

```
/*
```

```
select * from view_learningspecs_updateable;
```

```
insert into view_learningspecs_updateable(universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY,
STATE, ZIPCODE, EMAIL, PHONENO, USERTYPE_ID, STAFFTYPE)
values (20,'1','Yazan','Fake','street','city','state','zipcode', 'yna@email.com','12312321'
,'STAFF','LEARNINGSPECIALIST'); */
```

```
/* END LEARNING SPECIALISTS */
```

```
/* ADMINISTRATORS */
```

```
create or replace view view_administrators_updateable
AS select * from users natural join staff natural join ADMINISTRATORS;
```

```
/*
```

```
create or replace view view_administrators_updateable
AS select universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS, CITY,
STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID, STAFFTYPE, MANAGED_DEPARTMENTS from users natural join staff
natural join ADMINISTRATORS; */
```

```
create or replace trigger insert_admins_view
instead of insert
on view_administrators_updateable
for each row
declare
x users.userid%type;
begin
```

```

insert into users(universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS,
CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID)
values(:new.universityid, :new.PASSWORD, :new.FIRSTNAME, :new.LASTNAME, :new.ST
REETADDRESS, :new.CITY,
:new.STATE, :new.ZIPCODE,      :new.EMAIL, :new.PHONENO, :new.USERTYPE_ID)
RETURNING USERID INTO x;

insert into staff(userid,STAFFTYPE) values(x,:new.STAFFTYPE)
RETURNING USERID INTO x;

insert into ADMINISTRATORS(userid,MANAGED_DEPARTMENTS)
values(x,:new.MANAGED_DEPARTMENTS);

end;
/
/*
select * from view_administrators_updateable;

insert into view_administrators_updateable(universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY,
STATE, ZIPCODE, EMAIL, PHONENO, USERTYPE_ID, STAFFTYPE,
MANAGED_DEPARTMENTS)
values (25,'1','Yazan','Fake','street','city','state','zipcode', 'yna@email.com','12312321'
,'STAFF','ADMINISTRATOR','Financial'); */
/* END ADMINISTARTORS */

/* FRONTDESKS */
create or replace view view_frontdesks_updateable
AS select * from users natural join staff natural join FRONTDESKS;

/*
create or replace view view_frontdesks_updateable
AS select universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS, CITY,
STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID, STAFFTYPE from users natural join staff natural join FRONTDESKS;
*/

create or replace trigger insert_frontdesks_view
instead of insert

```

```

on view_frontdesks_updateable
for each row
declare
x users.userid%type;
begin

insert into users(universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS,
CITY, STATE, ZIPCODE, EMAIL, PHONENO,
USERTYPE_ID)
values(:new.universityid, :new.PASSWORD, :new.FIRSTNAME, :new.LASTNAME, :new.ST
REETADDRESS, :new.CITY,
:new.STATE, :new.ZIPCODE, :new.EMAIL, :new.PHONENO, :new.USERTYPE_ID)
RETURNING USERID INTO x;

insert into staff(userid,STAFFTYPE) values(x,:new.STAFFTYPE)
RETURNING USERID INTO x;

insert into FRONTDESKS(userid) values(x);

end;
/
/*
select * from view_frontdesks_updateable;

insert into view_frontdesks_updateable(universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY,
STATE, ZIPCODE, EMAIL, PHONENO, USERTYPE_ID, STAFFTYPE)
values (25,'1','Yazan','Fake','street','city','state','zipcode', 'yna@email.com','12312321'
,'STAFF','FRONTDESK'); */
/* END FRONTDESKS */

/* COUNSELORS */
create or replace view view_counselors_updateable
AS select * from users natural join staff natural join COUNSELORS ;

select * from view_counselors_updateable;

/*
create or replace view view_counselors_updateable

```

```

AS select universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS, CITY,
      STATE, ZIPCODE,  EMAIL, PHONENO,
USERTYPE_ID, STAFFTYPE from users natural join staff natural join COUNSELORS ;
*/

```

```

create or replace trigger insert_counselors_view
instead of insert
on view_counselors_updateable
for each row
declare
x users.userid%type;
begin

```

```

insert into users(universityid, PASSWORD, FIRSTNAME, LASTNAME, STREETADDRESS,
CITY, STATE, ZIPCODE,  EMAIL, PHONENO,
USERTYPE_ID)
values(:new.universityid, :new.PASSWORD, :new.FIRSTNAME, :new.LASTNAME, :new.ST
REETADDRESS, :new.CITY,
:new.STATE, :new.ZIPCODE,      :new.EMAIL, :new.PHONENO, :new.USERTYPE_ID)
RETURNING USERID INTO x;

```

```

insert into staff(userid,STAFFTYPE) values(x,:new.STAFFTYPE)
RETURNING USERID INTO x;

```

```

insert into COUNSELORS(userid) values(x);

```

```

end;

```

```

/

```

```

/*

```

```

select * from view_counselors_updateable;

```

```

insert into view_counselors_updateable(universityid, PASSWORD, FIRSTNAME,
LASTNAME, STREETADDRESS, CITY,
STATE, ZIPCODE,  EMAIL, PHONENO, USERTYPE_ID, STAFFTYPE)
values (109,'1','Yazan','Fake','street','city','state','zipcode', 'yna@email.com','12312321'
,'STAFF','COUNSELOR'); */
/* END COUNSELORS*/

```

```

/*****

```

```

END OF UPDATEABLE USER VIEWS

```

```
*****/
```

```
/* 4 updateable view for bookings
```

```
These work with a trigger to allow us to simplify adding bookings, which are spread across 2 tables. */
```

```
/* Appointments */
```

```
drop view view_appointments_updateable;  
create view view_appointments_updateable  
AS select * from bookings natural join appointments;
```

```
/*
```

```
create view view_appointments_updateable  
AS select BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME,  
END_TIME,  
BOOKING_DATE, USERID, tutorid from bookings natural join appointments;  
*/
```

```
create or replace trigger insert_appointment_view  
instead of insert  
on view_appointments_updateable  
for each row  
declare  
x APPOINTMENTS.BOOKINGID%type;  
begin
```

```
insert into BOOKINGS(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER,  
START_TIME, END_TIME, BOOKING_DATE, USERID)  
values(:new.BOOKINGTYPE_ID, :new.STATUS, :new.ROOM, :new.DURATION, :new.SEM  
ESTER, :new.START_TIME,  
:new.END_TIME, :new.BOOKING_DATE, :new.USERID)  
RETURNING bookingid INTO x;
```

```
insert into appointments(bookingid, tutorid) values(x,:new.tutorid);
```

```
end;
```

```
/
```

```
/*
```

```
select * from view_appointments_updateable;
```

```

insert into view_appointments_updateable(BOOKINGTYPE_ID, STATUS, ROOM,
DURATION, SEMESTER, START_TIME,
END_TIME, BOOKING_DATE, USERID, TUTORID)
values ('APPOINTMENT', NULL, 'L206', 1, 'Spring2016', '01-DEC-16',
'01-DEC-16', '10-MAR-16', '10000002', '21103942'); */
/* END Appointments */

/* EVENTS */
drop view view_events_updateable;
create view view_events_updateable
AS select * from bookings natural join events;

/*
create view view_events_updateable
AS select BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME,
END_TIME,
BOOKING_DATE, USERID, event_name, topic from bookings natural join events;
*/

create or replace trigger insert_events_view
instead of insert
on view_events_updateable
for each row
declare
x events.BOOKINGID%type;
begin

insert into BOOKINGS(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER,
START_TIME, END_TIME, BOOKING_DATE, USERID)
values(:new.BOOKINGTYPE_ID, :new.STATUS, :new.ROOM, :new.DURATION, :new.SEM
ESTER, :new.START_TIME,
:new.END_TIME, :new.BOOKING_DATE, :new.USERID)
RETURNING bookingid INTO x;

insert into events(bookingid, event_name, topic) values(x,:new.event_name, :new.topic);

end;
/

```

```

/*
select * from view_events_updateable;
insert into view_events_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION,
SEMESTER, START_TIME,
END_TIME, BOOKING_DATE, USERID, EVENT_NAME, TOPIC)
values ('APPOINTMENT', NULL,'L206', 1, 'Spring2016','01-DEC-16',
'01-DEC-16','10-MAR-16', '10000003','Test Name', 'Test Topic'); */
/* END EVENTS */

/* TRAINING_SESSIONS */
drop view view_trainingsess_updateable;
create view view_trainingsess_updateable
AS select * from bookings natural join training_sessions;

/*
create view view_trainingsess_updateable
AS select BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME,
END_TIME, BOOKING_DATE, USERID,
TRAINING_TITLE from bookings natural join training_sessions;
*/

create or replace trigger insert_trainingsess_view
instead of insert
on view_trainingsess_updateable
for each row
declare
x training_sessions.BOOKINGID%type;
begin

insert into BOOKINGS(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER,
START_TIME, END_TIME, BOOKING_DATE, USERID)
values(:new.BOOKINGTYPE_ID, :new.STATUS, :new.ROOM, :new.DURATION, :new.SEM
ESTER, :new.START_TIME,
:new.END_TIME, :new.BOOKING_DATE, :new.USERID)
RETURNING bookingid INTO x;

insert into TRAINING_SESSIONS(BOOKINGID, TRAINING_TITLE)
values(x, :new.TRAINING_TITLE);

```



```

end;
/

/*
select * from view_trainingsess_updateable;
insert into view_trainingsess_updateable(BOOKINGTYPE_ID, STATUS, ROOM,
DURATION, SEMESTER, START_TIME,
END_TIME, BOOKING_DATE, USERID, TRAINING_TITLE)
values ('APPOINTMENT', NULL,'L206', 1, 'Spring2016','01-DEC-16',
'01-DEC-16','10-MAR-16', '10000003','Test Title'); */
/* END TRAINING_SESSIONS */

/* RESERVATIONS */
drop view view_reservation_updateable;
create view view_reservation_updateable
AS select * from bookings natural join reservations;

/*
create view view_reservation_updateable
AS select BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER, START_TIME,
END_TIME, BOOKING_DATE, USERID
,RESERVATION_DETAIL from bookings natural join reservations;
*/

create or replace trigger insert_reservations_view
instead of insert
on view_reservation_updateable
for each row
declare
x reservations.BOOKINGID%type;
begin

insert into BOOKINGS(BOOKINGTYPE_ID, STATUS, ROOM, DURATION, SEMESTER,
START_TIME, END_TIME, BOOKING_DATE, USERID)
values(:new.BOOKINGTYPE_ID, :new.STATUS, :new.ROOM, :new.DURATION, :new.SEM
ESTER, :new.START_TIME,
:new.END_TIME, :new.BOOKING_DATE, :new.USERID)
RETURNING bookingid INTO x;

```

```

insert into RESERVATIONS(BOOKINGID, RESERVATION_DETAIL)
values(x, :new.RESERVATION_DETAIL);

end;
/

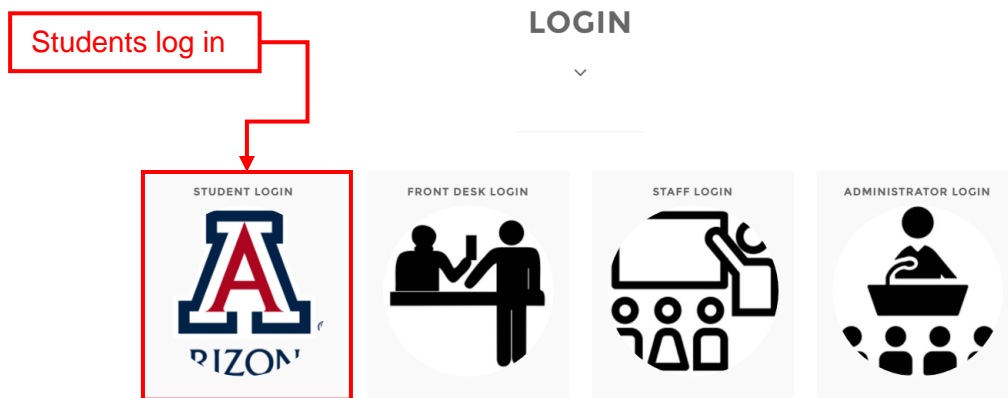
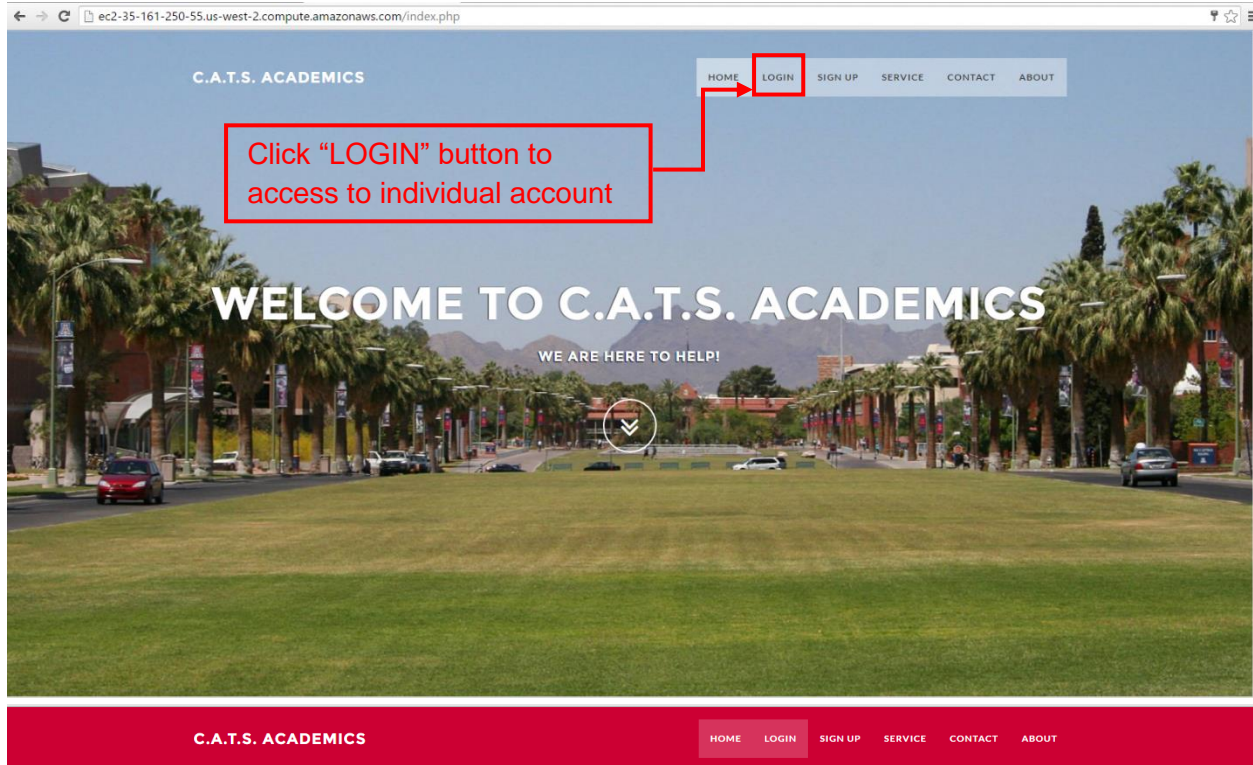
/*
select * from view_reservation_updateable;
insert into view_reservation_updateable(BOOKINGTYPE_ID, STATUS, ROOM, DURATION,
SEMESTER, START_TIME,
END_TIME, BOOKING_DATE, USERID, RESERVATION_DETAIL)
values ('APPOINTMENT', NULL,'L206', 1, 'Spring2016','01-DEC-16',
'01-DEC-16','10-MAR-16', '10000003','Test Reservation Detail');
*/
/* END RESERVATIONS */

/*****
END OF UPDATEABLE BOOKINGS VIEWS
*****/

```

Chapter 6.

After finishing all requirement analysis, conceptual and logical design, not only the SQL work we displayed in previous chapters, our team also began to develop our front-end demo. Then, following the feedback from our client, our finalized our front-end demo that met all basic functional needs this Wednesday (12/07/16). Now, we will give a systematic walkthrough to show how our front-end works, and at the end of this chapter, our team offers the URL link to this new online booking system.




Username: wanv1174
Password: llove416@
→ Log in

Username
wanv1174

Password
.....

STUDENT LOGIN



FR

ADMINISTRATOR LOGIN

Hello, Athlete!

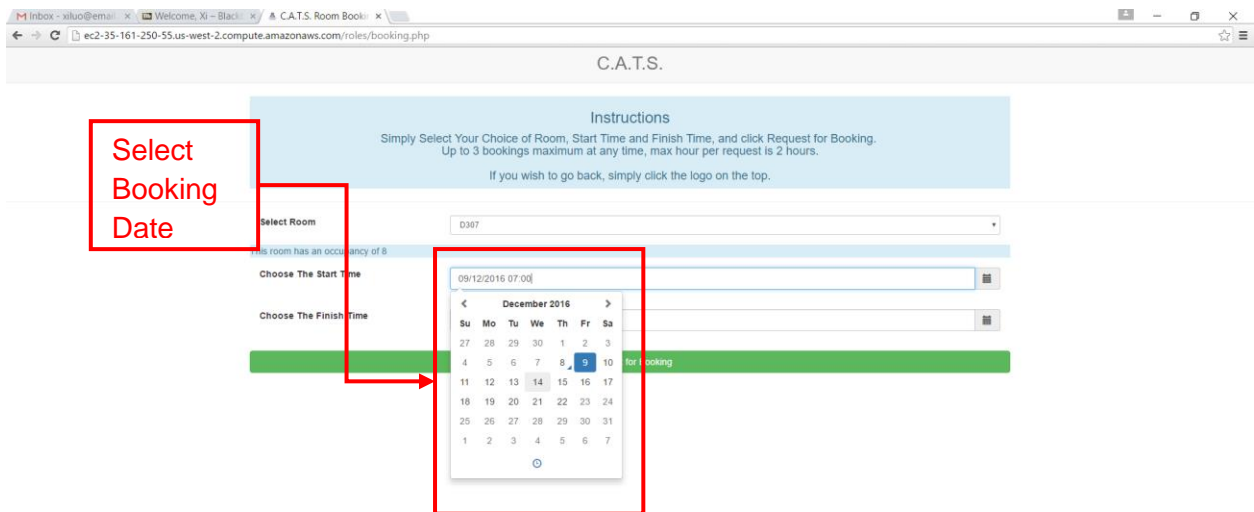
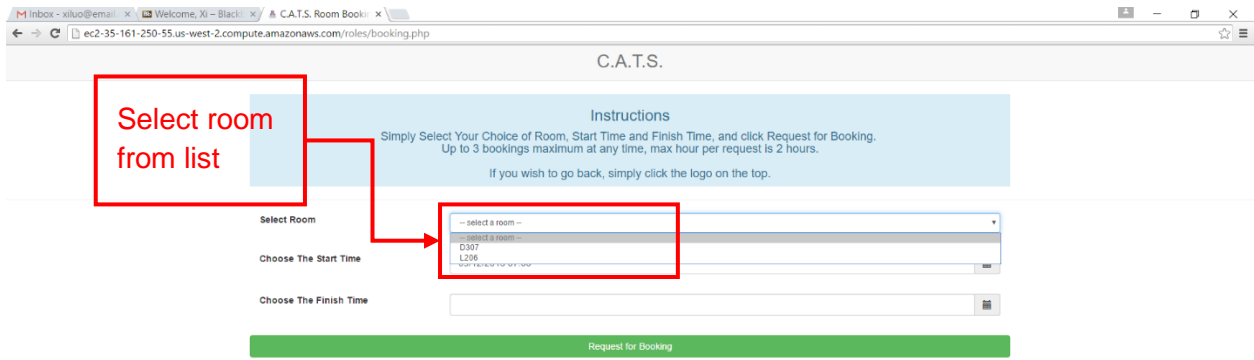
Click this button

Create New Booking

Create New Booking

Check Existing Requests

Check Existing Booking Requests



C.A.T.S.

Instructions

Simply Select Your Choice of Room, Start Time and Finish Time, and click Request for Booking.
Up to 3 bookings maximum at any time, max hour per request is 2 hours.
If you wish to go back, simply click the logo on the top.

Select
Booking
Time

Select Room: D307

This room has an occupancy of 8

Choose The Start Time: 10/12/2016 12:00

Choose The Finish Time: [empty]

Request for Booking

Submit
new
booking
request

Choose The Start Time: 10/12/2016 12:00

Choose The Finish Time: 10/12/2016 14:00

Request for Booking

...5-161-250-55-us-west-2.compute.amazonaws.com says:
Your booking request has been submitted, an administrator will review your request and contact you later!

OK

List Existing Bookings

| Request ID | Room Number | Start Time | End Time | Request Status | |
|------------|-------------|------------------|------------------|----------------|--------------------------------|
| 3 | L206 | 07/12/2016 07:00 | 07/12/2016 09:00 | Pending | Remove Request |
| 4 | L206 | 07/12/2016 08:00 | 07/12/2016 10:00 | Pending | Remove Request |
| 6 | D307 | 10/12/2016 12:00 | 10/12/2016 14:00 | Pending | Remove Request |

Click here to remove request

List Existing Bookings

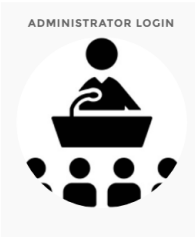
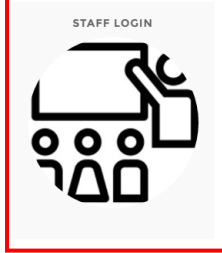
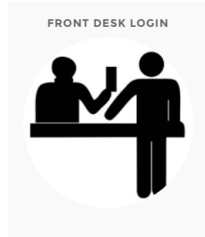
| Request ID | Room Number | Start Time | End Time | Request Status | |
|------------|-------------|------------------|------------------|----------------|--------------------------------|
| 3 | L206 | 07/12/2016 07:00 | 07/12/2016 09:00 | Pending | Remove Request |
| 4 | L206 | 07/12/2016 08:00 | 07/12/2016 10:00 | Pending | Remove Request |

Bookings list after removing request

LOGIN

Staff click here to log in

Username: wanv1173
Password: llove416@
→ Log in



Hello, Employee!

Check Existing Requests

Manage Student Booking Requests

Click here to manage the booking requests

List of Pending Requests

| Request ID | Room Number | Start Time | End Time | User ID | Request Status | | |
|------------|-------------|------------------|------------------|----------|----------------|-----------------|-----------------|
| 3 | L206 | 07/12/2016 07:00 | 07/12/2016 08:00 | wanv1174 | Pending | Approve Request | Decline Request |
| 4 | L206 | 07/12/2016 08:00 | 07/12/2016 10:00 | wanv1174 | Pending | Approve Request | Decline Request |
| 6 | D307 | 10/12/2016 12:00 | 10/12/2016 14:00 | wanv1174 | Pending | Approve Request | Decline Request |

...5-161-250-55.us-west-2.compute.amazonaws.com says:
Are you sure you want to approve this request?
OK Cancel

Staff, such as counselor, click to approve students' booking request

List of Pending Requests

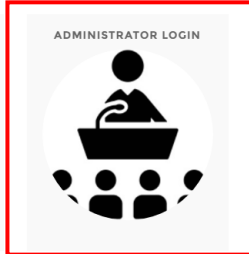
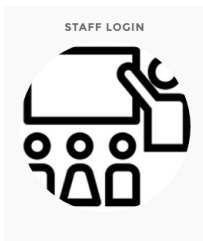
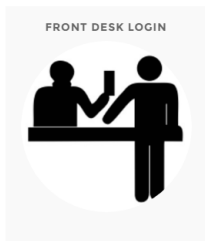
| Request ID | Room Number | Start Time | End Time | User ID | Request Status | | |
|------------|-------------|------------------|------------------|----------|----------------|-----------------|-----------------|
| 4 | L206 | 07/12/2016 08:00 | 07/12/2016 10:00 | wanv1174 | Pending | Approve Request | Decline Request |
| 6 | D307 | 10/12/2016 12:00 | 10/12/2016 14:00 | wanv1174 | Pending | Approve Request | Decline Request |

...5-161-250-55.us-west-2.compute.amazonaws.com says:
Are you sure you want to decline this request?
 Prevent this page from creating additional dialogs.
OK Cancel

Staff, such as counselor, click to decline students' booking request

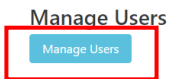
Username: wanv1172
Password: llove416@
→ Log in

LOGIN



Administrator click here to log in

Hello, Administrator!



Administrator click here to manage users

List Active Users

| Account | University ID | Name | User Address | Email | Phone Number | User Type | Accountstatus | Action |
|----------|---------------|---------------|--------------------------------------|------------------------------|--------------|---------------|---------------|-----------------|
| 10000001 | 1 | Joe Wong | 213 W Park Ave, Tucson, AZ, 85719 | JoeWong@catmail.arizona.edu | 5203312345 | STUDENT | Enabled | Disable Account |
| 10000002 | 2 | Patrick Wates | 1023 W Wetmore Rd, Tucson, AZ, 85705 | PWates@catmail.arizona.edu | 5203224341 | STAFF | Disabled | Disable Account |
| 10000003 | 23140000 | William H. | 103 S Oracle St, Tucson, AZ, 85721 | HWilliam@catmail.arizona.edu | 5203009345 | STAFF | Enabled | Disable Account |
| 10000004 | 4 | Mary JP. | 1108 8th St, Tucson, AZ, 85721 | MaryJP@catmail.arizona.edu | 5203172305 | STAFF | Enabled | Disable Account |
| 10000005 | 5 | Lucy S. | 2222 22nd St, Tucson, AZ, 85719 | LucyS@catmail.arizona.edu | 5203310891 | STAFF | Enabled | Disable Account |
| wanv1171 | 0 | W 1111 | AK, Adak, 99546, W@email.arizona.edu | 23147654 | 555 | Student | Enabled | Disable Account |
| wanv1172 | 21111111 | W W | W, AK, Adak, 99546 | W@W.com | 5 | Administrator | Enabled | Disable Account |
| wanv1173 | 23147111 | W W | 5, AK, Adak, 99546 | W@W.com | 5 | Counselor | Enabled | Disable Account |
| wanv1174 | 23147643 | W W | 5, AK, Adak, 99546 | W@W.com | 5 | Student | Enabled | Disable Account |

Display all active users

The screenshot shows the C.A.T.S. Administrator interface. At the top, there are navigation links: Administrator Dashboard, Manage Users, Manage Equipment, Manage Room, and Sign Out. The main content area is divided into two sections:

Generate Referral Key for User

This section contains a form with the following fields:

- Enter User University ID Number: 23478122
- Select User Type: A dropdown menu with options: Student, Student Front Desk, Learning Specialist, Counselor (highlighted), and Administrator.
- Get Referral Key: A green button.

Below the form is a table titled **List Active Referral Keys**:

| University ID | User Type | Referral Key | Action |
|---------------|---------------|---|------------|
| 21111111 | Administrator | \$2y\$10STGjJUXA5vVMhtGgI8a/b.P65v5Fqyow9Z.hwnkyztSICOXRTNK | Delete Key |

At the bottom of the page is the **List Active Users** table, which is identical to the one shown in the first image.

Get referral key to create a new user

Generate Referral Key for User

Enter User University ID Number

Select User Type

Referral key for uid 23478122 is
\$2y\$10\$InKd2.nZrnPDfbSBN5B4d.sFwPvwov8jEkeb1MxPP9sveqH5YRem

Referral key

C.A.T.S. Academics

Self Registration Portal

Enter Your University ID Number

Enter Your Referral Key

Select Your User Type

Input referral key, university id, and choose your user type to continue

Create new user account

C.A.T.S. Academics

Self Registration Portal

The screenshot shows a registration form with fields for User ID, Username, Password, First Name, Last Name, University Email Address, Phone Number, Street Address, and Select ST, CT, ZIP. A dropdown menu is open for the location selection, showing a list of locations including Adak, Akiachak, Akiak, Akutan, Alakanuk, Aleknagik, Allakaket, Ambler, Anaktuuuk Pass, Anchor Point, Anchorage, Anderson, Angoon, Aniak, Anvik, Arctic Village, Atka, Atkasuk, Auke Bay, and Barrow. A red box highlights the form, and a red arrow points from a text box to the dropdown menu.

Create new user account → set up user's information: can use drop-down list to select location

Here, to make the dropdown list of location, we added one more table just into our front-end, "ctstzip" table, which contains 41,755 rows.

Continuing to work on "Administrator" page:

The screenshot shows the 'C.A.T.S. Administrator Page' with navigation links: 'C.A.T.S. Home Page', 'Manage Users', 'Manage Equipment', 'Manage Rooms', and a 'Sign Out' button. A large grey banner displays 'Hello, Administrator!'.

Manage Users

Manage Users

Manage Equipment

Manage Equipment

Manage Rooms

Manage Rooms

Click here to manage equipment

List Equipment

Add new equipment

| Equipment ID | Equipment Condition | Equipment Type | Room | | |
|--------------|---------------------------|----------------|------|----------------------------------|----------------------------------|
| 100001 | Normal, purchased in 2014 | Computer1 | L206 | Update Equipment | Delete Equipment |
| 100003 | Slow, tested in 2015 | Computer1 | L206 | Update Equipment | Delete Equipment |

Equipment Condition

Equipment Type

Room

Insert Equipment

[Submit](#)

List Equipment

| Equipment ID | Equipment Condition | Equipment Type | Room | | |
|--------------|---------------------------|----------------|------|----------------------------------|----------------------------------|
| 100001 | Normal, purchased in 2014 | Computer1 | L206 | Update Equipment | Delete Equipment |

Equipment Condition

Equipment Type

Room

Insert Equipment

[Submit](#)

Update the status/condition of Equipment

Hello, Administrator!

Manage Users

Manage Users

Manage Equipment

Manage Equipment

Manage Rooms

Manage Rooms

Click here to manage rooms

List Rooms

| Room Number | Room Size | ROOMLOCKED | | |
|-------------|-----------|------------|-----------------------------|-----------------------------|
| D307 | 8 | N | Update Room | Delete Room |
| L206 | 20 | F | Update Room | Delete Room |

| | |
|--------------|---|
| Room Number | <input type="text" value="Room Number"/> |
| Room Locked? | <input type="text" value="Is Room Locked When Created?"/> |
| Room Size | <input type="text" value="Room Size > 0"/> |
| Insert Room | <input type="submit" value="Submit"/> |

Insert new room here and ensure input 'F' in "Room Locked" to guarantee this new-added room could be booked

List Rooms

| Room Number | Room Size | ROOMLOCKED | | |
|-------------|-----------|------------|-----------------|-----------------|
| D307 | 8 | N | Confirm Changes | Discard Changes |
| L206 | 20 | F | Update Room | Delete Room |

Room Number

Room Locked?

Room Size

Insert Room

Click "Update Room" to fill out the new information of existing room

URL Link: <http://ec2-35-161-250-55.us-west-2.compute.amazonaws.com/>

Chapter 7.

Implementation Plan

To develop this online Booking & Scheduling system, our team made the following detailed implementation plan at the beginning of this semester, and we show the status of each major task of our implementation plan.

| Detailed Implementation Plan of LIGHT^HOUSE team's Project | | | | | | |
|--|---|------------|----------|------------|---|--|
| | Task | Start Date | Duration | End Date | Status | |
| 1 | Team Organization | 8/22/2016 | 5 | 8/26/2016 | on time | |
| | Find out all teammates one EDM group requires | | | | | |
| | Define the name and project direction of our team | | | | | |
| | Determine the potential customer list | | | | | |
| 2 | Client Connection | 8/27/2016 | 4 | 8/30/2016 | on time | |
| 2.1 | Send out the request emails | | | | | |
| 2.2 | Schedule & Hold the first client meeting | | | | | |
| 3 | Requirement Analysis | 9/1/2016 | 9 | 9/9/2016 | on time | |
| 3.1 | Schedule & Hold the further meetings with different people in client team | | | | | |
| 3.2 | Summarize meeting notes | | | | | |
| 3.3 | Make requirement analysis | | | | | |
| 3.4 | Generate a list of client requirements & Construct background story of client | | | | | |
| 3.5 | Check with client and correct misunderstanding points | | | | | |
| 4 | Conceptual Design | 9/10/2016 | 16 | 9/25/2016 | on time | |
| 4.1 | Design the ERD based on client story | | | | | |
| 4.2 | Check with professor | | | | | |
| 4.3 | Correct the errors and send the copy of finalized ERD to client | | | | | |
| 5 | Logical Design | 9/20/2016 | 26 | 10/15/2016 | delayed, but not for the entire project | |
| 5.1 | Create the conceptual data dictionary | | | | | |
| 5.2 | Check with the professor | | | | | |
| 5.3 | Correct mistakes & Make normalization | | | | | |
| 5.4 | Design the integrity constrains | | | | | |
| 5.5 | Finish relational data dictionary | | | | | |
| 6 | SQL Work | 10/16/2016 | 37 | 11/22/2016 | delayed, but not for the entire project | |
| 6.1 | Set up table script in Oracle based on logical design | | | | | |
| 6.2 | Testing & Debugging of DB table script | | | | | |
| 6.3 | Generate query questions & Summarize the needs to triggers/procedures | | | | | |
| 6.4 | Write sql code for our query questions and needed triggers | | | | | |
| 6.5 | Testing & Debugging | | | | | |
| 7 | Front-end Demo | 11/1/2016 | 38 | 12/7/2016 | on time | |
| 7.1 | Determine the implementation platform (server) | | | | | |
| 7.2 | Hold group meeting to discuss about the design of front-end | | | | | |
| 7.3 | Convert all oracle sql coding into MySQL version, only for front-end | | | | | |
| 7.4 | Program the front-end | | | | | |
| 7.5 | Post our coding work on Amazon server | | | | | |
| 7.6 | Present the draft of our solution | | | | | |
| 7.7 | Improve our work based on comments from both classmates and professor | | | | | |
| 7.8 | Testing & Debugging of front-end | | | | | |
| 8 | Report Generation | 12/1/2016 | 9 | 12/9/2016 | on time | |
| 8.1 | Assign writing task(s) to each teammate | | | | | |
| 8.2 | Schedule & Hold the last client meeting to deliver our finalized solution | | | | | |
| 8.3 | Finish all writing work of final report | | | | | |
| 8.4 | Hold a group meeting to check all work and submit on time | | | | | |
| 8.5 | Send the source code, costing plan, and potential problems with recommendations to client | | | | | |

Costing Estimation

Not only the detailed implementation plan, our team also provides the expected costing plan of our project. The table below clearly list the price of each paid item and the estimated hours spent in both our implementation and future potential development.

| Expected Costing Plan of LIGHT^HOUSE team's Project | | |
|--|-----------------------------|-------------------|
| Task / Item | Labor Hours (in hrs) | Price (\$) |
| Team Organization | 5 | |
| Client Connection | 5 | |
| Requirement Analysis | 15 | |
| Conceptual Design | 30 | |
| Logical Design | 40 | |
| SQL Work | 60 | |
| Front-end Demo | 80 | |
| Report Generation | 30 | |
| Further development of advanced system features | 50 | |
| Maintenance | 1 hr/month | |
| | | |
| Amazon Web Server | | 1-year free |
| Contact Sheet | | \$0.1/piece |

Implementation Platform

Amazon Web Service offers one-year free trial, and it supports many kinds of programming languages (HTML, PHP, MySQL, CSS, etc., also which we used to develop our online booking system) and software. Therefore, our team selected Amazon Web Service (AWS) as our implementation platform. After the delivery of our final solution, we will provide the source code for our client. They can finally determine whether or not they will continue using AWS as their implementation platform.

Lesson Learnt

At the end of our project report, we want to talk about the key technological challenges our team experienced during this semester. These key challenges are front-end implementation and figuring out the differences between Oracle and MySQL. Most of our team are not familiar with MySQL before, but we need to convert all SQL work from oracle format into MySQL format because we selected Amazon Web Services (AWS) to finish our interface. However, through this process, our team learnt a lot about MySQL. It has several unique advantages comparing to Oracle, but not all functions of MySQL are friendly to users. For instance, MySQL has commands, "UPDATE CASCADE" and "DELETE RESTRICT", but Oracle doesn't have these two. It means that we need to create and add triggers if we want to realize synchronous update in Oracle SQL. Moreover, "DELETE RESTRICT" could help protect parent tables from automatically mistaken deletions in child tables, but "DELETE CASCADE" couldn't.

Secondly, it is not easy for our group to create a completed and fancy frontend within limited time even though it's not hard to start. To begin with, there are a variety of tasks required for this project, which means that it's impossible for all teammates to focus on frontend implementation. Furthermore, we spent a long time in the improvements of both ERD and Data Dictionary. That is, we need to guarantee our frontend implementation to follow with our conceptual and logical designs. At last, although some members of our team already had different levels of

programming skills, but we are not familiar with PHP and HTML before. Therefore, the learning process of new programming languages costed us much time. Especially, it is hard for us to build a real-time scheduling and booking function of our frontend system. However, our team really obtained many great opportunities to practice our programming skills.

The last not the least, not only the knowledge of programming languages, we also leant a lot from our teamwork. Firstly, each teammate was so busy with his/her own tasks, so we realized how important the time management is. Sometimes, it is difficult for us to meet together. Therefore, we created To-Do list together, which ordered what we need to finish. Moreover, our team was divided into several subgroups based on the preferences and strengths of each member. Then, each subgroup worked on different parts of our project, and it was easier for fewer members to meet together. All these are useful to guarantee all our work to be finished on time, which really helped us to save a lot of time and increase the efficiency of entire group.