PROJECT REPORT

SUBMITTED FOR DATABASE MANAGEMENT SYSTEM (UCS-310)

Submitted By

Name of the student:	**Roll No.**
Keshav Anand	101703284
Kunal Bajaj	101703297
Kunal Saraf	101703300

Batch: COE 14

Submitted To: Mr. Anil Vashisht

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY, PATIALA-147001, PUNJAB

JAN-MAY 2019

CONTENTS

ABSTRACT

The project primarily focuses on solving the problem of arranging rooms for extra classes and for society affairs in the university. Currently if any teacher wants to reschedule a class then he needs to arrange a room manually. Even certain societies require rooms for carrying out there affairs. Our system solves the entire problem by digitalizing this process. Through our system, a teacher can select a particular room on a date along with a time slot and then book it in case it is available.

In this system we have made use of PL-SQL for the entire procedural implementation. The tables are normalized in order to make sure that there are no duplicate entries in the table so that clashes aren’t there thereby providing a hassle free system to book rooms.

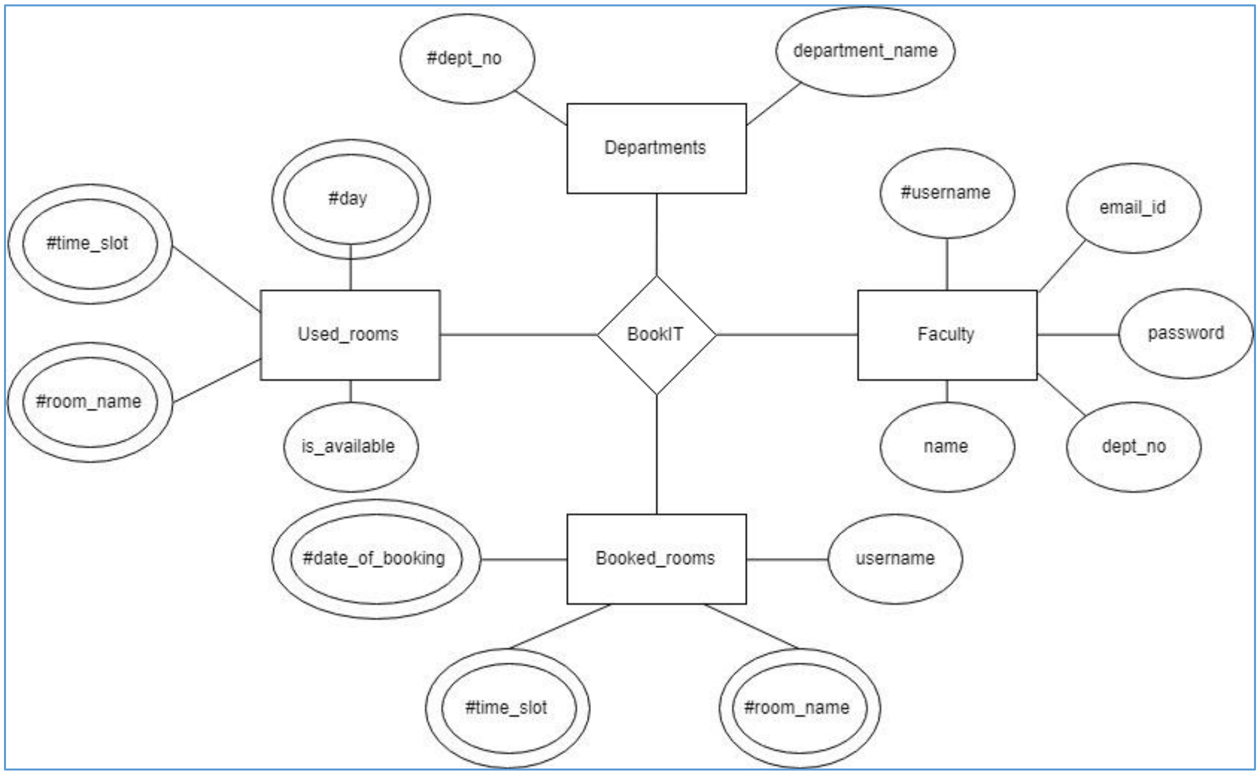
ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our Teacher Mr. Anil Vashisht for his able guidance and support in completing the Project.

This project would not have been completed without his enormous help and worthy experience. Whenever we needed him, he was behind us.

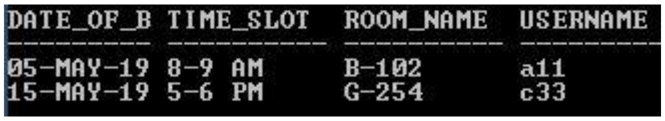
Although this report has been prepared with utmost care and deep routed interest, even then we accept respondent and imperfection.

ENTITY-RELATION DIAGRAM



NORMALIZATION

1. BOOKED_ROOMS



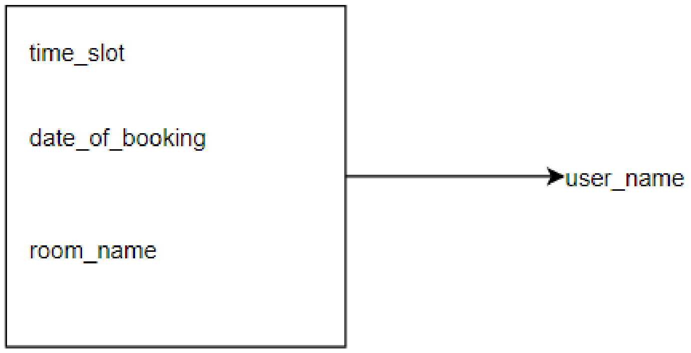
```
DATE_OF_B TIME_SLOT  ROOM_NAME  USERNAME
--------- ---------- ---------- ---------
05-MAY-19 8-9 AM     B-102      a11
15-MAY-19 5-6 PM     G-254      c33
```

1st NF - The above table contains attributes time_slot, date_of_booking, room_name which are multivalued attributes as

- a single user can book a room for different time slots on the same day
- a single user can book a room on different dates keeping the time slot same
- a single user can book different rooms keeping time_slot and date_of_booking same.

So in order to bring it in the 1st NF, we need to declare each of them separately that is why we need to flat the table.

2nd NF - The above table has time_slot, date_of_booking and room_name as composite primary key so we need to check the dependencies of these columns and for that we can draw a FD

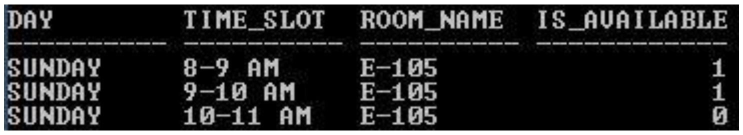


The FD shows that username has a complete dependency on the time_slot, date_of_booking and room_name. So the given table is in 2nd NF.

3rd NF - There aren't any indirect dependencies amongst the columns in the given table since username depends directly on the primary key of the table hence the given table is in 3rd NF.

Since the above table satisfies the conditions of 1NF, 2NF, 3NF hence the given table is normalized.

2. USED_ROOMS



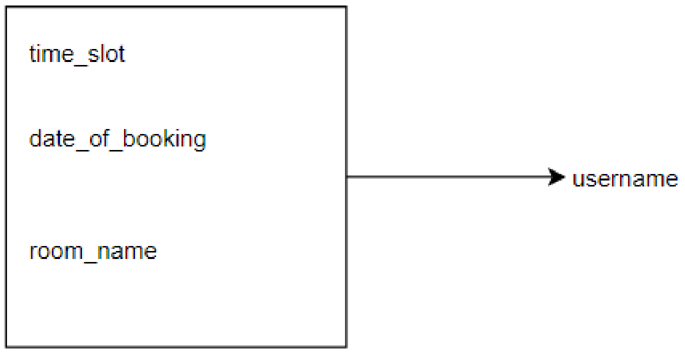
```
DAY        TIME_SLOT  ROOM_NAME  IS_AVAILABLE
---------- ---------- ---------- ------------
SUNDAY     8-9 AM     E-105                 1
SUNDAY     9-10 AM    E-105                 1
SUNDAY     10-11 AM   E-105                 0
```

1st NF - The above table contains attributes time_slot, day, room_name which are multivalued attributes as

- time_slot can have many values for a single day for a single room.
- day can have many values for a single time_slot and for a single room.
- room_name can have many values for single days and for a single time_slot.

So in order to bring it in the 1st NF, we need to declare each of them separately that is why we need to flat the table.

2nd NF - The above table has time_slot, date_of_booking, room_name as composite primary key so we need to check the dependencies of these columns for that we can draw a FD

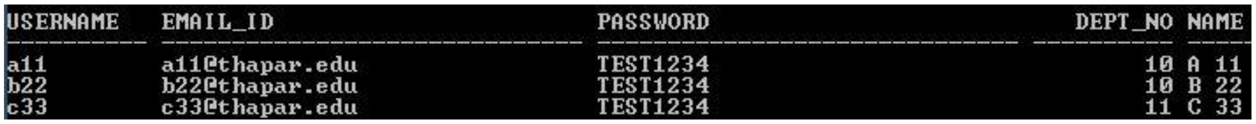


The FD shows that is_available has a complete dependency on the time_slot, date_of_booking, room_name. So the given table is in 2nd NF.

3rd NF - There aren't any indirect dependencies amongst the columns in the given table since is_available depends directly on the primary key of the table. Hence the given table is in 3rd NF.

Since the above table satisfies the conditions of 1NF, 2NF, 3NF hence the given table is normalized.

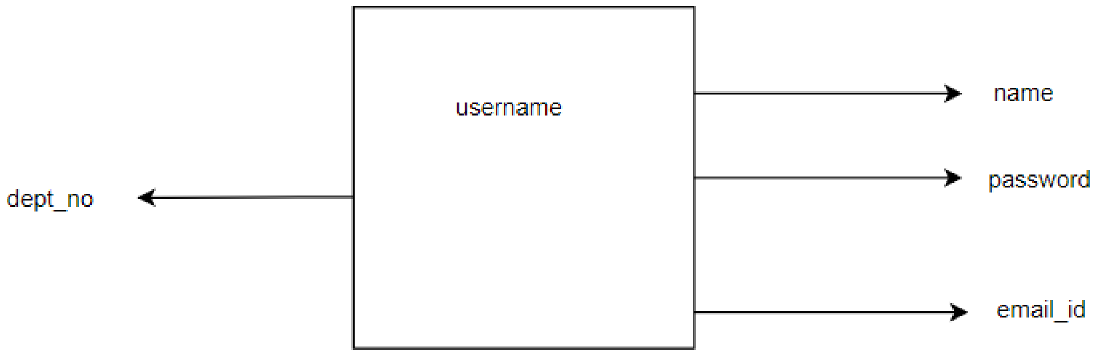
3. FACULTY



```
USERNAME   EMAIL_ID                        PASSWORD                           DEPT_NO NAME
---------- ------------------------------- ------------------------------ ---------- -----
a11        a11@thapar.edu                  TEST1234                               10 A 11
b22        b22@thapar.edu                  TEST1234                               10 B 22
c33        c33@thapar.edu                  TEST1234                               11 C 33
```

1st NF - The above table contains no multivalued attributes hence at the intersection of every column and row we will get one and only one answer. So the table is in **1NF.**

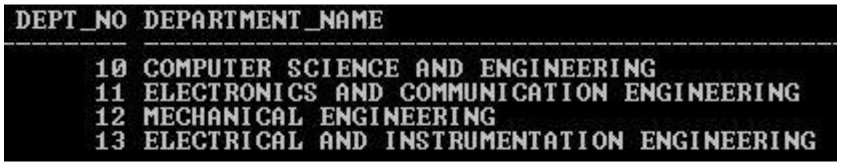
2nd NF - The primary key in this table is username so there is no composite primary key. So the given table is in **2NF.** moreover the below drawn FD depicts that all the columns are directly dependent on the primary key So in order to keep our table in 2nd NF



3rd NF - The above table has no indirect dependencies amongst its columns so the table is in 3NF also this thing could be determined through the FD. If we had included department_name in our faculty table, then it would have led to transitive dependency which would have resulted in the violation of the 3rd NF. But we removed that column and put it in another table **departments** and in order to avoid loss of any sort of information we linked it via foreign key on dept_no which is common in both the tables. Hence the given table is in 3NF.

Since the above table satisfies the conditions of 1NF, 2NF, 3NF hence the given table is normalized.

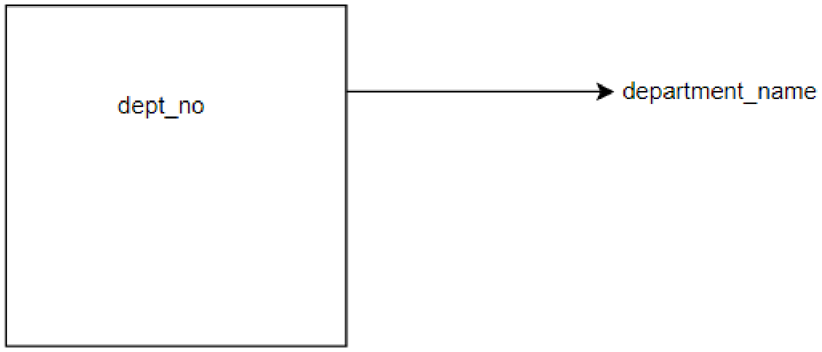
4. DEPARTMENTS



```
DEPT_NO DEPARTMENT_NAME
-------- ------------------------------------------------
      10 COMPUTER SCIENCE AND ENGINEERING
      11 ELECTRONICS AND COMMUNICATION ENGINEERING
      12 MECHANICAL ENGINEERING
      13 ELECTRICAL AND INSTRUMENTATION ENGINEERING
```

1st NF - The above table is in 1NF as all the columns are single valued attributes.

2nd NF - The above table is in 2NF because there is no composite primary key in the table.



3rd NF - The above table is in 3NF because there are no transitive dependencies in the table as shown by the FD above.

Since the above table satisfies the conditions of 1NF, 2NF, 3NF hence the given table is normalized.

DATABASE OBJECTS

Tables –

1. USED_ROOMS
2. DEPARTMENTS
3. FACULTY
4. BOOKED_ROOMS

Procedures –

1. ADD_FACULTY
2. ADD_DEPARTMENT
3. LOGIN
4. SHOW_BOOKINGS
5. SHOW_AVAILABLE_ROOMS
6. BOOK_A_ROOM

Triggers –

1. USED_ROOMS_CHECK
2. FACULTY_CHECK
3. BOOKED_ROOMS_CHECK
4. DEPARTMENTS_INSERT
5. BOOKED_ROOMS_INSERT
6. FACULTY_INSERT

Indexes –

1. BOOKINGS_PER_USERNAME

QUERIES

Create table queries –

```
CREATE TABLE USED_ROOMS(
        DAY VARCHAR(10),
        TIME_SLOT VARCHAR(10),
        ROOM_NAME VARCHAR(10),
        IS_AVAILABLE NUMBER(1),
        CONSTRAINT USED_ROOMS_PK PRIMARY KEY (DAY, TIME_SLOT, ROOM_NAME)
);

CREATE TABLE DEPARTMENTS(
        DEPT_NO NUMBER(5),
        DEPARTMENT_NAME VARCHAR(40),
        CONSTRAINT DEPARTMENTS_PK PRIMARY KEY(DEPT_NO)
);

CREATE TABLE FACULTY(
        USERNAME VARCHAR(10),
        EMAIL_ID VARCHAR(30),
        PASSWORD VARCHAR(30),
        DEPT_NO NUMBER(5),
        NAME VARCHAR(30),
        CONSTRAINT EMPLOYEES_PK PRIMARY KEY(USERNAME),
        CONSTRAINT EMPLOYEES_FK FOREIGN KEY(DEPT_NO) REFERENCES
DEPARTMENTS(DEPT_NO)
);

CREATE TABLE BOOKED_ROOMS(
        DATE_OF_BOOKING DATE,
        TIME_SLOT VARCHAR(10),
        ROOM_NAME VARCHAR(10),
        USERNAME VARCHAR(10),
        CONSTRAINT BOOKED_ROOMS_PK PRIMARY
KEY(DATE_OF_BOOKING,TIME_SLOT,ROOM_NAME),
        CONSTRAINT BOOKED_ROOMS_FK FOREIGN KEY(USERNAME) REFERENCES
FACULTY(USERNAME)
);
```

Create Index queries –

```
CREATE INDEX BOOKINGS_PER_USERNAME
ON BOOKED_ROOMS(USERNAME);
```

Create trigger queries –

```
CREATE OR REPLACE TRIGGER USED_ROOMS_CHECK
BEFORE INSERT OR UPDATE
ON USED_ROOMS
FOR EACH ROW
BEGIN
        IF UPPER(:NEW.DAY) NOT IN('MONDAY', 'TUESDAY', 'WEDNESDAY',
        'THURSDAY', 'FRIDAY', 'SATURDAY', 'SUNDAY') THEN
                RAISE_APPLICATION_ERROR(-20001,'INVALID DAY');
        ELSIF :NEW.TIME_SLOT NOT IN('8-9 AM', '9-10 AM', '10-11 AM',
        '11-12 PM', '12-1 PM', '1-2 PM', '2-3 PM', '3-4 PM', '4-5 PM',
        '5-6 PM', '6-7 PM') THEN
                RAISE_APPLICATION_ERROR(-20002,'INVALID TIME SLOT');
        ELSIF :NEW.IS_AVAILABLE NOT IN (0, 1) THEN
                RAISE_APPLICATION_ERROR(-20003,'INVALID AVAILIBILITY');
        END IF;
END;
/

CREATE OR REPLACE TRIGGER FACULTY_CHECK
BEFORE INSERT OR UPDATE
ON FACULTY
FOR EACH ROW
BEGIN
        IF LENGTH(:NEW.PASSWORD) < 8 THEN
                RAISE_APPLICATION_ERROR(-20004,'PASSWORD TOO SHORT');
        END IF;
END;
/

CREATE OR REPLACE TRIGGER BOOKED_ROOMS_CHECK
BEFORE INSERT OR UPDATE
ON BOOKED_ROOMS
FOR EACH ROW
BEGIN
        IF :NEW.DATE_OF_BOOKING < SYSDATE THEN
                RAISE_APPLICATION_ERROR(-20005,'CANNOT BOOK ROOMS IN PAST');
        ELSIF :NEW.TIME_SLOT NOT IN('8-9 AM', '9-10 AM', '10-11 AM',
        '11-12 PM', '12-1 PM', '1-2 PM', '2-3 PM', '3-4 PM', '4-5 PM',
        '5-6 PM', '6-7 PM') THEN
                RAISE_APPLICATION_ERROR(-20002,'INVALID TIME SLOT');
        END IF;
END;
/
```

```
CREATE OR REPLACE TRIGGER DEPARTMENTS_INSERT
AFTER INSERT
ON DEPARTMENTS
FOR EACH ROW
BEGIN
        DBMS_OUTPUT.PUT_LINE('DEPARTMENT ADDED SUCCESSFULLY');
END;
/

CREATE OR REPLACE TRIGGER BOOKED_ROOMS_INSERT
AFTER INSERT
ON BOOKED_ROOMS
FOR EACH ROW
BEGIN
        DBMS_OUTPUT.PUT_LINE('ROOM BOOKED SUCCESSFULLY');
END;
/

CREATE OR REPLACE TRIGGER FACULTY_INSERT
AFTER INSERT
ON FACULTY
FOR EACH ROW
BEGIN
        DBMS_OUTPUT.PUT_LINE('FACULTY ADDED SUCCESSFULLY');
END;
/
```

Create Procedure queries –

```
CREATE OR REPLACE PROCEDURE ADD_FACULTY(
        USERNAME_ IN VARCHAR,
        EMAIL_ID_ IN VARCHAR,
        PASSWORD_ IN VARCHAR,
        DEPT_NO_ IN NUMBER,
        NAME_ IN VARCHAR)
IS
BEGIN
        INSERT INTO FACULTY VALUES(USERNAME_, EMAIL_ID_, PASSWORD_,
        DEPT_NO_, NAME_);
END;
/

CREATE OR REPLACE PROCEDURE ADD_DEPARTMENT(
        DEPT_NO_ NUMBER,
        DEPARTMENT_NAME_ VARCHAR)
IS
BEGIN
        INSERT INTO DEPARTMENTS VALUES(DEPT_NO_, DEPARTMENT_NAME_);
END;
/

CREATE OR REPLACE PROCEDURE LOGIN(
        USERNAME_ IN VARCHAR,
        PASSWORD_ IN VARCHAR)
IS
        CURSOR AUTHENTICATION IS SELECT * FROM FACULTY;
        RECORD FACULTY%ROWTYPE;
        FLAG NUMBER := 0;
BEGIN
        OPEN AUTHENTICATION;
        LOOP
                EXIT WHEN AUTHENTICATION%NOTFOUND;
                FETCH AUTHENTICATION INTO RECORD;
                IF UPPER(RECORD.USERNAME) = UPPER(USERNAME_) AND
                RECORD.PASSWORD = PASSWORD_ THEN
                        DBMS_OUTPUT.PUT_LINE('LOGIN SUCCESSFULL');
                        FLAG := 1;
                END IF;
        END LOOP;
        CLOSE AUTHENTICATION;
        IF FLAG = 0 THEN
                DBMS_OUTPUT.PUT_LINE('INVALID USERNAME OR PASSWORD');
        END IF;
END;
/

CREATE OR REPLACE PROCEDURE SHOW_BOOKINGS(
        USER VARCHAR)
IS
        ONE_ROW BOOKED_ROOMS%ROWTYPE;
        CURSOR ALL_ROWS IS SELECT * FROM BOOKED_ROOMS WHERE
        UPPER(USERNAME) = UPPER(USER);
BEGIN
        OPEN ALL_ROWS;
        LOOP
                FETCH ALL_ROWS INTO ONE_ROW;
                EXIT WHEN ALL_ROWS%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE(ONE_ROW.DATE_OF_BOOKING || ' ' ||
                ONE_ROW.TIME_SLOT || ' ' || ONE_ROW.ROOM_NAME);
        END LOOP;
        IF ALL_ROWS%ROWCOUNT = 0 THEN
                DBMS_OUTPUT.PUT_LINE('SORRY! YOU HAVE NO BOOKINGS');
        END IF;
        CLOSE ALL_ROWS;
END;
/

CREATE OR REPLACE PROCEDURE SHOW_AVAILABLE_ROOMS(
        DATE_ DATE,
        TIME_SLOT_ VARCHAR)
IS
        CURSOR AVAILABLE_ROOMS IS SELECT ROOM_NAME FROM USED_ROOMS
        WHERE UPPER(DAY) = TO_CHAR(DATE_,'FMDAY') AND TIME_SLOT =
        TIME_SLOT_ AND IS_AVAILABLE = 1 AND   ROOM_NAME NOT IN
        (SELECT ROOM_NAME FROM BOOKED_ROOMS WHERE
        TO_CHAR(DATE_OF_BOOKING) = TO_CHAR(DATE_) AND TIME_SLOT =
        TIME_SLOT_);
        ONE_ROOM BOOKED_ROOMS.ROOM_NAME%TYPE;
BEGIN
        OPEN AVAILABLE_ROOMS;
        LOOP
                FETCH AVAILABLE_ROOMS INTO ONE_ROOM;
                EXIT WHEN AVAILABLE_ROOMS%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE(ONE_ROOM);
        END LOOP;
        IF(AVAILABlE_ROOMS%ROWCOUNT = 0) THEN
                DBMS_OUTPUT.PUT_LINE('SORRY! NO ROOMS AVAILABLE FOR
                SELECTED SLOT');
        END IF;
        CLOSE AVAILABLE_ROOMS;
END;
/


CREATE OR REPLACE PROCEDURE BOOK_A_ROOM(DATE_ DATE,
        TIME_SLOT_ VARCHAR,
        ROOM VARCHAR,
        USER VARCHAR)
IS
        CURSOR AVAILABLE_ROOMS IS SELECT ROOM_NAME FROM USED_ROOMS
        WHERE UPPER(DAY) = TO_CHAR(DATE_,'FMDAY') AND TIME_SLOT =
        TIME_SLOT_ AND IS_AVAILABLE = 1 AND ROOM_NAME NOT IN
        (SELECT ROOM_NAME FROM BOOKED_ROOMS WHERE
        TO_CHAR(DATE_OF_BOOKING) = TO_CHAR(DATE_) AND TIME_SLOT =
        TIME_SLOT_);
        ONE_ROOM BOOKED_ROOMS.ROOM_NAME%TYPE;
        NUMBER_OF_ROOMS NUMBER;
        FLAG NUMBER := 0;
BEGIN
        SELECT COUNT(ROOM_NAME) INTO NUMBER_OF_ROOMS FROM USED_ROOMS
        WHERE UPPER(DAY) = TO_CHAR(DATE_,'FMDAY') AND TIME_SLOT =
        TIME_SLOT_ AND IS_AVAILABLE = 1 AND ROOM_NAME NOT IN
        (SELECT ROOM_NAME FROM BOOKED_ROOMS WHERE
        TO_CHAR(DATE_OF_BOOKING) = TO_CHAR(DATE_) AND TIME_SLOT =
        TIME_SLOT_);
        IF NUMBER_OF_ROOMS = 0 THEN
                DBMS_OUTPUT.PUT_LINE('SORRY! NO ROOMS AVAILABLE FOR SELECTED
                SLOT');
        END IF;
        OPEN AVAILABLE_ROOMS;
        LOOP
                FETCH AVAILABLE_ROOMS INTO ONE_ROOM;
                EXIT WHEN AVAILABLE_ROOMS%NOTFOUND;
                IF ONE_ROOM = ROOM THEN
                        INSERT INTO BOOKED_ROOMS VALUES(DATE_, TIME_SLOT_,
ROOM,
                        LOWER(USER));
                        FLAG := 1;
                END IF;
        END LOOP;
        CLOSE AVAILABLE_ROOMS;
        IF FLAG = 0 THEN
                DBMS_OUTPUT.PUT_LINE('INVALID ROOM ENTERED');
        END IF;
END;
/
```

Some Insert queries –

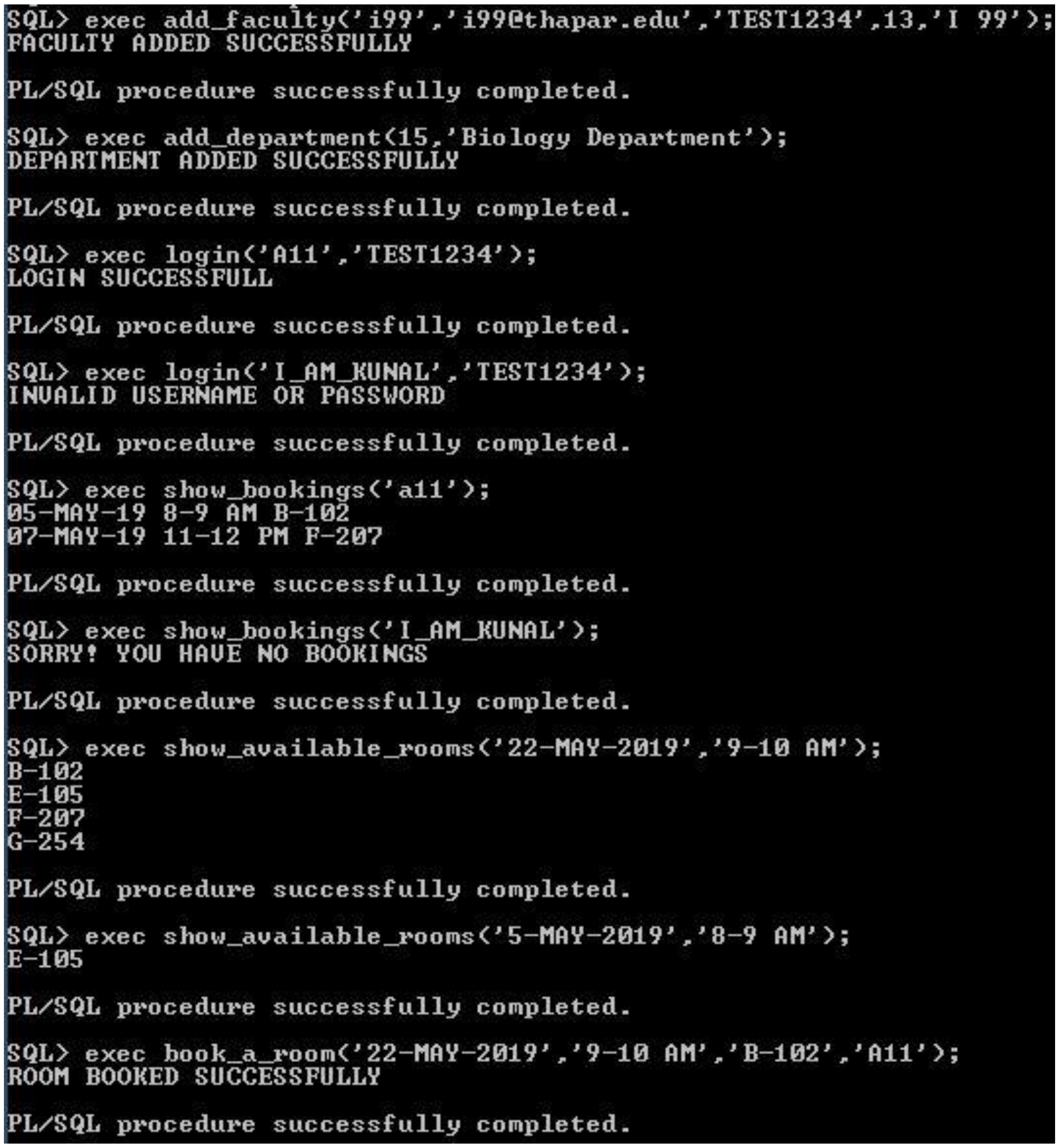
```
INSERT INTO DEPARTMENTS VALUES(10,'COMPUTER SCIENCE AND ENGINEERING');
INSERT INTO DEPARTMENTS VALUES(11,'ELECTRONICS AND COMMUNICATION
ENGINEERING');
INSERT INTO DEPARTMENTS VALUES(12,'MECHANICAL ENGINEERING');
INSERT INTO DEPARTMENTS VALUES(13,'ELECTRICAL AND INSTRUMENTATION
ENGINEERING');

INSERT INTO FACULTY VALUES('a11','a11@thapar.edu','TEST1234',10,'A 11');
INSERT INTO FACULTY VALUES('b22','b22@thapar.edu','TEST1234',10,'B 22');
INSERT INTO FACULTY VALUES('c33','c33@thapar.edu','TEST1234',11,'C 33');
INSERT INTO FACULTY VALUES('d44','d44@thapar.edu','TEST1234',11,'D 44');
INSERT INTO FACULTY VALUES('e55','e55@thapar.edu','TEST1234',12,'E 55');
INSERT INTO FACULTY VALUES('f66','f66@thapar.edu','TEST1234',12,'F 66');
INSERT INTO FACULTY VALUES('g77','g77@thapar.edu','TEST1234',13,'G 77');
INSERT INTO FACULTY VALUES('h88','h88@thapar.edu','TEST1234',13,'H 88');

INSERT INTO USED_ROOMS VALUES('MONDAY', '8-9 AM', 'B-102', 1);
INSERT INTO USED_ROOMS VALUES('MONDAY', '9-10 AM', 'B-102', 1);
INSERT INTO USED_ROOMS VALUES('MONDAY', '10-11 AM', 'B-102', 1);
INSERT INTO USED_ROOMS VALUES('MONDAY', '11-12 PM', 'B-102', 1);
INSERT INTO USED_ROOMS VALUES('MONDAY', '12-1 PM', 'B-102', 1);
INSERT INTO USED_ROOMS VALUES('MONDAY', '1-2 PM', 'B-102', 0);
INSERT INTO USED_ROOMS VALUES('MONDAY', '2-3 PM', 'B-102', 0);
INSERT INTO USED_ROOMS VALUES('MONDAY', '3-4 PM', 'B-102', 1);
INSERT INTO USED_ROOMS VALUES('MONDAY', '4-5 PM', 'B-102', 0);
INSERT INTO USED_ROOMS VALUES('MONDAY', '5-6 PM', 'B-102', 0);
INSERT INTO USED_ROOMS VALUES('MONDAY', '6-7 PM', 'B-102', 1);
INSERT INTO USED_ROOMS VALUES('MONDAY', '8-9 AM', 'E-105', 1);
INSERT INTO USED_ROOMS VALUES('MONDAY', '9-10 AM', 'E-105', 1);
INSERT INTO USED_ROOMS VALUES('MONDAY', '10-11 AM', 'E-105', 0);
INSERT INTO USED_ROOMS VALUES('MONDAY', '11-12 PM', 'E-105', 1);
INSERT INTO USED_ROOMS VALUES('MONDAY', '12-1 PM', 'E-105', 0);
INSERT INTO USED_ROOMS VALUES('MONDAY', '1-2 PM', 'E-105', 0);
INSERT INTO USED_ROOMS VALUES('MONDAY', '2-3 PM', 'E-105', 0);
INSERT INTO USED_ROOMS VALUES('MONDAY', '3-4 PM', 'E-105', 0);
INSERT INTO USED_ROOMS VALUES('MONDAY', '4-5 PM', 'E-105', 0);
INSERT INTO USED_ROOMS VALUES('MONDAY', '5-6 PM', 'E-105', 1);
INSERT INTO USED_ROOMS VALUES('MONDAY', '6-7 PM', 'E-105', 0);
INSERT INTO USED_ROOMS VALUES('MONDAY', '8-9 AM', 'F-207', 0);
INSERT INTO USED_ROOMS VALUES('MONDAY', '9-10 AM', 'F-207', 0);
INSERT INTO USED_ROOMS VALUES('MONDAY', '10-11 AM', 'F-207', 0);
INSERT INTO USED_ROOMS VALUES('MONDAY', '11-12 PM', 'F-207', 0);
INSERT INTO USED_ROOMS VALUES('MONDAY', '12-1 PM', 'F-207', 0);
```

SCREEN SHOTS

All 6 Procedures implemented in the following image -



```
SQL> exec add_faculty('i99','i99@thapar.edu','TEST1234',13,'I 99');
FACULTY ADDED SUCCESSFULLY

PL/SQL procedure successfully completed.

SQL> exec add_department(15,'Biology Department');
DEPARTMENT ADDED SUCCESSFULLY

PL/SQL procedure successfully completed.

SQL> exec login('A11','TEST1234');
LOGIN SUCCESSFULL

PL/SQL procedure successfully completed.

SQL> exec login('I_AM_KUNAL','TEST1234');
INVALID USERNAME OR PASSWORD

PL/SQL procedure successfully completed.

SQL> exec show_bookings('a11');
05-MAY-19 8-9 AM B-102
07-MAY-19 11-12 PM F-207

PL/SQL procedure successfully completed.

SQL> exec show_bookings('I_AM_KUNAL');
SORRY! YOU HAVE NO BOOKINGS

PL/SQL procedure successfully completed.

SQL> exec show_available_rooms('22-MAY-2019','9-10 AM');
B-102
E-105
F-207
G-254

PL/SQL procedure successfully completed.

SQL> exec show_available_rooms('5-MAY-2019','8-9 AM');
E-105

PL/SQL procedure successfully completed.

SQL> exec book_a_room('22-MAY-2019','9-10 AM','B-102','A11');
ROOM BOOKED SUCCESSFULLY

PL/SQL procedure successfully completed.
```