

DELHI TECHNOLOGICAL UNIVERSITY



DATABASE MANAGEMENT SYSTEM MC - 302

SUBMITTED TO:
Mr. ROHIT KUMAR
Ms. NEHA

SUBMITTED BY:
SACHIN DUHAN
2K17/MC/087

DELHI TECHNOLOGICAL UNIVERSITY

VISION

TO BE A WORLD CLASS UNIVERSITY THROUGH EDUCATION, INNOVATION AND
RESEARCH FOR THE SERVICE OF HUMANITY.

MISSION

- ◆ TO ESTABLISH CENTRES OF EXCELLENCE IN EMERGING AREAS OF SCIENCE, ENGINEERING, TECHNOLOGY, MANAGEMENT AND ALLIED AREAS.
- ◆ TO FOSTER AN ECOSYSTEM FOR INCUBATION, PRODUCT DEVELOPMENT, TRANSFER OF TECHNOLOGY AND ENTREPRENEURSHIP.
- ◆ TO CREATE ENVIRONMENT OF COLLABORATION, EXPERIMENTATION, IMAGINATION AND CREATIVITY.
- ◆ TO DEVELOP HUMAN POTENTIAL WITH ANALYTICAL ABILITIES, ETHICS AND INTEGRITY.
- ◆ TO PROVIDE ENVIRONMENT FRIENDLY, REASONABLE AND SUSTAINABLE SOLUTIONS FOR LOCAL AND GLOBAL NEEDS.

DEPARTMENT OF APPLIED MATHEMATICS

VISION

TO EMERGE AS A CENTRE OF EXCELLENCE AND EMINENCE BY IMPARTING
FUTURISTIC TECHNICAL EDUCATION WITH SOLID MATHEMATICAL
BACKGROUND IN KEEPING WITH GLOBAL STANDARDS, MAKING OUR STUDENTS
TECHNOLOGICALLY AND MATHEMATICALLY COMPETENT AND ETHICALLY
STRONG SO THAT THEY CAN READILY CONTRIBUTE TO THE RAPID
ADVANCEMENT OF SOCIETY AND MANKIND

MISSION

- ◆ TO ACHIEVE ACADEMIC EXCELLENCE THROUGH INNOVATIVE TEACHING AND LEARNING PRACTICES.
- ◆ TO IMPROVE THE RESEARCH COMPETENCE TO ADDRESS SOCIAL NEEDS.
- ◆ TO INCULCATE A CULTURE THAT SUPPORTS AND REINFORCES ETHICAL, PROFESSIONAL BEHAVIOURS FOR A HARMONIOUS AND PROSPEROUS SOCIETY.
- ◆ STRIVE TO MAKE STUDENTS TO UNDERSTAND, APPRECIATE AND GAIN MATHEMATICAL SKILLS AND DEVELOP LOGIC, SO THAT THEY ARE ABLE TO CONTRIBUTE INTELLIGENTLY IN DECISION MAKING WHICH CHARACTERISES OUR SCIENTIFIC AND TECHNOLOGICAL AGE.

PROGRAMME EDUCATIONAL OUTCOMES

- ◆ TO PREPARE GRADUATES WITH A SOLID FOUNDATION IN ENGINEERING, MATHEMATICAL SCIENCE AND TECHNOLOGY FOR A SUCCESSFUL CAREER IN MATHEMATICS AND COMPUTING / FINANCE / COMPUTER ENGINEERING FIELDS.
- ◆ TO PREPARE GRADUATES TO BECOME EFFECTIVE COLLABORATORS / INNOVATORS, WHO COULD ABLY ADDRESS TOMORROW'S SOCIAL, TECHNICAL AND ENGINEERING CHALLENGES.
- ◆ TO ENRICH GRADUATES WITH INTEGRITY AND ETHICAL VALUES SO THAT THEY BECOME RESPONSIBLE ENGINEERS.

PROGRAMME OUTCOMES

The POs are defined in line with the graduate attributes set by NBA.

- ◆ ENGINEERING KNOWLEDGE: THE GRADUATE OF MATHEMATICS & COMPUTING MUST HAVE AN ABILITY TO APPLY KNOWLEDGE OF MATHEMATICS, BASIC SCIENCE AND COMPUTER SCIENCE TO SOLVE ENGINEERING AND RELATED PROBLEMS.
- ◆ PROBLEM ANALYSIS: AN ABILITY TO IDENTIFY, ANALYZE AND FORMULATE COMPLEX ENGINEERING PROBLEMS TO REACH LOGICAL CONCLUSION.
- ◆ DESIGN/DEVELOPMENT OF SOLUTION: AN ABILITY TO DESIGN AND CONDUCT EXPERIMENTS, ANALYZE AND INTERPRET THE DATA.
- ◆ CONDUCT INVESTIGATIONS OF COMPLEX PROBLEMS: AN ABILITY TO USE RESEARCH BASED KNOWLEDGE AND APPLY RESEARCH METHODS TO PROVIDE VALID CONCLUSION.
- ◆ MODERN TOOL USAGES: AN ABILITY TO CREATE, SELECT AND IMPLEMENT APPROPRIATE TECHNIQUES, SUCH AS ARTIFICIAL INTELLIGENCE, NEURAL NETWORK TO MODEL COMPLEX COMPUTER ENGINEERING ACTIVITY.
- ◆ THE ENGINEER AND SOCIETY: AN ABILITY TO EXPLORE THE IMPACT OF ENGINEERING SOLUTIONS ON THE SOCIETY AND ALSO ON CONTEMPORARY ISSUES ON SOCIETAL AND ENVIRONMENTAL CONTEXT.
- ◆ ENVIRONMENT AND SUSTAINABILITY: AN ABILITY TO DESIGN A FEASIBLE SYSTEM, COMPONENT OR PROCESS WITHOUT VIOLATING NORMS FOR PUBLIC HEALTH AND SAFETY, CULTURAL, SOCIAL AND ENVIRONMENTAL ISSUES.
- ◆ ETHICS: AN ABILITY TO UNDERSTAND AND PRACTICE PROFESSIONAL AND ETHICAL RESPONSIBILITIES. 9. INDIVIDUAL AND TEAM WORKS : AN ABILITY TO FUNCTION EFFECTIVELY AS AN INTEGRAL MEMBER OR A LEADER IN A MULTIDISCIPLINARY TEAM.
- ◆ COMMUNICATION: AN ABILITY TO COMMUNICATE EFFECTIVELY IN BOTH ORAL AND WRITTEN FORM FOR EFFECTIVE TECHNICAL DECISION MAKING, REPORT MAKING AND PRESENTATION.
- ◆ PROJECT MANAGEMENT AND FINANCE: AN ABILITY TO DEMONSTRATE PRINCIPLE OF MANAGEMENT AND APPLY THEM TO SUITABLE PROJECTS.
- ◆ LIFE LONG LEARNING: AN ABILITY TO RECOGNIZE THE NEED FOR AND TO BE READY FOR LIFE LONG LEARNING TO KEEP UPDATED ON TECHNOLOGICAL CHANGES.

INDEX

PRACTICAL NO.	TITLE	PAGE NO.
1	SYNOPSIS AND ER DIAGRAM OF PROJECT	7
2	TO IMPLEMENT DDL STATEMENTS	9
3	TO IMPLEMENT DML STATEMENTS	12
4	TO IMPLEMENT SELECT STATEMENTS	14
5	TO IMPLEMENT NESTED QUERIES	17
6	INTRODUCTION TO PL/SQL	19
7	EXCEPTIONAL HANDLING IN PL/SQL	26
8	PROJECT REPORT	29

Practical 1: Synopsis and ER Diagram

SYNOPSIS

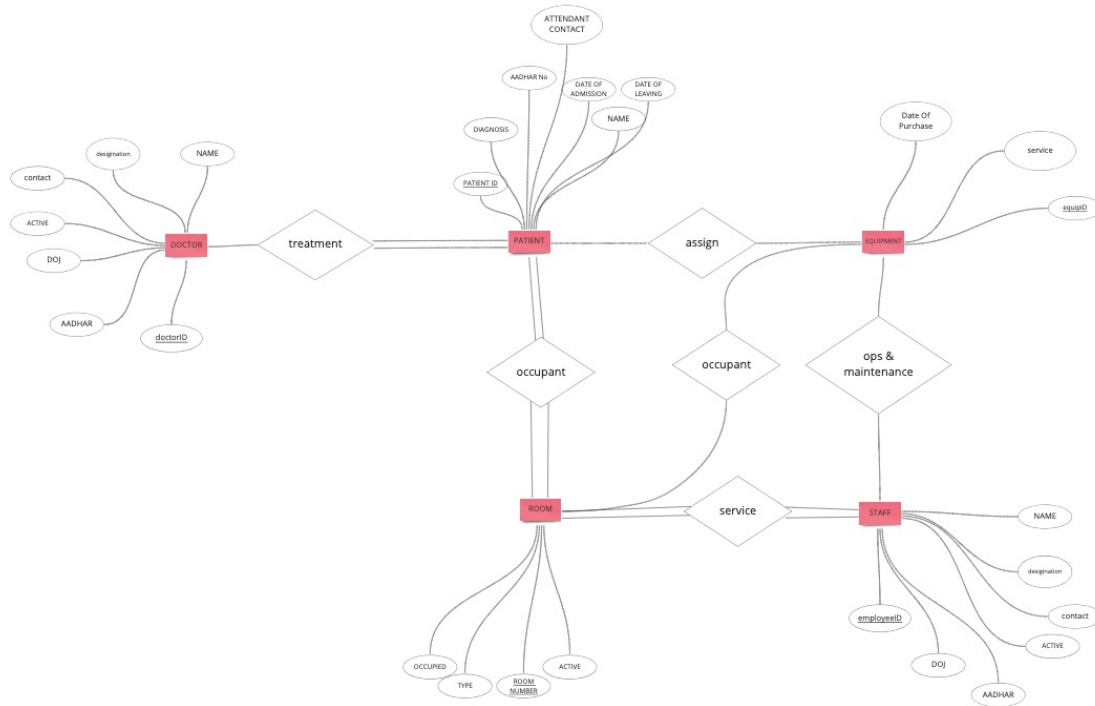
Title - HOSPITAL MANAGEMENT SYSTEM

The DBMS lab project being chosen is intended for the efficient and effective management of a given hospital system. A hospital management system (HMS) is a Database management system that facilitates managing the functioning of the hospital or any medical set up. This system or software will help in making the whole functioning paperless. The hospital records management software keeps a track of all the operations, stores the users' data, performs its query and generates documentation when required.

Such a system can be integrated for storing & processing information for managing all aspects of a hospital's operations such as medical, financial, administrative, legal, and compliance. It may also include electronic health records, business intelligence, and revenue cycle management so some details of the features supported are listed below:-

1. Maintaining a record of all doctors and staff of the hospital, including their rank and role in the organization.
2. Maintaining a record of all rooms, Machines and life-saving equipment in the hospital.
3. Record keeping of all patients visiting the hospital and details of their treatment and status of being admitted.
4. Maintaining a record of which equipment was utilized during the treatment of which patient.
5. Maintaining details of staff assigned to a room or equipment for cleaning and assisting respectively.
6. Details of list Doctors and shift of doctors assigned to a patient and same for hospital staff.

Entity - Relation Diagram



Practical 2: To implement DDL statements

1. CREATE
2. CREATE with constraints
3. ALTER TABLE (with all constraints)
4. DROP TABLE

CREATE TABLE

CREATE TABLE DOCTOR(doctorID int);

```
[mysql> create table doctor(doctorID int);
Query OK, 0 rows affected (0.02 sec)

[mysql> show tables;
+-----+
| Tables_in_dbms |
+-----+
| doctor          |
+-----+
1 row in set (0.01 sec)
```

CREATE with CONSTRAINTS

CREATE TABLE DOCTOR (doctorID int AUTO_INCREMENT NOT NULL PRIMARY KEY, fname varchar(200), lname varchar(200), AADHAR int(12) NOT NULL, designation ENUM('JR','SR','HOD','consultant','surgeon','Trainee'), DOJ DATE, contact int(10), isActive ENUM('0','1'));

```
mysql> clear
mysql> create table DOCTOR ( doctorID int AUTO_INCREMENT NOT NULL PRIMARY KEY, fname varchar(200), lname varchar(200), AADHAR int(12) NOT NULL, designation ENUM('JR','SR','HOD','consultant','surgeon','Trainee'), DOJ DATE, contact int(10), isActive ENUM('0','1'));
Query OK, 0 rows affected (0.02 sec)

mysql> show tables
-> ;
+-----+
| Tables_in_DBMSLAB |
+-----+
| DOCTOR             |
+-----+
1 row in set (0.00 sec)
```

ALTER TABLE

ALTER TABLE DOCTOR ADD COLUMN DEPARTMENT

ENUM('physician','cardiology','pediatrics','neurology','nephrology') NOT NULL;

Field	Type	Null	Key	Default	Extra
doctorID	int(11)	NO	PRI	NULL	auto_increment
fname	varchar(200)	YES		NULL	
lname	varchar(200)	YES		NULL	
AADHAR	int(12)	NO		NULL	
designation	enum('JR','SR','HOD','consultant','surgeon','Trainee')	YES		NULL	
DOJ	date	YES		NULL	
contact	int(10)	YES		NULL	
isActive	enum('0','1')	YES		NULL	
DEPARTMENT	enum('physician','cardiology','pediatrics','neurology','nephrology')	NO		NULL	

9 rows in set (0.00 sec)

ALTER TABLE ROOM ADD FOREIGN KEY (patientID) REFERENCES PATIENT(patientID);

Field	Type	Null	Key	Default	Extra
roomNo	int(11)	NO	PRI	NULL	auto_increment
isActive	enum('0','1')	YES		NULL	
TYPE	varchar(200)	YES		NULL	
OCCUPIED	enum('0','1')	YES		NULL	
patientID	int(11)	NO	MUL	NULL	

ALTER TABLE EQUIPMENT ADD patientID int NOT NULL;

ALTER TABLE EQUIPMENT ADD employeeID int NOT NULL;

ALTER TABLE EQUIPMENT ADD FOREIGN KEY (patientID) REFERENCES
PATIENT(patientID);

ALTER TABLE EQUIPMENT ADD FOREIGN KEY (employeeID) REFERENCES
STAFF(employeeID);

Field	Type	Null	Key	Default	Extra
equipID	int(11)	NO	PRI	NULL	auto_increment
DATE_OF_PURCHASE	date	YES		NULL	
NAME	varchar(200)	NO		NULL	
patientID	int(11)	NO	MUL	NULL	
employeeID	int(11)	NO	MUL	NULL	

ALTER TABLE ROOM ALTER COLUMN TYPE DROP DEFAULT;

```
[mysql> alter table room
-> alter column type drop default;
Query OK, 0 rows affected (0.01 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

ALTER TABLE ROOM RENAME COLUMN PATIENT TO OCCUPANT;

```
[mysql> alter table room rename column patient to occupant;
Query OK, 0 rows affected (0.01 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> describe room;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| roomID     | int(11)       | NO   | PRI | NULL    |       |
| occupant   | int(11)       | YES  | UNI | NULL    |       |
| type       | varchar(50)   | YES  |     | NULL    |       |
| isOccupied | enum('0','1') | YES  |     | NULL    |       |
| isActive   | enum('0','1') | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

DROP TABLE

DROP TABLE DOCTOR;

```
[mysql> show tables;
+-----+
| Tables_in_dbms |
+-----+
| doctor          |
| room            |
+-----+
2 rows in set (0.01 sec)

[mysql> drop table doctor;
Query OK, 0 rows affected (0.01 sec)

[mysql> show tables;
+-----+
| Tables_in_dbms |
+-----+
| room            |
+-----+
1 row in set (0.00 sec)
```

Practical 3: To implement DML statements

1. INSERT
2. UPDATE
3. DELETE
4. TRUNCATE

INSERT

INSERT INTO STAFF VALUES

(1,'SNAME1','NURSE',2222222223,986986986986,'2015-07-01','1');

```
mysql> insert into staff values (1,'sname1','nurse',2222222223,986986986986,'2015-07-01','1');
Query OK, 1 row affected (0.00 sec)

mysql> insert into staff values (2,'sname2','radiologist',2233322223,968965982988,'2013-03-01','1');
Query OK, 1 row affected (0.01 sec)

mysql> insert into ops values(2,2);
Query OK, 1 row affected (0.01 sec)
```

UPDATE

UPDATE EQUIPMENT SET ROOM=1 WHERE EQUIPMENTID=1;

```
[mysql> update equipment
[      -> set room =1
[      -> where equipmentID =1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

DELETE

DELETE FROM TREATMENT WHERE DOCTOR=3;

```
[mysql> delete from treatment where doctor=3;
Query OK, 2 rows affected (0.00 sec)
```

TRUNCATE
TRUNCATE OPS;

```
[mysql> select*from ops;
```

+	-----	+	-----	+
	staff		equipment	
+	-----	+	-----	+
	2		2	
+	-----	+	-----	+

1 row in set (0.00 sec)

```
[mysql> truncate ops;
```

Query OK, 0 rows affected (0.02 sec)

```
[mysql> select*from ops;
```

Empty set (0.01 sec)

Practical 4: To implement SELECT statements

1. Simple
2. WHERE clause + IN / NOT IN
3. Aggregate functions
4. Group By + Having
5. Order By
6. Views
7. In-Built functions (e.g. Date)

SIMPLE SELECT STATEMENT

*SELECT * FROM DOCTOR;*

```
mysql> select*from doctor;
+-----+-----+-----+-----+-----+-----+-----+
| doctorID | name   | aadhar   | DOJ      | contact  | isActive | department | designation |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | Doctor1 | 111112552223 | 2003-10-17 | 4597981891 | 1 | medicine | HOD |
| 2 | doctor2 | 564687416768 | 2006-10-17 | 4943478636 | 1 | Surgery | HOD |
| 3 | doctor3 | 676167465771 | 2010-01-01 | 8944746546 | 1 | surgery | SR |
| 4 | Doctor4 | 575168451565 | 2010-03-20 | 5646574541 | 1 | surgery | SR |
| 5 | doctor5 | 837897984877 | 2011-05-09 | 5475783597 | 1 | pediatrics | HOD |
| 6 | Doctor6 | 798165748189 | 2013-08-04 | 4479856321 | 1 | obs&gynae | HOD |
| 7 | Doctor7 | 754798737798 | 2015-09-07 | 1234894831 | 1 | pediatrics | JR |
| 8 | doctor8 | 135789754779 | 2015-09-01 | 9418764135 | 1 | pediatrics | SR |
| 9 | Doctor9 | 494874984987 | 2016-12-01 | 1614843819 | 1 | obs&gynae | JR |
| 10 | Doctor10 | 687189791897 | 2018-01-01 | 6465168744 | 1 | medicine | JR |
+-----+-----+-----+-----+-----+-----+-----+
10 rows in set (0.01 sec)
```

WHERE CLAUSE + NOT IN

Select roomID from room where roomID not in (select distinct roomID from equipment);

```
mysql> select (roomID) from room where roomID NOT IN (select distinct roomID from equipment);
+-----+
| roomID |
+-----+
| 1 |
| 2 |
| 3 |
| 4 |
| 13 |
| 14 |
+-----+
6 rows in set (0.00 sec)
```

AGGREGATE FUNCTION

Select count(DISTINCT department) from staff;

```
mysql> select count(DISTINCT department) from staff;
+-----+
| count(DISTINCT department) |
+-----+
| 8 |
+-----+
1 row in set (0.00 sec)
```

GROUP BY + HAVING

Select count(doctorID), department from doctor group by department;

```
[mysql> select count(doctorID), department from doctor group by department;
+-----+-----+
| count(doctorID) | department |
+-----+-----+
|                2 | medicine   |
|                3 | Surgery    |
|                3 | pediatrics |
|                2 | obs&gynae  |
+-----+-----+
4 rows in set (0.00 sec)
```

```
[mysql> select count(doctorID), department from doctor group by department having count(doctorID)>2;;
+-----+-----+
| count(doctorID) | department |
+-----+-----+
|                3 | Surgery    |
|                3 | pediatrics |
+-----+-----+
2 rows in set (0.00 sec)
```

ORDER BY

Select doctorID, department, designation from doctor order by department;

```
[mysql> select doctorID, department, designation from doctor order by department;
+-----+-----+-----+
| doctorID | department | designation |
+-----+-----+-----+
|         1 | medicine   | HOD         |
|        10 | medicine   | JR          |
|         6 | obs&gynae  | HOD         |
|         9 | obs&gynae  | JR          |
|         5 | pediatrics | HOD         |
|         7 | pediatrics | JR          |
|         8 | pediatrics | SR          |
|         2 | Surgery    | HOD         |
|         3 | surgery    | SR          |
|         4 | surgery    | SR          |
+-----+-----+-----+
10 rows in set (0.01 sec)
```

VIEWS

Create view HOD as select doctorID, department from doctor where designation='HOD';

```
[mysql> create view HOD as select doctorID, department from doctor where designation='HOD';
Query OK, 0 rows affected (0.01 sec)

[mysql> select * from HOD;
+-----+-----+
| doctorID | department |
+-----+-----+
|         1 | medicine   |
|         2 | Surgery    |
|         5 | pediatrics |
|         6 | obs&gynae  |
+-----+-----+
4 rows in set (0.01 sec)
```

IN-BUILT FUNCTIONS (DATE)

Select DOP from equipment where eqID=1;

Select DATEDIFF(curdate(), (select DOP from equipment where eqID=1));

```
[mysql> select DOP from equipment where eqID=1;
+-----+
| DOP      |
+-----+
| 2003-11-01 |
+-----+
1 row in set (0.00 sec)

[mysql> select datediff(curdate(), (select DOP from equipment where eqID=1));
+-----+
| datediff(curdate(), (select DOP from equipment where eqID=1)) |
+-----+
| 6021 |
+-----+
1 row in set (0.00 sec)
```


Practical 5: TO IMPLEMENT NESTED QUERIES

1. INNER JOIN
2. LEFT JOIN
3. RIGHT JOIN

INNER JOIN

Select department, equipment.name from staff inner join ops using (staffID) inner join equipment using (eqID);

```
mysql> SELECT department,equipment.name from staff INNER JOIN ops USING (staffID) INNER JOIN equipment USING (eqID);
```

department	name
Pathology	microscope
Pathology	microscope
Pathology	microscope
Radiology	Xray
Radiology	MRI
Radiology	CT
Radiology	Xray

7 rows in set (0.00 sec)

LEFT OUTER JOIN

Select staff.staffID, department, eqID from staff left join ops on staff.staffID=ops.staffID;

```
mysql> select staff.staffID, department, eqID from staff left outer join ops on staff.staffID=ops.staffID;
```

staffID	department	eqID
1	Surgery	NULL
2	surgery	NULL
3	obs&gynae	NULL
4	obs&gynae	NULL
5	pediatrics	NULL
6	medicine	NULL
7	obs&gynae	NULL
8	surgery	NULL
9	Surgery	NULL
10	obs&gynae	NULL
11	Pathology	16
12	Pathology	18
13	Pathology	17
14	Radiology	4
15	Radiology	5
16	Radiology	6
17	Radiology	7
18	Radiology	NULL
19	Accounts	NULL
20	housekeeping	NULL
21	Accounts	NULL

21 rows in set (0.00 sec)

RIGHT OUTER JOIN

select staffID, equipment.eqID from ops right outer join join equipment using (eqID);

```
mysql> select staffID,equipment.eqID from ops right outer join join equipment using (eqID);
+-----+-----+
| staffID | eqID |
+-----+-----+
| NULL    | 8    |
| NULL    | 9    |
| NULL    | 10   |
| NULL    | 13   |
| 14      | 4    |
| 17      | 7    |
| 15      | 5    |
| 16      | 6    |
| 11      | 16   |
| 13      | 17   |
| 12      | 18   |
| NULL    | 19   |
| NULL    | 12   |
| NULL    | 2    |
| NULL    | 11   |
| NULL    | 20   |
| NULL    | 3    |
| NULL    | 14   |
| NULL    | 21   |
| NULL    | 1    |
| NULL    | 22   |
| NULL    | 15   |
+-----+-----+
22 rows in set (0.01 sec)
```

Practical 6: INTRODUCTION TO PL/SQL

CREATE A PL/SQL BLOCK AND IMPLEMENT THE FOLLOWING:

1. VARIABLES
2. PROCEDURES
3. FUNCTIONS
4. PACKAGES

VARIABLE

```
/* INTRODUCTION TO PL SQL */

DECLARE
    -- variable declaration
    message varchar2(20) := 'Hello, World!';
BEGIN
    /*
    * PL/SQL executable statement(s)
    */
    dbms_output.put_line(message);
END;
/
```

```
/* OUTPUT */

Hello World

PL/SQL procedure successfully completed.
```

```
/* VARIABLE AND CODE EXAMPLE 2*/

DECLARE
    a integer := 10;
    b integer := 20;
    c integer;
    f real;
BEGIN
    c := a + b;
    dbms_output.put_line('Value of c: ' || c);
    f := 70.0/3.0;
    dbms_output.put_line('Value of f: ' || f);
END;
/
```

```
/* OUTPUT */
Value of c: 30
Value of f: 23.333333333333333333

PL/SQL procedure successfully completed.
PI CONSTANT NUMBER := 3.141592654;
```

```
/*EXAMPLE 3 */

DECLARE
    -- constant declaration
    pi constant number := 3.141592654;
    -- other declarations
    radius number(5,2);
    dia number(5,2);
    circumference number(7, 2);
    area number (10, 2);
BEGIN
    -- processing
    radius := 9.5;
    dia := radius * 2;
    circumference := 2.0 * pi * radius;
    area := pi * radius * radius;
    -- output
    dbms_output.put_line('Radius: ' || radius);
    dbms_output.put_line('Diameter: ' || dia);
    dbms_output.put_line('Circumference: ' || circumference);
    dbms_output.put_line('Area: ' || area);
END;
/
```

```
/* GENERAL QUERY EXAMPLE */

DECLARE
    d_id DOCTORS.id%type := 1;
    d_name DOCTORS.name%type;
    d_addr DOCTORS.address%type;
    d_sal DOCTORS.salary%type;
BEGIN
    SELECT name, address, salary INTO d_name, d_addr, d_sal
    FROM DOCTORS
    WHERE id = d_id;
    dbms_output.put_line('Doctor is ' || d_name || ' from ' || d_addr || ' earns '
END;
/
```

PROCEDURES

```
/*PROCEDURE*/

CREATE OR REPLACE PROCEDURE greetings
AS
BEGIN
    dbms_output.put_line('Hello World!');
END;
/

EXECUTE greetings;
```

```
/*OUTPUT*/
Hello World!

PL/SQL procedure successfully completed.
```

```
/* CODE EXAMPLE 2 */

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:= 23;
    b:= 45;
    findMin(a, b, c);
    dbms_output.put_line('Minimum of (23, 45) : ' || c);
END;
/
```

```
/* OUTPUT */
Minimum of (23, 45) : 23

PL/SQL procedure successfully completed.
```

FUNCTIONS

```
/*FUNCTIONS*/

CREATE OR REPLACE FUNCTION totalDoctors
RETURN number IS
    total number(2) := 0;
BEGIN
    SELECT count(*) into total
    FROM DOCTOR;

    RETURN total;
END;
/
```

```
/* OUTPUT */
Function created.

/* EXAMPLE*/
DECLARE
    c number(2);
BEGIN
    c := totalDoctors();
    dbms_output.put_line('Total no. of Doctors: ' || c);
END;
/
```

```
/* OUTPUT */

Total no. of Doctors: 6

PL/SQL procedure successfully completed.
```

```
/* WRITE A FUNCTION TO COMPUTE THE FACTORIAL OF A NUMBER */

DECLARE
    num number;
    factorial number;

FUNCTION fact(x number)
RETURN number
IS
    f number;
BEGIN
    IF x=0 THEN
        f := 1;
    ELSE
        f := x * fact(x-1);
    END IF;
RETURN f;
END;

BEGIN
    num:= 6;
    factorial := fact(num);
    dbms_output.put_line(' Factorial ' || num || ' is ' || factorial);
END;
/
```

```
/* OUTPUT */
```

```
Factorial 6 is 720
```

```
PL/SQL procedure successfully completed.
```

PACKAGES

```
/*PACKAGES*/

CREATE OR REPLACE PACKAGE BODY dod_sal AS

    PROCEDURE find_sal(d_id DOCTORS.id%TYPE) IS
        d_sal DOCTORS.salary%TYPE;
    BEGIN
        SELECT salary INTO d_sal
        FROM DOCTORS
        WHERE id = d_id;
        dbms_output.put_line('Salary: ' || d_sal);
    END find_sal;
END dod_sal;
/
```

```
/* OUTPUT */
Package body created.
```

```
/* EXAMPLE 1 */

CREATE OR REPLACE PACKAGE c_package AS
    -- Adds a doctor
    PROCEDURE addDoctor(d_id DOCTORS.id%type,
        d_name DOCTORS.No.ame%type,
        d_age DOCTORS.age%type,
        d_addr DOCTORS.address%type,
        d_sal DOCTORS.salary%type);

    -- Removes a doctor
    PROCEDURE delDoctor(d_id DOCTORS.id%TYPE);
    --Lists all DOCTORS
    PROCEDURE listdoctor;

END c_package;
/
```



```

/* CREATING A PACKAGE BODY */

CREATE OR REPLACE PACKAGE BODY c_package AS
  PROCEDURE addDoctor(d_id DOCTORS.id%type,
    d_name DOCTORS.No.ame%type,
    d_age DOCTORS.age%type,
    d_addr DOCTORS.address%type,
    d_sal DOCTORS.salary%type)
  IS
  BEGIN
    INSERT INTO DOCTORS (id,name,age,address,salary)
      VALUES(d_id, d_name, d_age, d_addr, d_sal);
  END addDoctor;

  PROCEDURE delDoctor(d_id DOCTORS.id%type) IS
  BEGIN
    DELETE FROM DOCTORS
      WHERE id = d_id;
  END delDoctor;

  PROCEDURE listdoctor IS
  CURSOR c_DOCTORS is
    SELECT name FROM DOCTORS;
  TYPE d_list is TABLE OF DOCTORS.Name%type;
  name_list d_list := d_list();
  counter integer :=0;
  BEGIN
    FOR n IN c_DOCTORS LOOP
      counter := counter +1;
      name_list.extend;
      name_list(counter) := n.name;
      dbms_output.put_line('doctor(' ||counter|| ')'||name_list(counter));
    END LOOP;
  END listdoctor;

END c_package;
/

```

```

/* USING THE PACKAGE */

DECLARE
  code DOCTORS.id%type:= 8;
BEGIN
  c_package.addDoctor(7, 'Rajnish', 25, 'Chennai', 3500);
  c_package.addDoctor(8, 'Subham', 32, 'Delhi', 7500);
  c_package.delDoctor(code);
  c_package.listdoctor;
END;
/

```

Practical 7: EXCEPTIONAL HANDLING IN PL/SQL

```
DECLARE
    d_id DOCTORS.id%type := 812; /* Docter_id is give here */
    d_name DOCTORS.Name%type;
BEGIN
    SELECT name INTO d_name
    FROM DOCTORS
    WHERE id = d_id;
    DBMS_OUTPUT.PUT_LINE ('Name: ' || d_name);

EXCEPTION
    WHEN no_data_found THEN
        dbms_output.put_line('No such Doctor!');
    WHEN others THEN
        dbms_output.put_line('Error!');
END;
/
```

```
/* OUTPUT */
```

```
No such Doctor!
```

```
PL/SQL procedure successfully completed.
```

```
/* RAISING EXCEPTION IN PL SQL */
```

```
--- defining my expection
```

```
DECLARE
    invalid-id-exception EXCEPTION;
```

```

DECLARE
    d_id DOCTORS.id%type := &cd_id;
    d_name DOCTORS.Name%type;
    -- user defined exception
    invalid-id-exception EXCEPTION;
BEGIN
    IF d_id <= 0 THEN
        RAISE invalid-id-exception;
    ELSE
        SELECT name INTO d_name
        FROM DOCTORS
        WHERE id = d_id;
        DBMS_OUTPUT.PUT_LINE ('Name: ' || d_name);
    END IF;
EXCEPTION
    WHEN invalid-id-exception THEN
        dbms_output.put_line('ID must be greater than zero!');
    WHEN no_data_found THEN
        dbms_output.put_line('No such Doctor!');
    WHEN others THEN
        dbms_output.put_line('Error!');
END;
/

```

/* OUTPUT */

```

Enter value for cd_id: -6
ID must be greater than zero!

```

PL/SQL procedure successfully completed.

```

Enter value for cd_id: 12921
No such Doctor!

```

PL/SQL procedure successfully completed.

```

DECLARE
    d_name DOCTORS.NAME%TYPE;
    d_id DOCTORS.doctor_id%TYPE := 12; /*doctor_id is given here!*/
BEGIN
    -- get the Doctor
    SELECT NAME INTO d_name
    FROM DOCTORS
    WHERE doctor_id > d_id;

    -- show the Doctor name
    dbms_output.put_line('Doctor name is ' || d_name);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        dbms_output.put_line('Doctor ' || d_id || ' does not exist');
    WHEN TOO_MANY_ROWS THEN
        dbms_output.put_line('The database returns more than one Doctor');
END;
/

```

/* OUTPUT */

Doctor 12 does not exist

PL/SQL procedure successfully completed.

~
~

Practical 8: PROJECT REPORT

The aim of this project is to simulate a Hospital Management System in accordance with the following requirements:

1. Maintaining a record of all doctors and staff of the hospital, including their rank and role in the organization.
2. Maintaining a record of all rooms, Machines and life-saving equipment in the hospital.
3. Record keeping of all patients visiting the hospital and details of their treatment and status of being admitted.
4. Maintaining a record of which equipment was utilized during the treatment of which patient.
5. Maintaining details of staff assigned to a room or equipment for cleaning and assisting respectively.
6. Details of list Doctors and shift of doctors assigned to a patient and same for hospital staff.

The database is implemented in SQL technology

Tables

```
[mysql> show tables;
+-----+
| Tables_in_dbms |
+-----+
| doctor          |
| equipment       |
| hod             |
| ops             |
| patient         |
| room           |
| roomoccupation |
| service         |
| staff           |
| supporttreatment |
| treatment       |
+-----+
```

Staff details

```
mysql> select * from staff;
```

staffID	name	aadhar	contact	DOJ	profile	isActive	department
1	Nurse1	143646549844	5645642198	2003-10-17	Nurse	1	Surgery
2	Nurse2	641987487844	6571657657	2006-10-17	Nurse	1	surgery
3	nurse3	657895876479	1654789165	2010-01-01	Nurse	1	obs&gynae
4	nurse4	648968789778	9816543589	2010-03-20	Nurse	1	obs&gynae
5	nurse5	679498187377	4651673489	2011-05-09	Nurse	1	pediatrics
6	nurse6	564891676379	5647561654	2013-08-04	Nurse	1	medicine
7	janitor2	451894897657	1616546513	2015-09-07	janitor	1	housekeeing
8	nurse8	651484567995	4547644946	2015-09-01	Nurse	1	surgery
9	Nurse9	125461357189	4654497987	2016-12-01	Nurse	1	Surgery
10	Nurse10	446461894548	1676574654	2018-01-01	Nurse	1	obs&gynae
11	tech1	189434894844	6848489184	2003-10-17	Pathologist	1	Pathology
12	Tech2	364654841897	4874564789	2003-10-17	Pathologist	1	Pathology
13	tech3	456789123654	3489719876	2004-05-05	Pathologist	1	Pathology
14	Radio1	147852369852	6478489787	2010-06-06	Xray	1	Radiology
15	radio2	159753486159	7654897745	2015-01-01	MRI	1	Radiology
16	radio3	564347216813	5789795489	2016-01-08	CT	1	Radiology
17	radio4	489746518654	4871987984	2017-08-01	Xray	1	Radiology
18	radio5	564315645486	9846598714	2003-01-01	ultrasound	1	Radiology
19	Reception1	148918357894	4897913489	2003-10-17	Receptionist	1	Accounts
20	Janitor1	489489498489	6765498484	2003-10-17	Janitor	1	housekeeping
21	cashier1	654498198489	6518971651	2003-10-17	Cashier	1	Accounts

Doctors details

```
mysql> select * from doctor;
```

doctorID	name	aadhar	DOJ	contact	isActive	department	designation
1	Doctor1	111112552223	2003-10-17	4597981891	1	medicine	HOD
2	doctor2	564687416768	2006-10-17	4943478636	1	Surgery	HOD
3	doctor3	676167465771	2010-01-01	8944746546	1	surgery	SR
4	Doctor4	575168451565	2010-03-20	5646574541	1	surgery	SR
5	doctor5	837897984877	2011-05-09	5475783597	1	pediatrics	HOD
6	Doctor6	798165748189	2013-08-04	4479856321	1	obs&gynae	HOD
7	Doctor7	754798737798	2015-09-07	1234894831	1	pediatrics	JR
8	doctor8	135789754779	2015-09-01	9418764135	1	pediatrics	SR
9	Doctor9	494874984987	2016-12-01	1614843819	1	obs&gynae	JR
10	Doctor10	687189791897	2018-01-01	6465168744	1	medicine	SR

Details of patients' diagnosis and attending doctor

patientID	doctorID	diagnosis
5	10	Influenza
7	1	Tuberculosis
10	3	Hernia
15	2	Desmoid tumor
3	2	Joint replacement
2	4	Traffic accident
4	9	Pregnancy II term
9	9	Pregnancy III term
21	6	Hormonal fluctuation
24	7	Sore throat
26	8	Rhinolith
27	5	Acute separative Otitis Media

Details of equipment in the hospital

eqID	DOP	name	roomID
2	2003-11-02	ECG	11
3	2003-11-03	ECG	12
4	2005-01-01	Xray	6
5	2010-01-01	MRI	7
6	2010-01-01	CT	7
7	2015-01-01	Xray	6
8	2003-10-17	deliveryTable	5
13	2003-11-01	Ventilator	5
14	2003-11-01	Incubator	12
15	2003-11-01	Ventilator	16
16	2003-11-01	microscope	8

Details of room occupancy and equipment assigned to the patient

roomID	patientID	DOA	DOD	eqID
11	6	2018-01-25	2018-01-28	12
10	14	2018-02-07	2018-02-12	14
3	17	2018-03-16	2018-03-16	1
4	30	2019-12-12	2019-12-14	22
11	6	2018-01-25	2018-01-28	12
10	14	2018-02-07	2018-02-12	14
3	17	2018-03-16	2018-03-16	1
4	30	2019-12-12	2019-12-14	22

The Hospital management system is implemented in MySQL using the relational database model.

Operations and functionalities are as discussed in the previous practicals.