



AMERICAN INTERNATIONAL UNIVERSITY- BANGLADESH

where leaders are created

Project Name: Hospital Management System
Course Name: Advance Database Management System
Section: B

Serial No	ID	Name
1	18-38113-2	Md Asifur Rahman
2	18-38573-2	Md. Abu Bakar Siddique Sadi
3	17-35531-3	Shahnaz Alam
4	18-37350-1	Mst. Maksuratun Tabassum
5	18-37334-1	Mishu,Munia Mostary

Contents

1.Introduction:	3
2.Project Proposal:	3
3.Class Diagram	4
4.Use case Diagram	5
5.Activity Diagram	6
6.User Interface	7
7.Scenario Description:	12
8.ER Diagram	13
9.Normalization	14
10.Schema Diagram	15
11.Table Creation	16
12.Data Insertion	17
13.Query Writing	21
(i)SINGLE ROW FUNCTION	21
(ii)GROUP FUNCTION	21
(iii)SUB QUERY	21
(iv)JOINING	22
(v)VIEW	22
14.PL/SQL	23
(i)FUNCTION:	23
(ii)PROCEDURE:	27
(iii)RECORD	31
(iv)CURSOR	35
(v)TRIGGER	40
(vi)PACKAGE	44
15.Conclusion	57

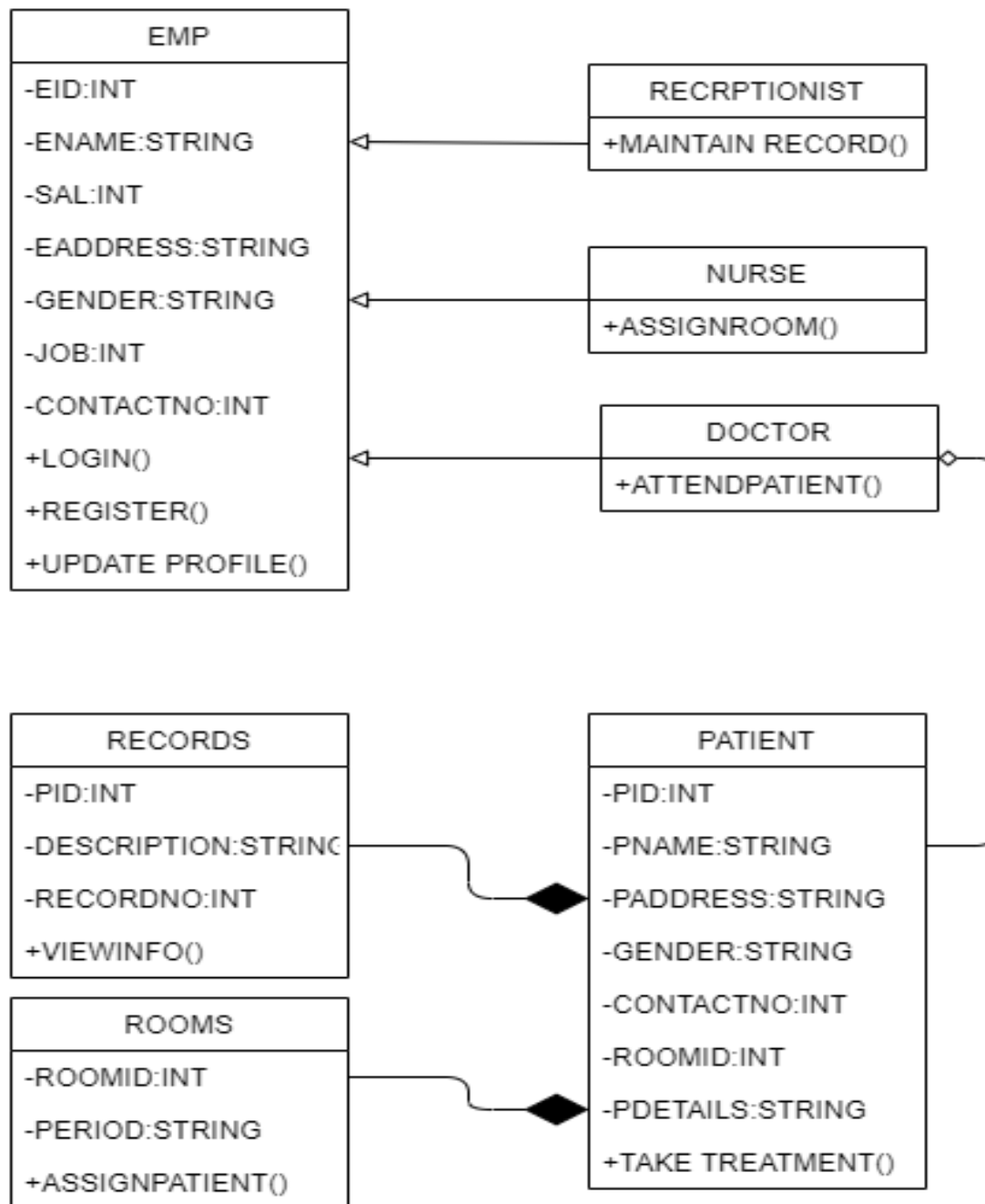
1.Introduction:

Our project Hospital Management System shows all the visual instrument of database tables and the relations between Patient, employees, Rooms, Records etc. It used structure data and to define the relationships between structured data groups of Hospital Management System functionalities.

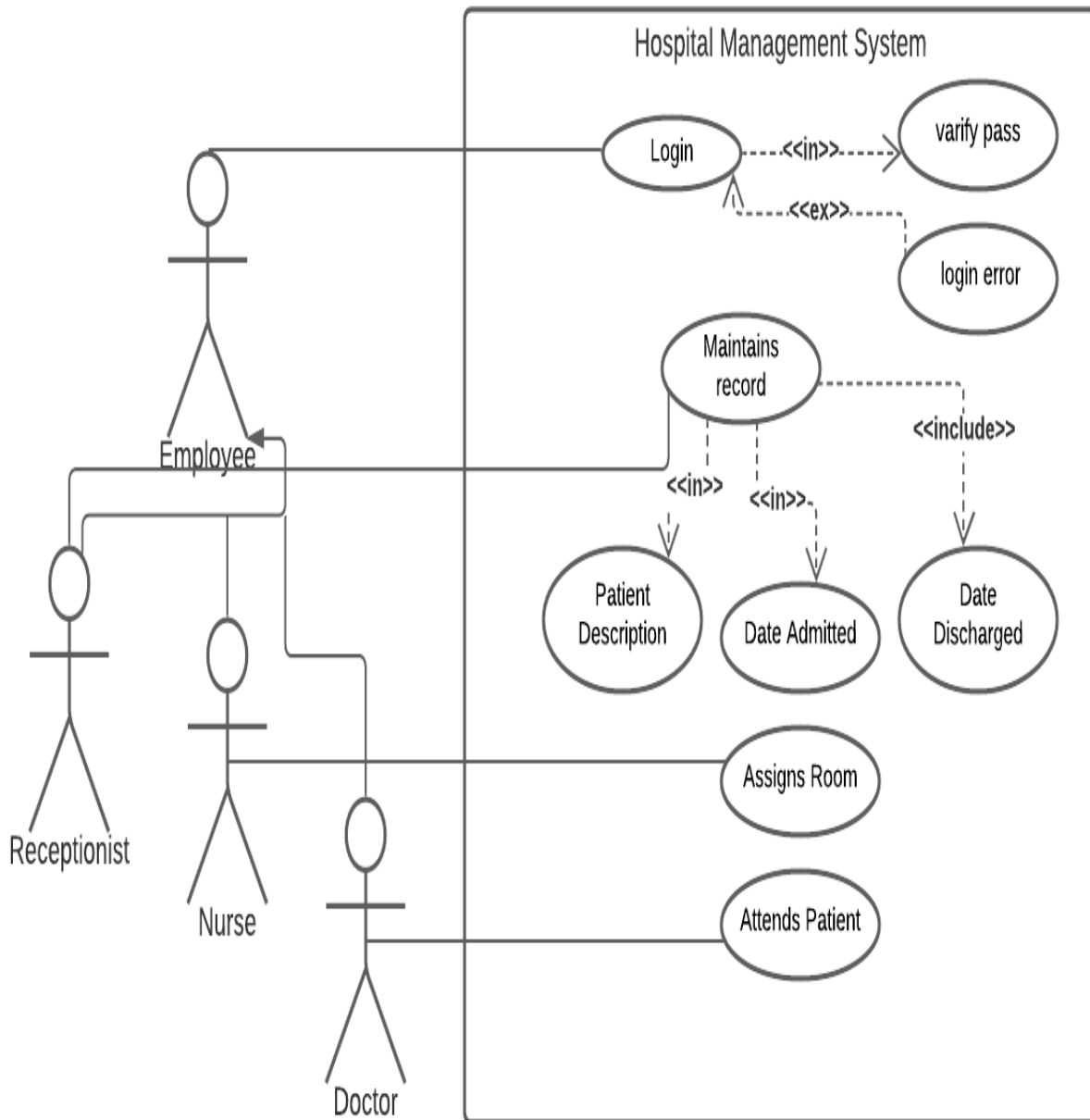
2.Project Proposal:

Our project Hospital Management System includes admission of patients, storing their details into the system. It has the facility to give a unique id for every patient and stores the details of every patient and the employee automatically. It includes a search facility to know the current status of each room. We can search availability of a employee and the details of a patient using the id. It is accessible by receptionist. Only they can add data into the database. The data can be retrieved easily. The interface is very user-friendly. The data are well protected for personal use and makes the data processing very fast.

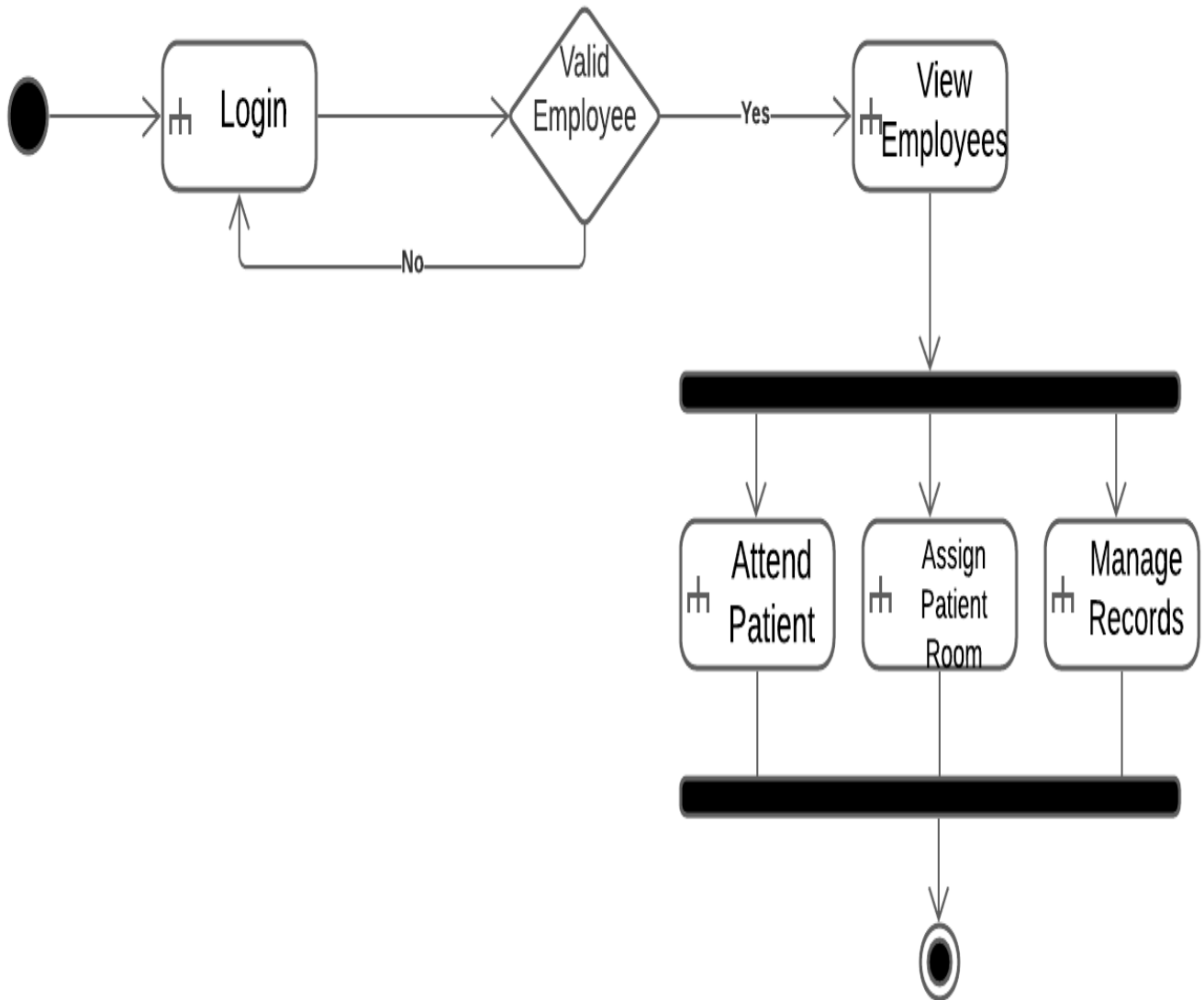
3. Class Diagram



4. Use case Diagram



5. Activity Diagram



6.User Interface

LOGIN



User Interface

[Back](#)

Registration

Name

Username

Email

Password

Confirm Password

User type:

☐ Customer

☐ Seller

User Interface

Welcome to Home page

[Patients Details](#)

[Employees Details](#)

[Logout](#)

User Interface

PATIENTS DETAILS

ID	NAME	CONTACT NO.	ADDRESS	DATE OF ADMIT	DATE OF DISCHARGE
1					
2					
3					
4					
5					

User Interface

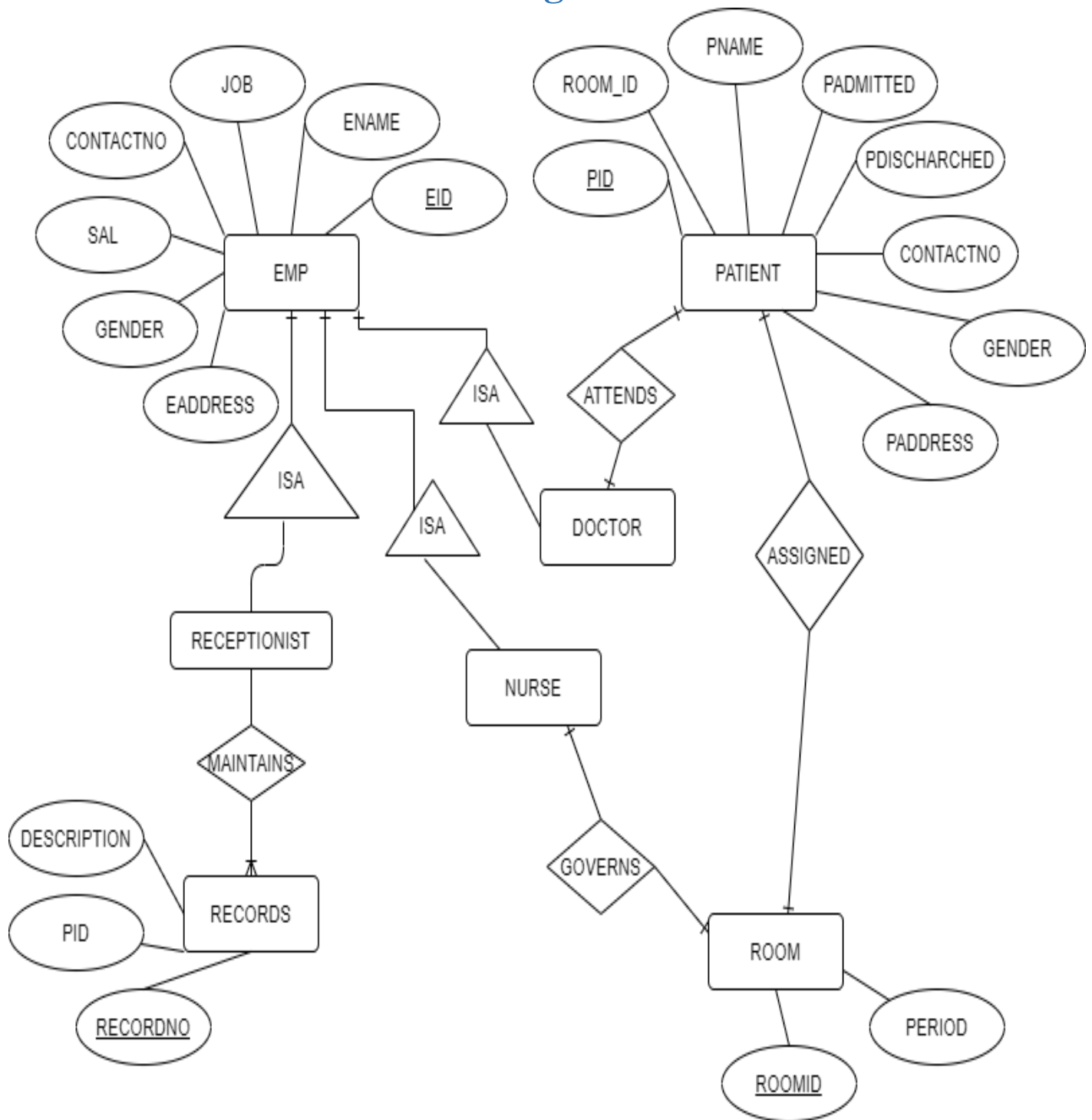
EMPLOYEES DETAILS

ID	NAME	CONTACT NO.	ADDRESS	SALARY	NID NO.
1					
2					
3					
4					
5					

7.Scenario Description:

In this diagram, there is Employee entity which has three categories like Doctor , Nurse and Receptionist. Employee entity has unique employee ID, name, salary, address, gender,NID and contact no. We can store details of every employee who are working in this hospital and search their details with their unique id automatically. Here, Doctor can attend patient. Patient entity has different attributes like patient name, unique ID, room ID, contact no ,address, gender, patient details, admission date and discharged date. It will be easy to find patient details with their unique ID where every details of patient is stored. Patient can be assigned Room. Room entity has a unique Room ID and period attribute. So we know the details that which patient will be assigned in which room and duration. Room is being governed by Nurse. Receptionist maintains record of patient. Record entity has three attributes like unique patient ID, description and Record no. So we will know the details of every patient with their record number whenever it is needed.

8.ER Diagram



9.Normalization

Bill (code, price, quantity, PID, name, gender, address, date admitted, date discharged, contact no)

1NF: contact no

2NF: code, price, quantity

PID, name, gender, address, date admitted, date discharged, contact no

3NF: code, price, quantity

PID, name, gender, address, P_details, contact no

P_details, date admitted, date discharged

Assigned (PID, name, gender, address, date admitted, date discharged, contact no, Room_ID, period)

1NF: contact no

2NF: PID, name, gender, address, date admitted, date discharged, contact no

Room_ID, period

3NF: PID, name, gender, address, P_details, contact no

Room_ID, period

P_details, date admitted, date discharged

Governs (Room_ID, period)

1NF:

2NF: Room_ID, period

3NF: Room_ID, period

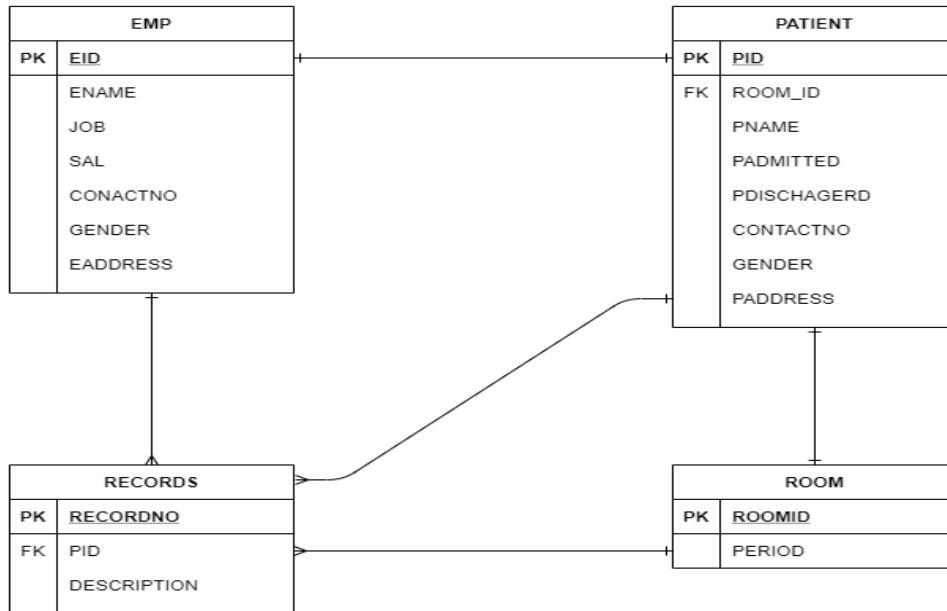
Total Table:

1. code, price, quantity
2. PID, name, gender, address, P_details, contact no
3. P_details, date admitted, date discharged
4. PID, name, gender, address, P_details, contact no
5. Room_ID, period
6. P_details, date admitted, date discharged
7. Room_ID, period

Final Table:

1. code, price, quantity
2. PID, name, gender, address, P_details, contact no
3. P_details, date admitted, date discharged
4. Room_ID, period

10.Schema Diagram



11. Table Creation

```
❑ CREATE TABLE EMP
(EID INT NOT NULL PRIMARY KEY,
 ENAME VARCHAR2(10),
 JOB VARCHAR2(9),
 SAL NUMBER(7),
 CONTACTNO NUMBER(11),
 GENDER VARCHAR2(8),
 EADDRESS VARCHAR2(50))
```

```
❑ CREATE TABLE ROOMS
(ROOMID INT NOT NULL PRIMARY KEY,
 PERIOD_DAYS NUMBER(4))
```

```
❑ CREATE TABLE PATIENT
(PID INT NOT NULL PRIMARY KEY,
 PNAME VARCHAR2(10),
 PADMitted DATE,
 PDISCHARGED DATE,
 CONTACTNO NUMBER(11),
 GENDER VARCHAR2(8),
 PADDRESS VARCHAR2(50),
 ROOM_ID INT references ROOMS (ROOMID))
```

```
❑ CREATE TABLE RECORDS
(RECORDNO INT NOT NULL PRIMARY KEY,
 DESCRIPTION VARCHAR2(80),
 PID INT references PATIENT (PID))
```


12.Data Insertion

INSERT INTO EMP VALUES

(1001,'ALEX','DOCTOR',7000,001100110,'MALE','BOSTON')

INSERT INTO EMP VALUES

(1002,'AVA','DOCTOR',7500,000000110,'FEMALE','DETROIT')

INSERT INTO EMP VALUES

(1003,'DREW','DOCTOR',7800,011000110,'MALE','DETROIT')

INSERT INTO EMP VALUES

(1004,'RYAN','DOCTOR',68000,010000110,'MALE','BOSTON')

INSERT INTO EMP VALUES

(1005,'AVERY','DOCTOR',7000,000000111,'FEMALE','TEXAS')

INSERT INTO EMP VALUES

(1006,'LOGAN','DOCTOR',7200,000110000,'MALE','FLORIDA')

INSERT INTO EMP VALUES

(1007,'EMERSON','DOCTOR',8000,100110000,'FEMALE','NEW-YORK')

INSERT INTO EMP VALUES

(1008,'BROOKLYN','DOCTOR',9000,000110010,'FEMALE','NEW-YORK')

INSERT INTO EMP VALUES

(1009,'TONY','DOCTOR',9500,010110000,'MALE','CALIFORNIA')

INSERT INTO EMP VALUES

(1010,'PERKER','DOCTOR',9000,010110100,'MALE','NEW-YORK')

INSERT INTO EMP VALUES

(1011,'AUSTIN','NURSE',5500,010100100,'MALE','NEW-YORK')

INSERT INTO EMP VALUES

(1012,'ERZA','NURSE',5400,010100100,'FEMALE','BOSTON')

INSERT INTO EMP VALUES

(1013,'ADAM','NURSE',5400,010111100,'MALE','TEXAS')

INSERT INTO EMP VALUES

(1014,'PENNEY','NURSE',5500,110111101,'FEMALE','TEXAS')

INSERT INTO EMP VALUES

(1015,'AMY','RECEPTION',4800,110111101,'FEMALE','NEW-YORK')

INSERT INTO EMP VALUES

(1016,'KALEY','NURSE',5200,110000001,'FEMALE','TEXAS')

INSERT INTO EMP VALUES

(1017,'BARNI','RECEPTION',5200,110010001,'FEMALE','TEXAS')

INSERT INTO ROOMS VALUES

(101,5)

INSERT INTO ROOMS VALUES

(102,9)

INSERT INTO ROOMS VALUES

(103,5)

INSERT INTO ROOMS VALUES

(104,15)

INSERT INTO ROOMS VALUES
(105,3)

INSERT INTO ROOMS VALUES
(106,9)

INSERT INTO ROOMS VALUES
(107,12)

INSERT INTO PATIENT VALUES
(8001,'ADAM',to_date('17-12-2020','dd-mm-yyyy'),to_date('17-01-2021','dd-mm-yyyy'),80080074570,'MALE','KURIL',101);

INSERT INTO PATIENT VALUES
(8002,'MIA',to_date('15-1-2021','dd-mm-yyyy'),to_date('04-02-2021','dd-mm-yyyy'),80008970070,'FEMALE','UTTARA',103);

INSERT INTO PATIENT VALUES
(8003,'CARTER',to_date('20-01-2021','dd-mm-yyyy'),to_date('04-02-2021','dd-mm-yyyy'),80088970070,'MALE','DHANMONDI',102);

INSERT INTO PATIENT VALUES
(8004,'BLAKE',to_date('30-03-2021','dd-mm-yyyy'),to_date('07-04-2021','dd-mm-yyyy'),8076070070,'MALE','MIRPUR',101);

INSERT INTO PATIENT VALUES
(8005,'BRENT',to_date('03-02-2021','dd-mm-yyyy'),to_date('27-02-2021','dd-mm-yyyy'),80080897000,'MALE','NORDA',104);

INSERT INTO PATIENT VALUES
(8006,'ROSE',to_date('09-03-2021','dd-mm-yyyy'),to_date('27-03-2021','dd-mm-yyyy'),80080897001,'FEMALE','FARMGATE',105);

INSERT INTO PATIENT VALUES

(8007,'RICK',to_date('20-03-2021','dd-mm-yyyy'),to_date('27-03-2021','dd-mm-yyyy'),80080897078,'MALE','FARMGATE',103);

INSERT INTO RECORDS VALUES

(101,'Acute upper respiratory infections ',8001)

INSERT INTO RECORDS VALUES

(102,'Diabetes Mellitus',8002)

INSERT INTO RECORDS VALUES

(103,'Dislocated bone',8003)

INSERT INTO RECORDS VALUES

(104,'Acute upper respiratory infections',8004)

INSERT INTO RECORDS VALUES

(105,'Osteoarthritis',8005)

INSERT INTO RECORDS VALUES

(106,'Infectious diseases',8006)

INSERT INTO RECORDS VALUES

(107,'Fever',8006)

INSERT INTO RECORDS VALUES

(108,'Broken Bone',8005)

13.Query Writing

(i)SINGLE ROW FUNCTION

SELECT CONCAT(ENAME,SAL) FROM EMP;

SELECT ENAME,LENGTH(ENAME)FROM EMP;

SELECT ENAME,INSTR(ENAME,'A')FROM EMP;

(ii)GROUP FUNCTION

SELECT SUM(SAL) TOTAL_SAL FROM EMP;

SELECT COUNT(EID), EADDRESS FROM EMP GROUP BY
EADDRESS ORDER BY COUNT(EID) DESC;

SELECT MAX (PADMITTED) LATEST FROM PATIENT;

(iii)SUB QUERY

SELECT * FROM PATIENT WHERE ROOM_ID= '103'

SELECT * FROM EMP WHERE EID IN (SELECT EID FROM EMP
WHERE SAL > 7500);

SELECT * FROM PATIENT WHERE PADDRESS (SELECT
PADDRESS FROM PATIENT WHERE PADDRESS='MIRPUR');

(iv)JOINING

```
SELECT PNAME, CONTACTNO, PADMITTED FROM PATIENT,  
ROOMS WHERE PATIENT.ROOM_ID = ROOMS.ROOMID;
```

```
SELECT PNAME,CONTACTNO FROM PATIENT INNER JOIN  
RECORDSON PATIENT.PID = RECORDS.PID;
```

```
SELECT PID, DESCRIPTOIN, RECORDNO  
FROM PATIENT O RIGHT JOIN RECORDS C  
ON O.PID = C.PID  
WHERE PID>8001
```

(v)VIEW

```
CREATE VIEW DETAILS AS SELECT ENAME, EADDRESS FROM  
EMP WHERE EID < 1008;  
SELECT * FROM DETAILS
```

```
CREATE VIEW DETAILS_VIEW AS SELECT EID, ENAME, SAL  
FROM EMP ORDER BY SAL;  
SELECT * FROM DETAILS_VIEW
```

```
CREATE VIEW PATIENT_DETAIL AS SELECT PATIENT.PNAME,  
RECORDS.DESCRPTION FROM PATIENT, RECORDS WHERE  
PATIENT.PID = RECORDS.PID;  
SELECT * FROM PATIENT_DETAIL
```

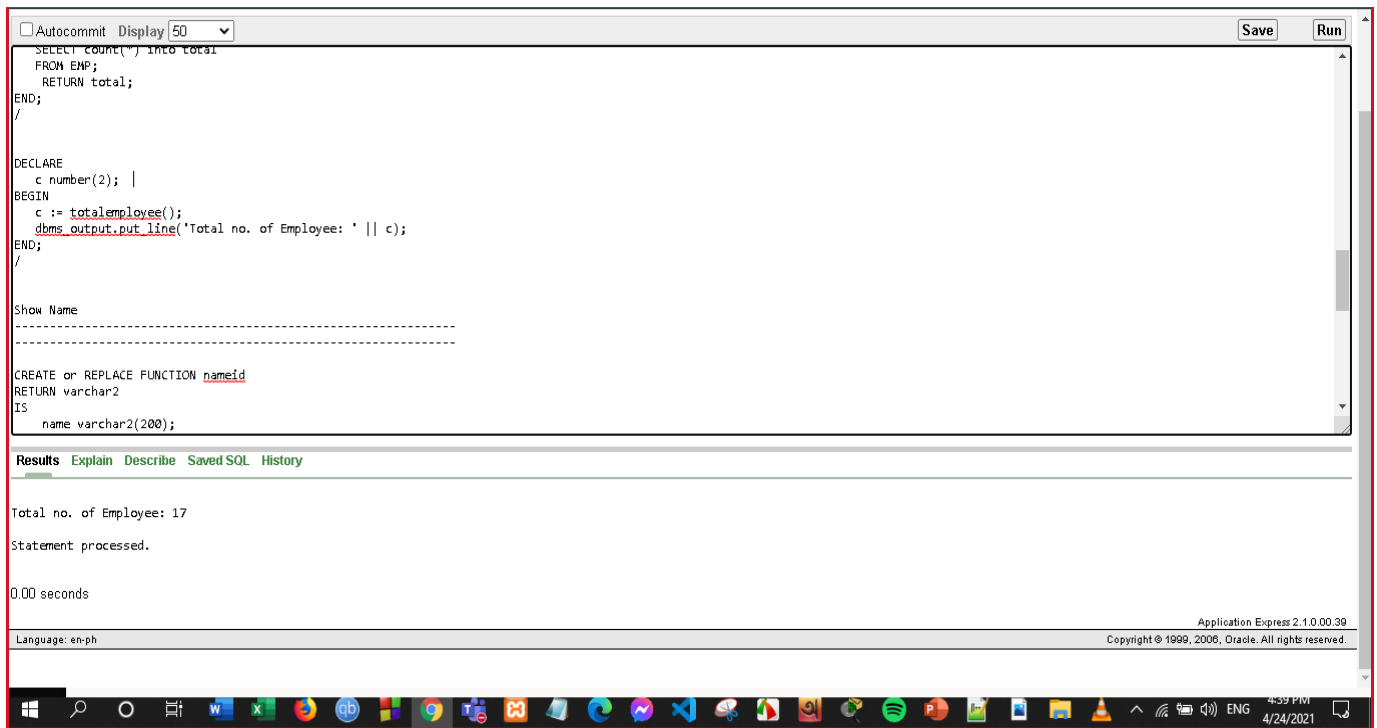
14.PL/SQL

(i)FUNCTION:

1. Find the total number of employee

```
CREATE OR REPLACE FUNCTION totalemployee  
RETURN number IS  
    total number(2) := 0;  
BEGIN  
    SELECT count(*) into total  
    FROM EMP;  
    RETURN total;  
END;  
/
```

```
DECLARE  
    c number(2);  
BEGIN  
    c := totalemployee();  
    dbms_output.put_line('Total no. of Employee: ' || c);  
END;  
/
```



2. Findout the total salary of employees.

CREATE OR REPLACE FUNCTION sumsal

RETURN number IS

total number := 0;

BEGIN

SELECT SUM(SAL) into total

FROM EMP;

RETURN total;

END;

/

DECLARE

 c number;

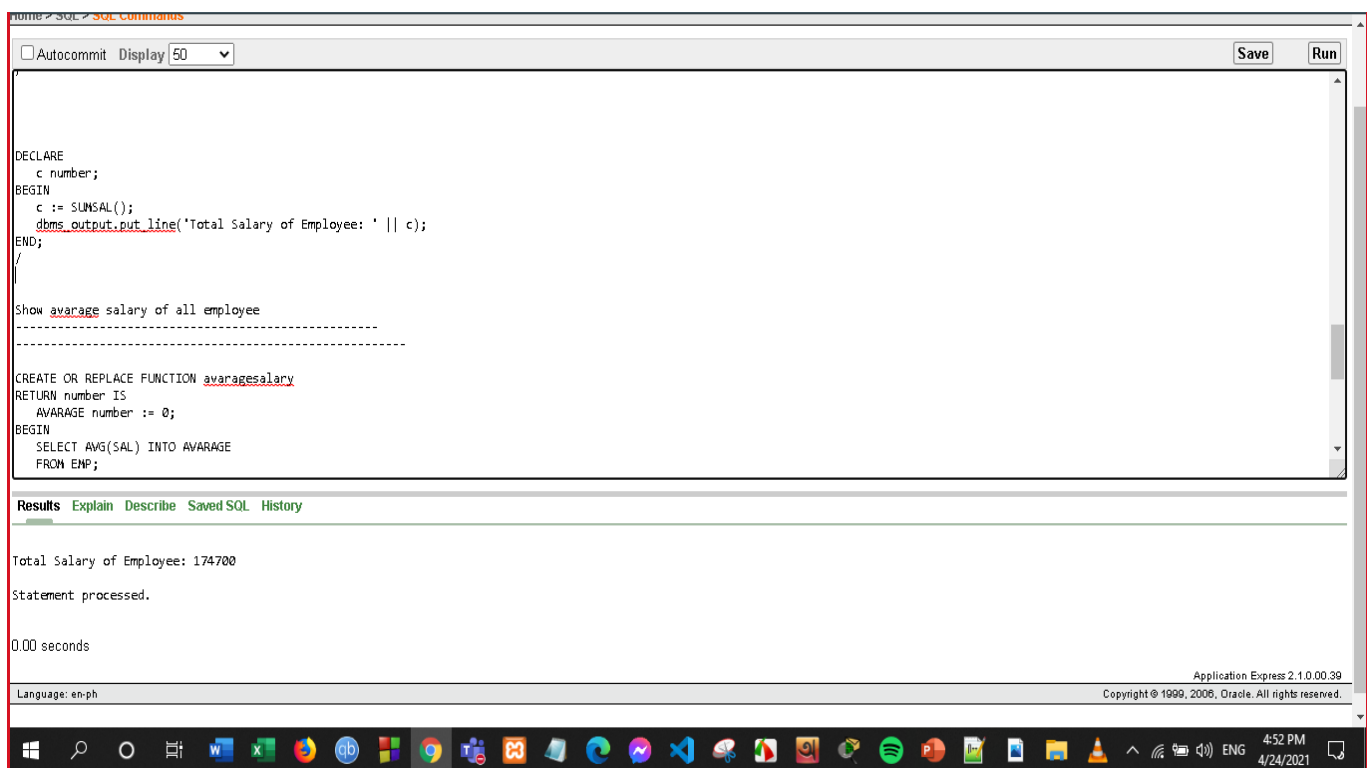
BEGIN

 c := SUMSAL();

 dbms_output.put_line('Total Salary of Employee: ' || c);

END;

/



The screenshot shows a window titled "SQL Commands" with a toolbar containing "Autocommit", "Display" (set to 50), "Save", and "Run". The main text area contains the following SQL code:

```
DECLARE
  c number;
BEGIN
  c := SUMSAL();
  dbms_output.put_line('Total Salary of Employee: ' || c);
END;
/
```

Below the code, there is a section titled "Show average salary of all employee" followed by a dashed line. The "Results" tab is active, displaying the output of the SQL commands:

```
Total Salary of Employee: 174700
Statement processed.
0.00 seconds
```

The bottom status bar shows "Language: en-ph", "Application Express 2.1.0.00.39", and "Copyright © 1999, 2006, Oracle. All rights reserved." The Windows taskbar is visible at the bottom with various application icons and a system clock showing 4:52 PM on 4/24/2021.

3. Show average salary of all employees.

```
CREATE OR REPLACE FUNCTION avaragesalary  
RETURN number IS  
    AVERAGE number := 0;  
BEGIN  
    SELECT AVG(SAL) INTO AVERAGE  
    FROM EMP;  
  
    RETURN AVERAGE;  
END;  
/
```

```
DECLARE  
    c number;  
BEGIN  
    c := avaragesalary();  
    dbms_output.put_line('AVERAGE SALARY: ' || c);  
END;  
/
```

The screenshot shows the SQL Developer 'SQL Commands' window. The script defines a function `avaragesalary` that calculates the average salary from the `EMP` table. It then declares a variable `c` and calls the function, outputting the result. The 'Results' pane shows the output: 'AVARAGE SALARY: 10276.4705882352941176470588235294117647'. The status bar indicates 'Statement processed.' and '0.00 seconds'.

```
CREATE OR REPLACE FUNCTION avaragesalary
RETURN number IS
  AVARAGE number := 0;
BEGIN
  SELECT AVG(SAL) INTO AVARAGE
  FROM EMP;

  RETURN AVARAGE;
END;
/

DECLARE
  c number;
BEGIN
  c := avaragesalary();
  dbms_output.put_line('AVARAGE SALARY: ' || c);
END;
/

-----
-----

Results Explain Describe Saved SQL History

AVARAGE SALARY: 10276.4705882352941176470588235294117647
Statement processed.
0.00 seconds

Language: en-ph
Application Express 2.1.0.00.39
Copyright © 1999, 2006, Oracle. All rights reserved.
```

(ii)PROCEDURE:

1 . Findout the minimum number using procedure.

DECLARE

a number;

b number;

c number;

PROCEDURE findMin(x IN number, y IN number, z OUT number) IS

BEGIN

IF x < y THEN

z:= x;

ELSE

z:= y;

END IF;

```

END;

BEGIN

    a:= 21;

    b:= 45;

    findMin(a, b, c);

    dbms_output.put_line(' Minimum of (21, 45) : ' || c);

END;

/

```

The screenshot shows the Oracle SQL Developer interface. The top toolbar includes buttons for 'Save' and 'Run'. The main text area contains the following PL/SQL code:

```

DECLARE
  a number;
  b number;
  c number;
PROCEDURE findMin(x IN number, y IN number, z OUT number) IS
BEGIN
  IF x < y THEN
    z:= x;
  ELSE
    z:= y;
  END IF;
END;
BEGIN
  a:= 21;
  b:= 45;
  findMin(a, b, c);
  dbms_output.put_line(' Minimum of (21, 45) : ' || c);
END;
/

```

Below the code editor, the 'Results' tab is active, displaying the output of the script:

```

Minimum of (21, 45) : 21
Statement processed.
0.00 seconds

```

The bottom status bar indicates the language is 'enph' and the application is 'Application Express 2.1.0.00.39'. The Windows taskbar at the bottom shows the time as 9:15 PM on 4/24/2021.

2 . Findout the square of a number.

```

DECLARE

    a number;

PROCEDURE squareNum(x IN OUT number) IS

```

BEGIN

 x := x * x;

END;

BEGIN

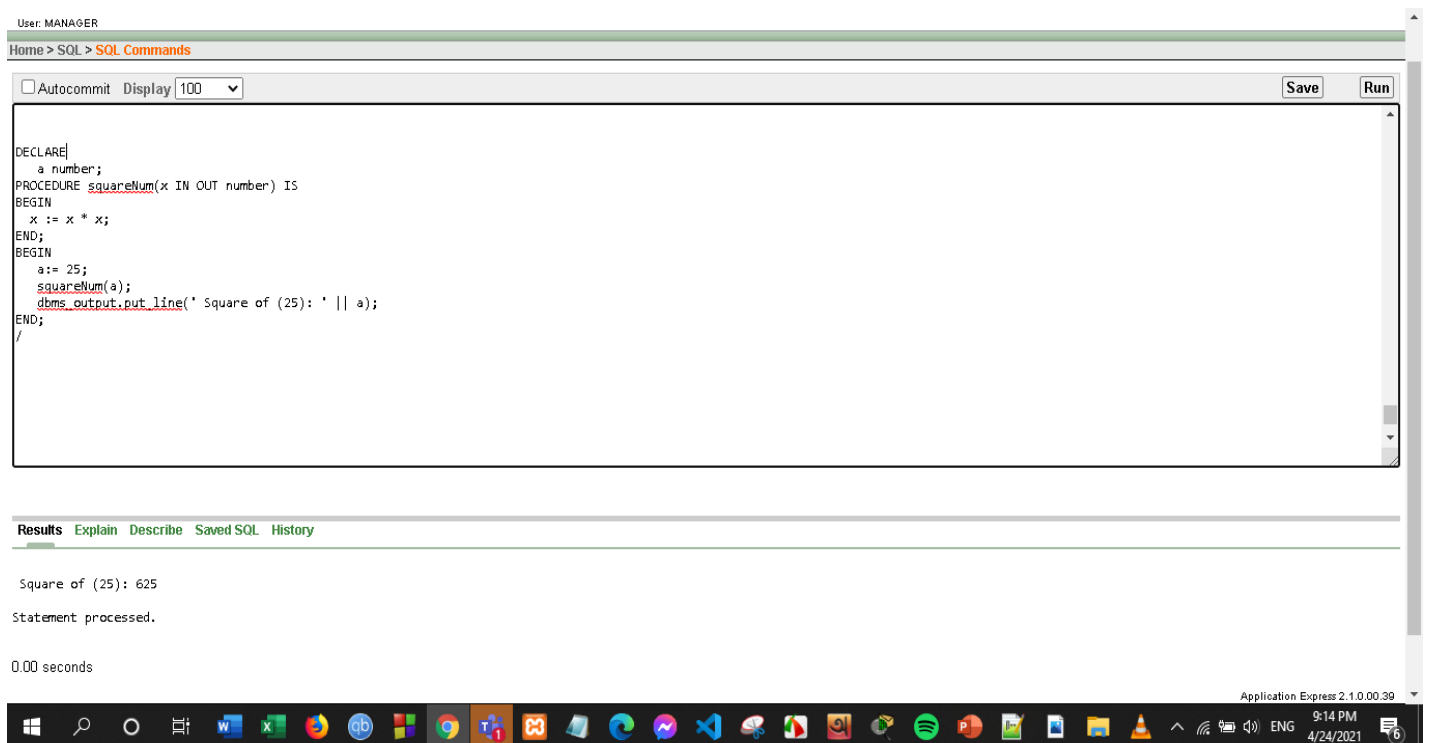
 a:= 25;

 squareNum(a);

 dbms_output.put_line(' Square of (25): ' || a);

END;

/



The screenshot shows the Oracle SQL Developer interface. At the top, the user is identified as 'MANAGER'. The breadcrumb navigation shows 'Home > SQL > SQL Commands'. Below this, there are controls for 'Autocommit' (unchecked), 'Display' (set to 100), and buttons for 'Save' and 'Run'. The main text area contains the following PL/SQL code:

```
DECLARE
  a number;
PROCEDURE squareNum(x IN OUT number) IS
BEGIN
  x := x * x;
END;
BEGIN
  a:= 25;
  squareNum(a);
  dbms_output.put_line(' Square of (25): ' || a);
END;
/
```

Below the code editor, the 'Results' tab is active, displaying the output of the script:

```
Square of (25): 625
Statement processed.
0.00 seconds
```

The bottom of the image shows the Windows taskbar with various application icons and the system clock indicating 9:14 PM on 4/24/2021.

3 . Findout the square of a number.

DECLARE

```
b number;
```

```
PROCEDURE squareNum(x IN OUT number) IS
```

```
BEGIN
```

```
  x := x * x;
```

```
END;
```

```
BEGIN
```

```
  b:= 10;
```

```
  squareNum(b);
```

```
  dbms_output.put_line(' Square of (10): ' || b);
```

```
END;
```

```
/
```

The screenshot shows the SQL Developer application window. The top toolbar includes buttons for Autocommit, Display (set to 100), Save, and Run. The main text area contains the following PL/SQL code:

```
DECLARE
  b number;
PROCEDURE squareNum(x IN OUT number) IS
BEGIN
  x := x * x;
END;
BEGIN
  b:= 10;
  squareNum(b);
  dbms_output.put_line(' Square of (10): ' || b);
END;
/
```

Below the code editor, the 'Results' tab is active, displaying the output of the execution:

```
Square of (10): 100
Statement processed.
0.00 seconds
```

The bottom status bar shows the language as 'en-ph', the application version as 'Application Express 2.1.0.00.39', and the copyright notice 'Copyright © 1999, 2006, Oracle. All rights reserved.' The Windows taskbar at the very bottom shows various application icons and the system clock indicating 9:17 PM on 4/24/2021.

(iii)RECORD

1 . Show roomid and period days using record.

declare

room_rec rooms%rowtype;

begin

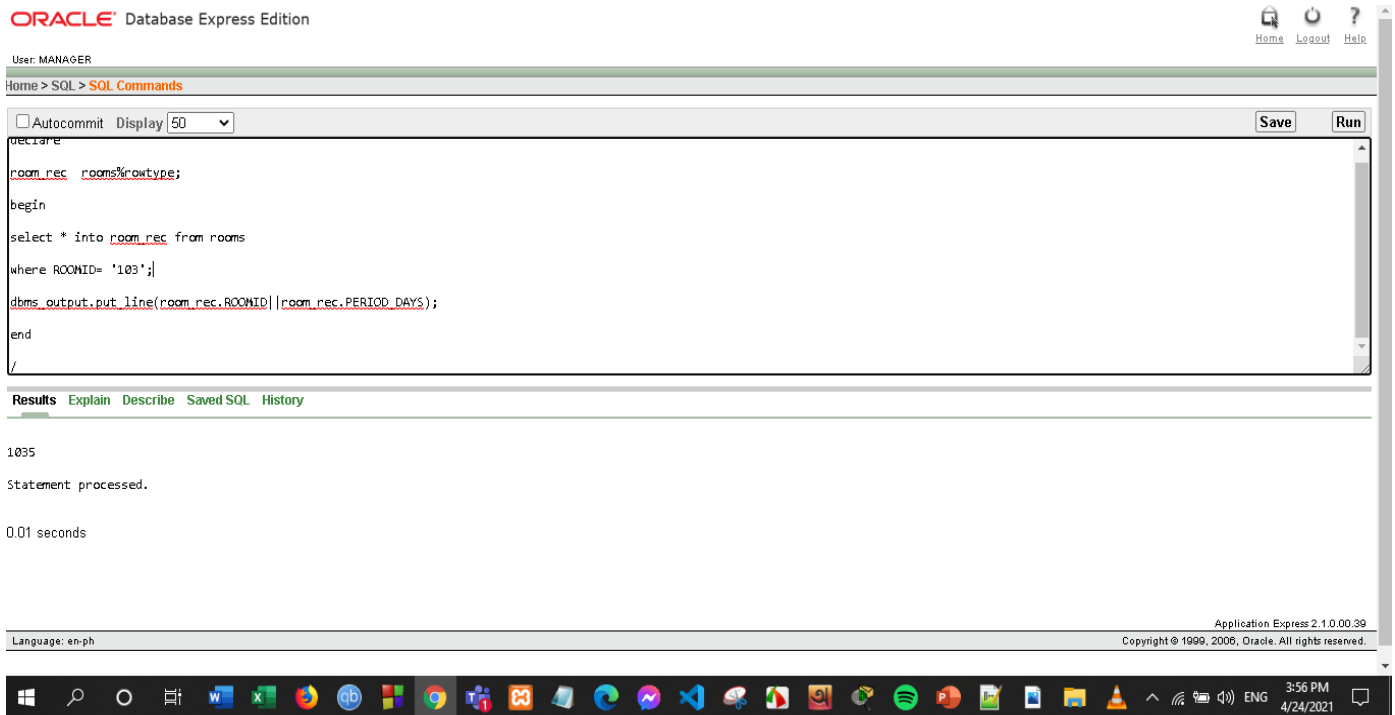
select * into room_rec from rooms

where ROOMID= '103';

dbms_output.put_line(room_rec.ROOMID || ' '||room_rec.PERIOD_DAYS);

end

/

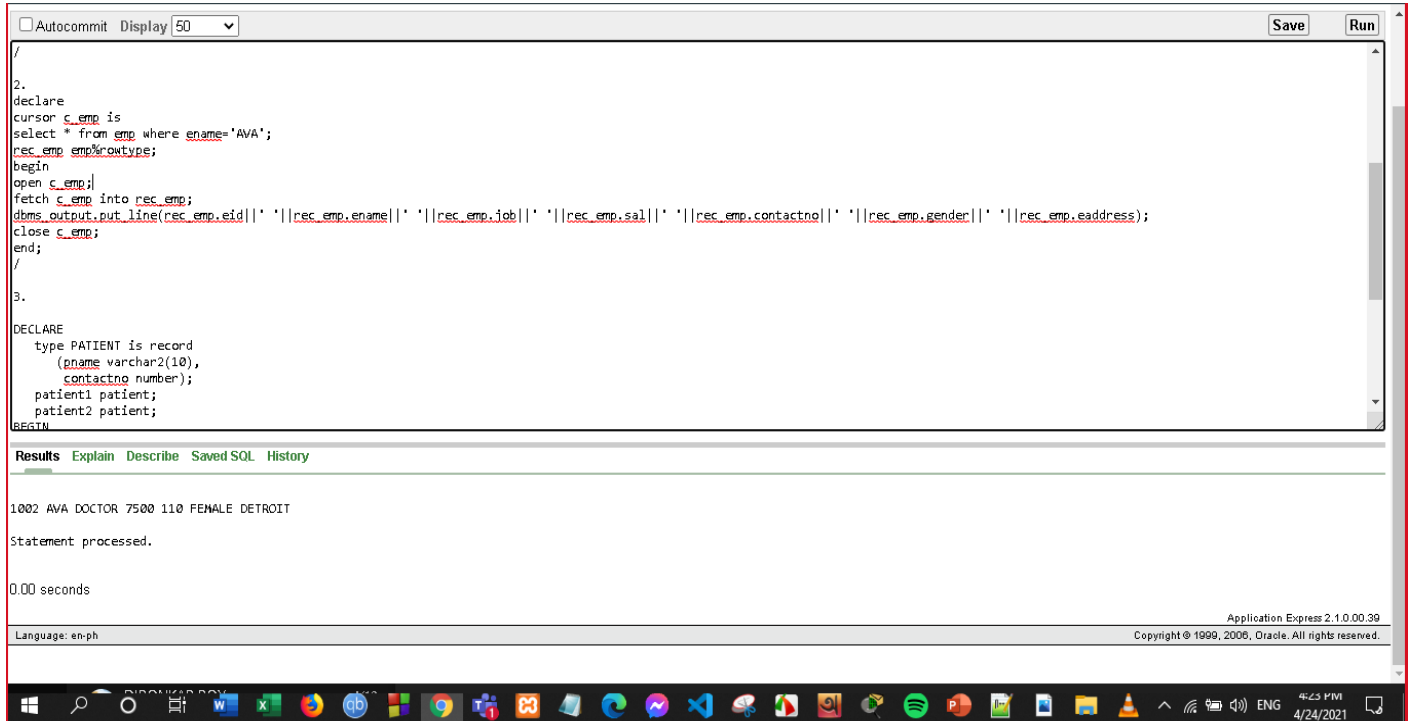


2 . Show the row from emp table where ename='AVA'

```
declare
cursor c_emp is
select * from emp where ename='AVA';
rec_emp emp%rowtype;
begin
open c_emp;
fetch c_emp into rec_emp;
dbms_output.put_line(rec_emp.eid||' '||rec_emp.ename||' '||rec_emp.job||' '||rec_emp.sal||'
'||rec_emp.contactno||' '||rec_emp.gender||' '||rec_emp.eaddress);
close c_emp;
```


end;

/



The screenshot shows the Oracle SQL Developer interface. The top toolbar includes buttons for Autocommit, Display (set to 50), Save, and Run. The main editor contains a PL/SQL script with two parts. The first part is a cursor-based query for employee data. The second part is a DECLARE statement for a PATIENT record type and two variables. The Results pane at the bottom shows a single row of data: 1002 AVA DOCTOR 7500 110 FEMALE DETROIT. Below the results, it states 'Statement processed.' and '0.00 seconds'. The status bar at the bottom indicates 'Language: en-ph' and 'Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.'

```
2.  
declare  
cursor c_emp is  
select * from emp where ename='AVA';  
rec_emp emp%rowtype;  
begin  
open c_emp;  
fetch c_emp into rec_emp;  
dbms_output.put_line(rec_emp.eid||' '||rec_emp.ename||' '||rec_emp.job||' '||rec_emp.sal||' '||rec_emp.contactno||' '||rec_emp.gender||' '||rec_emp.eaddress);  
close c_emp;  
end;  
/  
3.  
DECLARE  
type PATIENT is record  
  (pname varchar2(10),  
   contactno number);  
patient1 patient;  
patient2 patient;  
BEGIN
```

Results Explain Describe Saved SQL History

1002 AVA DOCTOR 7500 110 FEMALE DETROIT

Statement processed.

0.00 seconds

Language: en-ph Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.

3 . Show pname and contactno from patient table.

DECLARE

type PATIENT is record

(pname varchar2(10),

contactno number);

patient1 patient;

patient2 patient;

BEGIN

-- patient 1 specification

patient1.pname := 'ADAM';

```
patient1.contactno := '80080074570';
```

```
-- patient 2 specification
```

```
Patient2.pname := 'Mia';
```

```
Patient2.contactno := '80008970070';
```

```
-- Print patient 1 record
```

```
dbms_output.put_line('Name : ' || patient1.pname);
```

```
dbms_output.put_line('contactno : ' || patient1.contactno);
```

```
-- Print patient 2 record
```

```
dbms_output.put_line('Name : ' || patient2.pname);
```

```
dbms_output.put_line(' contactno : ' || patient2.contactno);
```

```
END;
```

```
/
```

```
DECLARE
  type PATIENT is record
    (pname varchar2(10),
    contactno number);
  patient1 patient;
  patient2 patient;
BEGIN
  -- patient 1 specification
  patient1.pname := 'ADAM';
  patient1.contactno := '80080074570';
  -- patient 2 specification
  Patient2.pname := 'Mia';
  Patient2.contactno := '80008970070';

  -- Print patient 1 record
  dbms_output.put_line('Name : ' || patient1.pname);
  dbms_output.put_line('contactno : ' || patient1.contactno);

  -- Print patient 2 record
  dbms_output.put_line('Name : ' || patient2.pname);
  dbms_output.put_line(' contactno : ' || patient2.contactno);
END;
```

Results Explain Describe Saved SQL History

```
Name : ADAM
contactno : 80080074570
Name : Mia
contactno : 80008970070
```

Statement processed.

0.00 seconds

Language: en-ph

Application Express 2.1.0.00.39
Copyright © 1999, 2006, Oracle. All rights reserved.

(iv)CURSOR

1 . Show ename and sal from emp table using cursor.

```
DECLARE
```

```
    CURSOR C1 IS SELECT ENAME , SAL FROM EMP;
```

```
    VNAME EMP.ENAME%TYPE;
```

```
    VSAL EMP.SAL%TYPE;
```

```
BEGIN
```

```
    OPEN C1;
```

```
LOOP
```

```
    FETCH C1 INTO VNAME, VSAL;
```

```
    EXIT WHEN C1%NOTFOUND;
```

```
    DBMS_OUTPUT.PUT_LINE (VNAME ||' '||VSAL);
```

```
END LOOP;
```

CLOSE C1;

END;

/

```
FETCH C1 INTO VNAME, VSAL;

EXIT WHEN C1%NOTFOUND;

DBMS_OUTPUT.PUT_LINE (VNAME || ' ' || VSAL);

END LOOP;

CLOSE C1;

END;

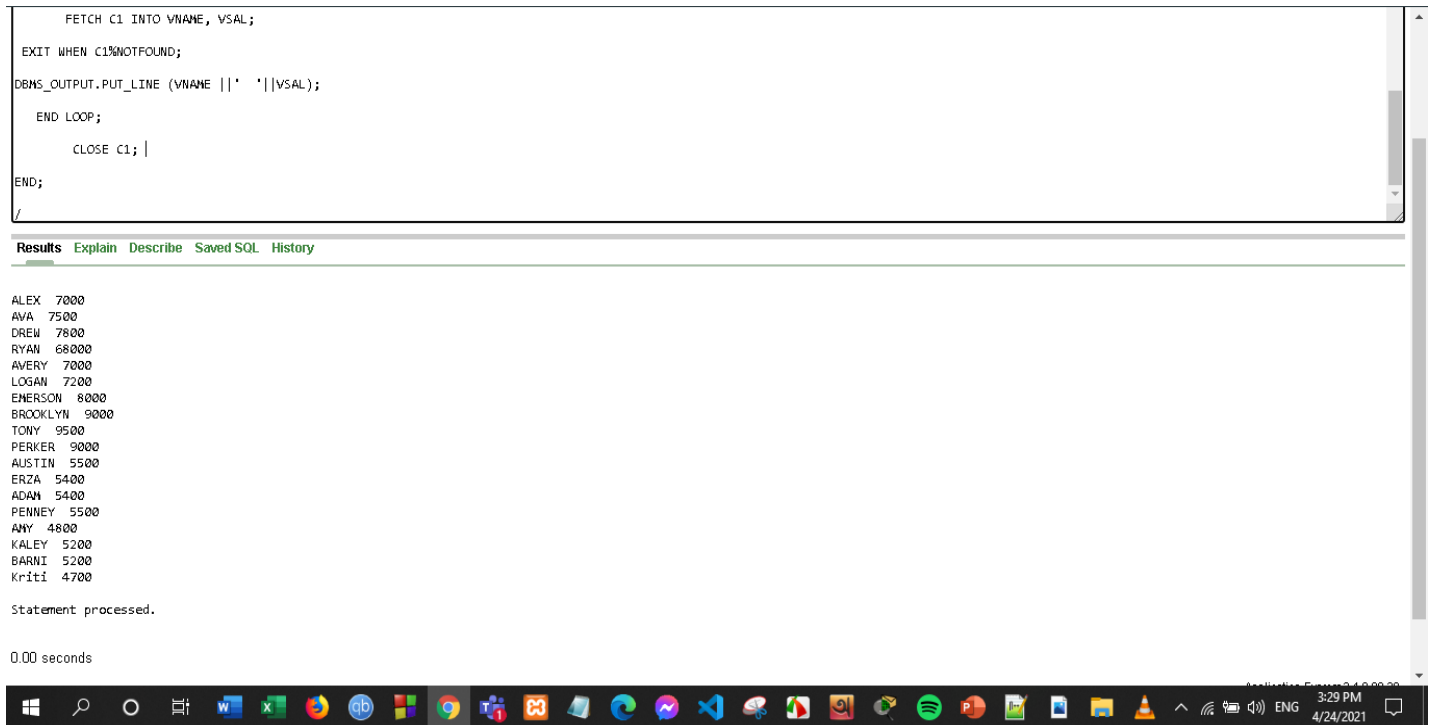
/
```

Results Explain Describe Saved SQL History

ALEX	7000
AVA	7500
DREW	7800
RYAN	68000
AVERY	7000
LOGAN	7200
EMERSON	8000
BROOKLYN	9000
TONY	9500
PERKER	9000
AUSTIN	5500
ERZA	5400
ADAM	5400
PENNEY	5500
AMY	4800
KALEY	5200
BARNI	5200
Kriti	4700

Statement processed.

0.00 seconds



2 . Show pname and contactno from patient table.

DECLARE

CURSOR C2 IS SELECT PNAME , CONTACTNO FROM PATIENT;

QNAME PATIENT.PNAME%TYPE;

QCONTACTNO PATIENT.CONTACTNO%TYPE;

BEGIN

OPEN C2;

LOOP

FETCH C2 INTO QNAME, QCONTACTNO ;

EXIT WHEN C2%NOTFOUND;

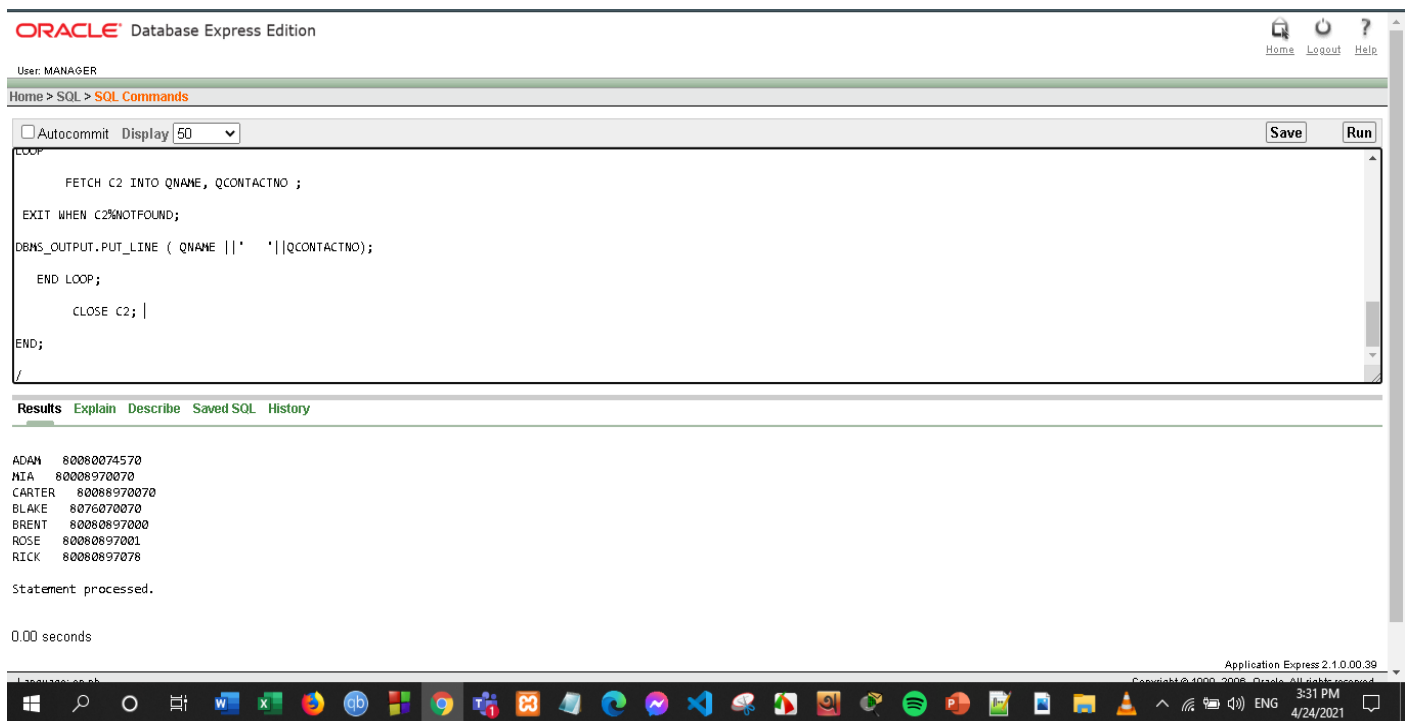
DBMS_OUTPUT.PUT_LINE (QNAME ||' '||QCONTACTNO);

END LOOP;

CLOSE C2;

END;

/



3 . Delete using cursor.

DECLARE

XYZ EMP.EID%TYPE;

BEGIN

XYZ := 1001;

DELETE FROM EMP WHERE EID=XYZ;

IF SQL%FOUND THEN

```
DBMS_OUTPUT.PUT_LINE ('RECORD DELETED');
```

```
ELSE
```

```
DBMS_OUTPUT.PUT_LINE ('NO EMPLOYEE AVAILABLE FOR THIS ID');
```

```
END IF;
```

```
COMMIT;
```

```
END;
```

```
/
```

The screenshot displays the Oracle Database Express Edition interface. At the top, the title bar reads "ORACLE Database Express Edition". Below it, the user is identified as "MANAGER". The breadcrumb navigation shows "Home > SQL > SQL Commands". The main text area contains the following SQL script:

```
☐ Autocommit  Display 50 [Save] [Run]
IF SQL%FOUND THEN
    DBMS_OUTPUT.PUT_LINE ('RECORD DELETED');
ELSE
    DBMS_OUTPUT.PUT_LINE ('NO EMPLOYEE AVAILABLE FOR THIS ID');
END IF;
COMMIT;
END;
/
```

Below the script, the "Results" tab is active, showing the output of the execution:

```
RECORD DELETED
1 row(s) deleted.
0.00 seconds
```

The status bar at the bottom indicates "Language: emph" and "Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved." The Windows taskbar is visible at the very bottom of the image.

(v)TRIGGER

1 . Create and Fire trigger on emp table while Insert and update

```
CREATE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON EMP
FOR EACH ROW
WHEN (NEW.EMPNO > 0)
DECLARE
    sal_diff number;
BEGIN
    sal_diff := :NEW.SAL - :OLD.SAL;
    dbms_output.put_line('Old salary: ' || :OLD.SAL);
    dbms_output.put_line('New salary: ' || :NEW.SAL);
    dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/

//INSERTING

INSERT INTO EMP VALUES (7950, 'Kriti', 'NURSE', 7700,'08/07/1982',1500, 600,
30 );

//UPDATING

UPDATE EMP
SET SAL = SAL + 500
WHERE EMPNO = 7950;
```


User: SCOTT

Home > SQL > SQL Commands

☐ Autocommit Display 50

Save

Run

```
//TRIGGERS
GRANT CREATE ANY TRIGGER TO EMP;

CREATE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON EMP
FOR EACH ROW
WHEN (NEW.EMPNO > 0)
DECLARE
    sal_diff number;
BEGIN
    sal_diff := :NEW.SAL - :OLD.SAL;
    dbms_output.put_line('Old salary: ' || :OLD.SAL);
    dbms_output.put_line('New salary: ' || :NEW.SAL);
```

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

Trigger created.

0.52 seconds

Language: en-ph

Application Express 2.1.0.00.39
Copyright © 1999, 2006, Oracle. All rights reserved.

User: SCOTT

Home > SQL > SQL Commands

☐ Autocommit Display 50

Save

Run

```
DECLARE
    sal_diff number;
BEGIN
    sal_diff := :NEW.SAL - :OLD.SAL;
    dbms_output.put_line('Old salary: ' || :OLD.SAL);
    dbms_output.put_line('New salary: ' || :NEW.SAL);
    dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/

//INSERTING

INSERT INTO EMP VALUES (7950, 'Kriti', 'NURSE', 7700, '08/07/1982', 1500, 600, 30 ); |
```

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)Old salary:
New salary: 1500
Salary difference:

1 row(s) inserted.

0.03 seconds

Language: en-ph

Application Express 2.1.0.00.39
Copyright © 1999, 2006, Oracle. All rights reserved.

User: SCOTT

Home > SQL > SQL Commands

☐ Autocommit
 Display 50

Save

Run

```

dbms_output.put_line('Salary difference: || sal_diff');
END;
/

//INSERTING
INSERT INTO EMP VALUES (7950, 'Kriti', 'NURSE', 7700,'08/07/1982',1500, 600, 30 );

//UPDATING
UPDATE EMP
SET SAL = SAL + 500
WHERE EMPNO = 7950;

```

[Results](#)
[Explain](#)
[Describe](#)
[Saved SQL](#)
[History](#)

Old salary: 1500
 New salary: 2000
 Salary difference: 500

1 row(s) updated.

0.00 seconds

Language: en-ph

Application Express 2.1.0.00.39

Copyright © 1999, 2006, Oracle. All rights reserved.

2 . Fire Trigger while inserting new values to dept table

CREATE OR REPLACE TRIGGER dept_added

after INSERT ON dept

FOR EACH ROW

WHEN (NEW.deptno > 0)

BEGIN

dbms_output.put_line('New Department Added');

END;

/

select * from dept;

insert into dept values ('50','TEACHING','KURIL');

The screenshot shows the Oracle Database Express Edition web interface. At the top, it says "ORACLE Database Express Edition". The user is logged in as "SCOTT". The breadcrumb navigation shows "Home > SQL > SQL Commands". Below this, there's a toolbar with "Autocommit" (unchecked), "Display" (set to 50), "Save", and "Run" buttons. The main text area contains the following SQL code:

```
WHEN (NEW.deptno > 0)
BEGIN
    dbms_output.put_line('New Department Added');
END;
/
select * from dept;
insert into dept values ('50','TEACHING','KURIL');
```

Below the code area, there are tabs for "Results", "Explain", "Describe", "Saved SQL", and "History". The "Results" tab is active, showing the output of the execution:

```
New Department Added
1 row(s) inserted.

0.04 seconds
```

At the bottom of the interface, it shows "Language: en-ph" and "Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved." The Windows taskbar is visible at the very bottom of the image.

3 . Fire trigger while finding salary difference of emp table.

CREATE TRIGGER display_salary_changes

BEFORE UPDATE ON EMP

FOR EACH ROW

WHEN (NEW.EMPNO > 0)

DECLARE

sal_diff number;

BEGIN

sal_diff := :NEW.SAL - :OLD.SAL;

dbms_output.put_line('Old salary: ' || :OLD.SAL);

dbms_output.put_line('New salary: ' || :NEW.SAL);

dbms_output.put_line('Salary difference: ' || sal_diff);

END;

/

//UPDATING

UPDATE EMP

SET SAL = SAL + 1000

WHERE EMPNO = 7369;

The screenshot shows the Oracle Database Express Edition web interface. At the top, it says "ORACLE Database Express Edition" and "User: SCOTT". Below this, there's a navigation bar with "Home", "Logout", and "Help". The main area is titled "Home > SQL > SQL Commands". It features a text editor with a "Save" and "Run" button. The SQL commands entered are:

```
drop trigger display_salary_changes;
/
//UPDATING
UPDATE EMP
SET SAL = SAL + 1000
WHERE EMPNO = 7369;
/
```

Below the editor, there's a "Results" tab selected, showing the output of the commands:

```
Old salary: 800
New salary: 1800
Salary difference: 1000
1 row(s) updated.
0.00 seconds
```

At the bottom, there's a status bar with "Language: en-ph" and "Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved." The Windows taskbar is visible at the very bottom.

(vi)PACKAGE

1 . Show ename from emp table serialwise using package.

CREATE OR REPLACE PACKAGE c_package AS

PROCEDURE addEmp(c_eid emp.eid%type,

c_ename emp.ename%type,

```
c_job emp.job%type,  
c_sal emp.sal%type,  
c_contactno emp.contactno%type,  
c_gender emp.gender%type,  
c_eaddress emp.eaddress%type);
```

```
PROCEDURE delEmp(c_eid emp.eid%TYPE);
```

```
PROCEDURE listEmp;
```

```
END c_package;
```

```
/
```

```
//NEW PACKAGE BODY
```

```
CREATE OR REPLACE PACKAGE BODY c_package AS
```

```
PROCEDURE addEmp(c_eid emp.eid%type,
```

```
    c_ename emp.ename%type,  
    c_job emp.job%type,  
    c_sal emp.sal%type,  
    c_contactno emp.contactno%type,  
    c_gender emp.gender%type,
```

```
c_eaddress emp.eaddress%type)
IS
BEGIN
    INSERT INTO emp (eid,ename,job,sal,contactno,gender,eaddress)
        VALUES(c_eid, c_ename, c_job,c_sal,c_contactno,c_gender, c_eaddress);
END addEmp;
```

```
PROCEDURE delEmp(c_eid emp.eid%type) IS
BEGIN
    DELETE FROM emp
    WHERE eid = c_eid;
END delEmp;
```

```
PROCEDURE listEmp IS
CURSOR c_emp is
    SELECT  ename FROM emp;
TYPE c_list is TABLE OF emp.ename%type;
ename_list c_list := c_list();
counter integer :=0;
BEGIN
    FOR n IN c_emp LOOP
        counter := counter +1;
        ename_list.extend;
        ename_list(counter) := n.ename;
        dbms_output.put_line('Emp(' ||counter|| ')||ename_list(counter));
```

```
END LOOP;  
END listEmp;
```

```
END c_package;
```

```
/
```

```
//Using Package
```

```
DECLARE
```

```
code emp.eid%type:= 21;
```

```
BEGIN
```

```
c_package.addEmp(1021,  
'Rajnish','DOCTOR',7900,111100011,'MALE','CHATTOGRAM');
```

```
c_package.addEmp(1022, 'Subham', 'DOCTOR',8300,1110101,'FEMALE',  
'SYLHET');
```

```
c_package.listEmp;
```

```
c_package.delEmp(code);
```

```
c_package.listEmp;
```

```
END;
```

```
/
```

Home > SQL > SQL Commands

☐ Autocommit Display 100000 Save Run

```
*****NEW PACKAGE*****
select * from EMP;

1.
CREATE OR REPLACE PACKAGE c_package AS
  PROCEDURE addEmp(c_eid emp.eid%type,
    c_ename emp.ename%type,
    c_job emp.job%type,
    c_sal emp.sal%type,
    c_contactno emp.contactno%type,
    c_gender emp.gender%type,
    c_eaddress emp.eaddress%type);


  PROCEDURE delEmp(c_eid emp.eid%TYPE);
  PROCEDURE listEmp;
END c_package;
/
```

Results Explain Describe Saved SQL History

Package created.

0.00 seconds

Language: en-ph Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.



Home > SQL > SQL Commands

☐ Autocommit Display 100000 Save Run

```
//NEW PACKAGE BODY

CREATE OR REPLACE PACKAGE BODY c_package AS
  PROCEDURE addEmp(c_eid emp.eid%type,
    c_ename emp.ename%type,
    c_job emp.job%type,
    c_sal emp.sal%type,
    c_contactno emp.contactno%type,
    c_gender emp.gender%type,
    c_eaddress emp.eaddress%type)
  IS
  BEGIN
    INSERT INTO emp (eid,ename,job,sal,contactno,gender,eaddress)
    VALUES(c_eid, c_ename, c_job,c_sal,c_contactno,c_gender, c_eaddress);
  END addEmp;

  PROCEDURE delEmp(c_eid emp.eid%type) IS
  BEGIN
    DELETE FROM emp
    WHERE eid = c_eid;
  END delEmp;


```

Results Explain Describe Saved SQL History

Package Body created.

0.00 seconds

Language: en-ph Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.




```
Emp (1)Rajnish  
Emp (2)AWA  
Emp (3)DREW  
Emp (4)RYAN  
Emp (5)AVERY  
Emp (6)LOGAN  
Emp (7)ENERSON  
Emp (8)BROCKLYN  
Emp (9)TONY  
Emp (10)PERKER  
Emp (11)AUSTIN  
Emp (12)ERZA  
Emp (13)ADAN  
Emp (14)PENNEY  
Emp (15)ANY  
Emp (16)KALEY  
Emp (17)BARNI  
Emp (18)k:riti  
Emp (19)Rajnish  
Emp (20)Subham  
Emp (21)Subham  
Emp (1)Rajnish  
Emp (2)AWA  
Emp (3)DREW  
Emp (4)RYAN  
Emp (5)AVERY  
Emp (6)LOGAN  
Emp (7)ENERSON  
Emp (8)BROCKLYN  
Emp (9)TONY  
Emp (10)PERKER  
Emp (11)AUSTIN  
Emp (12)ERZA  
Emp (13)ADAN  
Emp (14)PENNEY  
Emp (15)ANY  
Emp (16)KALEY  
Emp (17)BARNI  
Emp (18)k:riti  
Emp (19)Rajnish  
Emp (20)Subham  
Emp (21)Subham
```

2 . Show serialwise pname from patient table using package.

CREATE OR REPLACE PACKAGE c_package AS

```
PROCEDURE addPatient(c_pid patient.pid%type,  
c_pname patient.pname%type,  
c_padmitted patient.padmitted%type,  
c_pdischarged patient.pdischarged%type,  
c_contactno patient.contactno%type,  
c_gender patient.gender%type,  
c_paddress patient.paddress%type,  
c_room_id patient.room_id%type);
```

```
PROCEDURE delPatient(c_pid patient.pid%TYPE);
```

```
PROCEDURE listPatient;
```

```
END c_package;
```

```
/
```

```
//NEW PACKAGE BODY
```

```
CREATE OR REPLACE PACKAGE BODY c_package AS
```

```
    PROCEDURE addPatient(c_pid patient.pid%type,
```

```
        c_pname patient.pname%type,
```

```
        c_padmitted patient.padmitted%type,
```

```
        c_pdischarged patient.pdischarged%type,
```

```
        c_contactno patient.contactno%type,
```

```
        c_gender patient.gender%type,
```

```
        c_paddress patient.paddress%type,
```

```
        c_room_id patient.room_id%type)
```

```
IS
```

```
BEGIN
```

```
    INSERT INTO patient
```

```
(pid,pname,padmitted,pdischarged,contactno,gender,paddress,room_id)
```

```
VALUES(c_pid,c_pname,c_padmitted,c_pdischarged,c_contactno,c_gender,c_paddresses,c_room_id);
```

```
END addPatient;
```

```
PROCEDURE delPatient(c_pid patient.pid%type) IS
```

```
BEGIN
```

```
DELETE FROM patient
```

```
WHERE pid = c_pid;
```

```
END delPatient;
```

```
PROCEDURE listPatient IS
```

```
CURSOR c_patient is
```

```
SELECT pname FROM patient;
```

```
TYPE c_list is TABLE OF patient.pname%type;
```

```
pname_list c_list := c_list();
```

```
counter integer :=0;
```

```
BEGIN
```

```
FOR n IN c_patient LOOP
```

```
counter := counter +1;
```

```
pname_list.extend;
```

```
pname_list(counter) := n.pname;
```

```
dbms_output.put_line('Patient(' ||counter|| ')||pname_list(counter));
```

```
END LOOP;
```

```
END listPatient;
```

```
END c_package;
```

```
/
```

```
//Using Package
```

```
DECLARE
```

```
code patient.pid%type:= 9;
```

```
BEGIN
```

```
c_package.addPatient(8008,  
'Shuprov','12/03/2020','12/03/2021',111100011,'MALE','CHATTOGRAM',104);
```

```
c_package.addPatient(8009, 'Saadi','12/03/2021','12/04/2021',1110101,'FEMALE',  
'SYLHET',105);
```

```
c_package.listPatient;
```

```
c_package.delPatient(code);
```

```
c_package.listPatient;
```

```
END;
```

```
/
```

```

BEGIN
c_package.addPatient(8008, 'Shuprov','12/03/2020','12/03/2021',111100011,'MALE','CHATTOSGRAM',104);
c_package.addPatient(8009, 'Saadi','12/03/2021','12/04/2021',1110101,'FEMALE', 'SYLHET',105);
c_package.listPatient;
c_package.delPatient(code);
c_package.listPatient;
END;
/

```

Results Explain Describe Saved SQL History

```

Patient(1)ADAM
Patient(2)NIA
Patient(3)CARTER
Patient(4)BLAKE
Patient(5)BRENT
Patient(6)ROSE
Patient(7)RICK
Patient(8)Shuprov
Patient(9)Saadi
Patient(1)ADAM
Patient(2)NIA
Patient(3)CARTER
Patient(4)BLAKE
Patient(5)BRENT
Patient(6)ROSE
Patient(7)RICK
Patient(8)Shuprov
Patient(9)Saadi

```

Statement processed.

3 . Show serialwise description of patients from records table using package .

CREATE OR REPLACE PACKAGE c_package AS

```

PROCEDURE addRecords(c_recordno records.recordno%type,
c_description records.description%type,
c_pid records.pid%type);

```

```

PROCEDURE delRecords(c_recordno records.recordno%TYPE);

```

```

PROCEDURE listRecords;

```

```

END c_package;

```

/

//NEW PACKAGE BODY

CREATE OR REPLACE PACKAGE BODY c_package AS

 PROCEDURE addRecords(c_recordno records.recordno%type,
 c_description records.description%type,
 c_pid records.pid%type)

 IS

 BEGIN

 INSERT INTO records (recordno,description,pid)

 VALUES(c_recordno,c_description,c_pid);

 END addRecords;

 PROCEDURE delRecords(c_recordno records.recordno%type) IS

 BEGIN

 DELETE FROM records

 WHERE recordno = c_recordno;

 END delRecords;

 PROCEDURE listRecords IS

 CURSOR c_records is

 SELECT description FROM records;

```

TYPE c_list is TABLE OF records.description%type;
description_list c_list := c_list();
counter integer :=0;
BEGIN
    FOR n IN c_records LOOP
        counter := counter +1;
        description_list.extend;
        description_list(counter) := n.description;
        dbms_output.put_line('Records(' ||counter|| ')'||description_list(counter));
    END LOOP;
END listRecords;

END c_package;
/

```

```
//Using Package
```

```

DECLARE
    code records.recordno%type:= 10;
BEGIN
    c_package.addRecords(109, 'Broken Thigh',8003);
    c_package.addRecords(110, 'Jaundice',8005);
    c_package.listRecords;

```

```
c_package.delRecords(code);
```

```
c_package.listRecords;
```

```
END;
```

```
/
```

```
c_package.delRecords(code);  
c_package.listRecords;  
END;  
/
```

```
select * from records;
```

Results Explain Describe Saved SQL History

```
Records(1)Acute upper respiratory infections  
Records(2)Diabetes Mellitus  
Records(3)Dislocated bone  
Records(4)Acute upper respiratory infections  
Records(5)Osteoarthritis  
Records(6)Infectious diseases  
Records(7)Fever  
Records(8)Broken Bone  
Records(9)Broken Thigh  
Records(10)Jaundice  
Records(1)Acute upper respiratory infections  
Records(2)Diabetes Mellitus  
Records(3)Dislocated bone  
Records(4)Acute upper respiratory infections  
Records(5)Osteoarthritis  
Records(6)Infectious diseases  
Records(7)Fever  
Records(8)Broken Bone  
Records(9)Broken Thigh  
Records(10)Jaundice
```

Statement processed.



15.Conclusion

We have gone through various aspects and areas for completing our project. We have discovered various features while doing this project. We have some exciting features which we are willing to add in the future.