PRANVEER SINGH INSTITUTE OF TECHNOLOGY, KANPUR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Odd Semester 2023-24

B. Tech.- Third Year

Semester- V

Lab File

Database Management System

(KCS551)

Submitted To :

Faculty Name :_____

Designation :_____

Submitted By :

Name : Aakarshit Srivastava

Roll No. : 2101641520001

Section : CS-AI-3A

Table of Contents

Department Vision Statement

To be a recognized Department of Computer Science & Engineering that produces versatile computer engineers, capable of adapting to the changing needs of computer and related industry.

Department Mission Statements

The mission of the Department of Computer Science and Engineering is:

i. To provide broad based quality education with knowledge and attitude to succeed in Computer Science & Engineering careers.

ii. To prepare students for emerging trends in computer and related industry.

iii. To develop competence in students by providing them skills and aptitude to foster culture of continuous and lifelong learning.

iv. To develop practicing engineers who investigate research, design, and find workable solutions to complex engineering problems with awareness & concern for society as well as environment.

Program Educational Objectives (PEOs)

i. The graduates will be efficient leading professionals with knowledge of computer science & engineering discipline that enables them to pursue higher education and/or successful careers in various domains.

ii. Graduates will possess capability of designing successful innovative solutions to real life problems that are technically sound, economically viable and socially acceptable.

iii. Graduates will be competent team leaders, effective communicators and capable of working in multidisciplinary teams following ethical values.

iv. The graduates will be capable of adapting to new technologies/tools and constantly upgrading their knowledge and skills with an attitude for lifelong learning

Department Program Outcomes (POs)

The students of Computer Science and Engineering Department will be able:

1. Engineering knowledge: Apply the knowledge of mathematics, science, Computer Science & Engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and Computer Science & Engineering sciences.

3. Design/development of solutions: Design solutions for complex Computer Science & Engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Investigation: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex Computer Science & Engineering activities with an understanding of the limitations.

6. The Engineering and Society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice in the field of Computer Science and Engineering.

7. Environment and sustainability: Understand the impact of the professional Computer Science & Engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the Computer Science & Engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex Computer Science & Engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the Computer Science & Engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Department Program Specific Outcomes (PSOs)

The students will be able to:

1. Use algorithms, data structures/management, software design, concepts of programming languages and computer organization and architecture.

2. Understand the processes that support the delivery and management of information systems within a specific application environment.

Course Outcomes

*Level of Bloom's Taxonomy	Level to be met	*Level of Bloom's Taxonomy	Level to be met
L1: Remember	1	L2: Understand	2
L3: Apply	3	L4: Analyze	4
L5: Evaluate	5	L6: Create	6

CO Number	Course Outcomes
KCS-551.1	Able to learn brief knowledge [**L1: Remember**] about modeling and Data definition language (DDL) operations on table.
KCS-551.2	Able to handle Data Base by applying [**L3: Apply**] data manipulation operations (DML) operations on table.
KCS-551.3	Able to analyze [**L4: Analyze**] normalization of data and uses of cursor trigger and views operations on table.

List of Experiments

Lab No.	Lab Experiment	Corresponding CO
0	Write different syntaxes of DDL, DML, DCL, TCL with example.	**CO1**
1	DDL Statement: Create Command (For Hospital Management System Schema)	**CO1**
2	DDL Statement: ALTER, DROP and TRUNCATE	**CO2**
3	DML Statement: Insert, update, delete and select	**CO2**
4	Use of Operators and Aggregate function (min, max, sum, count and average)	**CO2**
5	Use of Group by, Order by and Having clause	**CO2**
6	Use of JOIN Operations	**CO2**
7	Use of Sub Query	**CO2**
8	Use of Co-Related sub queries	**CO2**
9	PL /SQL – Use of basic PL/SQL	**CO2**
10	PL /SQL – Use of Procedure, cursor and trigger	CO3

INDEX

S No	Lab Experiment	Date of Experiment	Date of Submission	Marks	Faculty Signature
0	Write different syntaxes of DDL, DML, DCL, TCLwith example.				
1	DDL Statement: Create Command (For Hospital Management System Schema)				
2	DDL Statement: ALTER, DROP and TRUNCATE				
3	DML Statement: Insert, update, delete and select				
4	Use of Operators and Aggregate function (min, max, sum, count, and average)				
5	Use of Group by, Order by and Having clause				
6	Use of JOIN Operations				
7	Use of Sub Query				
8	Use of Co-Related sub queries				
9	PL /SQL – Use of basic PL/SQL				
10	PL /SQL – Use of Procedure, cursor, and trigger				

Lab No. :- 0

Objective : -

Write different syntaxes of DDL, DML, DCL, TCLwith example.

Data Definition Language (DDL) :

1. CREATE TABLE: This command is used to create the database Table.

   Syntax:

   ```
   CREATE TABLE TableName (
      Column1 DataType,
      Column2 DataType,
      ...
   );
   ```

   Example:

   ```
   CREATE TABLE Employees (
      EmployeeID INT PRIMARY KEY,
      FirstName VARCHAR(50),
      LastName VARCHAR(50),
      Salary DECIMAL(10, 2)
   );
   ```

2. ALTER TABLE: This is used to alter the existing structure of the database.

   Syntax:

   ```
   ALTER TABLE TableName
   ADD COLUMN NewColumn DataType;
   ```

   Example:

   ```
   ALTER TABLE Employees
   ADD COLUMN Department VARCHAR(50);
   ```

3. DROP TABLE: This command is used to remove objects from the database.

   Syntax:

   DROP TABLE TableName;

   Example:

   DROP TABLE Employees;

4. TRUNCATE : This is used to remove all records from a table, including all spaces allocated for the records are removed.

   Syntax:

   TRUNCATE TABLE TableName;

   Example:

   TRUNCATE TABLE Employees;

Data Manipulation Language (DML) :

1. SELECT : It is used to retrieve data from the database.

   Syntax:

   SELECT Column1, Column2
   FROM TableName
   WHERE Condition;

   Example:

   SELECT FirstName, LastName
   FROM Employees
   WHERE Salary > 50000;

2. INSERT : It is used to insert new data into a table.

   Syntax:

   ```
   INSERT INTO TableName (Column1, Column2, ...)
   VALUES (Value1, Value2, ...);
   ```

   Example:

   ```
   INSERT INTO Employees (FirstName, LastName, Salary)
   VALUES ('John', 'Doe', 60000);
   ```

3. UPDATE : It is used to update existing data within a table.

   Syntax:

   ```
   UPDATE TableName
   SET Column1 = Value1, Column2 = Value2
   WHERE Condition;
   ```

   Example:

   ```
   UPDATE Employees
   SET Salary = 65000
   WHERE LastName = 'Doe';
   ```

4. DELETE : It is used to delete records from a database table.

   Syntax:

   ```
   DELETE FROM TableName
   WHERE Condition;
   ```

   Example:

   ```
   DELETE FROM Employees
   WHERE Salary < 50000;
   ```

Data Control Language (DCL) :

1. GRANT : This command gives users access privileges to the database.

   Syntax:

   GRANT Permission
   ON TableName
   TO User;

   Example:

   GRANT SELECT, INSERT
   ON Employees
   TO Manager;

2. REVOKE : This command withdraws the user's access privileges given by using the GRANT command.

   Syntax:

   REVOKE Permission
   ON TableName
   FROM User;

   Example:

   REVOKE INSERT
   ON Employees
   FROM Manager;

Transaction Control Language (TCL) :

1. COMMIT: Commits a Transaction to make transaction effect permanent

   Syntax:

   COMMIT;

   Example:

   COMMIT;

2. ROLLBACK: Rollbacks a transaction in case of any error occurs.

   Syntax:

   ROLLBACK;

   Example:

   ROLLBACK;

3. SAVEPOINT: Sets a save point within a transaction.

   Syntax:

   SAVEPOINT SavepointName;

   Example:

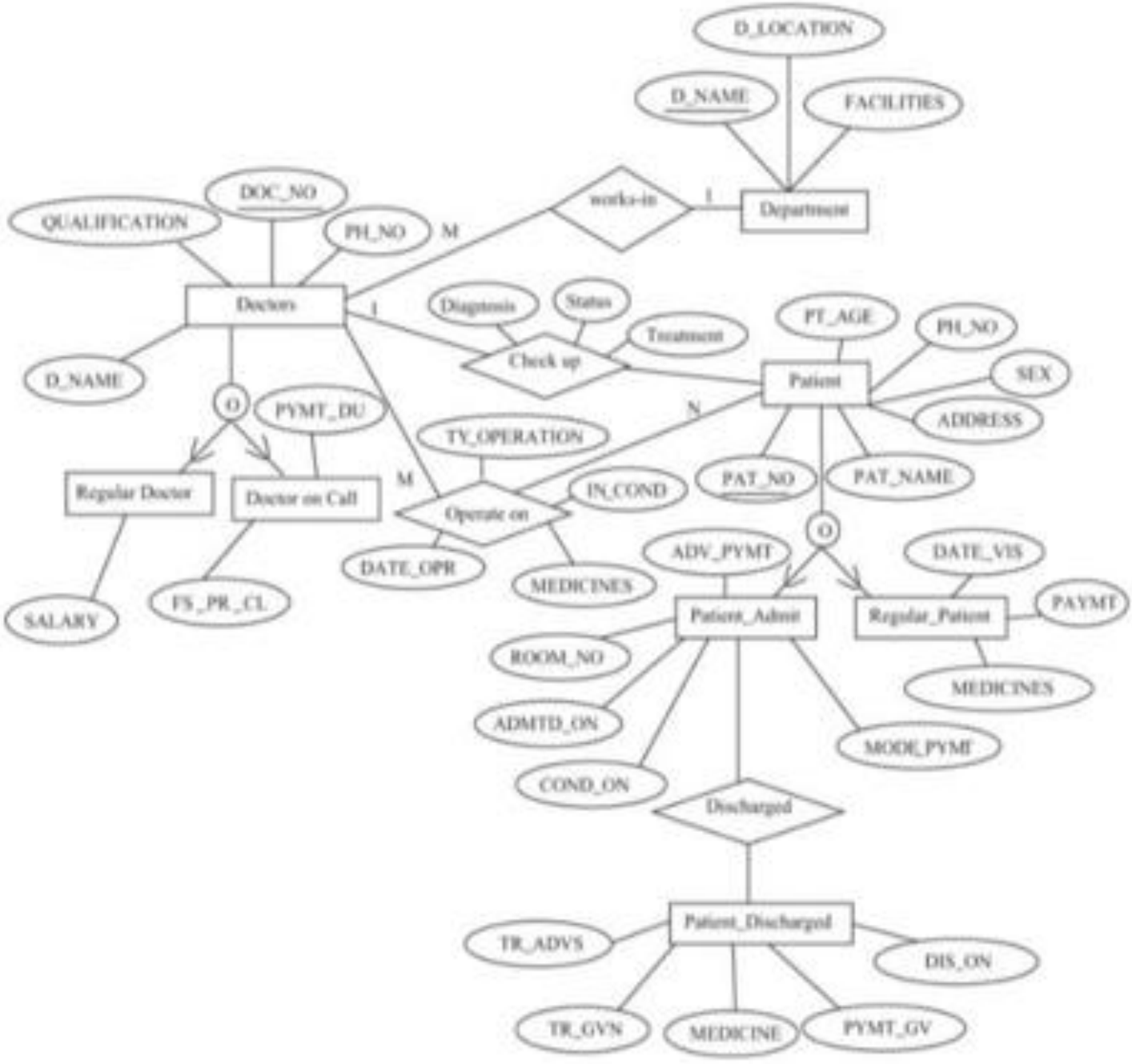
   SAVEPOINT mySavepoint;

Lab No. :- 1

Objective : -

DDL Statement: Create Command (For Hospital Management System Schema)

E-R Diagram



Relational Database Schema for Case Study

The relational database schema for *Hospital Management* database is as follows:

1. DEPARTMENT (D_NAME, D_LOCATION, FACILITIES)
2. ALL_DOCTORS (DOC_NO, DEPARTMENT)
3. DOC_REG(DOC_NO, D_NAME, QUALIFICATION, SALARY, EN_TIME, EX_TIME, ADDRESS, PH_NO, DOJ)
4. DOC_ON_CALL (DOC_NO, D_NAME, QUALIFICATION, FS_PR_CL, PYMT_DU, ADDRESS, PH_NO)
5. PAT_ENTRY (PAT_NO, PAT_NAME, CHKUP_DT, PT_AGE, SEX, RFRG_CSTNT, DIAGNOSIS, RFD, ADDRESS, CITY, PH_NO, DEPARTMENT)
6. PAT_CHKUP (PAT_NO, DOC_NO, DIAGNOSIS, STATUS, TREATMENT)
7. PAT_ADMIT (PAT_NO, ADV_PYMT, MODE_PYMT, ROOM_NO, DEPTNAME, ADMTD_ON, COND_ON, INVSTGTN_DN, TRMT_SDT, ATTDNT_NM)
8. PAT_DIS (PAT_NO, TR_ADVS, TR_GVN, MEDICINES, PYMT_GV, DIS_ON)
9. PAT_REG (PAT_NO, DATE_VIS, CONDITION, TREATMENT, MEDICINES, DOC_NO, PAYMT)
10. PAT_OPR (PAT_NO, DATE_OPR, IN_COND, AFOP_COND, TY_OPERATION, MEDICINES, DOC_NO, OPTH_NO, OTHER_SUG)
11. ROOM_DETAILS (ROOM_NO, TYPE, STATUS, RM_DL_CRG, OTHER_CRG)

Create the tables as per schema: -

CREATE TABLE DEPARTMENT (
D_NAME varchar2(50), D_LOCATION varchar2(20), FACILITIES varchar2(50))

CREATE TABLE DOCTOR (DOC_ID varchar2(5), DEPARTMENT varchar2(50))

CREATE TABLE PATIENT (
P_ID varchar2(5), P_NAME varchar2(50), P_AGE number(2), P_SEX varchar2(1),
P_ADDRESS varchar2(50), P_CITY varchar2(20), P_CONTACT number(10),
P_CHECKUP_DATE date, P_DIAGNOSIS varchar2(50), P_REFDOC varchar2(5),
DEPARTMENT varchar2(50))

CREATE TABLE ROOM_DETAILS (
ROOM_NO number(5), R_TYPE varchar2(10), R_STATUS varchar2(1), P_ID varchar2(5),
DAILY_CHARGE number(5))

CREATE TABLE DOC_REG (
DOC_ID varchar2(50), DOC_NAME varchar2(50), QUALIFICATION varchar2(20),
SALARY number(10), CONTACT_NO number(10), DOJ date)

```
CREATE TABLE DOC_ON_CALL (
DOC_ID varchar2(50), DOC_NAME varchar2(50), QUALIFICATION varchar2(20),
FEE_PER_CALL number(5), CONTACT_NO number(10))

CREATE TABLE PAT_CHECKUP (
P_ID varchar2(5),  DOC_ID varchar2(50), P_DIAGNOSIS varchar2(50),  STATUS varchar2(25),
TREATMENT varchar2(50))

CREATE TABLE PAT_ADMIT (
P_ID varchar2(5),  DOC_ID varchar2(50), P_DIAGNOSIS varchar2(50),  STATUS varchar2(25),
TREATMENT varchar2(50), ADMT_ON date, ROOM_NO number(5))

CREATE TABLE PAT_DISCHRG (
P_ID varchar2(5),  DOC_ID varchar2(50), P_DIAGNOSIS varchar2(50),
TRMNT_GVN varchar2(50),  PYMNT number(10),  DSCHRG_ON date)

CREATE TABLE PAT_REG (
P_ID varchar2(5), DATE_VISIT date,  DIAGNOSIS varchar2(50),  TREATMENT varchar2(50),
MEDICINES_RECMND varchar2(50))

CREATE TABLE PAT_OPR (
P_ID varchar2(5), DOC_ID varchar2(50), DATE_ADMIT date,  DATE_OPR date,
OPRTN_TYPE varchar2(5), OPTH_NO varchar2(5))
```

Lab No. :- 2

Objective : -

DDL Statement: ALTER, DROP and TRUNCATE

1. **Apply the alter command to make the following changes to the tables:**

   a) **Add constraint for Department Table: d_name as primary key.**

      Alter table department add constraint D_name_pk PRIMARY KEY(D_NAME)

   b) **Add constraint for Department and Doctor Table: doc_id as primary key constraint in Doctor Table and department as foreign key constraint referencing d_name from Department Table**.

      Alter table Doctor add constraint Doc_id_pk PRIMARY KEY(DOC_ID)
      Alter table Doctor add constraint FOREIGN KEY(DEPARTMENT) dept_fk+ REFERENCES Department(D_NAME)

   c) **Add constraint for Patient Table: p_id as primary key constraint, department as foreign key constraint referencing d_name from Department Table and p_refdoc as foreign key constraint referencing doc_id from Doctor Table.**

      Alter table Patient add constraint Pat_id_pk PRIMARY KEY(P_ID)
      Alter table Patient add constraint pat_dept_fk FOREIGN KEY(DEPARTMENT) REFERENCES Department(D_NAME)
      Alter table Patient add constraint pat_refdoc_fk FOREIGN KEY(P_REFDOC) REFERENCES DOCTOR(DOC_ID)

   d) **Implement a constraint in Pat_dischrg table to ensure the Payment cannot be Null.**

      Alter table PAT_DISCHRG MODIFY PYMNT number(10) NOT NULL

   e) **Implement a constraint in Room_details table so that Room status is either ‘Y’ or ‘N’.**

      Alter table Room_details ADD CONSTRAINT R_det_status Check(R_STATUS IN('Y','N'))

   f) **Implement a constraint in Pat_admit table to make Admt_on as Not Null.**

      Alter table PAT_ADMIT MODIFY ADMT_ON DATE NOT NULL

   g) **Alter table Doc_on_call table so that fee per call cannot be more than 2000.**

      Alter table DOC_ON_Call ADD CONSTRAINT Fee_PC_ck Check(FEE_PER_CALL<=2000)

   h) **Implement a constraint in Doc_reg table to make date of joining cannot be before 2000.**

      Alter table DOC_REG ADD CONSTRAINT DOJ_ck Check(DOJ>='01/JAN/2000')

i) Implement a constraint on Doc_on_call to make qualification MBBS as default.

ALTER TABLE DOC_ON_CALL MODIFY QUALIFICATION DEFAULT 'MBBS'

j) Modify Patient Details to make Contact Number and Age as Not Null.

Alter table PATIENT MODIFY P_AGE number(2) NOT NULL

Lab No. :- 3

Objective : -

DML Statement: Insert, update, delete and select

1. Perform the following insert commands to fill the records in the created tables.

```
INSERT ALL
into Department values ('Anaesthesia','Floor 1','Critical Care & Pain Management')
into Department values ('Cardiac','Floor 2','Surgery')
into Department values ('Diagnostics','Floor 3','Diagnosis')
into Department values ('ENT','Floor 4','Medicine')
into Department values ('General Surgery','Floor 5','Surgery')
into Department values ('Neuro Sciences','Floor 6','Diagnosis with Surgery')
into Department values ('Physiotherapy','Floor 7','Critical care & Pain Management')
into Department values ('Psychiatry','Floor 8','Diagnosis')
into Department values ('Orthopaedic','Floor 9','Critical Care & Pain Management')
into Department values ('Pulmonary','Floor 10','Reserved')
into Department values ('General','Floor 11','Reserved')
SELECT * from DUAL
```

11 row(s) inserted.

```
INSERT ALL
into Doctor values ('DR01','Anaesthesia')
into Doctor values ('DR02','Anaesthesia')
into Doctor values ('DC01','Anaesthesia')
into Doctor values ('DC02','Anaesthesia')
into Doctor values ('DR03','Cardiac')
into Doctor values ('DR04','Cardiac')
into Doctor values ('DC03','Cardiac')
into Doctor values ('DC04','Cardiac')
into Doctor values ('DR05','Diagnostics')
into Doctor values ('DR06','Diagnostics')
into Doctor values ('DC05','Diagnostics')
into Doctor values ('DC06','Diagnostics')
into Doctor values ('DR07','ENT')
into Doctor values ('DR08','ENT')
into Doctor values ('DC07','ENT')
into Doctor values ('DC08','ENT')
into Doctor values ('DR09','General Surgery')
into Doctor values ('DR10','General Surgery')
into Doctor values ('DC09','General Surgery')
into Doctor values ('DC10','General Surgery')
into Doctor values ('DR11','Neuro Sciences')
into Doctor values ('DR12','Neuro Sciences')
into Doctor values ('DC11','Neuro Sciences')
```

```
into Doctor values ('DC12','Neuro Sciences')
into Doctor values ('DR13','Physiotherapy')
into Doctor values ('DR14','Physiotherapy')
into Doctor values ('DC13','Physiotherapy')
into Doctor values ('DC14','Physiotherapy')
into Doctor values ('DR15','Psychiatry')
into Doctor values ('DR16','Psychiatry')
into Doctor values ('DC15','Psychiatry')
into Doctor values ('DC16','Psychiatry')
into Doctor values ('DR17','Orthopaedic')
into Doctor values ('DR18','Orthopaedic')
into Doctor values ('DC17','Orthopaedic')
into Doctor values ('DC18','Orthopaedic')
into Doctor values ('DR19','Pulmonary')
into Doctor values ('DR20','Pulmonary')
into Doctor values ('DC19','Pulmonary')
into Doctor values ('DC20','Pulmonary')
SELECT * from DUAL
```

40 row(s) inserted.

```
INSERT ALL
into Patient values ('PT001','AAA',35,'M','Civil Lines','Kanpur',9080706051,
'01-JUN-2016','Cardiac Problem','DR03','Cardiac')
into Patient values ('PT002','AAB',40,'F','Kalyanpur','Kanpur',9080706052,'02-JUN-
2016','Physio   Problem','DR13','Physiotherapy')
into Patient values ('PT003','AAC',45,'M','Parade','Kanpur',9080706053,'01-JUN-2016','ENT
Problem','DC13','ENT')
into Patient values ('PT004','AAD',50,'F','Rawatpur','Kanpur',9080706054,'02-JUN-
2016','Diagnostics Problem','DR05','Diagnostics')
into Patient values ('PT005','AAE',55,'M','Harjinder Nagar','Kanpur',9080706055,'03-JUN-
2016','Neuro Problem','DR11','Neuro Sciences')
into Patient values ('PT006','BAA',35,'M','Civil Lines','Lucknow',9080706061,'01-JUN-
2016','Ortho    Problem','DC17','Orthopaedic')
into Patient values ('PT007','BAB',40,'F','Charbagh','Lucknow',9080706062,'02-JUN-
2016','Surgery','DC09','General Surgery')
into Patient values ('PT008','BAC',45,'M','Alambagh','Lucknow',9080706063,'01-JUN-
2016','ENT     Problem','DC13','ENT')
into Patient values ('PT009','BAD',50,'F','Gomti Nagar','Lucknow',9080706064,'02-JUN-
2016','Surgery','DR09','General Surgery')
into Patient values ('PT010','BAE',55,'M','Hazrat Ganj','Lucknow',9080706065,'03-JUN-
2016','Neuro Problem','DR11','Neuro Sciences')
into Patient values ('PT011','CAA',35,'M','Civil Lines','New Delhi',9080706071,'01-JUN-
2016','Ortho Problem','DC17','Orthopaedic')
into Patient values ('PT012','CAB',40,'F','Charbagh','New Delhi',9080706072,'02-JUN-
2016','Surgery','DC09','General Surgery')
into Patient values ('PT013','CAC',45,'M','Alambagh','New Delhi',9080706073,'01-JUN-
2016','ENT     Problem','DC13','ENT')
into Patient values ('PT014','DAD',50,'F','Gomti Nagar','New Delhi',9080706074,'02-JUN-
2016','Surgery','DR09','General Surgery')
```

```
into Patient values ('PT015','DAE',55,'M','Hazrat Ganj','New Delhi',9080706075,'03-JUN-
2016','Neuro Problem','DR11','Neuro Sciences')
SELECT * from DUAL
```

15 row(s) inserted.

```
INSERT ALL
into ROOM_DETAILS values (101,'P AC','N','PT001',5000)
into ROOM_DETAILS values (102,'P AC','N','PT003',5000)
into ROOM_DETAILS values (103,'P AC','N','PT010',5000)
into ROOM_DETAILS values (104,'P AC','N','PT019',5000)
into ROOM_DETAILS values (105,'P AC','N','PT020',5000)
into ROOM_DETAILS values (201,'P NON-AC','N','PT007',3000)
into ROOM_DETAILS values (202,'P NON-AC','N','PT005',3000)
into ROOM_DETAILS values (203,'P NON-AC','Y','',3000)
into ROOM_DETAILS values (204,'P NON-AC','Y','',3000)
into ROOM_DETAILS values (205,'P NON-AC','Y','',3000)
into ROOM_DETAILS values (301,'G AC','N','PT009',3000)
into ROOM_DETAILS values (302,'G AC','N','PT012',3000)
into ROOM_DETAILS values (303,'G AC','N','PT014',3000)
into ROOM_DETAILS values (304,'G AC','Y','',3000)
into ROOM_DETAILS values (305,'G AC','Y','',3000)
into ROOM_DETAILS values (401,'G NON-AC','Y','PT011',2000)
into ROOM_DETAILS values (402,'G NON-AC','Y','PT017',2000)
into ROOM_DETAILS values (403,'G NON-AC','Y','',2000)
into ROOM_DETAILS values (404,'G NON-AC','Y','',2000)
into ROOM_DETAILS values (405,'G NON-AC','Y','',2000)
SELECT * from DUAL
```

20 row(s) inserted.

```
INSERT ALL
into DOC_REG values ('DR01','Dr. A','MD',80000,8090607011,'01-Jan-2004')
into DOC_REG values ('DR02','Dr. B','MD',60000,8090607012,'01-Mar-2007')
into DOC_REG values ('DR03','Dr. C','MBBS',100000,8090607013,'01-May-2008')
into DOC_REG values ('DR04','Dr. D','MBBS',120000,8090607014,'01-Jul-2010')
into DOC_REG values ('DR05','Dr. E','MD',50000,8090607015,'01-Sep-2006')
into DOC_REG values ('DR06','Dr. F','MD',60000,8090607016,'01-Nov-2012')
into DOC_REG values ('DR07','Dr. G','MBBS',80000,8090607017,'01-Feb-2010')
into DOC_REG values ('DR08','Dr. H','MBBS',90000,8090607018,'01-Apr-2013')
into DOC_REG values ('DR09','Dr. I','MBBS',120000,8090607019,'01-Jun-2010')
into DOC_REG values ('DR10','Dr. J','MBBS',150000,8090607020,'01-Aug-2015')
into DOC_REG values ('DR11','Dr. K','MBBS',140000,8090607031,'01-Oct-2012')
into DOC_REG values ('DR12','Dr. L','MD',120000,8090607032,'01-Dec-2010')
into DOC_REG values ('DR13','Dr. M','MBBS',80000,8090607033,'01-Nov-2011')
into DOC_REG values ('DR14','Dr. N','MD',60000,8090607034,'01-Aug-2014')
into DOC_REG values ('DR15','Dr. O','MD',60000,8090607035,'01-Apr-2015')
into DOC_REG values ('DR16','Dr. P','MD',60000,8090607036,'01-Jan-2010')
```

```
into DOC_REG values ('DR17','Dr. Q','MBBS',90000,8090607037,'01-Mar-2007')
into DOC_REG values ('DR18','Dr. R','MD',70000,8090607038,'01-Jun-2009')
into DOC_REG values ('DR19','Dr. S','MD',650000,8090607039,'01-Sep-2010')
into DOC_REG values ('DR20','Dr. T','MD',70000,8090607040,'01-Dec-2012')
SELECT * from DUAL
```

20 row(s) inserted.

```
INSERT ALL
into DOC_ON_CALL values ('DC01','Dr. AT','MD',800,8090607051)
into DOC_ON_CALL values ('DC02','Dr. BS','MD',600,8090607052)
into DOC_ON_CALL values ('DC03','Dr. CR','MBBS',1000,8090607053)
into DOC_ON_CALL values ('DC04','Dr. DQ','MBBS',1200,8090607054)
into DOC_ON_CALL values ('DC05','Dr. EP','MD',500,8090607055)
into DOC_ON_CALL values ('DC06','Dr. FO','MD',600,8090607056)
into DOC_ON_CALL values ('DC07','Dr. GN','MBBS',800,8090607057)
into DOC_ON_CALL values ('DC08','Dr. HM','MBBS',900,8090607058)
into DOC_ON_CALL values ('DC09','Dr. IL','MBBS',1200,8090607059)
into DOC_ON_CALL values ('DC10','Dr. JK','MBBS',1500,8090607060)
into DOC_ON_CALL values ('DC11','Dr. KJ','MBBS',1400,8090607081)
into DOC_ON_CALL values ('DC12','Dr. LI','MD',1200,8090607082)
into DOC_ON_CALL values ('DC13','Dr. MH','MBBS',800,8090607083)
into DOC_ON_CALL values ('DC14','Dr. NG','MD',600,8090607084)
into DOC_ON_CALL values ('DC15','Dr. OF','MD',600,8090607085)
into DOC_ON_CALL values ('DC16','Dr. PE','MD',600,8090607086)
into DOC_ON_CALL values ('DC17','Dr. QD','MBBS',900,8090607087)
into DOC_ON_CALL values ('DC18','Dr. RC','MD',700,8090607088)
into DOC_ON_CALL values ('DC19','Dr. SB','MD',500,8090607089)
into DOC_ON_CALL values ('DC20','Dr. TA','MD',600,8090607090)
SELECT * from DUAL
```

20 row(s) inserted.

```
INSERT ALL
into Pat_checkup values ('PT001','DR03','Cardiac Problem','Referred for Operation','Preliminary')
into Pat_checkup values ('PT002','DR13','Physio Problem','RegularPatient','Regular Treatment')
into Pat_checkup values ('PT003','DC13','ENT Problem','Admitted','Adviced Treatment')
into Pat_checkup values ('PT004','DR05','Diagnostics Problem','RegularPatient','Regular Treatment')
into Pat_checkup values ('PT005','DR11','Neuro Problem','Referred for Operation','Preliminary')
into Pat_checkup values ('PT006','DC17','Ortho Problem','RegularPatient','Regular Treatment')
into Pat_checkup values ('PT007','DC09','Surgery','Referred for Operation','Preliminary')
into Pat_checkup values ('PT008','DC13','ENT Problem','RegularPatient','Regular Treatment')
into Pat_checkup values ('PT009','DR09','Surgery','Referred for Operation','Preliminary')
into Pat_checkup values ('PT010','DR11','Neuro Problem','Admitted','Adviced Treatment')
into Pat_checkup values ('PT011','DR19','Pulmonary Problem','Admitted','Adviced Treatment')
into Pat_checkup values ('PT012','DC09','Surgery','Referred for Operation','Preliminary')
into Pat_checkup values ('PT013','DC13','ENT Problem','RegularPatient','Regular Treatment')
into Pat_checkup values ('PT014','DR09','Surgery','Referred for Operation','Preliminary')
```

```
into Pat_checkup values ('PT015','DR11','Neuro Problem','RegularPatient','Regular Treatment')
into Pat_checkup values ('PT016','DR19','Pulmonary Problem','RegularPatient','Regular Treatment')
into Pat_checkup values ('PT017','DR09','Surgery','Referred for Operation','Preliminary')
into Pat_checkup values ('PT018','DR18','Ortho Problem','RegularPatient','Regular Treatment')
into Pat_checkup values ('PT019','DR10','Surgery','Referred for Operation','Preliminary')
into Pat_checkup values ('PT020','DC12','Neuro Problem','Admitted','Adviced Treatment')
SELECT * from DUAL
```

20 row(s) inserted.

```
INSERT ALL
into Pat_Admit values ('PT001','DR03','Cardiac Problem','Referred for Operation','Preliminary',
'31-May-2016',101)
into Pat_Admit values ('PT003','DC13','ENT Problem','Admitted','Adviced Treatment',
'29-May-2016',102)
into Pat_Admit values ('PT005','DR11','Neuro Problem','Referred for Operation','Preliminary',
'30-May-2016',202)
into Pat_Admit values ('PT007','DC09','Surgery','Referred for Operation','Preliminary',
'30-May-2016',201)
into Pat_Admit values ('PT009','DR09','Surgery','Referred for Operation','Preliminary',
'29-May-2016',301)
into Pat_Admit values ('PT010','DR11','Neuro Problem','Admitted','Adviced Treatment',
'03-Jun-2016',103)
into Pat_Admit values ('PT011','DR19','Pulmonary Problem','Admitted','Adviced Treatment',
'30-May-2016',401)
into Pat_Admit values ('PT012','DC09','Surgery','Referred for Operation','Preliminary',
'31-May-2016',302)
into Pat_Admit values ('PT014','DR09','Surgery','Referred for Operation','Preliminary',
'30-May-2016',303)
into Pat_Admit values ('PT017','DR09','Surgery','Referred for Operation','Preliminary',
'31-May-2016',402)
into Pat_Admit values ('PT019','DR10','Surgery','Referred for Operation','Preliminary',
'31-May-2016',104)
into Pat_Admit values ('PT020','DC12','Neuro Problem','Admitted','Adviced Treatment',
'30-May-2016',105)
SELECT * from DUAL
```

12 row(s) inserted.

```
INSERT ALL
into Pat_Reg values ('PT001','21-Feb-2016','Cardiac Problem','Preliminary','Regular')
into Pat_Reg values ('PT002','31-Mar-2016','Physio Problem','RegularTreatment','Basic')
into Pat_Reg values ('PT003','12-Feb-2016','ENT Problem','AdvicedTreatment','Basic')
into Pat_Reg values ('PT004','05-Apr-2016','Diagnostics Problem','RegularTreatment','Basic')
into Pat_Reg values ('PT005','11-May-2016','Neuro Problem','Preliminary','Regular')
into Pat_Reg values ('PT006','17-May-2016','Ortho Problem','RegularTreatment','Regular')
into Pat_Reg values ('PT007','30-May-2016','Surgery','Preliminary','Cured')
into Pat_Reg values ('PT008','13-Mar-2016','ENT Problem','RegularTreatment','Regular')
```

```
into Pat_Reg values ('PT009','19-May-2016','Surgery','Preliminary','Cured')
into Pat_Reg values ('PT010','11-May-2016','Neuro Problem','AdvicedTreatment','Regular')
into Pat_Reg values ('PT011','19-Feb-2016','Pulmonary Problem','AdvicedTreatment','Basic')
into Pat_Reg values ('PT012','30-May-2016','Surgery','Preliminary','Cured')
into Pat_Reg values ('PT013','13-Apr-2016','ENT Problem','RegularTreatment','Basic')
into Pat_Reg values ('PT014','27-May-2016','Surgery','Preliminary','Cured')
into Pat_Reg values ('PT015','11-Mar-2016','Neuro Problem','RegularTreatment','Regular')
into Pat_Reg values ('PT016','19-Apr-2016','Pulmonary Problem','RegularTreatment','Basic')
into Pat_Reg values ('PT017','29-May-2016','Surgery','Preliminary','Cured')
into Pat_Reg values ('PT018','18-May-2016','Ortho Problem','RegularTreatment','Regular')
into Pat_Reg values ('PT019','31-May-2016','Surgery','Preliminary','Cured')
into Pat_Reg values ('PT020','02-Jun-2016','Neuro Problem','AdvicedTreatment','Regular')
SELECT * from DUAL
```

20 row(s) inserted.

```
INSERT ALL
into Pat_Oprtn values ('PT001','DR03','31-May-16','01-Jun-2016','Major','OT 1')
into Pat_Oprtn values ('PT005','DR11','30-May-16','01-Jun-2016','Major','OT 2')
into Pat_Oprtn values ('PT007','DC09','30-May-16','02-Jun-2016','Minor','OT 1')
into Pat_Oprtn values ('PT009','DR09','29-May-16','31-May-2016','Minor','OT 2')
into Pat_Oprtn values ('PT012','DC09','31-May-16','02-Jun-2016','Minor','OT 1')
into Pat_Oprtn values ('PT014','DR09','30-May-16','03-Jun-2016','Major','OT 2')
into Pat_Oprtn values ('PT017','DR09','31-May-16','03-Jun-2016','Minor','OT 1')
into Pat_Oprtn values ('PT019','DR10','31-May-16','31-May-2016','Major','OT 2')
SELECT * from DUAL
```

8 row(s) inserted.

```
INSERT ALL
into Pat_Dischrg values ('PT001','DR03','Cardiac Problem','Regular',154165,'06-Jun-16')
into Pat_Dischrg values ('PT003','DC13','ENT Problem','Short Term',52304,'04-Jun-16')
into Pat_Dischrg values ('PT005','DR11','Neuro Problem','Long Term',184234,'05-Jun-16')
into Pat_Dischrg values ('PT007','DC09','Surgery','Short Term',41652,'05-Jun-16')
into Pat_Dischrg values ('PT009','DR09','Surgery','Regular',35485,'03-Jun-16')
into Pat_Dischrg values ('PT010','DR11','Neuro Problem','Regular',166168,'07-Jun-16')
into Pat_Dischrg values ('PT011','DR19','Pulmonary Problem','Long Term',55262,'03-Jun-16')
into Pat_Dischrg values ('PT012','DC09','Surgery','Short Term',54316,'04-Jun-16')
into Pat_Dischrg values ('PT014','DR09','Surgery','Long Term',124285,'08-Jun-16')
into Pat_Dischrg values ('PT017','DR09','Surgery','Short Term',34165,'06-Jun-16')
into Pat_Dischrg values ('PT019','DR10','Surgery','Long term',24165,'02-Jun-16')
into Pat_Dischrg values ('PT020','DC12','Neuro Problem','Regular',124784,'04-Jun-16')
SELECT * from DUAL
```

12 row(s) inserted.

2. **Write a query to update a record of Department table from General to Medicine where location of department is at Floor 11.**

   Update Department Set D_name='Medicine' where D_Location='Floor 11'

3. **Write a query to delete a record of Department table where location of department is at Floor 11.**

   DELETE from DEPARTMENT where D_Location='Floor 11'

Lab No. :- 4

Objective : -

Use of Operators and Aggregate function (min, max, sum, count, and average)

1. Calculate the total number of Patients in the Cardiac department.

```
SELECT COUNT(*) as Total_Cardiac_Patients
FROM PATIENT
WHERE DEPARTMENT = 'Cardiac';
```

2. Calculate the average salary of all the regular Doctors.

```
SELECT AVG(SALARY) as Average_Salary
FROM DOC_REG
WHERE QUALIFICATION = 'MD' OR QUALIFICATION = 'MBBS';
```

3. Find the Details of Doctors whose id starts with DC.

```
SELECT *
FROM DOC_REG
WHERE DOC_ID LIKE 'DC%';
```

4. Find details of all doctors who are NOT MBBS.

```
SELECT *
FROM DOC_REG
WHERE QUALIFICATION <> 'MBBS';
```

5. Find the details of patients with age greater than 45.

```
SELECT *
FROM PATIENT
WHERE P_AGE > 45;
```

6. Find the details of Doctor-on-call with the maximum fee.

```
SELECT *
FROM DOC_ON_CALL
WHERE FEE_PER_CALL = (SELECT MAX(FEE_PER_CALL) FROM DOC_ON_CALL);
```

7. Find the details of patients being treated for neuro problem.

```
SELECT *
FROM PATIENT
WHERE P_DIAGNOSIS = 'Neuro Problem';
```

8. Find the total number of patients visiting for general surgery.

```
SELECT COUNT(*) as General_Surgery_Patients
FROM PATIENT
WHERE DEPARTMENT = 'General Surgery';
```

9. Calculate the average payment done by patients admitted to the hospital.

```
SELECT AVG(PYMNT) as Average_Payment
FROM PAT_DISCHRG;
```

10. Find the details of the room allocated in the hospital with a General room without AC or NON-AC.

```
SELECT *
FROM ROOM_DETAILS
WHERE R_TYPE = 'G' AND (R_STATUS = 'N' OR R_STATUS = 'Y');
```

11. Find the names of Doctors-on-call whose fee is above Rs. 1000.

```
SELECT DOC_NAME, FEE_PER_CALL
FROM DOC_ON_CALL
WHERE FEE_PER_CALL > 1000;
```

12. Find the details of the top 5 highest-paid Doctors.

```
SELECT *
FROM DOC_REG
ORDER BY SALARY DESC
FETCH FIRST 5 ROWS ONLY;
```

13. Find details of doctors whose salary is greater than 75000 and less than 100000.

```
SELECT *
FROM DOC_REG
WHERE SALARY > 75000 AND SALARY < 100000;
```

14. Find the list of patients admitted to the hospital between 1st June 2016 to 3rd June 2016.

```
SELECT *
FROM PAT_ADMIT
WHERE ADMT_ON BETWEEN TO_DATE('01-JUN-2016', 'DD-MON-YYYY') AND TO_DATE('03-JUN-2016', 'DD-MON-YYYY');
```

15. Find the second maximum salary of Doctor in the Orthopaedic department.

```
SELECT MAX(SALARY) as Second_Max_Salary
FROM DOC_REG
WHERE QUALIFICATION = 'MD' AND DOC_ID IN (
        SELECT DOC_ID FROM DOCTOR WHERE DEPARTMENT = 'Orthopaedic'
)
```

```
AND SALARY < (SELECT MAX(SALARY) FROM DOC_REG WHERE QUALIFICATION = 'MD'
AND DOC_ID IN (SELECT DOC_ID FROM DOCTOR WHERE DEPARTMENT = 'Orthopaedic'));
```

16. Find the list of rooms vacant having a facility of a Private room with AC.

```
SELECT *
FROM ROOM_DETAILS
WHERE R_TYPE = 'P' AND R_STATUS = 'Y' AND P_ID IS NULL;
```

17. Find the count of patients who are female and are not diagnosed to have surgery.

```
SELECT COUNT(*) as Female_Non_Surgery_Patients
FROM PATIENT
WHERE P_SEX = 'F' AND DEPARTMENT NOT LIKE '%Surgery%';
```

18. Find the count of ENT patients in the Hospital for the year 2016.

```
SELECT COUNT(*) as ENT_Patients_2016
FROM PATIENT
WHERE DEPARTMENT = 'ENT' AND EXTRACT(YEAR FROM P_CHECKUP_DATE) = 2016;
```

19. Find the total count of regular Doctors in the hospital who are in cardiac, ENT, and Orthopaedic department.

```
SELECT COUNT(*) as Total_Regular_Doctors
FROM DOC_REG
WHERE QUALIFICATION = 'MD' AND DOC_ID IN (SELECT DOC_ID FROM DOCTOR WHERE
DEPARTMENT IN ('Cardiac', 'ENT', 'Orthopaedic'));
```

Lab No. :- 5

Objective : -

Use of Group by, Order by and Having clause

1. **Display the details of patient in decreasing order as per their check-up date and in increasing order of age.**

```
SELECT P.P_ID, P.P_NAME, P.P_AGE, P.P_SEX, P.P_ADDRESS, P.P_CITY, P.P_CONTACT,
P.P_CHECKUP_DATE, P.P_DIAGNOSIS, P.P_REFDOC, P.DEPARTMENT
FROM PATIENT P
ORDER BY P.P_CHECKUP_DATE DESC, P.P_AGE;
```

2. **Sort salary of doctor's department-wise.**

```
SELECT DOC_ID, DOC_NAME, SALARY, QUALIFICATION, CONTACT_NO, DOJ
FROM DOC_REG
ORDER BY DEPARTMENT, SALARY;
```

3. **Find the department where the number of patients is more than 3.**

```
SELECT DEPARTMENT, COUNT(*) as Patient_Count
FROM PATIENT
GROUP BY DEPARTMENT
HAVING COUNT(*) > 3;
```

4. **Retrieve the total number of patients admitted to each department.**

```
SELECT DEPARTMENT, COUNT(P_ID) AS TOTAL_PATIENTS
FROM PAT_ADMIT
GROUP BY DEPARTMENT;
```

5. **List all doctors in the 'Orthopaedic' department, ordered alphabetically by their names.**

```
SELECT DOC_ID, DOC_NAME
FROM DOC_REG
WHERE DOC_ID IN (SELECT DOC_ID FROM DOCTOR WHERE DEPARTMENT =
'Orthopaedic')
ORDER BY DOC_NAME;
```

6. **Find departments where the average salary of doctors is greater than 90000.**

```
SELECT DEPARTMENT FROM DOC_REG
GROUP BY DEPARTMENT
HAVING AVG(SALARY) > 90000;
```

7. Count the number of patients in each city.

```
SELECT P_CITY, COUNT(P_ID) AS PATIENT_COUNT
FROM PATIENT
GROUP BY P_CITY;
```

8. Display the patient names and ages in descending order of their ages.

```
SELECT P_NAME, P_AGE
FROM PATIENT
ORDER BY P_AGE DESC;
```

9. Identify departments with more than five doctors.

```
SELECT DEPARTMENT
FROM DOCTOR
GROUP BY DEPARTMENT
HAVING COUNT(DOC_ID) > 5;
```

10. Calculate the average daily charge for each room type.

```
SELECT R_TYPE, AVG(DAILY_CHARGE) AS AVG_DAILY_CHARGE
FROM ROOM_DETAILS
GROUP BY R_TYPE;
```

11. List patients who had checkups, ordered by the checkup date in ascending order.

```
SELECT P_NAME, P_CHECKUP_DATE
FROM PATIENT
WHERE P_CHECKUP_DATE IS NOT NULL
ORDER BY P_CHECKUP_DATE ASC;
```

12. Find doctors who have more than two patients admitted.

```
SELECT DOC_ID, DOC_NAME
FROM DOC_REG
WHERE DOC_ID IN (SELECT DOC_ID FROM PAT_ADMIT GROUP BY DOC_ID HAVING
COUNT(P_ID) > 2);
```

13. Count the number of patients for each diagnosis.

```
SELECT P_DIAGNOSIS, COUNT(P_ID) AS PATIENT_COUNT
FROM PATIENT
GROUP BY P_DIAGNOSIS;
```

14. Retrieve patient details ordered by the diagnosis and then by the treatment.

```
SELECT *
FROM PATIENT
ORDER BY P_DIAGNOSIS, TREATMENT;
```

15. Identify departments with an average patient age greater than 40.

```
SELECT DEPARTMENT
FROM PATIENT
GROUP BY DEPARTMENT
HAVING AVG(P_AGE) > 40;
```

16. Find the total number of patients admitted to each room type.

```
SELECT R_TYPE, COUNT(P_ID) AS PATIENT_COUNT
FROM ROOM_DETAILS
WHERE P_ID IS NOT NULL
GROUP BY R_TYPE;
```

17. List doctors who are on call, ordered by their fee per call in descending order.

```
SELECT DOC_ID, DOC_NAME, FEE_PER_CALL
FROM DOC_ON_CALL
ORDER BY FEE_PER_CALL DESC;
```

18. Identify doctors who have performed surgeries and have more than one patient with a diagnosis of 'Surgery'.

```
SELECT D.DOC_ID, D.DOC_NAME
FROM DOC_REG D
JOIN PAT_CHECKUP PC ON D.DOC_ID = PC.DOC_ID
WHERE PC.TREATMENT = 'Surgery'
GROUP BY D.DOC_ID, D.DOC_NAME
HAVING COUNT(DISTINCT PC.P_ID) > 1;
```

Lab No. :- 6

Objective : -

Use of JOIN Operations

1. **Find Doctor ID with department name of those doctors who are called by hospital.**

```
SELECT DISTINCT DC.DOC_ID, D.DEPARTMENT
FROM DOC_ON_CALL DC
JOIN DOCTOR D ON DC.DOC_ID = D.DOC_ID;
```

2. **Find the name, doctor id and its concerned department of all the doctors.**

```
SELECT D.DOC_ID, D.DEPARTMENT, DR.DOC_NAME
FROM DOCTOR D
JOIN DOC_REG DR ON D.DOC_ID = DR.DOC_ID;
```

3. **Find Patient's Id and Name who has been discharged on 05th June 2016.**

```
SELECT P.P_ID, P.P_NAME
FROM PAT_DISCHRG PD
JOIN PATIENT P ON PD.P_ID = P.P_ID
WHERE PD.DSCHRG_ON = TO_DATE('05-JUN-2016', 'DD-MON-YYYY');
```

4. **List of all the patient details with room number who is an admitted to the hospital for treatment.**

```
SELECT P.*, R.ROOM_NO
FROM PAT_ADMIT PA
JOIN PATIENT P ON PA.P_ID = P.P_ID
JOIN ROOM_DETAILS R ON PA.ROOM_NO = R.ROOM_NO;
```

5. **Find id and name of patient who visit hospital and undergoing for 'Regular Treatment'.**

```
SELECT P.P_ID, P.P_NAME
FROM PAT_REG PR
JOIN PATIENT P ON PR.P_ID = P.P_ID
WHERE PR.TREATMENT = 'RegularTreatment';
```

6. **Give the distinct department name of doctors who are not handling any patients.**

```
SELECT DISTINCT D.DEPARTMENT
FROM DOCTOR D
WHERE D.DOC_ID NOT IN (SELECT DISTINCT DOC_ID FROM PAT_CHECKUP);
```

7. **Give the details of doctors (Doctor id, doctor name) who operated any patient or have done surgery.**

```
SELECT DISTINCT D.DOC_ID, DR.DOC_NAME
FROM DOC_REG DR
JOIN PAT_OPR PO ON DR.DOC_ID = PO.DOC_ID
UNION
SELECT DISTINCT D.DOC_ID, DR.DOC_NAME
FROM DOC_REG DR
JOIN PAT_CHECKUP PC ON DR.DOC_ID = PC.DOC_ID
WHERE PC.TREATMENT = 'Surgery';
```

8. **Find the details of patient who is referred for operation.**

```
SELECT P.*
FROM PAT_CHECKUP PC
JOIN PATIENT P ON PC.P_ID = P.P_ID
WHERE PC.STATUS = 'Referred for Operation';
```

9. **Give the details of patient with concerned department referred.**

```
SELECT P.*, D.DEPARTMENT
FROM PAT_CHECKUP PC
JOIN PATIENT P ON PC.P_ID = P.P_ID
JOIN DOCTOR D ON PC.DOC_ID = D.DOC_ID;
```

10. **Give the patient id and name that is discharged after payment of amount greater than Rs. 80000.**

```
SELECT P.P_ID, P.P_NAME
FROM PAT_DISCHRG PD
JOIN PATIENT P ON PD.P_ID = P.P_ID
WHERE PD.PYMNT > 80000;
```

11. **Details of patient being treated by which doctor of which department.**

```
SELECT P.P_ID, P.P_NAME, P.DIAGNOSIS, D.DOC_ID, D.DOC_NAME, D.DEPARTMENT
FROM PAT_CHECKUP PC
JOIN DOCTOR D ON PC.DOC_ID = D.DOC_ID
JOIN PATIENT P ON PC.P_ID = P.P_ID;
```

12. **Find the patient’s name with treatment prescribed to the regular patients.**

```
SELECT P.P_ID, P.P_NAME, PR.TREATMENT
FROM PAT_REG PR
JOIN PATIENT P ON PR.P_ID = P.P_ID
WHERE PR.STATUS = 'RegularTreatment';
```

13. Find the details of highest paid Regular doctor.

```
SELECT D.DOC_ID, D.DOC_NAME, D.QUALIFICATION, D.SALARY
FROM DOC_REG D
WHERE D.DOC_ID IN (SELECT DOC_ID FROM DOC_ON_CALL)
ORDER BY D.SALARY DESC
FETCH FIRST 1 ROWS ONLY;
```

14. Give the details of operated patient with concerned doctor.

```
SELECT P.P_ID, P.P_NAME, O.DATE_OPR, O.OPRTN_TYPE, O.OPTH_NO, D.DOC_ID,
D.DOC_NAME, D.DEPARTMENT
FROM PAT_OPR O
JOIN PATIENT P ON O.P_ID = P.P_ID
JOIN DOCTOR D ON O.DOC_ID = D.DOC_ID;
```

15. Find the details of treatment offered to patient under Cardiac department.

```
SELECT P.P_ID, P.P_NAME, PC.P_DIAGNOSIS, PC.TREATMENT
FROM PAT_CHECKUP PC
JOIN DOCTOR D ON PC.DOC_ID = D.DOC_ID
JOIN PATIENT P ON PC.P_ID = P.P_ID
WHERE D.DEPARTMENT = 'Cardiac';
```

16. Find detail of all the patients who have either been admitted or referred for operation.

```
SELECT P.P_ID, P.P_NAME, A.ADMT_ON, A.STATUS AS ADMISSION_STATUS,
O.DATE_OPR, O.STATUS AS OPERATION_STATUS
FROM PAT_ADMIT A
FULL OUTER JOIN PAT_OPR O ON A.P_ID = O.P_ID
JOIN PATIENT P ON P.P_ID = COALESCE(A.P_ID, O.P_ID);
```

17. Find detail of patients with the date of operation and by the concerned doctor.

```
SELECT P.P_ID, P.P_NAME, O.DATE_OPR, O.OPRTN_TYPE, O.OPTH_NO, D.DOC_ID,
D.DOC_NAME, D.DEPARTMENT
FROM PAT_OPR O
JOIN PATIENT P ON O.P_ID = P.P_ID
JOIN DOCTOR D ON O.DOC_ID = D.DOC_ID;
```

18. Find the details of room occupied by patient having diagnosed as ‘Neuro Problem’.

```
SELECT R.ROOM_NO, R.R_TYPE, R.R_STATUS, R.P_ID, P.P_NAME
FROM ROOM_DETAILS R
JOIN PATIENT P ON R.P_ID = P.P_ID
JOIN PAT_CHECKUP PC ON P.P_ID = PC.P_ID
WHERE PC.P_DIAGNOSIS = 'Neuro Problem';
```

Lab No. :- 7

Objective : -

Use of Sub Query

1. **Display the name and ID of the regular doctors of each department.**

```
SELECT D_NAME, DOC_ID
FROM Department
WHERE D_NAME IN (
        SELECT DEPARTMENT FROM Doctor
);
```

2. **Display the name and ID of the regular doctors according to the descending order of their date of joining.**

```
SELECT DOC_NAME, DOC_ID
FROM DOC_REG
WHERE DOC_ID IN (
        SELECT DOC_ID  FROM DOC_REG  ORDER BY DOJ DESC
);
```

3. **Find the name and ID of doctors of a particular area of specialization which can be called.**

```
SELECT DOC_NAME, DOC_ID
FROM DOC_ON_CALL
WHERE QUALIFICATION IN (
        SELECT QUALIFICATION  FROM DOC_ON_CALL WHERE QUALIFICATION = 'MD'
);
```

4. **Display the name and ID of doctors as per their department and salary.**

```
SELECT D_NAME,
(SELECT DOC_NAME FROM Doctor WHERE D_NAME = Department.D_NAME) AS
DOC_NAME,
(SELECT DOC_ID FROM Doctor WHERE D_NAME = Department.D_NAME) AS DOC_ID,
(SELECT SALARY FROM DOC_REG WHERE DOC_ID = (SELECT DOC_ID FROM Doctor
WHERE D_NAME = Department.D_NAME)) AS SALARY
FROM Department;
```

5. **Display the name and ID of the regular doctors who have joined the hospital in the 2010.**

```
SELECT DOC_NAME, DOC_ID
FROM DOC_REG  WHERE EXTRACT(YEAR FROM DOJ) = 2010;
```

6. **Find the name and patient ID of those patients that are admitted to the hospital.**

```
SELECT P_NAME, P_ID FROM PAT_ADMIT;
```

7. **Display the name and patient ID of those patients that are treated by 'Dr. I'.**

```
SELECT P_NAME, P_ID
FROM PAT_CHECKUP
WHERE DOC_ID = (SELECT DOC_ID FROM DOC_ON_CALL WHERE DOC_NAME = 'Dr. I');
```

8. **Display the name and patient ID of those patients that are diagnosed a 'ENT Problem'.**

```
SELECT P_NAME, P_ID
FROM PAT_CHECKUP
WHERE P_DIAGNOSIS = 'ENT Problem';
```

9. **Find the name and patient ID of those patients that are regular patient to the hospital.**

```
SELECT P_NAME, P_ID
FROM PAT_REG
WHERE P_ID IN (
        SELECT P_ID FROM PAT_REG WHERE TREATMENT = 'Regular'
);
```

10. **Print the name and patient ID of those patients that are admitted to the hospital on '30-May-2016'.**

```
SELECT P_NAME, P_ID
FROM PAT_ADMIT
WHERE ADMT_ON = TO_DATE('30-May-2016', 'DD-MON-YYYY');
```

11. **Display the name and patient ID of those male patients that have 'Ortho Problem', 'Neuro Problem' and 'ENT Problem'.**

```
SELECT P_NAME, P_ID
FROM PAT_CHECKUP
WHERE P_SEX = 'M'
AND P_DIAGNOSIS IN ('Ortho Problem', 'Neuro Problem', 'ENT Problem')
AND P_ID IN (
        SELECT P_ID FROM PAT_REG WHERE TREATMENT = 'Regular'
);
```

12. **Display the name and patient ID of those patients that are treated by 'Dr. MH'.**

```
SELECT P_NAME, P_ID
FROM PAT_CHECKUP
WHERE DOC_ID = (
        SELECT DOC_ID FROM DOC_ON_CALL WHERE DOC_NAME = 'Dr. MH'
);
```

13. **Print the name and ID of the lowest and highest paid regular doctor.**

```
SELECT MIN(DOC_NAME) AS Lowest_Paid_Doctor, MIN(DOC_ID) AS Lowest_Paid_ID
FROM DOC_REG
WHERE DOC_ID IN (
        SELECT DOC_ID FROM Doctor
)
```

```
UNION
SELECT MAX(DOC_NAME) AS Highest_Paid_Doctor, MAX(DOC_ID) AS Highest_Paid_ID
FROM DOC_REG
WHERE DOC_ID IN (
        SELECT DOC_ID FROM Doctor
);
```

14. Find the name and ID of those patients whose is admitted to hospital having 'Neuro Problem'.

```
SELECT P_NAME, P_ID
FROM PAT_ADMIT
WHERE P_DIAGNOSIS = 'Neuro Problem';
```

15. Print the distinct number of diseases that are diagnosed in order.

```
SELECT DISTINCT P_DIAGNOSIS
FROM PAT_CHECKUP
ORDER BY P_DIAGNOSIS;
```

16. Print the number of patients having 'Ortho Problem' in 2016.

```
SELECT COUNT(P_ID) AS Num_Patients_Ortho_2016
FROM PAT_CHECKUP
WHERE P_DIAGNOSIS = 'Ortho Problem' AND EXTRACT(YEAR FROM
P_CHECKUP_DATE) = 2016;
```

17. Display the names and ID of that patient that have ‘Minor’ type of operation and is treated by 'Dr. I'.

```
SELECT P.P_NAME, P.P_ID
FROM PAT_OPR PO
JOIN PATIENT P ON PO.P_ID = P.P_ID
WHERE PO.OPRTN_TYPE = 'Minor'
  AND PO.DOC_ID = (
        SELECT DOC_ID FROM DOC_ON_CALL WHERE DOC_NAME = 'Dr. I'
);
```

18. Find the ID number of the doctor who has been called up by at least 1 patient.

```
SELECT DISTINCT DOC_ID
FROM DOC_ON_CALL
WHERE DOC_ID IN (
        SELECT DISTINCT DOC_ID FROM PAT_CHECKUP
);
```

19. Find the name and ID number of the doctor who has not been called up by any of the patients.

```
SELECT DOC_NAME, DOC_ID
FROM DOC_ON_CALL
WHERE DOC_ID NOT IN (
        SELECT DISTINCT DOC_ID FROM PAT_CHECKUP
);
```

20. Find the name of the doctor and their contact number who has been called up by at least 1 patient.

```
SELECT DOC_NAME, CONTACT_NO
FROM DOC_ON_CALL
WHERE DOC_ID IN (
        SELECT DISTINCT DOC_ID FROM PAT_CHECKUP
);
```

21. Calculate and print the amount earned by the doctor 'Dr. MH'.

```
SELECT DOC_NAME, FEE_PER_CALL * (SELECT COUNT(*) FROM PAT_CHECKUP
WHERE DOC_ID = (
        SELECT DOC_ID FROM DOC_ON_CALL WHERE DOC_NAME = 'Dr. MH')
        ) AS Earnings
FROM DOC_ON_CALL
WHERE DOC_NAME = 'Dr. MH';
```

22. Display department ID, department name, with the number of doctors associated with them.

```
SELECT D.D_NAME, D.D_LOCATION, (SELECT COUNT(DOC_ID)
FROM DOCTOR
WHERE DEPARTMENT = D.D_NAME) AS Num_Doctors
FROM DEPARTMENT D;
```

23. Find the name and ID number of the patients that have gone 'Major' type of operation in Operation theatre 'OT 2'.

```
SELECT P.P_NAME, P.P_ID
FROM PAT_OPR PO
JOIN PATIENT P ON PO.P_ID = P.P_ID
WHERE PO.OPRTN_TYPE = 'Major' AND PO.OPTH_NO = 'OT 2';
```

24. Find the name and ID number of the patients that are admitted in 'May' and discharged in 'June'.

```
SELECT P.P_NAME, P.P_ID
FROM PAT_ADMIT PA
JOIN PATIENT P ON PA.P_ID = P.P_ID
JOIN PAT_DISCHRG PD ON PA.P_ID = PD.P_ID
WHERE EXTRACT(MONTH FROM PA.ADMT_ON) = 5 AND EXTRACT(MONTH FROM
PD.DSCHRG_ON) = 6;
```

25. Make the salary of those doctors 1.2 * salary who earn less than Rs. 70000.

```
UPDATE DOC_REG
SET SALARY = SALARY * 1.2
WHERE SALARY < 70000;
```

Lab No. :- 8

Objective : -

Use of Co-Related sub queries

1. **Retrieve the names of doctors who belong to the "Cardiac" department.**

```
SELECT DOC_NAME
FROM DOC_REG D
WHERE D.DEPARTMENT = 'Cardiac';
```

2. **Find the qualifications of doctors who are on call and have a fee per call less than 800.**

```
SELECT QUALIFICATION
FROM DOC_ON_CALL DOC
WHERE FEE_PER_CALL < 800 AND DOC.DOC_ID IN (
        SELECT DOC_ID FROM   DOC_REG
        WHERE QUALIFICATION = DOC.QUALIFICATION
);
```

3. **List the patients who have been admitted to a room with a daily charge greater than 4000.**

```
SELECT P_NAME
FROM PATIENT P
WHERE P.P_ID IN (
        SELECT P_ID FROM ROOM_DETAILS WHERE DAILY_CHARGE > 4000
);
```

4. **Get the total number of patients admitted to rooms with AC facilities.**

```
SELECT COUNT(*)
FROM PATIENT P
WHERE P.ROOM_NO IN (
        SELECT ROOM_NO FROM ROOM_DETAILS WHERE R_TYPE LIKE '%AC%'
);
```

5. **Retrieve the departments where the average doctor's salary is more than 70000.**

```
SELECT DEPARTMENT
FROM DOC_REG D
GROUP BY DEPARTMENT
HAVING AVG(SALARY) > 70000;
```

6. **Find the total number of patients who had surgery as their treatment.**

```
SELECT COUNT(*)
FROM PATIENT
WHERE P_ID IN (
        SELECT P_ID FROM PAT_CHECKUP WHERE TREATMENT = 'Surgery'
);
```

7. **List the details of patients who were admitted on or after 30-May-2016.**

```
SELECT * FROM PATIENT
WHERE P_ID IN (
        SELECT P_ID FROM PAT_ADMIT WHERE ADMT_ON >= '30-MAY-2016'
);
```

8. **Retrieve the names of doctors who are on call and have a qualification of "MD."**

```
SELECT DOC_NAME
FROM DOC_ON_CALL
WHERE QUALIFICATION = 'MD';
```

9. **Find the room numbers and types where patients with pulmonary problems were admitted.**

```
SELECT ROOM_NO, R_TYPE
FROM ROOM_DETAILS
WHERE P_ID IN (
        SELECT P_ID FROM PATIENT WHERE P_DIAGNOSIS = 'Pulmonary'
);
```

10. **Get the names of doctors who have recommended medicines for patients with a diagnosis of "RegularTreatment."**

```
SELECT DOC_NAME
FROM DOC_REG
WHERE DOC_ID IN (
        SELECT DOC_ID FROM PAT_CHECKUP
        WHERE TREATMENT = 'RegularTreatment'
);
```

11. **Retrieve the patient names who had surgery and were discharged on or before 05-Jun-2016.**

```
SELECT P_NAME
FROM PATIENT
WHERE P_ID IN (
        SELECT P_ID FROM PAT_CHECKUP WHERE TREATMENT = 'Surgery' AND
        STATUS = 'Discharged' AND P_DIAGNOSIS <= '05-Jun-2016'
);
```

12. **Find the total number of patients who had operations performed by a doctor with the qualification "MD."**

```
SELECT COUNT(DISTINCT P_ID)
FROM PAT_CHECKUP
WHERE DOC_ID IN (
        SELECT DOC_ID FROM DOC_REG WHERE QUALIFICATION = 'MD'
);
```

13. **List the details of patients who were admitted and have a diagnosis of "Neuro Problem."**

```
SELECT *
FROM PATIENT
WHERE P_DIAGNOSIS = 'Neuro Problem' AND P_ID IN (
        SELECT P_ID FROM PAT_ADMIT
);
```

14. Retrieve the names of doctors who have a salary greater than the average salary of doctors in the "Cardiac" department.

```
SELECT DOC_NAME
FROM DOC_REG
WHERE SALARY > (
        SELECT AVG(SALARY) FROM DOC_REG WHERE DEPARTMENT = 'Cardiac'
);
```

15. Get the patient names and room numbers for those who were admitted but have not been discharged yet.

```
SELECT P_NAME, ROOM_NO
FROM PATIENT
WHERE P_ID IN (
        SELECT P_ID FROM PAT_ADMIT WHERE DSCHRG_ON IS NULL
);
```

16. Find the qualifications of doctors who have a fee per call greater than 1000 and are on call for "Pulmonary" cases.

```
SELECT DISTINCT QUALIFICATION
FROM DOC_REG
WHERE DOC_ID IN (
        SELECT DOC_ID FROM DOC_ON_CALL
        WHERE FEE_PER_CALL > 1000
        AND DOC_ID IN (
                SELECT DOC_ID FROM DOC_REG
                WHERE DEPARTMENT = 'Pulmonary'
        )
);
```

17. List the patients who had surgery and were discharged with a payment less than 50000.

```
SELECT P_NAME
FROM PATIENT
WHERE P_ID IN (
        SELECT P_ID FROM PAT_CHECKUP
        WHERE TREATMENT = 'Surgery'
        AND STATUS = 'Discharged' AND PYMNT < 50000
);
```

18. Retrieve the department names where the total number of patients is greater than 5.

```
SELECT DEPARTMENT
FROM DOC_REG
WHERE DEPARTMENT IN (
        SELECT DEPARTMENT  FROM PATIENT
        GROUP BY DEPARTMENT
        HAVING COUNT(*) > 5
);
```

19. Find the patient names and contact numbers who have been recommended physiotherapy.

```
SELECT P_NAME, P_CONTACT
FROM PATIENT
WHERE P_ID IN (
        SELECT P_ID
        FROM PAT_CHECKUP
        WHERE TREATMENT LIKE '%Physiotherapy%'
);
```

20. Get the names of doctors who have recommended medicines for patients admitted to room number 103.

```
SELECT DOC_NAME
FROM DOC_REG
WHERE DOC_ID IN (
        SELECT DOC_ID
        FROM PAT_CHECKUP
        WHERE P_ID IN (
                SELECT P_ID
                FROM PAT_ADMIT
                WHERE ROOM_NO = 103
        )
        AND TREATMENT LIKE '%Medicine%'
);
```

Lab No. :- 9

Objective : -

PL /SQL – Use of basic PL/SQL

1. Write a PL/SQL code to print your name.

```
DECLARE
        my_name VARCHAR2(50);
BEGIN
        my_name := 'ChatGPT';
        DBMS_OUTPUT.PUT_LINE('My name is: ' || my_name);
END;
/
```

2. Write a PL/SQL code to find and average of three numbers.

```
DECLARE
        num1 NUMBER := 10; -- replace with your first number
        num2 NUMBER := 20; -- replace with your second number
        num3 NUMBER := 30; -- replace with your third number
        average NUMBER;

BEGIN
        average := (num1 + num2 + num3) / 3;

DBMS_OUTPUT.PUT_LINE('The average of ' || num1 || ', ' || num2 || ', and ' || num3 || ' is: ' ||
                                                                                        average);
END;
/
```

3. Write a PL/SQL code to find factorial of a given number.

```
DECLARE
        num NUMBER := 5; -- replace with your desired number
        factorial NUMBER := 1;
        i NUMBER;

BEGIN
        IF num < 0 THEN
                DBMS_OUTPUT.PUT_LINE('Factorial is not defined for negative numbers.');
        ELSE
                FOR i IN 1...num LOOP
                        factorial := factorial * i;
                END LOOP;
                DBMS_OUTPUT.PUT_LINE('The factorial of ' || num || ' is: ' || factorial);
        END IF;
END;
/
```

4. Write a PL/SQL code to find simple interest.

```
DECLARE
    principal_amount NUMBER := 1000;
    interest_rate NUMBER := 5;
    time_period NUMBER := 2;
    simple_interest NUMBER;

BEGIN
    simple_interest := (principal_amount * interest_rate * time_period) / 100;
    DBMS_OUTPUT.PUT_LINE('The simple interest is: ' || simple_interest);
END;
/
```

5. Write a PL/SQL code to enter a record in a new table (EMP_New) from the existing table (EMP)

```
DECLARE
    v_emp_id    EMP.emp_id%TYPE;
    v_emp_name  EMP.emp_name%TYPE;
    v_salary    EMP.salary%TYPE;
BEGIN
    v_emp_id := 101;
    v_emp_name := 'John Doe';
    v_salary := 50000;

    INSERT INTO EMP_New (emp_id, emp_name, salary)
    VALUES (v_emp_id, v_emp_name, v_salary);
    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Record inserted successfully into EMP_New table.');
END;
/
```

Lab No. :- 10

Objective : -

PL /SQL – Use of Procedure, cursor, and trigger

Question-1:

1) Create table rad_vals:

a) Add Following Attribute Radius of type number

```
CREATE TABLE rad_vals (
        Radius NUMBER
);
```

b) Insert some records in the above table

```
INSERT INTO rad_vals (Radius) VALUES (5.2);
INSERT INTO rad_vals (Radius) VALUES (8.7);
INSERT INTO rad_vals (Radius) VALUES (12.1);
```

2) Create table Area:

a) Add following Attributes

- **Radius of type number with precision**
- **Area of type number with scale and precision**
- **Perimeter of type number with scale and precision**

```
CREATE TABLE Area (
    Radius NUMBER (10, 2),
    Area NUMBER (10, 2),
     Perimeter NUMBER (10, 2)
);
```

Using Cursors Find the area and perimeter of Circle and insert the value of radius and calculated area and perimeter in the area table

```
-- Declare variables
DECLARE
   v_radius NUMBER(10,2);
   v_area NUMBER(10,2);
   v_perimeter NUMBER(10,2);

-- Declare cursor
CURSOR c_circle_data IS
   SELECT Radius FROM rad_vals;
```

```
BEGIN
   -- Open cursor
   OPEN c_circle_data;

   -- Loop through cursor
   FOR circle_rec IN c_circle_data
   LOOP
      -- Calculate area and perimeter
      v_radius := circle_rec.Radius;
      v_area := 3.141592653589793 * v_radius * v_radius;
      v_perimeter := 2 * 3.141592653589793 * v_radius;

      -- Insert values into the Area table
      INSERT INTO Area (Radius, Area, Perimeter) VALUES (v_radius, v_area, v_perimeter);
   END LOOP;

   -- Close cursor
   CLOSE c_circle_data;
END;
/
```

Question-2:

Suppose we have the following table:

```
Create table mylog(
        who varchar2(30),
        logon_num number
);
```

Write a procedure to keep track of how many times someone logged on to the DB. When running, if user is already in table, increment logon_num. Otherwise, add user into the table.

```
CREATE OR REPLACE PROCEDURE logon_tracking(p_who VARCHAR2) IS
   v_logon_count NUMBER;

BEGIN
   -- Check if the user already exists in the mylog table
   SELECT logon_num INTO v_logon_count
   FROM mylog
   WHERE who = p_who;

   -- If the user exists, increment logon_num; otherwise, insert a new record
   IF v_logon_count IS NOT NULL THEN
      UPDATE mylog
      SET logon_num = v_logon_count + 1
      WHERE who = p_who;


   ELSE
```

```
      INSERT INTO mylog (who, logon_num)
      VALUES (p_who, 1);
   END IF;
   COMMIT; -- Commit the transaction

   DBMS_OUTPUT.PUT_LINE('Logon tracking completed for ' || p_who);
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
      ROLLBACK; -- Rollback the transaction in case of an error
END logon_tracking;
/
```

Question-3:

An HR system has an employee table that holds a row for each employee within the company. Each record in the table has a manager field (mgr) that holds the id for the employee's manager.

Write a trigger so that when a manager record is deleted, the mgr field of that manager's employees is set to NULL.

In other words, implement the SQL statement as:

WHEN AN EMPLOYEE IS DELETED,
UPDATE employee
SET mgr = null
WHERE mgr = employee id of the deleted employee

```
CREATE OR REPLACE TRIGGER update_mgr_on_delete
AFTER DELETE ON employee
FOR EACH ROW
BEGIN
   UPDATE employee
   SET mgr = NULL
   WHERE mgr = :old.employee_id;

   COMMIT; -- Commit the transaction
END update_mgr_on_delete;
/
```