AMERICAN INTERNATIONAL UNIVERSITY – BANGLADESH



PROJECT: HOSPITAL MANAGEMENT SYSTEM

Course Name: Advance Database Management System

Section: B2

Submitted to: Rezwan Ahmed

Group Name: #Semicolon

Group members:

<u>Name</u> <u>ID</u>

Zaman, Isra 15-28938-1

Showmik, MD. Jannatul Baki 15-28584-1

TABLE OF CONTENTS

1.	Project Objectives	 3
2.	Implementation	 4
3.	Future Work	 13
4.	Conclusion	 14
5	Datahasa Scrints	 15

Project Objectives

The project is aimed to develop a hospital management system. It is designed to achieve the following objectives:

- To computerize all details regarding patient details and hospital details.
- Scheduling the appointment of patient with doctors to make it convenient for both.
- Scheduling the services of specialized doctors and emergency properly so that the facilities provided by the hospital are fully utilized in effective and efficient manner.
- The information of the patients should be kept up to date and there should be kept in the system for hospital purposes.

From the developer's side, the aim is to learn to implement PL/SQL, procedure, function, view, sequence, trigger and package.

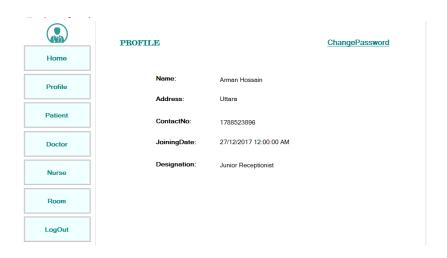
Implementation



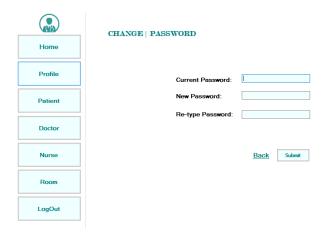
This is the login page.



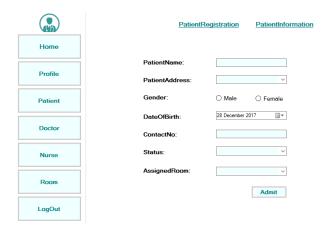
This is the home page.



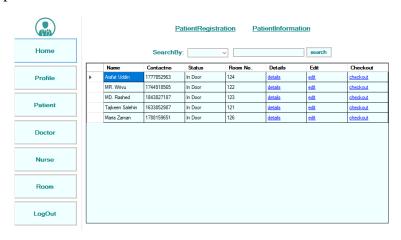
This is the user's profile page.



This page is for changing the account's password.



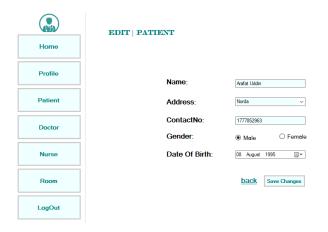
Admitting new patient.



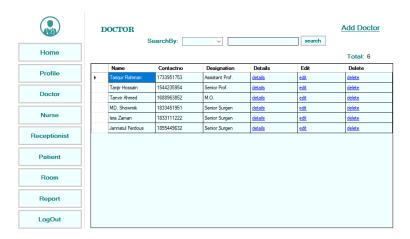
Patient list. From here, patient's details can be seen, patient can be edited and can be checked out when released by doctor.



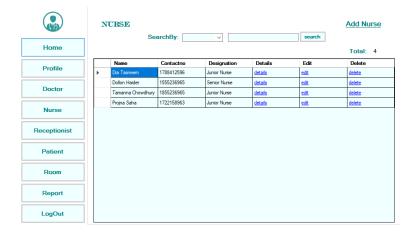
This page is for patient's details.



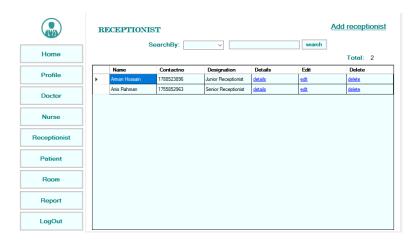
Edit patient's information from this page.



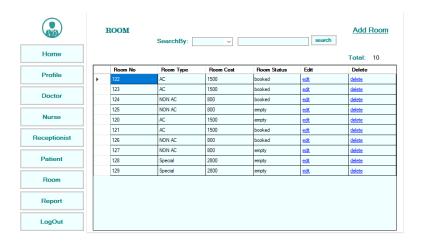
See doctors list from this page.



See nurses list from this page.



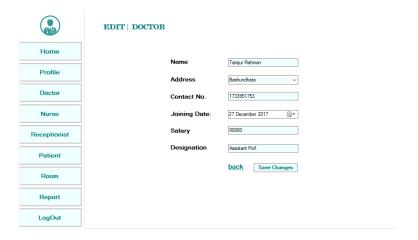
Receptionist list.



Rooms list.



This page is to show doctor's detailed information.



Edit doctor's information.



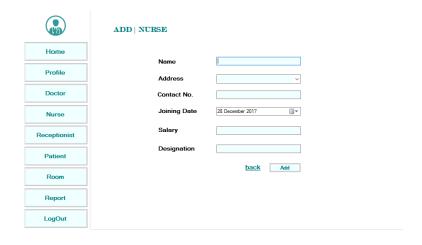
Add new doctor.



This page is to show nurse's detailed information.



Edit nurse's information.



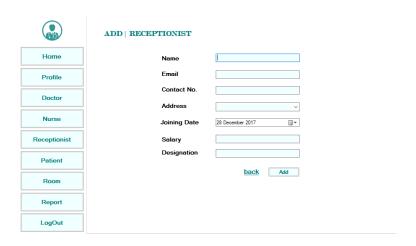
Add new nurse.



This page is to show receptionist detailed information.



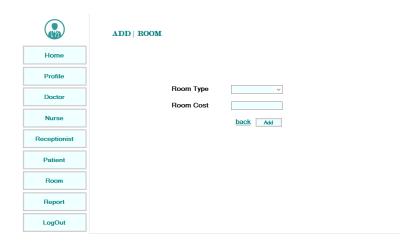
Edit receptionist.



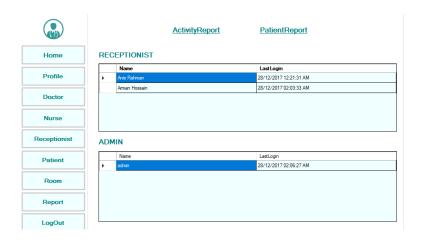
Add new receptionist.



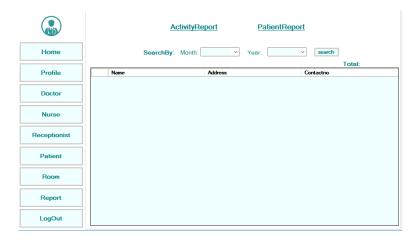
Edit room's information.



Add new room.



This page is to show activity log.



This page is to show patient's monthly report.

Future Work

Hospital management system is a vast system. In the system there are lot of entities, users, functionality. Here, we worked using some of the entities, users and functionality. The future work will be including more users and functionality.

Conclusion

Here, we tried to build a hospital management system that would be helpful to maintain a hospital and keep varies information. We used oracle database. In the back end, we used PL/SQL, procedure, function, trigger, sequence, view and page.

Database Scripts

Procedures

```
create or replace PROCEDURE PATIENT_ADMIT(pname in PATIENT.P_NAME%type,paddress in
PATIENT.P ADDRESS%type,pgender in PATIENT.P GENDER%type,pdob in
PATIENT.P_DOB%type,pcontact in PATIENT.P_CONTACT_NO%type,pstatus in
PATIENT.P_STATUS%type,loginname in VARCHAR2,roomno in ROOM.ROOM_NO%type :=
null) AS
maxid PATIENT.P_ID%type;
status PATIENT.P_STATUS%type;
rec_id NUMBER(5);
BEGIN
 INSERT INTO PATIENT
values(PATIENT_SEQ.NEXTVAL,pname,paddress,pgender,pdob,pcontact,pstatus,'Not Released');
 SELECT P STATUS INTO status FROM PATIENT WHERE P ID=(SELECT MAX(P ID)
FROM PATIENT);
 SELECT R_ID INTO rec_id FROM RECEPTIONIST WHERE R_EMAIL=loginname;
 SELECT P_ID INTO maxid FROM PATIENT WHERE P_ID=(SELECT MAX(P_ID) FROM
PATIENT);
 INSERT INTO ASSIST_PATIENT VALUES(ASSIST_PATIENT_SEQ.NEXTVAL,maxid,rec_id);
 IF(status='In Door') then
 INSERT INTO ASSIGNED ROOM
VALUES(ASSIGNED_ROOM_SEQ.NEXTVAL,maxid,roomno);
 UPDATE ROOM SET ROOM_STATUS='booked' WHERE ROOM_NO=roomno;
 END IF;
END PATIENT_ADMIT;
create or replace PROCEDURE PATIENT_CHECKOUT
(
 PATIENTID IN NUMBER
) AS
roomid ROOM.ROOM_NO%TYPE;
BEGIN
```

UPDATE PATIENT SET P_CONDITION='Released' WHERE P_ID=PATIENTID;

SELECT R.ROOM_NO INTO roomid FROM ROOM R,ASSIGNED_ROOM A,PATIENT P WHERE R.ROOM_NO=A.ROOM_NO AND A.P_ID=P.P_ID AND A.P_ID=PATIENTID;

UPDATE ROOM SET ROOM_STATUS='empty' WHERE ROOM_NO=roomid;

UPDATE LOG_PATIENT SET RELEASE_DATE=SYSDATE WHERE PATIENT ID=PATIENTID;

END PATIENT_CHECKOUT;

create or replace PROCEDURE ADMIN_CHANGE_PASSWORD(UPASSWORD in USER_TYPE.U_PASSWORD%TYPE, UEMAIL in USER_TYPE.U_EMAIL%TYPE) AS BEGIN

UPDATE USER_TYPE SET U_PASSWORD=UPASSWORD WHERE U_EMAIL=UEMAIL; END ADMIN_CHANGE_PASSWORD;

CREATE OR REPLACE PROCEDURE ADD_DOCTOR(dname in DOCTOR.D_NAME%type, daddress in DOCTOR.D_ADDRESS%type, dcontact in DOCTOR.D_CONTACT_NO%type, djoindate in DOCTOR.D_JOINING_DATE%type, dsalary in DOCTOR.D_SALARY%type, ddesignation in DOCTOR.D_DESIGNATION%type) AS

BEGIN

INSERT INTO DOCTOR values(DOCTOR_SEQ.NEXTVAL,dname,daddress,dcontact,djoindate,dsalary,ddesignation); END ADD_DOCTOR;

CREATE OR REPLACE PROCEDURE ADD_NURSE(nname in NURSE.N_NAME%type, naddress in NURSE.N_ADDRESS%type, ncontact in NURSE.N_CONTACT_NO%type, njoindate in NURSE.N_JOINING_DATE%type, nsalary in NURSE.N_SALARY%type, ndesignation in NURSE.N_DESIGNATION%type) AS

BEGIN

INSERT INTO NURSE values(NURSE_SEQ.NEXTVAL,nname,naddress,ncontact,njoindate,nsalary,ndesignation); END ADD_NURSE;

create or replace PROCEDURE ADD_RECEPTIONIST(rname in RECEPTIONIST.R_NAME%type, remail in RECEPTIONIST.R_EMAIL%type, raddress in RECEPTIONIST.R_ADDRESS%type, rcontact in RECEPTIONIST.R_CONTACT_NO%type, rjoindate in

RECEPTIONIST.R_JOINING_DATE%type, rsalary in RECEPTIONIST.R_SALARY%type, rdesignation in RECEPTIONIST.R_DESIGNATION%type) AS

maxid USER_TYPE.ID%TYPE;

email USER_TYPE.U_EMAIL%TYPE;

BEGIN

INSERT INTO RECEPTIONIST

values(RECEPTIONIST_SEQ.NEXTVAL,rname,remail,raddress,rcontact,rjoindate,rsalary,rdesignation);

SELECT R_ID,R_EMAIL INTO maxid,email FROM RECEPTIONIST WHERE R_ID=(SELECT MAX(R_ID) FROM RECEPTIONIST);

INSERT INTO USER_TYPE

values(USER_TYPE_SEQ.NEXTVAL, 'RECEPTIONIST', email, 123456, maxid);

END ADD_RECEPTIONIST;

CREATE OR REPLACE PROCEDURE ADD_ROOM(roomtype IN ROOM.ROOM_TYPE%TYPE, roomcost IN ROOM.ROOM_COST%TYPE) AS

BEGIN

INSERT INTO ROOM values(ROOM_SEQ.NEXTVAL, roomtype, roomcost, 'empty');

END ADD_ROOM;

create or replace PROCEDURE ADMIN_PROFILE_UPDATE(UEMAIL IN ADMIN.ADMIN_EMAIL%TYPE, ANAME IN ADMIN.ADMIN_NAME%TYPE, AADDRESS IN ADMIN.ADMIN_ADDRESS%TYPE, ACONTACT IN ADMIN.ADMIN_CONTACT_NO%TYPE, ADOB IN ADMIN.DATE_OF_BIRTH%TYPE) AS

BEGIN

UPDATE ADMIN SET ADMIN_NAME=ANAME, ADMIN_ADDRESS=AADDRESS, ADMIN_CONTACT_NO=ACONTACT, DATE_OF_BIRTH=ADOB WHERE ADMIN_EMAIL=UEMAIL;

END ADMIN_PROFILE_UPDATE;

create or replace PROCEDURE LOGIN_INFO_PROCEDURE(email in USER_TYPE.U_EMAIL%TYPE) AS

id USER TYPE.ID%TYPE;

old USER_TYPE.U_EMAIL%TYPE;

```
CURSOR REC IS
SELECT LOGGED_PERSON FROM LOGIN_INFO;
FLAG NUMBER(2):=0;
BEGIN
SELECT ID INTO id FROM USER_TYPE WHERE U_EMAIL=email;
FOR I IN REC LOOP
 IF I.LOGGED_PERSON=id THEN
  FLAG:=1;
 END IF;
END LOOP;
IF FLAG=1 THEN
      UPDATE LOGIN_INFO SET LOGIN_DATE=SYSDATE WHERE
LOGGED_PERSON=id;
ELSIF FLAG=0 THEN
 INSERT INTO LOGIN_INFO VALUES(LOGIN_INFO_SEQ.NEXTVAL,id,sysdate);
END IF;
END LOGIN_INFO_PROCEDURE;
```

Functions

```
create or replace FUNCTION FIND_DOCTOR_ID(doctorid IN DOCTOR.D_ID%TYPE)

RETURN BOOLEAN AS

FLAG NUMBER(5):=0;

CURSOR DOC_REC IS

SELECT D_ID FROM ATTEND_PATIENT;

BEGIN

FOR I IN DOC_REC LOOP

IF I.D_ID=doctorid THEN

FLAG:=1;

END IF;

END LOOP;
```

```
IF FLAG=1 THEN
 RETURN TRUE;
 ELSE
 RETURN FALSE;
END IF;
END FIND_DOCTOR_ID;
create or replace FUNCTION FIND_NURSE_ID(nurseid IN NURSE.N_ID% TYPE)
RETURN BOOLEAN AS
FLAG NUMBER(5):=0;
CURSOR NURSE_REC IS
SELECT N_ID FROM GOVERNS_ROOM;
BEGIN
FOR I IN NURSE_REC LOOP
 IF I.N_ID=nurseid THEN
   FLAG:=1;
 END IF;
 END LOOP;
 IF FLAG=1 THEN
 RETURN TRUE;
 ELSE
 RETURN FALSE;
 END IF;
END FIND_NURSE_ID;
create or replace FUNCTION FIND_PATIENT_ASSIGNED_ROOM(patientid IN
ASSIGNED_ROOM.P_ID%TYPE)
RETURN BOOLEAN AS
FLAG NUMBER(5):=0;
CURSOR P_REC IS
SELECT P_ID FROM ASSIGNED_ROOM;
```

```
BEGIN
FOR I IN P_REC LOOP
 IF I.P_ID=patientid THEN
   FLAG:=1;
 END IF;
 END LOOP;
 IF FLAG=1 THEN
 RETURN TRUE;
 ELSE
 RETURN FALSE;
END IF;
END FIND_PATIENT_ASSIGNED_ROOM;
create or replace FUNCTION FIND_PATIENT_ASSIST_PATIENT(patientid IN
ASSIST_PATIENT.P_ID%TYPE)
RETURN BOOLEAN AS
FLAG NUMBER(5):=0;
CURSOR P_REC IS
SELECT P_ID FROM ASSIST_PATIENT;
BEGIN
FOR I IN P_REC LOOP
 IF I.P_ID=patientid THEN
   FLAG:=1;
 END IF;
 END LOOP;
 IF FLAG=1 THEN
 RETURN TRUE;
 ELSE
 RETURN FALSE;
END IF;
END FIND_PATIENT_ASSIST_PATIENT;
```

```
create or replace FUNCTION FIND_PATIENT_ATTEND_PATIENT(patientid IN
ATTEND_PATIENT.P_ID%TYPE)
RETURN BOOLEAN AS
FLAG NUMBER(5):=0;
CURSOR P_REC IS
SELECT P_ID FROM ATTEND_PATIENT;
BEGIN
FOR I IN P_REC LOOP
 IF I.P_ID=patientid THEN
   FLAG:=1;
 END IF;
 END LOOP;
 IF FLAG=1 THEN
 RETURN TRUE;
 ELSE
 RETURN FALSE;
END IF;
END FIND_PATIENT_ATTEND_PATIENT;
create or replace FUNCTION FIND_PATIENT_LOG_PATIENT(patientid IN
LOG_PATIENT.PATIENT_ID%TYPE)
RETURN BOOLEAN AS
FLAG NUMBER(5):=0;
CURSOR P_REC IS
SELECT PATIENT_ID FROM LOG_PATIENT;
BEGIN
FOR I IN P_REC LOOP
 IF I.PATIENT_ID=patientid THEN
   FLAG:=1;
 END IF;
 END LOOP;
```

```
IF FLAG=1 THEN
 RETURN TRUE;
ELSE
 RETURN FALSE;
END IF;
END FIND_PATIENT_LOG_PATIENT;
create or replace FUNCTION FIND_RECEPTIONIST_ID(rid in RECEPTIONIST.R_ID%TYPE)
RETURN BOOLEAN AS
FLAG NUMBER(5):=0;
CURSOR RECEPTIONIST_REC IS
SELECT R_ID FROM ASSIST_PATIENT;
BEGIN
FOR I IN RECEPTIONIST_REC LOOP
 IF I.R_ID=rid THEN
   FLAG:=1;
 END IF;
END LOOP;
IF FLAG=1 THEN
 RETURN TRUE;
ELSE
 RETURN FALSE;
END IF;
END FIND_RECEPTIONIST_ID;
create or replace FUNCTION FIND_ROOMNO_ASSIGNEDROOM(roomid IN
ROOM.ROOM_NO%TYPE)
RETURN BOOLEAN AS
FLAG NUMBER(5):=0;
CURSOR ROOM_REC IS
SELECT ROOM_NO FROM ASSIGNED_ROOM;
BEGIN
```

```
FOR I IN ROOM_REC LOOP
 IF I.ROOM_NO=roomid THEN
   FLAG:=1;
 END IF;
END LOOP;
IF FLAG=1 THEN
 RETURN TRUE;
ELSE
 RETURN FALSE;
END IF;
END FIND_ROOMNO_ASSIGNEDROOM;
create or replace FUNCTION FIND_ROOMNO_GOVERNSROOM(roomid IN
ROOM.ROOM_NO%TYPE)
RETURN BOOLEAN AS
FLAG NUMBER(5):=0;
CURSOR ROOM_REC IS
SELECT ROOM_NO FROM GOVERNS_ROOM;
BEGIN
FOR I IN ROOM_REC LOOP
 IF I.ROOM_NO=roomid THEN
   FLAG:=1;
 END IF;
END LOOP;
IF FLAG=1 THEN
 RETURN TRUE;
ELSE
 RETURN FALSE;
END IF;
END FIND_ROOMNO_GOVERNSROOM;
```

create or replace FUNCTION RECEPTIONIST_EMAIL(uid in RECEPTIONIST.R_EMAIL%TYPE)

RETURN RECEPTIONIST.R_EMAIL% TYPE AS

email RECEPTIONIST.R_EMAIL%TYPE;

BEGIN

SELECT R_EMAIL INTO email FROM RECEPTIONIST WHERE R_ID=uid;

RETURN email;

END RECEPTIONIST_EMAIL;

create or replace FUNCTION USER_EMAIL(uid in USER_TYPE.U_EMAIL%TYPE)

RETURN USER_TYPE.U_EMAIL%TYPE AS

email USER_TYPE.U_EMAIL%TYPE;

BEGIN

SELECT U_EMAIL INTO email FROM USER_TYPE WHERE ID=uid;

RETURN email;

END USER_EMAIL;

Trigger

CREATE OR REPLACE TRIGGER PATIENT_LOG

AFTER INSERT ON PATIENT

REFERENCING OLD AS OLD NEW AS NEW

FOR EACH ROW

BEGIN

INSERT INTO LOG_PATIENT(PATIENT_ID,ADMIT_DATE) VALUES(:NEW.P_ID,SYSDATE);

END;

```
create or replace TRIGGER ADDRESS_TRG
BEFORE INSERT ON ADDRESS
FOR EACH ROW
BEGIN
 IF INSERTING AND :NEW.ADDRESS_ID IS NULL THEN
  SELECT ADDRESS_SEQ.NEXTVAL INTO :NEW.ADDRESS_ID FROM SYS.DUAL;
 END IF;
END COLUMN_SEQUENCES;
END;
create or replace TRIGGER ADMIN_TRG
BEFORE INSERT ON ADMIN
FOR EACH ROW
BEGIN
 IF INSERTING AND :NEW.ADMIN_ID IS NULL THEN
  SELECT ADMIN_SEQ.NEXTVAL INTO :NEW.ADMIN_ID FROM SYS.DUAL;
 END IF;
END COLUMN_SEQUENCES;
END;
create or replace TRIGGER DOCTOR_TRG
BEFORE INSERT ON DOCTOR
FOR EACH ROW
BEGIN
 IF INSERTING AND :NEW.D_ID IS NULL THEN
  SELECT DOCTOR_SEQ.NEXTVAL INTO :NEW.D_ID FROM SYS.DUAL;
 END IF;
END COLUMN_SEQUENCES;
END;
```

```
create or replace TRIGGER LOG_PATIENT_TRG
BEFORE INSERT ON LOG_PATIENT
FOR EACH ROW
BEGIN
 IF INSERTING AND :NEW.LOGID IS NULL THEN
  SELECT LOG_PATIENT_SEQ.NEXTVAL INTO :NEW.LOGID FROM SYS.DUAL;
 END IF;
END COLUMN_SEQUENCES;
END;
create or replace TRIGGER RECEPTIONIST_TRG
BEFORE INSERT ON RECEPTIONIST
FOR EACH ROW
BEGIN
 IF INSERTING AND :NEW.R_ID IS NULL THEN
  SELECT RECEPTIONIST_SEQ.NEXTVAL INTO: NEW.R_ID FROM SYS.DUAL;
 END IF;
END COLUMN_SEQUENCES;
END;
create or replace TRIGGER LOGIN_INFO_TRG
BEFORE INSERT ON LOGIN INFO
FOR EACH ROW
BEGIN
 IF INSERTING AND :NEW.LID IS NULL THEN
  SELECT LOGIN_INFO_SEQ.NEXTVAL INTO :NEW.LID FROM SYS.DUAL;
 END IF;
END COLUMN_SEQUENCES;
END;
```

Views

CREATE OR REPLACE FORCE VIEW "PATIENT_VIEW" ("P_ID", "P_NAME", "P_ADDRESS", "P_GENDER", "P_DOB", "P_CONTACT_NO", "P_STATUS", "P_CONDITION", "ADMIT_DATE", "RELEASE_DATE")

SELECT

P."P_ID",P."P_NAME",P."P_ADDRESS",P."P_GENDER",P."P_DOB",P."P_CONTACT_NO",P."P STATUS",P."P CONDITION",L.ADMIT DATE,L.RELEASE DATE

FROM PATIENT P,LOG PATIENT L

WHERE P.P ID=L.PATIENT ID;

CREATE OR REPLACE FORCE VIEW ASSIST_PATIENT_VIEW("P_ID", "P_NAME", "R_ID", "R_NAME") AS

SELECT P.P_ID,P_NAME,R.R_ID,R_NAME

FROM PATIENT P, RECEPTIONIST R, ASSIST_PATIENT A

WHERE P.P_ID=A.P_ID AND R.R_ID=A.R_ID;

CREATE OR REPLACE FORCE VIEW PATIENT_ROOM_VIEW("P_ID", "P_NAME", "P_ADDRESS", "P_GENDER", "P_DOB", "P_CONTACT_NO", "P_STATUS", "P_CONDITION", "ROOM_NO") AS

select

P."P_ID",P."P_NAME",P."P_ADDRESS",P."P_GENDER",P."P_DOB",P."P_CONTACT_NO",P."P_STATUS",P."P_CONDITION", AR.ROOM_NO

from PATIENT P, ASSIGNED_ROOM AR

where P.P_ID=AR.P_ID AND P.P_CONDITION='Not Released';

CREATE OR REPLACE FORCE VIEW LOGIN_VIEW_REC("U_TYPE", "LOGIN_DATE", "R_NAME", "R_ID") AS

SELECT U.U_TYPE,L.LOGIN_DATE,R.R_NAME,R_ID

FROM USER_TYPE U,LOGIN_INFO L,RECEPTIONIST R

WHERE U.ID=L.LOGGED PERSON AND U.ID=R.R ID AND L.LOGGED PERSON=R.R ID;

CREATE OR REPLACE FORCE VIEW LOGIN_VIEW_ADMIN("U_TYPE", "LOGIN_DATE", "ADMIN_NAME") AS

SELECT U.U_TYPE,L.LOGIN_DATE,A.ADMIN_NAME

FROM USER_TYPE U,LOGIN_INFO L,ADMIN A

WHERE U.ID=L.LOGGED_PERSON AND U.ID=A.ADMIN_ID AND L.LOGGED_PERSON=A.ADMIN_ID;

Packages

create or replace PACKAGE DELETE_PACKAGE AS

 $PROCEDURE\ DELETE_DOCTOR(doctorid\ in\ DOCTOR.D_ID\%\ TYPE)\ ;$

PROCEDURE DELETE_NURSE(nurseid in NURSE.N_ID%TYPE);

PROCEDURE DELETE_RECEPTIONIST(recid in RECEPTIONIST.R_ID%TYPE);

PROCEDURE DELETE_ROOM(roomid in ROOM.ROOM_NO%TYPE);

PROCEDURE DELETE_PATIENT(patientid IN PATIENT.P_ID%TYPE);

END DELETE_PACKAGE;

create or replace PACKAGE BODY DELETE_PACKAGE AS

PROCEDURE DELETE_DOCTOR(doctorid in DOCTOR.D_ID%TYPE) AS

BEGIN

DELETE FROM DOCTOR WHERE D ID=doctorid;

IF FIND_DOCTOR_ID(doctorid)=TRUE THEN

DELETE FROM ATTEND_PATIENT WHERE D_ID=doctorid;

END IF;

END DELETE_DOCTOR;

```
PROCEDURE DELETE_NURSE(nurseid in NURSE.N_ID%TYPE) AS
BEGIN
DELETE FROM NURSE WHERE N_ID=nurseid;
IF FIND_NURSE_ID(nurseid)=TRUE THEN
  DELETE FROM GOVERNS_ROOM WHERE N_ID=nurseid;
END IF:
END DELETE_NURSE;
PROCEDURE DELETE_RECEPTIONIST(recid in RECEPTIONIST.R_ID%TYPE) AS
BEGIN
DELETE FROM RECEPTIONIST WHERE R_ID=recid;
DELETE FROM USER_TYPE WHERE ID=recid;
IF FIND_RECEPTIONIST_ID(recid)=TRUE THEN
  DELETE FROM ASSIST_PATIENT WHERE R_ID=recid;
END IF;
END DELETE_RECEPTIONIST;
PROCEDURE DELETE_ROOM(roomid IN ROOM.ROOM_NO%TYPE) AS
roomid_assignedroom ROOM.ROOM_NO%TYPE;
roomid_governsroom ROOM.ROOM_NO%TYPE;
BEGIN
 DELETE FROM ROOM WHERE ROOM_NO=roomid;
 IF FIND_ROOMNO_ASSIGNEDROOM(roomid)=TRUE THEN
   DELETE FROM ASSIGNED_ROOM WHERE ROOM_NO=roomid;
 END IF;
 IF FIND ROOMNO GOVERNSROOM(roomid)=TRUE THEN
   DELETE FROM GOVERNS_ROOM WHERE ROOM_NO=roomid;
 END IF;
END DELETE_ROOM;
```

PROCEDURE DELETE_PATIENT(patientid IN PATIENT.P_ID%TYPE) AS BEGIN

DELETE FROM PATIENT WHERE P_ID=patientid;

IF FIND_PATIENT_ASSIGNED_ROOM(patientid)=TRUE THEN

DELETE FROM ASSIGNED_ROOM WHERE P_ID=patientid;

END IF;

IF FIND_PATIENT_ASSIST_PATIENT(patientid)=TRUE THEN

DELETE FROM ASSIST PATIENT WHERE P ID=patientid;

END IF:

IF FIND_PATIENT_ATTEND_PATIENT(patientid)=TRUE THEN

DELETE FROM ATTEND_PATIENT WHERE P_ID=patientid;

END IF:

IF FIND_PATIENT_LOG_PATIENT(patientid)=TRUE THEN

DELETE FROM LOG_PATIENT WHERE PATIENT_ID=patientid;

END IF;

END DELETE_PATIENT;

END DELETE_PACKAGE;

create or replace PACKAGE EDIT_PACKAGE AS

PROCEDURE EDIT_DOCTOR(id in DOCTOR.D_ID%TYPE,name in DOCTOR.D_NAME%TYPE,address in DOCTOR.D_ADDRESS%TYPE,contact in DOCTOR.D_CONTACT_NO%TYPE,salary in DOCTOR.D_SALARY%TYPE,dob in DOCTOR.D_JOINING_DATE%TYPE,designation in DOCTOR.D_DESIGNATION%TYPE);

PROCEDURE EDIT_NURSE(id in NURSE.N_ID%TYPE,name in NURSE.N_NAME%TYPE,address in NURSE.N_ADDRESS%TYPE,contact in NURSE.N_CONTACT_NO%TYPE,salary in NURSE.N_SALARY%TYPE,dob in NURSE.N_JOINING_DATE%TYPE,designation in NURSE.N_DESIGNATION%TYPE);

PROCEDURE EDIT_RECEPTIONIST(id in RECEPTIONIST.R_ID%TYPE,email in RECEPTIONIST.R_EMAIL%TYPE,name in RECEPTIONIST.R_NAME%TYPE,address in RECEPTIONIST.R_ADDRESS%TYPE,contact in RECEPTIONIST.R_CONTACT_NO%TYPE,salary in

RECEPTIONIST.R_SALARY%TYPE,dob in RECEPTIONIST.R_JOINING_DATE%TYPE,designation in RECEPTIONIST.R_DESIGNATION%TYPE);

PROCEDURE EDIT_ROOM(roomid in ROOM.ROOM_NO%TYPE,roomtype in ROOM.ROOM_TYPE%TYPE,roomcost IN ROOM.ROOM_COST%TYPE);

PROCEDURE EDIT_PATIENT(id in PATIENT.P_ID%TYPE,name in PATIENT.P_NAME%TYPE,contact in PATIENT.P_CONTACT_NO%TYPE,address in PATIENT.P_ADDRESS%TYPE,gender in PATIENT.P_GENDER%TYPE,dob IN PATIENT.P_DOB%TYPE);

END EDIT_PACKAGE;

create or replace PACKAGE BODY EDIT_PACKAGE AS

PROCEDURE EDIT_DOCTOR(id in DOCTOR.D_ID%TYPE,name in DOCTOR.D_NAME%TYPE,address in DOCTOR.D_ADDRESS%TYPE,contact in DOCTOR.D_CONTACT_NO%TYPE,salary in DOCTOR.D_SALARY%TYPE,dob in DOCTOR.D_JOINING_DATE%TYPE,designation in DOCTOR.D_DESIGNATION%TYPE) AS

BEGIN

UPDATE DOCTOR SET D_NAME=name, D_ADDRESS=address, D_CONTACT_NO=contact,D_SALARY=salary,D_JOINING_DATE=DOB,D_DESIGNAT ION=designation WHERE D_ID=id;

END EDIT DOCTOR;

PROCEDURE EDIT_NURSE(id in NURSE.N_ID%TYPE,name in NURSE.N_NAME%TYPE,address in NURSE.N_ADDRESS%TYPE,contact in NURSE.N_CONTACT_NO%TYPE,salary in NURSE.N_SALARY%TYPE,dob in NURSE.N_JOINING_DATE%TYPE,designation in NURSE.N_DESIGNATION%TYPE) AS

BEGIN

UPDATE NURSE SET N_NAME=name, N_ADDRESS=address, N_CONTACT_NO=contact,N_SALARY=salary,N_JOINING_DATE=DOB,N_DESIGNAT ION=designation WHERE N_ID=id;

END EDIT_NURSE;

PROCEDURE EDIT_RECEPTIONIST(id in RECEPTIONIST.R_ID%TYPE,email in

RECEPTIONIST.R_EMAIL%TYPE,name in RECEPTIONIST.R_NAME%TYPE,address in

RECEPTIONIST.R_ADDRESS%TYPE,contact in

RECEPTIONIST.R_CONTACT_NO%TYPE, salary in

RECEPTIONIST.R SALARY%TYPE,dob in

RECEPTIONIST.R_JOINING_DATE% TYPE, designation in

RECEPTIONIST.R_DESIGNATION%TYPE) AS

oldemail RECEPTIONIST.R_EMAIL%TYPE;

newemail RECEPTIONIST.R_EMAIL%TYPE;

userid USER_TYPE.ID%TYPE:=id;

BEGIN

oldemail:=RECEPTIONIST_EMAIL(id);

UPDATE RECEPTIONIST SET R_NAME=name,R_EMAIL=email,

R ADDRESS=address,

R_CONTACT_NO=contact,R_SALARY=salary,R_JOINING_DATE=DOB,R_DESIGNATI ON=designation WHERE R_ID=id;

newemail:=RECEPTIONIST_EMAIL(id);

IF oldemail!=newemail THEN

UPDATE USER_TYPE SET U_EMAIL=newemail WHERE ID=userid;

END IF;

END EDIT_RECEPTIONIST;

PROCEDURE EDIT_ROOM(roomid in ROOM.ROOM_NO%TYPE,roomtype in ROOM.ROOM TYPE%TYPE,roomcost IN ROOM.ROOM COST%TYPE) AS

BEGIN

UPDATE ROOM_SET ROOM_TYPE=roomtype,ROOM_COST=roomcost WHERE ROOM_NO=roomid;

END EDIT_ROOM;

PROCEDURE EDIT_PATIENT(id in PATIENT.P_ID%TYPE,name in PATIENT.P_NAME%TYPE,contact in PATIENT.P_CONTACT_NO%TYPE,address in PATIENT.P_ADDRESS%TYPE,gender in PATIENT.P_GENDER%TYPE,dob IN PATIENT.P_DOB%TYPE) AS

BEGIN

UPDATE PATIENT SET

 $\label{eq:pname} P_NAME = name, P_CONTACT_NO = contact, P_ADDRESS = address, P_GENDER = gender, P_DOB = dob\ where\ P_ID = id;$

END EDIT_PATIENT;

END EDIT_PACKAGE;