SVRENGINEERINGCOLLEGE

AYYALURUMETTA(V),NANDYAL,KURNOOLDT.A
NDHRAPRADESH–518502

2019–2020

LABORATORYMANUAL

OF

DATABASE MANAGEMENT SYSTEMSLABORATORY

(15A05303)

(R-15 REGULATION)

Preparedby

Mr.Naga mallikarjunareddy
Asst. Professor

For

B.Tech IIYEAR–ISEM.(CSE)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SVR ENGINEERING COLLEGE

(AFFILIATED TOJNTUAANANTHAPURAM-AICITE-INDIA)
AYYALURUMETTA(V),NANDYAL,KURNOOLDT.A
NDHRAPRADESH–518502

LABMANUALCONTENT

DATABASE MANAGEMENT SYSTEMSLABORATORY

(15A05303**)**

Institute Vision & Mission, Department Vision & Mission

1. PO, PEO& PSO Statements.
2. List of Experiments
3. CO-PO Attainment
4. Experiment Code and Outputs

1. Institute Vision & Mission, Department Vision & Mission

Institute Vision:

To produce Competent Engineering Graduates & Managers with a strong base of Technical & Managerial Knowledge and the Complementary Skills needed to be Successful Professional Engineers & Managers.

Institute Mission:

To fulfill the vision by imparting Quality Technical & Management Education to the Aspiring Students, by creating Effective Teaching/Learning Environment and providing State – of the – Art Infrastructure and Resources.

Department Vision:

To produce Industry ready Software Engineers to meet the challenges of 21st Century.

Department Mission:

- Impart core knowledge and necessary skills in Computer Science and Engineering through innovative teaching and learning methodology.
- Inculcate critical thinking, ethics, lifelong learning and creativity needed for industry and society.
- Cultivate the students with all-round competencies, for career, higher education and self-employability.

2. PO, PEO& PSO Statements

PROGRAMME OUTCOMES (POs)

PO-1: Engineering knowledge - Apply the knowledge of mathematics, science, engineering fundamentals of Computer Science& Engineering to solve complex real-life engineering problems related to CSE.

PO-2: Problem analysis - Identify, formulate, review research literature, and analyze complex engineering problems related to CSE and reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO-3: Design/development of solutions - Design solutions for complex engineering problems related to CSE and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, cultural, societal and environmental considerations.

PO-4: Conduct investigations of complex problems - 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.

PO-5: Modern tool usage - Select/Create and apply appropriate techniques, resources and modern engineering and IT tools and technologies for rapidly changing computing needs, including prediction and modeling to complex engineering activities, with an understanding of the limitations.

PO-6: The engineer 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 CSE professional engineering practice.

PO-7: Environment and Sustainability - Understand the impact of the CSE professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of, and need for sustainable development.

PO-8: Ethics - Apply ethical principles and commit to professional ethics and responsibilities and norms of the relevant engineering practices.

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

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

PO-11: Project management and finance - Demonstrate knowledge and understanding of the 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.

PO-12: Life-long learning - Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadcast context of technological changes.

Program Educational Objectives (PEOs):

PEO 1:Graduates will be prepared for analyzing, designing, developing and testing the software solutions and products with creativity and sustainability.

PEO 2: Graduates will be skilled in the use of modern tools for critical problem solvingand analyzing industrial and societal requirements.

PEO 3:Graduates will be prepared with managerial and leadership skills for career and starting up own firms.

Program Specific Outcomes (PSOs):

PSO 1: Develop creative solutions by adapting emerging technologies / tools for real time applications.

PSO 2: Apply the acquired knowledge to develop software solutions and innovative mobile apps for various automation applications

2.1SubjectTimeTable

<table>
<tr><td colspan="9">SVRENGINEERINGCOLLEGE::NANDYAL
DEPARTMENTOFCSE</td></tr>
<tr><td colspan="6">M N Mallikarjuna</td><td colspan="3">II-I</td></tr>
<tr><td>Day/ Time</td><td>9:30 AM</td><td>10:20 AM</td><td>11:30 AM</td><td>12:20 PM-</td><td rowspan="8">LUNCHBREAK</td><td>02:00 PM</td><td>02:50 PM</td><td>03:40 PM</td></tr>
<tr><td></td><td>10:20 AM</td><td>11:10AM</td><td>12:20 PM</td><td>01:10 PM</td><td>02:50 PM</td><td>03:40 PM</td><td>04:30 PM</td></tr>
<tr><td>MON</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>TUE</td><td></td><td></td><td></td><td></td><td colspan="3">DBMS LAB</td></tr>
<tr><td>WED</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>THU</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>FRI</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>SAT</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

LISTOFEXPERIMENTS(SYLLABUS)

JAWAHARLALNEHRUTECHNOLOGICALUNIVERSITYANANTAPUR
B.Tech–II-ISem
15A05303DATABASE MANAGEMENT SYSTEMSLABORATORY
(CommontoCSE&IT)

Course Objectives:

- To create a database and query it using SQL, design forms and generate reports.
- Understand the significance of integrity constraints, referential integrity constraints, triggers, assertions.

Laboratory Experiments

1. Practice session: Students should be allowed to choose appropriate DBMS software, install it, configure it and start working on it. Create sample tables, execute some queries, use SQLPLUS features, use PL/SQL features like cursors on sample database. Students should be permitted to practice appropriate User interface creation tool and Report generation tool.
2. A college consists of number of employees working in different departments. In this context, create two tables employee and department. Employee consists of columns empno, empname, basic, hra, da, deductions, gross, net, date-of-birth. The calculation of hra,da are as per the rules of the college. Initially only empno, empname, basic have valid values. Other values are to be computed and updated later. Department contains deptno, deptname, and description columns. Deptno is the primary key in department table and referential integrity constraint exists between employee and department tables. Perform the following operations on the the database:
   - Create tables department and employee with required constraints.
   - Initially only the few columns (essential) are to be added. Add the remaining columns separately by using appropriate SQL command
   - Basic column should not be null
   - Add constraint that basic should not be less than 5000.
   - Calculate hra,da,gross and net by using PL/SQL program.
   - Whenever salary is updated and its value becomes less than 5000 a trigger has to be raised preventing the operation.
   - The assertions are: hra should not be less than 10% of basic and da should not be less than 50% of basic.
   - The percentage of hra and da are to be stored separately.
   - When the da becomes more than 100%, a message has to be generated and with user permission da has to be merged with basic.
   - Empno should be unique and has to be generated automatically.
   - If the employee is going to retire in a particular month, automatically a message has to be generated.
   - The default value for date-of-birth is 1 jan, 1970.
   - When the employees called daily-wagers are to be added the constraint that salary should be greater than or equal to 5000 should be dropped.
   - Display the information of the employees and departments with description of the fields.
   - Display the average salary of all the departments.
   - Display the average salary department wise.
   - Display the maximum salary of each department and also all departments put together.
   - Commit the changes whenever required and rollback if necessary.
   - Use substitution variables to insert values repeatedly.
   - Assume some of the employees have given wrong information about date-of-birth. Update the corresponding tables to change the value.
   - Find the employees whose salary is between 5000 and 10000 but not exactly 7500.
   - Find the employees whose name contains ‘en’.

- Try to delete a particular deptno. What happens if there are employees in it and if there are no employees.
- Create alias for columns and use them in queries.
- List the employees according to ascending order of salary.
- List the employees according to ascending order of salary in each department.
- Use '&&' wherever necessary
- Amount 6000 has to be deducted as CM relief fund in a particular month which has to be accepted as input from the user. Whenever the salary becomes negative it has to be maintained as 1000 and the deduction amount for those employees is reduced appropriately.
- The retirement age is 60 years. Display the retirement day of all the employees.
- If salary of all the employees is increased by 10% every year, what is the salary of all the employees at retirement time.
- Find the employees who are born in leap year.
- Find the employees who are born on Feb 29.
- Find the departments where the salary of atleast one employee is more than 20000.
- Find the departments where the salary of all the employees is less than 20000.
- On first January of every year a bonus of 10% has to be given to all the employees. The amount has to be deducted equally in the next 5 months. Write procedures for it.
- As a designer identify the views that may have to be supported and create views.
- As a designer identify the PL/SQL procedures necessary and create them using cursors.

Use appropriate Visual programming tools like oracle forms and reports, visual basic etc to create user interface screens and generate reports.

Note: As a designer identifies other operations that may be required and add to the above list. The above operations are not in order. Order them appropriately. Use SQL or PL/SQL depending on the requirement.

3. Students may be divided into batches and the following experiments may be given to them to better understand the DBMS concepts. Students should gather the required information, draw ER diagrams, map them to tables, normalize, create tables, triggers, procedures, execute queries, create user interfaces, and generate reports.
    - Student information system
    - APSRTC reservation system
    - Hostel management
    - Library management
    - Indian Railways reservation
    - Super market management
    - Postal system
    - Banking system
    - Courier system
    - Publishing house system

Week-1: CREATION OF TABLES

Exp 1:

Aim: To create a table called Employee with the following Structure and Answer the following queries.

Name	Type
Empno	Number
Ename	Varchar2(20)
Job	Varchar2(20)
Mgr	Number
Sal	Number

Sql>createtable Employee (Empnonumber,Ename varchar2(20),job varchar2(20), Mgrnumber,Sal number);
Or

Sql>createtable Employee (Empnonumber,Ename varchar2(20),job varchar2(20), Mgrnumber,Sal number); constraintpk_employeesprimarykey (empno), constraintfk_employees_deptnoforeignkey (deptno) references DEPARTMENTS(deptno));

Sql> Select * from Employee;

Output:

a. Add a column commission with domain to the Employee

table Sql> Altertable employee add commissionnumber;

Output:

b. Insert any five records in to thetable.

Sql> INSERT INTO Employee VALUES (1, 'King', 'ITmanager', '100', '20000'); Sql>

INSERT INTO Employee VALUES (5, 'blake', 'IT', '200', '30000');

Sql> INSERT INTO Employee VALUES (9, 'raj', 'manager', '300', '40000'); Sql>

INSERT INTO Employee VALUES (19, 'clarke', 'Assistant', '400', '50000'); Sql>

INSERT INTO Employee VALUES (25, 'mohan', 'clerk', '500', '60000');

Output:

c. Update the column details of job

Sql> UPDATE EMPLOYEE SET JOB = 'MANAGER'WHERE JOB IS NULL;

Output:

d. Rename the column of Employ table using alter command.

Sql>ALTER TABLE Employee RENAME COLUMN Ename TO Employname;

Output:

e. Delete the employee whose empno is19.

Sql>DELETEempno FROM Employee WHERE empno=19;

Output:

Exp 2:

Aim: Create department table with the following structure and answer the following quries.

Name	Type
Deptno	Number
Deptname	Varchar2(20)
location	Varchar2(20)

Sql>CREATE TABLE dept (Deptnonumber,Deptnamevarchar2(20), location varchar2(20));**or**

create table dept(deptno number(2,0), dname varchar2(14), locvarchar2(13),

constraintpk_dept primary key (deptno));

Output:

a. Add column designation to the department

table.

Sql>Alter table det add

designationvarchar2(20); Output:

b. Insert values into thetable.

Sql>insertintodeptvalues(101,„cse‟,„nellore‟,„assistant‟);

Sql>insertintodeptvalues(102,„Ece‟,„tpty‟,„assistant‟);

Sql>insertintodeptvalues(103,„eee‟,„banglore‟,„HR‟); Sql>

insert into dept values(104, „civil‟, „Hyd‟, „manager‟);

Sql>insertintodeptvalues(101,„cse‟,„chittoor‟,„assistant‟);

Output:

c. List the records of emp table grouped by dept no

Sql>SELECT empno from emp, dept , GROUP BY deptno;

Output:

d. Update the record where dept nois 9.

Sql>Updatetabledeptsetdeptno=9wherelocation=„tpty‟;

Output:

e. Delete any column data from the

table Sql>DELETE location FROM dept;

Output:

Exp 3:

Aim: To create a table called Customer table and answer the following queries.

Name	Name Type
Custname	Varchar2(20)
custstreet	Varchar2(20)

custcity	Varchar2(20)

Sql>CREATE TABLE customer (custnamevarchar2(20), custstreetvarchar2(20), custcityvarchar2(20));

a. Insert records into thetable

Sql> insert into customer values("kumar", "4street", "hyd");

Sql> insert into customer values("rmesh", "avenue", "hyd");

Sql> insert into customer values("Mahesh", "amerpet", "hyd");

Sql> insert into customer values("vasu","marthali", "Banglore");

Sql> insert into customer values("hari", "siliconcity", "Banglore"); Output:

b. Add salary column to thetable

Sql> Update table customer add salary

number; Output:

c. Alter the table column domain.

Sql>Altertablecustomersetcustname=„cname";

Output:

d. Drop salary column of the customer table.

Sql> Alter table customer drop columnsalary;

Output:

e. Deletetherowsofcustomertablewhoseust_cityis"hyd".

Sql>DELETEFROMcustomerWHERE custcity="hyd"; Output:

f. Create a table called branchtable.

Name	Name Type
branchname	Varchar2(20)
Branch	Varchar2(20)
asserts	Varchar2(20)

Sql> Create table branch (branchname varchar2(20), Branch varchar2(20), asserts varchar2(20); Output:

Exp 4:

Aim: To increase the size of data type for asserts to the branch and answer the following queries

a) Add and drop a column to the branch table.
Sql> Alter table branch add branchidnumber;
Output:

Sql> Alter table branch drop column branchid;
Output:

b) Insert values to thetable.
Sql> insert into branch values(„kukatpally‟, „Iron‟,„Iron_rods‟);Sql>insertintobranchvalues(„amerpet‟,„steel‟,„steel_plates‟);
Sql>insertintobranchvalues(„SRNagar‟,„soap‟,„soapplant‟);
Output:

c) Update the branch namecolumn
Sql>updatetablebranchsetbranchname=„bname‟;
Output:

d) Delete any two columns from thetable
Sql> Alter table branch drop(bname, asserts); Output:

Exp 5:

Aim: Create a table called sailor table and answer the following queries

Sailors(sid: integer, sname: string, rating: integer, age:real);

SQL>CREATE TABLE sailors (sid integer not null,sname varchar(32),rating integer,CONSTRAINTPK_sailors PRIMARY KEY (sid));

a.Add column age to the sailortable.

Sql>altertablesailorsaddcolumnagereal;

Output:

b. Insert values into thesailortable.

Sql>INSERTINTOsailors(sid,sname,rating,age)VALUES(22,'Dustin',7,45.0);

Sql>INSERTINTOsailors(sid,sname,rating,age)VALUES(22,'brutes',9,.60.0);

Sql>INSERTINTOsailors(sid,sname,rating,age)VALUES(22,'luber',8,58.0);

c. Delete the row withrating>8.

Sql> delete from sailors where

ratting>8; Output:

d. Update the column detailsofsailor.

Sql>Updatetablesailorssetsname=„sailorname";

Output:

e. Insert null values intothetable.

Sql>INSERTINTOsailors(sid,sname,rating,age)VALUES(22,'Dustin',,45.0);

Sql>INSERTINTOsailors(sid,sname,rating,age)VALUES(22,'',7,45.0);

Output:

Exp 6:

Aim: To Create a table called reserves table and answer the following queries

Reserves(sid:integer,bid:integer,day:date)

Sql>CREATETABLEreserves(sidintegernotnull,bidintegernotnull,daydatetimenotnull, CONSTRAINTPK_reservesPRIMARYKEY(sid,bid,day),FOREIGNKEY(sid)REFERENCESsailors(sid), FOREIGNKEY(bid)REFERENCESboats(bid));

a. Insertvaluesintothereservestable.

Sql>INSERTINTOreserves(sid,bid,day)VALUES(22,101,'1998-10-10');

Sql>INSERTINTOreserves(sid,bid,day)VALUES(31,101,'1998-10-10');

Sql>INSERTINTOreserves(sid,bid,day)VALUES(22,102,'1998-10-09');

Sql>INSERTINTOreserves(sid,bid,day)VALUES(64,102,'1998-10-08');

Output:

b. Addcolumntimetothereservestable.

Sql>Altertablereservesaddcolumnbnamevarchar2(20);

Output:

c. Alterthecolumndaydatatypetodate.

Sql>Altertablereservesmodifydaydate;

Output:

d. Dropthecolumntimeinthetable.

Sql>Altertablereservesdropcolumnsidwhereday='1998-10-10';

Output:

e. Deletetherowofthetablewithsomecondition.

Sql>Deletetablereserves;

Output:

Week 2: QUERIES USING DDL AND DML

SQL (Structured Query Language):

Structured Query Language is a databasecomputer language designed for managing data in relational database management systems(RDBMS), and originally based upon Relational Algebra. Its scope includes data query and update, schema creation and modification, and data access control. SQL was one of the first languages for Edgar F.Codd'srelational model in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks"[3] and became the most widely used language for relationaldatabases.

- IBM developed SQL in mid of1970‟s.
- Oracle incorporated in the year 1979.
- SQL used by IBM/DB2 and DS DatabaseSystems.
- SQL adopted as standard language for RDBS by ASNI in1989.

DATA TYPES:

1. **CHAR (Size):** This data type is used to store character strings values of fixed length. The size in brackets determines the number of characters the cell can hold. The maximum number of character is 255characters.
2. **VARCHAR (Size) / VERCHAR2 (Size)**: This data type is used to store variable length alphanumeric data. The maximum character can hold is 2000character.
3. **NUMBER (P, S):** The NUMBER data type is used to store number (fixed or floating point). Number of virtually any magnitude may be stored up to 38 digits of precision. Number as large as $9.99 * 10^{124}$. The precision (p) determines the number of places to the right of the decimal. If scale is omitted then the default is zero. If precision is omitted, values are stored with their original precision up to the maximum of 38digits.
4. **DATE:** This data type is used to represent date and time. The standard format is DD-MM- YY as in 17-SEP-2009. To enter dates other than the standard format, use the appropriate functions. Date time stores date in the 24-Hours format. By default the time in a date field is 12:00:00 am, if no time portion is specified. The default date for a date field is the first

day the currentmonth.

5. **LONG:** This data type is used to store variable length character strings containing up to 2GB. Long data can be used to store arrays of binary data in ASCII format. LONG values cannot be indexed, and the normal character functions such as SUBSTR cannot beapplied.
6. **RAW:** The RAW data type is used to store binary data, such as digitized picture or image. Dataloadedintocolumnsofthesedatatypesarestoredwithoutanyfurtherconversion.

   RAW data type can have a maximum length of 255 bytes. LONG RAW data type can contain up to2GB.

There are five types of SQL statements. They are:

1. DATA DEFINITION LANGUAGE(DDL)
2. DATA MANIPULATION LANGUAGE(DML)
3. DATA RETRIEVAL LANGUAGE(DRL)
4. TRANSATIONAL CONTROL LANGUAGE(TCL)
5. DATA CONTROL LANGUAGE(DCL)

EXP 7: TO PRACTICE DDL COMMANDS USING ORACLE

1. DATA DEFINITION LANGUAGE (DDL): The Data Definition Language (DDL) is used to create and destroy databases and database objects. These commands will primarily be used by database administrators during the setup and removal phases of a database project. Let's take a look at the structure and usage of four basic DDLcommands:

1. CREATE 2. ALTER 3. DROP 4.RENAME

1. CREATE:

(a) CREATE TABLE: This is used to create a new relation and thecorresponding

Syntax: CREATE TABLE relation_name

(field_1 data_type(Size),field_2 data_type(Size), .. .);

Example:

SQL>CREATE TABLE Student (snoNUMBER(3),snameCHAR(10),class CHAR(5));

(b) CREATETABLE..ASSELECT ..... Thisisusedtocreatethestructureofanewrelationfrom the structure of an existing relation.

Syntax: CREATE TABLE (relation_name_1, field_1,field_2,.....field_n) AS SELECT field_1,field_2,.....field_nFROM relation_name_2;

Example: SQL>CREATE TABLE std(rno,sname) AS SELECT sno,snameFROM student;

Output:

SQL>Descstd;

Output:

2. ALTER:

(a) ALTER TABLE ...ADD...: This is used to add some extra fields into existingrelation.

Syntax:ALTER TABLE relation_nameADD(new field_1 data_type(size), new field_2 data_type(size),..);

Example :SQL>ALTER TABLE std ADD(Address CHAR(10));

(b) ALTER TABLE...MODIFY...: This is used to change the width as well as data type of fieldsof existingrelations.

Syntax:ALTER TABLE relation_nameMODIFY (field_1 newdata_type(Size), field_2newdata_type(Size),. .....field_newdata_type(Size));

Example:SQL>ALTER TABLE student MODIFY(snameVARCHAR(10),class VARCHAR(5));

Output:

3. DROP TABLE: This is used to delete the structure of a relation. It permanently deletes the records in thetable.

Syntax: DROP TABLErelation_name;

Example: SQL>DROP TABLE std;

Output:

4. RENAME: It is used to modify the name of the existing databaseobject.

Syntax: RENAME TABLE old_relation_name TOnew_relation_name;

Example: SQL>RENAME TABLE stdTOstd1;

Output:

5. TRUNCATE: This command will remove the data permanently. But structure will not be removed.

Syntax: TRUNCATE TABLE<Tablename>

Example TRUNCATE TABLEstudent;

Output:

EXP8:

AIM: TO PRACTICE DML COMMANDS USING ORACLE

DATA MANIPULATION LANGUAGE (DML): The Data Manipulation Language (DML) is used to retrieve, insert and modify database information. These commands will be used by all database users during the routine operation of the database. Let's take a brief look at the basic DML commands:

1. INSERT 2.UPDATE 3.DELETE

1. INSERT INTO: This is used to add records into a relation. These are three type of INSERT INTO queries which areas

a) Inserting a singlerecord

Syntax: INSERT INTOrelationname(field_1,field_2,.field_n)VALUES (data_1, data_2,…..data_n);

Example: SQL>INSERT INTO student
(sno,sname,class,address)VALUES(1,'Ravi','M.Tech','Palakol');

Output:

b) Inserting all records from anotherrelation

Syntax: INSERT INTO relation_name_1 SELECTField_1,field_2,field_n FROM

relation_name_2 WHERE field_x=data;

Example: SQL>INSERT INTO stdSELECTsno,snameFROM student WHERE

name = 'Ramu';

Output:

c) Inserting multiplerecords

Syntax: INSERT INTO relation_namefield_1,field_2,.....field_n) VALUES (&data_1,&data_2,.

..... &data_n);

Example: SQL>INSERT INTO student(sno,sname,class,address)

VALUES(&sno,'&sname','&class','&address');

Enter value for sno: 101 Enter value for

name: Ravi Enter value for class:

M.Tech Enter value for name:Palakol

Output:

2. UPDATE-SET-WHERE: This is used to update the content of a record in arelation.

Syntax: SQL>UPDATE relation name SET Field_name1=data,field_name2=data, WHERE

field_name=data;

Example: SQL>UPDATE student SETsname = 'kumar' WHEREsno=1;

Output:

3. DELETE-FROM: This is used to delete all the records of a relation but it will retain the structure of thatrelation.

a) DELETE-FROM: This is used to delete all the records ofrelation.

Syntax: SQL>DELETE FROM relation_name;

Example: SQL>DELETE FROMstd;

Output:

b) DELETE -FROM-WHERE: This is used to delete a selected record from arelation.

Syntax: SQL>DELETE FROM relation_name WHERE condition;

Example: SQL>DELETE FROM student WHERE sno =2;

Output:

DRL(DATA RETRIEVAL LANGUAGE): Retrieves data from one or more tables.

1. SELECT FROM: To display all fields for allrecords.

Syntax: SELECT * FROM relation_name;

Example : SQL> select * from dept; Output:

2. SELECT FROM: To display a set of fields for all records ofrelation.

Syntax: SELECT a set of fields FROM relation_name;

Example: SQL> select deptno, dname from dept;

Output:

3.SELECT - FROM -WHERE: This query is used to display a selected set of fields for a selected set of records of a relation.

Syntax: SELECT a set offieldsFROMrelation_name WHERE condition;

Example: SQL> select * FROM dept WHERE deptno<=20;

Output:

4. SELECT - FROM -GROUP BY: This query is used to group to all the records in a relation together for each and every value of a specific key(s) and then display them for a selected set of fields therelation.

Syntax: SELECT a set of fields FROM relation_nameGROUP BY field_name;

Example: SQL> SELECT EMPNO, SUM (SALARY) FROM EMP GROUP BYEMPNO;

Output:

5. SELECT - FROM -ORDER BY: This query is used to display a selected set of fields from a relation in an ordered manner base on somefield.

Syntax: SELECT a set of fields FROM relation_name ORDER BY

field_name;

Example: SQL> SELECT empno,ename,job FROM emp ORDER BY job;

Output:

6. JOIN using SELECT - FROM - ORDER BY: This query is used to display a set of fields from two relations by matching a common field in them in an ordered manner based on somefields.

Syntax: SELECT a set of fields from both relations FROM relation_1, relation_2 WHERE relation_1.field_x = relation_2.field_y ORDER BY field_z;

Example: SQL>SELECT empno,ename,job,dname FROM emp,deptWHEREemp.deptno = 20 ORDER BY job;

Output:

7. JOIN using SELECT - FROM - GROUP BY: This query is used to display a set of fields from two relations by matching a common field in them and also group the corresponding records for each and every value of a specified key(s) whiledisplaying.

Syntax: SELECT a set of fields from both relations FROM relation_1,relation_2 WHERE relation_1.field-x=relation_2.field-y GROUP BY field-z;

Example: SQL> SELECT empno,SUM(SALARY) FROM emp,deptWHEREemp.deptno =20 GROUP BY empno;

Output:

8. UNION: This query is used to display the combined rows of two different queries, which are having the same structure, without duplicaterows.

Syntax: SELECT field_1,field_2,..... FROM relation_1 WHERE (Condition) UNION SELECT field_1,field_2,..... FROM relation_2 WHERE (Condition);

Example:

SQL> SELECT * FROM STUDENT;

Output:

SQL> SELECT * FROM STD;

Output:

SQL> SELECT * FROM student UNION SELECT * FROM std;

Output:

INTERSET: This query is used to display the common rows of two different queries, which are having the same structure, and to display a selected set of fields out ofthem.

Syntax: SELECT field_1,field_2,.. FROM relation_1 WHERE

(Condition) INTERSECT SELECT field_1,field_2,.. FROM relation_2 WHERE(Condition);

Example :SQL> SELECT * FROM student INTERSECT SELECT * FROM std;

Output:

9. MINUS: This query is used to display all the rows in relation_1,which are not having in the relation_2.

Syntax:SELECTfield_1,field_2,.....FROM relation_1 WHERE(Condition)MINUS

SELECTfield_1,field_2,.....

FROM relation_2 WHERE(Conditon);

SQL>SELECT * FROM student MINUS SELECT * FROM std;

Output:

TRANSATIONAL CONTROL LANGUAGE (T.C.L):

A transaction is a logical unit of work. All changes made to the database can be referred to as a transaction. Transaction changes can be mode permanent to the database only if they are committed a transaction begins with an executable SQL statement & ends explicitly with either role back or commit statement.

1. COMMIT: This command is used to end a transaction only with the help of the commit command transaction changes can be made permanent to thedatabase.

Syntax:SQL>COMMIT;

Example: SQL>COMMIT;

Output:

2. SAVE POINT: Save points are like marks to divide a very lengthy transaction to smaller once. They are used to identify a point in a transaction to which we can latter role back. Thus, save point is used in conjunction with rollback.

Syntax: SQL>SAVE POINT ID;

Example: SQL>SAVE POINTxyz;

Output:

3. ROLL BACK: A role back command is used to undo the current transactions. We can roll back the entire transaction so that all changes made by SQL statements are undo (or) role back a transaction to a save point so that the SQL statements after the save point are rollback.

Syntax: ROLLBACK(current transaction can be roll back) ROLL BACK to

save pointID;

Example: SQL>ROLLBACK;

SQL>ROLL BACK TO SAVE POINT xyz;

Output:

DATA CONTROL LANGUAGE (D.C.L):

DCL provides uses with privilege commands the owner of database objects (tables), has the soul authority ollas them. The owner (data base administrators) can allow other data base uses to access the objects as per their requirement

1. GRANT: The GRANT command allows granting various privileges to other users and allowing them to perform operations with in theirprivileges

For Example, if a uses is granted as 'SELECT' privilege then he/she can only view data but cannot perform any other DML operations on the data base object GRANTED privileges can also be withdrawn by the DBA at anytime

Syntax: SQL>GRANT PRIVILEGES on object_nameTouser_name;

Example: SQL>GRANT SELECT, UPDATE on empTohemanth;

Output:

2. REVOKE: To with draw the privileges that has been GRANTED to a uses, we use the REVOKEcommand

Syntax: SQL>REVOKE PRIVILEGES ON object-name FROMuser_name;

Example: SQL>REVOKE SELECT, UPDATE ONemp FROMravi;

Output:

1. Creation, altering and dropping of tables and inserting rows into a table(use constraints while creating tables) examples using SELECTcommand.

2. CREATE:

(a)CREATE TABLE: This is used to create a new relation

Syntax: CREATE TABLE relation_name

(field_1 data_type(Size),field_2 data_type(Size), .. .);

Example:

SQL>CREATE TABLE Student (snoNUMBER(3)**PRIMARY KEY**,snameCHAR(10),classCHAR(5));

Output:

3. ALTER:

(a) ALTER TABLE ...ADD...: This is used to add some extra fields into existingrelation.

Syntax:ALTER TABLE relation_nameADD(new field_1 data_type(size), new field_2data_type(size),..);

Example:SQL>ALTER TABLE stdADD(Address CHAR(10));

Output:

(b) ALTER TABLE...MODIFY...: This is used to change the width as well as data type of fieldsof existingrelations.

Syntax:ALTER TABLE relation_nameMODIFY (field_1 newdata_type(Size), field_2newdata_type(Size),. .....field_newdata_type(Size));

Example:SQL>ALTER TABLE student MODIFY(snameVARCHAR(10),classVARCHAR(5));

Output:

4. DROP TABLE: This is used to delete the structure of a relation. It permanently deletes the records in thetable.

Syntax: DROP TABLErelation_name;

Example: SQL>DROP TABLE std;

Output:

5. INSERT:

Syntax: INSERT INTO relation_namefield_1,field_2,.....field_n) VALUES (&data_1,&data_2,. ..... &data_n);

Example: SQL>INSERT INTO student(sno,sname,class,address)

VALUES(&sno,'&sname','&class','&address');

Output:

6. SELECT FROM: To display all fields for allrecords.

Syntax: SELECT * FROM relation_name;

Example : SQL> select * from student; Output:

2. SELECT FROM: To display a set of fields for all records ofrelation.

Syntax: SELECT a set of fields FROM relation_name; ***Example:***

SQL> select sno, sname from student; Output:

3.SELECT - FROM -WHERE: This query is used to display a selected set of fields for a selected set of records of a relation.

Syntax: SELECT a set offieldsFROMrelation_nameWHERE condition;

Example: SQL> select * FROM student WHERE class='CSE';

Output:

There are 5 constraints available in ORACLE:

1. NOT NULL: When a column is defined as NOTNULL, then that column becomes a mandatory column. It implies that a value must be entered into the column if the record is to be accepted for storage in thetable.

Syntax:

CREATE TABLE Table_Name(column_namedata_type(*size*) **NOT NULL,**);

Example:

CREATE TABLE student (sno**NUMBER(3)NOT NULL,** name**CHAR**(**10**));

Output:

2. UNIQUE: The purpose of a unique key is to ensure that information in the column(s) isunique i.e. a value entered in column(s) defined in the unique constraint must not be repeated across the column(s). A table may have many uniquekeys.

Syntax:

CREATE TABLE Table_Name(column_namedata_type(*size*) **UNIQUE, ….**);

Example:

CREATE TABLE student (sno**NUMBER(3) UNIQUE,** name **CHAR**(**10**));

Output:

3. CHECK: Specifies a condition that each row in the table must satisfy. To satisfy the constraint, each row in the table must make the condition either TRUE or unknown (due to anull).

Syntax:

CREATE TABLE Table_Name(column_namedata_type(*size*) **CHECK(*logical expression*), ….**);

Example: **CREATE TABLE** student (sno**NUMBER (3),** name**CHAR**(**10**),class **CHAR(5),CHECK**(class **IN**('CSE','CAD','VLSI'));

Output:

4. PRIMARY KEY: A field which is used to identify a record uniquely. A column or combination of columns can be created as primary key, which can be used as a reference from other tables. A table contains primary key is known as MasterTable.

- ✓ It must uniquely identify each record in atable.
- ✓ It must contain uniquevalues.
- ✓ It cannot be a nullfield.
- ✓ It cannot be multiportfield.
- ✓ It should contain a minimum no. of fields necessary to be calledunique.

Syntax:

CREATE TABLE Table_Name(column_namedata_type(*size*) **PRIMARY KEY, ….**);

Example:

CREATE TABLE faculty (fcode**NUMBER(3) PRIMARY KEY,**fname**CHAR**(**10**));

Output:

5. FOREIGN KEY: It is a table level constraint. We cannot add this at column level. To reference any primary key column from other table this constraint can be used. The table in which the foreign key is defined is called a **detail table**. The table that defines the primary key and is referenced by the foreign key is called the **mastertable**.

Syntax: **CREATE TABLE**Table_Name(column_namedata_type(*size*) **FOREIGN KEY**(column_name)**REFERENCES**table_name);

Example:

CREATE TABLE subject (scode**NUMBER (3) PRIMARY KEY,**
subname**CHAR**(**10**),fcode**NUMBER(3), FOREIGN KEY**(fcode) **REFERENCE** faculty);

Output:

Defining integrity constraints in the alter table command:

Syntax: **ALTER TABLE**Table_Name**ADDPRIMARY KEY** (column_name);

Example: **ALTER TABLE** student **ADDPRIMARY KEY**(sno);

(Or)

Syntax: **ALTER TABLE** table_name**ADD CONSTRAINT** constraint_name **PRIMARY KEY**(colname)

Example: **ALTER TABLE** student **ADD CONSTRAINT** SN **PRIMARY KEY(**SNO**)**

Output:

Dropping integrity constraints in the alter table command:

Syntax: **ALTER TABLE**Table_Name**DROP**constraint_name;

Example: **ALTER TABLE** student **DROPPRIMARYKEY**;

(or)

Syntax: **ALTER TABLE** student **DROP CONSTRAINT** constraint_name**;**

Example: **ALTER TABLE** student **DROP CONSTRAINT**SN**;**

Output:

Week-3:QUERIES USING AGGREGATE FUNCTIONS

Exp 9:

Aim:To Practice Aggregate functions using oracle.

Aggregative operators: In addition to simply retrieving data, we often want to perform some computation or summarization. SQL allows the use of arithmetic expressions. We now consider a powerful class of constructs for computing aggregate values such as MIN andSUM.

1. Count: COUNT following by a column name returns the count of tuple in that column. If DISTINCT keyword is used then it will return only the count of unique tuple in the column. Otherwise, it will return count of all the tuples (including duplicates) count (*) indicates all the tuples of thecolumn.

Syntax: COUNT (Column name)

Example:SELECT COUNT (Sal) FROM emp;

Output:

2. SUM: SUM followed by a column name returns the sum of all the values in thatcolumn.

Syntax:SUM (Column name)

Example: SELECT SUM (Sal) From emp;

Output:

3. AVG: AVG followed by a column name returns the average value of that columnvalues.

Syntax:AVG (n1,n2..)

Example: Select AVG(10, 15, 30) FROM DUAL;

Output:

4. MAX: MAX followed by a column name returns the maximum value of thatcolumn.

Syntax: MAX (Column name)

Example: SELECT MAX (Sal) FROM emp;

Output:

SQL> select deptno,max(sal) from emp group by deptno; Output:

SQL> select deptno,max(sal) from emp group by deptno having max(sal)<3000;

Output:

5. MIN: MIN followed by column name returns the minimum value of thatcolumn.

Syntax: MIN (Column name)

Example: SELECT MIN (Sal) FROM emp;

SQL>select deptno,min(sal) from emp group by deptno having min(sal)>1000;

Output:

VIEW: In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL functions, WHERE, and JOIN statements to a view and present the

data as if the data were coming from one single table.

A view is a virtual table, which consists of a set of columns from one or more tables. It is similar to a table but it doesn't not store in the database. View is a query stored as an object.

Syntax: CREATE VIEW view_name AS SELECT set of fields FROM relation_name WHERE (Condition)

SQL>CREATE VIEW employee AS SELECT empno,ename,jobFROM EMP WHERE

job = 'clerk';

OUTPUT:

SQL> SELECT * FROM EMPLOYEE;

OUTPUT:

3. Example:

CREATE VIEW [Current Product List] AS SELECT
ProductID,ProductName
FROM Products
WHERE Discontinued=No

DROP VIEW:This query is used to delete a view , which has been already created.

Syntax: DROP VIEWView_name;

Example : SQL> DROP VIEWEMPLOYEE;

Output:

Exp10:

Aim: To practice String & Character functions using sql

CONVERSION FUNCTIONS:

To_char: TO_CHAR (number) converts n to a value of VARCHAR2 data type, using the optional number format fmt. The value n can be of type NUMBER, BINARY_FLOAT, or BINARY_DOUBLE.

SQL>select to_char(65,'RN')from dual; LXV

To_number :TO_NUMBER converts expr to a value of NUMBER data type.

SQL>Select to_number ('1234.64') from Dual;

Output:

To_date:TO_DATE converts char of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 data type to a value of DATE data type.

SQL>SELECT TO_DATE('January 15, 1989, 11:00 A.M.')FROM DUAL;

Output:

STRING FUNCTIONS:

Concat: CONCAT returns char1 concatenated with char2. Both char1 and char2 can be any of the datatypes

SQL>SELECT CONCAT(„ORACLE",‟CORPORATION")FROMDUAL;
Output:

Lpad: LPAD returns expr1, left-padded to length n characters with the sequence of characters in expr2.

SQL>SELECT LPAD(„ORACLE",15,"*")FROM DUAL;

Output:

Rpad: RPAD returns expr1, right-padded to length n characters with expr2, replicated as many times as necessary.

SQL>SELECT RPAD („ORACLE‟,15,‟*‟)FROM DUAL;
Output:

Ltrim: Returns a character expression after removing leading blanks.

SQL>SELECT LTRIM(„SSMITHSS‟,‟S‟)FROM DUAL;
Output:

Rtrim: Returns a character string after truncating all trailing blanks

SQL>SELECT RTRIM(„SSMITHSS‟,‟S‟)FROM DUAL;
Output:

Lower: Returns a character expression after converting uppercase character data

to lowercase.

SQL>SELECT LOWER(„DBMS‟)FROM DUAL;
Output:

Upper: Returns a character expression with lowercase character data converted to

uppercase SQL>SELECT UPPER(„dbms‟)FROM DUAL;
Output:

Length: Returns the number of characters, rather than the number of bytes, of

the given string expression, excluding trailing blanks.

SQL>SELECT LENGTH(„DATABASE‟)FROM DUAL;
Output:

Substr: Returns part of a character, binary, text, or image expression.

SQL>SELECT SUBSTR(„ABCDEFGHIJ‟3,4) FROM DUAL;
Output:

Instr: The INSTR functions search string for substring. The function returns an integer

indicating the position of the character in string that is the first character of this

occurrence. SQL>SELECT INSTR('CORPORATE FLOOR','OR',3,2) FROM DUAL;
Output:

DATE FUNCTIONS:

Sysdate:

SQL>SELECT SYSDATE FROM DUAL; 29-DEC-08

Output:

next_day:

SQL>SELECT NEXT_DAY(SYSDATE,"WED")FROM DUAL;

Output:

add_months:

SQL>SELECT ADD_MONTHS(SYSDATE,2)FROM DUAL;

Output:

last_day:

SQL>SELECT LAST_DAY(SYSDATE)FROM DUAL;

Output:

months_between:

SQL>SELECT MONTHS_BETWEEN(SYSDATE,HIREDATE)FROM EMP;

Output:

Least:

SQL>SELECT LEAST('10-JAN-07','12-OCT-07')FROM DUAL;

Output:

Greatest:

SQL>SELECT GREATEST('10-JAN-07','12-OCT-07')FROM DUAL;

Output:

Trunc:

SQL>SELECT TRUNC(SYSDATE,'DAY')FROM DUAL;

Output:

Round:

SQL>SELECT ROUND(SYSDATE,'DAY')FROM DUAL;

Output:

to_char:

SQL> select to_char(sysdate, "dd\mm\yy") from dual;

Output:

to_date:

SQL> select to date (sysdate, "dd\mm\yy") from

dual; Output:

Truncate:

SQL>SELECT TRUNC(SYSDATE,'DAY')FROM DUAL;
Output:

Round:

SQL>SELECT ROUND(SYSDATE,'DAY')FROM DUAL;
Output:

to_char:

SQL> select to_char(sysdate, "dd\mm\yy") from dual;
Output:

to_date:

SQL> select to date (sysdate, "dd\mm\yy") from
dual; Output:

Exp11:

Aim: Consider the following schema:

Sailors (sid,sname, rating, age)

Boats (bid, bname, color)

Reserves (sid, bid,day(date))

Write subquery statement for the following queries**.**

```
create table sailors
( sidint primary key,
sname varchar(38),
   ratingint,
   age float check (age > 16 and age < 110)
);

create table boats (
   bid int primarykey,
bname varchar(25),
   color varchar(21)
);
```

```
create table reserves
( sidint,
   bidint,
   day
   date,
   foreign key (sid) references sailors (sid),
   foreign key (bid) references boats (bid)
);
```

1. Find the names of the sailors who have reserved both a red or a yellow boat. SQL> select s.sname from sailors s, boats b, reserves r where s.sid=r.sid and b.bid=r.bid and (b.color='red' orb.color='yellow');

Output:

2. Find all sids of sailors who have a rating of 10 or have reserved boat111.

```
SQL> select s.sid from sailors s where s.rating = 10 union select r.sid from reserves r
        where r.bid = 111;
```

Output:

3. Find all sids of sailors who have reserved red boats but not yellowboats

```
SQL> select s.sid from sailors s, boats b, reserves r where s.sid = r.sid and r.bid = b.bid and
b.color = 'red'
        minus
        select s2.sid from sailors s2, boats b2, reserves r2 where s2.sid = r2.sid and r2.bid =
        b2.bid and b2.color = 'yellow';
```

Output:

4. Find the names of the sailors who have reserved both a red and a yellowboat.

```
SQL> select s.sname from sailors s, boats b, reserves r where s.sid = r.sid and r.bid =
b.bid and b.color = 'red'
        intersect
        select s2.sname from sailors s2, boats b2, reserves r2 where s2.sid = r2.sid and
        r2.bid = b2.bid and b2.color = 'yellow';
```

Output:

5. Find the names of sailors who have reserved a red boat, and list in the order of age.
SQL>SELECT S.sname, S.ageFROM Sailors S, Reserves R, BoatsB
WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = "red" ORDER BY S.age; Output:

6. Find the names of sailors who have reserved at least one boat. SQL>SELECT snameFROM Sailors S, Reserves RWHERE S.sid =R.sid;

Output:

7. Find the ids and names of sailors who have reserved two different boats on the same day.
SQL>SELECT DISTINCT S.sid, S.snameFROM Sailors S, Reserves R1, Reserves R2WHERE S.sid = R1.sid AND S.sid = R2.sidAND R1.day = R2.day AND R1.bid <> R2.bid;

Output:

8. Using Expressions and Strings in the SELECT Command.
SQL>ELECT sname, age, rating + 1 assth

FROM Sailors
WHERE 2* rating – 1 < 10 AND sname like „B_%b"; Output:

9. NestedQuery
Find the names of sailors who have reserved boat 103.

SQL>SELECT S.snameFROM Sailors SWHERE S.sid
IN (SELECTR.sidFROM Reserves RWHERE R.bid = 103) Output:

SQL>SELECT S.snameFROM Sailors SWHERE
EXISTS (SELECT *FROM Reserves RWHERE R.bid = 103AND R.sid
= S.sid)

Output:

10. Find the name and the age of the youngestsailor.

```
SQL>SELECT S.sname, S.ageFROM Sailors SWHERE S.age<= ALL (
        SELECTageFROM Sailors )
```

Output:

Week-4: PROGRAMS ON PL/SQL

Exp 12:

PL/SQL Introduction

PL/SQL stands for Procedural Language extension of SQL. PL/SQL is a combination of SQL along with the procedural features of programming languages. It was developed by Oracle Corporation in the early 90's to enhance the capabilities of SQL.

Oracle uses a PL/SQL engine to processes the PL/SQL statements. A PL/SQL code can be stored in the client system (client-side) or in the database (server-side).

Advantages of PL/SQL:

- ***Block Structures*:** PL SQL consists of blocks of code, which can be nested within each other. Each block forms a unit of a task or a logical module. PL/SQL Blocks can be stored in the database andreused.
- ***Procedural Language Capability*:** PL SQL consists of procedural language constructs such as conditional statements (if else statements) and loops like (FORloops).
- ***Better Performance*:** PLSQL engine processes multiple SQL statements simultaneously as a single block, thereby reducing networktraffic.
- ***Error Handling*:** PL/SQL handles errors or exceptions effectively during the execution of a PL/SQL program. Once an exception is caught, specific actions can be taken depending upon the type of the exception or it can be displayed to the user with a message.

Syntax of PL/SQL program:

```
Declare
        Variable declaration;
Begin
        Executable statements;
end;
```

Conditional Statements in PL/SQL

As the name implies, PL/SQL supports programming language features like conditional statements, iterative statements.The programming constructs are similar to how you use in programming languages like Java and C++. In this section I will provide you syntax of how to use conditional statements in PL/SQL programming.

IF THEN ELSE STATEMENT:

```
IF condition THEN
        statement 1;

ELSE
statement 2; END IF;

IF condition 1 THEN
        statement1;
        statement2;
ELSIF condtion2 THEN
        statement 3;

ELSE
        statement 4;

END IF
```

1. Write a PL/SQL program to find the total and average of 4 subjects and display thegrade

```
declare
    java number(10);
    dbms number(10); co
    number(10); mfcs
    number(10); total
    number(10); avgs
    number(10); per
    number(10);
```

```
d
b
m
s
_
o
u
t
p
u
t
.
p
u
t
_
l
i
n
e
(
'
E
N
T
E
   begin   R THE MARKS'); java:=&java;
           dbms:=&dbms;
           co:=&co;
           mfcs:=&mfcsl;
           total:=(java+dbms+co+mfcs);
           per:=(total/600)*100;
           if java<40 or dbms<40 or co<40 or mfcs<40 then
                   dbms_output.put_line('FAIL');
             if per>75 then
                     dbms_output.put_line('GRADE
             A'); elsif per>65 and per<75 then
             dbms_output.put_line('GRADE B'); elsif
             per>55 and per<65then
                     dbms_output.put_line('GRADE
                     C'); else
                             dbms_output.put_line('INVALID INPUT');
             end if;
             dbms_output.put_line('PERCENTAGE IS '||per);
     end;
     /
```

OUTPUT:

;

2. Write a PL/SQL program to find the largest of three enumbers

begin

declare

a

n

u

m

b

e

r

;

b

n

u

m

b

e

r

```
c                       dbms_output.put_line('ALL ARE EQUAL');
number  elsif a>b and a>c then
;


a:=&a;
b:=&b;
c:=&c;
i
        f

        a
        =
        b

        a
        n
        d

        b
        =
        c

        a
        n
        d

        c
        =
        a

        t
        h
        e
        n
```

```
dbms_output.put_line('A IS GREATER'); elsif b>c
then dbms_output.put_line('B ISGREATER');
                elsedbms_output.put_line('C
ISGREATER');
        end if;
end;
/
```

OUTPUT:

Loops in PL/SQL

There are three types of loops in PL/SQL:

1. Simple Loop
2. While Loop
3. ForLoop

1. Simple Loop:A Simple Loop is used when a set of statements is to be executed at least once before the loop terminates. An EXIT condition must be specified in the loop, otherwise the loop will get into an infinite number of iterations. When the EXIT condition is satisfied the process exits from theloop.

Syntax:

```
LOOP
        statements;
 EXIT;
        {or EXIT WHEN condition ;}
```

condition is evaluated at the beginning of each iteration. The iteration continues until the condition becomesfalse.

Syntax:

WHILE <condition>

LOOP

statements; END

LOOP;

While Loop :A WHILE LOOP is used when a set of statements has to be executed as long as a condition is true. The

2. FOR Loop:A FOR LOOP is used to execute a set of statements for a predetermined number of times. Iteration occurs between the start and end integer values given. The counter is always incremented by 1. The loop exits when the counter reaches the value of the endinteger.

Syntax:

FOR counter IN

val1..val2

LOOP

statemen

ts;

END LOOP;

3. Write a PL/SQL program to generate Fibonacciseries

declare

a

nu

m

```
b number; n
c number; n
number;    i
number;
;
n:=&n;
a:=0;
b:=1;
dbms_output.put_line(a);
dbms_output.put_line(b);  for  i
in 1..n-2
loop

begin
        c:=a+b;
        dbms_output.put_line(c); a:=b;
        b:=c;
    end loop;

end;
/
```

OUTPUT:

```
a number; rev
number; d
number;
```

4. Write a PL/SQL Program to display the number in Reverse Order

```
a:=&a;
rev:=0;
while a>0
loop
declare
begin
        d:=mod(a,10);
        rev:=(rev*10)+d;
        a:=trunc(a/10);
    end loop;
    dbms_output.put_line('no is'|| rev);
end;
/
```

OUTPUT:

Week-5: PROCEDURES AND FUNCTIONS

Exp 13:

A procedures or function is a group or set of SQL and PL/SQL statements that perform a specific task. A function and procedure are a named PL/SQL Block which is similar. The major difference between a procedure and a function is, a function must always return a value, but a procedure may or may not return avalue.

Procedures:

A procedure is a named PL/SQL block which performs one or more specific task. This is similar to a procedure in other programming languages. A procedure has a header and a body. The header consists of the name of the procedure and the parameters or variables passed to the procedure.

The body consists or declaration section, execution section and exception section similar to a general PL/SQL Block. A procedure is similar to an anonymous PL/SQL Block but it is named for repeatedusage.

We can pass parameters to procedures in three ways:

Parameters Description

INtype These types of parameters are used to send values to storedprocedures.

OUTtype These types of parameters are used to get values from stored procedures. This is similar to a return type in functions.

IN OUT type These types of parameters are used to send values and get values from stored procedures.

"A procedure may or may not return any value."

Syntax:

CREATE [OR REPLACE] PROCEDURE procedure_name (<Argument> {IN, OUT,INOUT}
<Datatype>,…)

IS

Declaration section<variable, constant> ; BEGIN

Execution section

EXCEPTION

Exception section END

or

```
CREATE OR REPLACE PROCEDURE <procedure_name> (
        <parameterl IN/OUT <datatype>
        ..
        .
        )
[ IS | AS ]
        <declaration_part
> BEGIN
        <execution part>
EXCEPTION
        <exception handling part>
END;
```

Example:

```
CREATE OR REPLACE PROCEDURE welcome_msg (p_name IN VARCHAR2) IS
BEGIN
dbms_output.put_line („Welcome '|| p_name);
END;
/

Output:EXECwelcome_msg („Guru99“);
```

1. create table named emp have two column id and salary with numberdatatype.

```
CREATE OR REPLACE PROCEDURE p1(id IN NUMBER, sal IN NUMBER) AS
BEGIN
  INSERT INTO empVALUES(id, sal);
DBMD_OUTPUT.PUT_LINE('VALUE INSERTED.'); END;
/
```

Output

:

Functions:

A function is a named PL/SQL Block which is similar to a procedure. The major difference between a procedure and a function is, a function must always return a value, but a procedure may or may not return avalue.

Syntax:

```
CREATE [OR REPLACE] FUNCTION function_name [parameters]
RETURN return_datatype; {IS, AS}
Declaration_section<variable,constant> ;
 BEGIN
Execution_section
   Returnreturn_variable;
 EXCEPTION
  exception section
   Return
   return_variable;
 END;
```

Example:

```
create or replace function getsal (no IN number) return
number is
sal number(5);
begin
 select salary into sal from emp where
 id=no; return sal;
end;
/
```

Output

Week-6: TRIGGERS

Exp 13:

1. Create a row level trigger for the customers table that would fire for INSERT orUPDATE or DELETEoperationsperformedontheCUSTOMERStable.Thistriggerwilldisplaythesalarydifference between the old values and new values:

CUSTOMERS table:

ID	NAME	AGE	ADDRESS	SALARY
1	Alive	24	Khammam	2000
2	Bob	27	Kadappa	3000
3	Catri	25	Guntur	4000
4	Dena	28	Hyderabad	5000
5	Eeshwar	27	Kurnool	6000
6	Farooq	28	Nellur	7000

Output:

2. Creation of insert trigger, delete trigger, update trigger practice triggers using the empdatabase.

```
SQL> create table emp (name varchar(10),empno number(3),age number(3));
```

```
Table
created
. SQL>
```

```
createorreplacetriggert2beforeinsertonempforea
chrow
when(new.age>100)
begin
RAISE_APPLICATION_ERROR(-20998,'INVALID ERROR'); 6*
end;
```

Output:

3. The following program creates a row-level trigger for the customers table that would fire forINSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old values and newvalues.

```
CREATE OR REPLACE TRIGGER display_salary_changes

BEFORE DELETE OR INSERT OR UPDATE ON customers FOR
EACH ROW
WHEN (NEW.ID > 0)
DECLARE
sal_diff number;
BEGIN
sal_diff := :NEW.salary - :OLD.salary;
dbms_output.put_line('Old salary: ' || :OLD.salary);
dbms_output.put_line('New salary: ' ||
:NEW.salary); dbms_output.put_line('Salary
difference: ' || sal_diff); END;
/
Trigger is created
```

Output:

Week-7: PROCEDURES

Exp 14:

1. Create the procedure for palindrome of givennumber.

```
DECLARE
  n number;
  m
  number;
  temp
  number:=0; rem
  number;
BEGIN
  n :=12321;
  m :=n;
  while n>0
  loop
    rem := mod(n,10);
    temp :=
    (temp*10)+rem; n :=
    trunc(n/10);
  end loop;
  if m = temp
  then
dbms_output.put_line('Palindrome');
  else
dbms_output.put_line('Not
  Palindrome'); end if;
END;
```

Output:

2. Create the procedure for GCD: Program should load two registers with two Numbers and then apply the logic for GCD of twonumbers.

```
DECLARE

   -- declare variable num1, num2 and t
   -- and these three variables datatype are
   integer num1 INTEGER;
   num2INTEGER; t
       INTEGER;
BEGIN
   num1 := 8;

   num2 := 48;

   WHILE MOD(num2, num1) != 0 LOOP
```

```
        t := MOD(num2, num1);

        num2 := num1;

        num1 := t;

    ENDLOOP;

dbms_output.Put_line('GCD of ' ||num1 ||' and ' ||num2 ||' is '
||num1); END;
```

Output:

3. Write the PL/SQL programs to create the procedure for factorial ofgivennumber.

```
Delimiter //
 CREATE PROCEDURE fact(IN x INT)
BEGIN
DECLARE result INT;
DECLARE iINT;
SET result = 1;
SET i = 1; WHILE
i<= xDO
SET result = result * i;
SET i = i + 1;
END WHILE;
SELECT x AS Number, result as Factorial;
    END//
Query OK, 0 rows affected (0.17 sec)
```

Output:

4. Write the PL/SQL programs to create the procedure to find sum of Nnaturalnumber.

```
SQL>Declare i
number:=0;
nnumber;
 sum1 number:=0;
 Begin
 n:=&n;
 while i
 loop
 sum1:=sum1+i;
dbms_output.put_line(i)
```

```
; i:=i+1;
 end loop;
dbms_output.put_line('The sum
 is:'||sum1); End;
 /
```

Output:

5.Write the PL/SQL programs to create the procedure to find Fibonacciseries.

```
declare
    first number:=0;
    second
    number:=1; third
    number;
    n number:=&n;
    i number;

begin
    dbms_output.put_line('Fibonacci series
    is:'); dbms_output.put_line(first);
    dbms_output.put_line(second);

    fori in 2..n
    loop
          third:=first+second;
          first:=second;
          second:=third;
          dbms_output.put_line(third
          );
    end
loop; end;
/
```

Output

6.Write the PL/SQL programs to create the procedure to check the given number isperfect ornot

```
declare
j number(10);
n
number(10);
```

```
sum number(10):= 0;
begin
j:=&num;
for n in 1..j-1 loop
if mod(j,n)=0 then
sum=sum+n;
end if;
end
loop;
if sum=j then
dbms_output.put_line("perfect
no."); else
dbma_output.put_line("not
perfect"); end if;
end;/
```

Output:

Week-8: CURSORS

Exp 15:

1. Write a PL/SQL block that will display the name, dept no, salary of fist highestpaid employees.

```
DECLARE
   CURSOR dpt_cur IS
    SELECTd.department_id        id,
department_namedptname,
          city,
Nvl(first_name, '...')
    manager FROM
    departments d
          left outer join employees e
                ON ( d.manager_id = e.employee_id )
          join locations l USING(location_id)
    ORDER BY 2;
emp_nameemployees.first_name%TYPE;
emp_max_salaryemployees.salary%TYPE; BEGIN
   FOR dept_all IN dpt_cur LOOP
      SELECT Max(salary)
      INTO emp_max_salary
```

```
      FROMemployees
      WHERE department_id = dept_all.id;

      IF emp_max_salary IS NULL THEN
emp_name := '...';
      ELSE
        SELECT first_name
        INTO emp_name
        FROMemployees
        WHERE department_id = dept_all.id
              AND salary = emp_max_salary;
      END IF;

dbms_output.Put_line(Rpad(dept_all.dptname, 20)
                     || Rpad(dept_all.manager, 15)
                     || Rpad(dept_all.city,20)
                     || Rpad(emp_name,20));
  END LOOP;
END;
/
```

Output:

2.Write a PL/SQL block that will display the employee details along with salary using Cursors

```
DECLARE
z_employeeemployees%ROWTYPE;
BEGIN
  SELECT *
  INTO z_employee -- INTO clause always notifies only single row can be fetch
  FROMemployees
  WHERE employee_id = 149;

dbms_output.Put_line('Employee Details : ID:'
                    ||z_employee.employee_id
                    ||' Name: '
                    ||z_employee.first_name
                    ||' '
                    ||z_employee.last_name
                    ||' Salary: '
                    ||z_employee.salary);
END; /
```

Output:

Lab Exercise:

3. To write a Cursor to display the list of employees who are working as a Managersor Analyst.

4. To write a Cursor to find employee with given job and dept no.

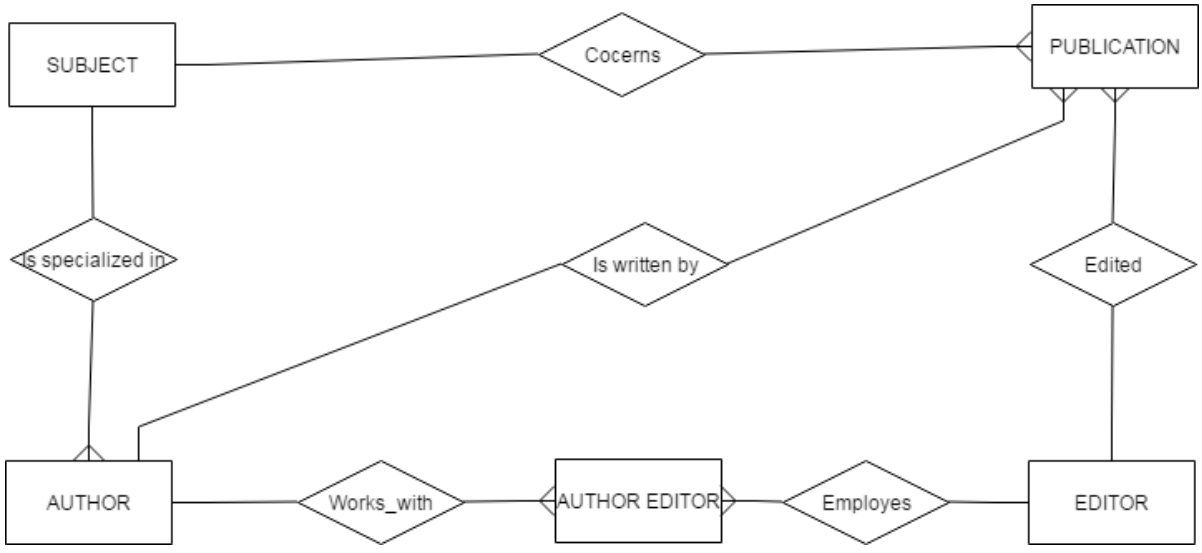
5. Write a PL/SQL block using implicit cursor that will display message, the salaries of alltheemployeesinthe„employee"tableareupdated.Ifnoneoftheemployee"ssalaryareupdatedweget a message 'None of the salaries were updated'. Else we get a message likefor example, 'Salaries for1000 employeesareupdated'if there are1000rowsin„employee" table

WEEK-9

CASE STUDY: BOOK PUBLISHING COMPANY

AIM: A publishing company produces scientific books on various subjects. The books are written by authors who specialize in one particular subject. The company employs editors who, not necessarily being specialists in a particular area, each take sole responsibility for editing one or more publications.

A publication covers essentially one of the specialist subjects and is normally written by a single author. When writing a particular book, each author works with on editor, but may submit another work for publication to be supervised by other editors. To improve their competitiveness, the company tries to employ a variety of authors, more than one author being a specialist in a particular subject.



LAB ASSIGNMENT:

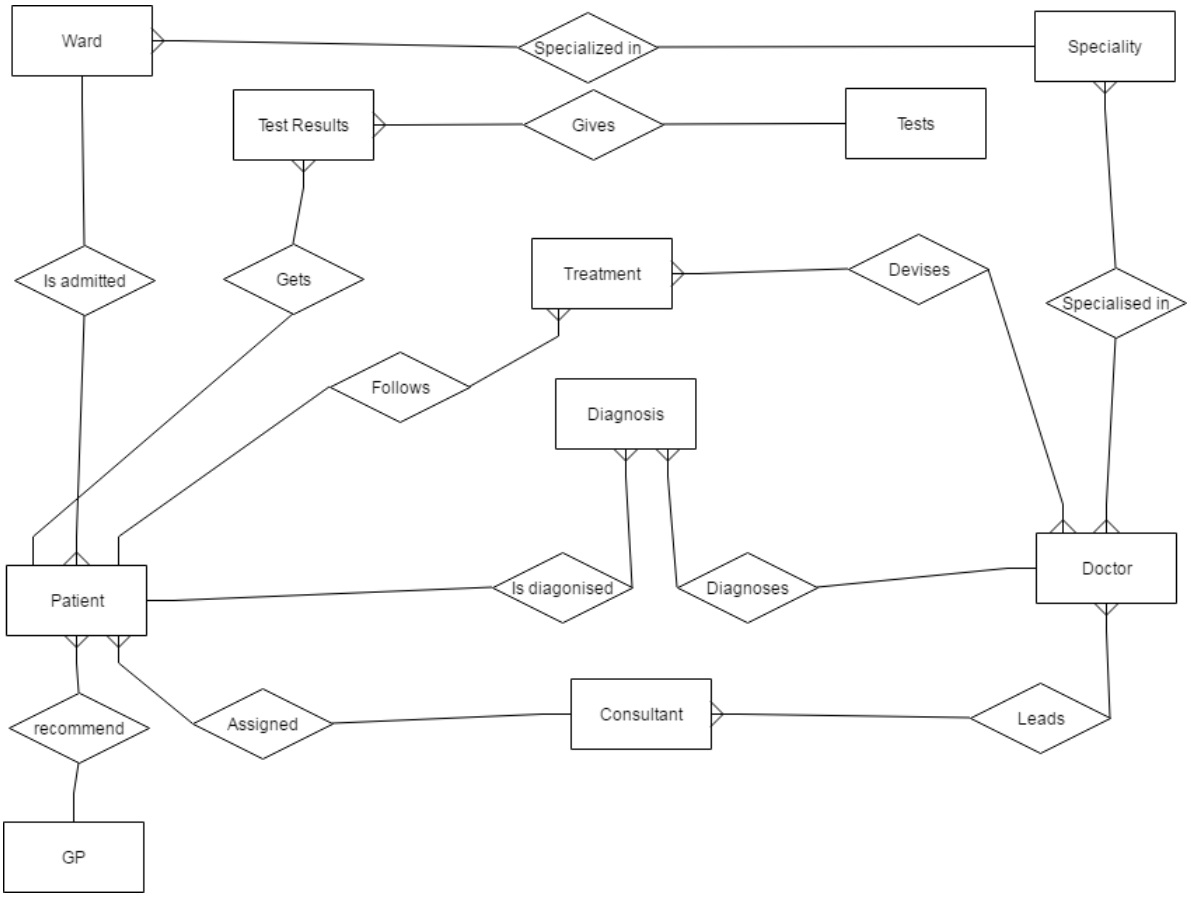
1. Analyze thedatarequired.
2. Normalizetheattributes.
3. CreatethelogicaldatamodelusingE-Rdiagrams

WEEK -10

CASE STUDY: GENERAL HOSPITAL

AIM: A General Hospital consists of a number of specialized wards (such as Maternity, Paediatry,Oncology,etc).Eachwardhostsanumberofpatients,whowereadmittedonthe Recommendation of their own GP and confirmed by a consultant employed by the Hospital. On admission, the personal details of every patient are recorded.

A separate register is to be held to store the information of the tests undertaken and the results of a prescribed treatment. A number of tests may be conducted for each patient. Each patient is assigned to one leading consultant but may be examined by another doctor, if required. Doctors are specialists in some branch of medicine and may be leading consultants for a number of patients, not necessarily from the sameward.



LAB ASSIGNMENT:

1. Analyze thedatarequired.
2. Normalizetheattributes.
3. CreatethelogicaldatamodelusingE-Rdiagrams

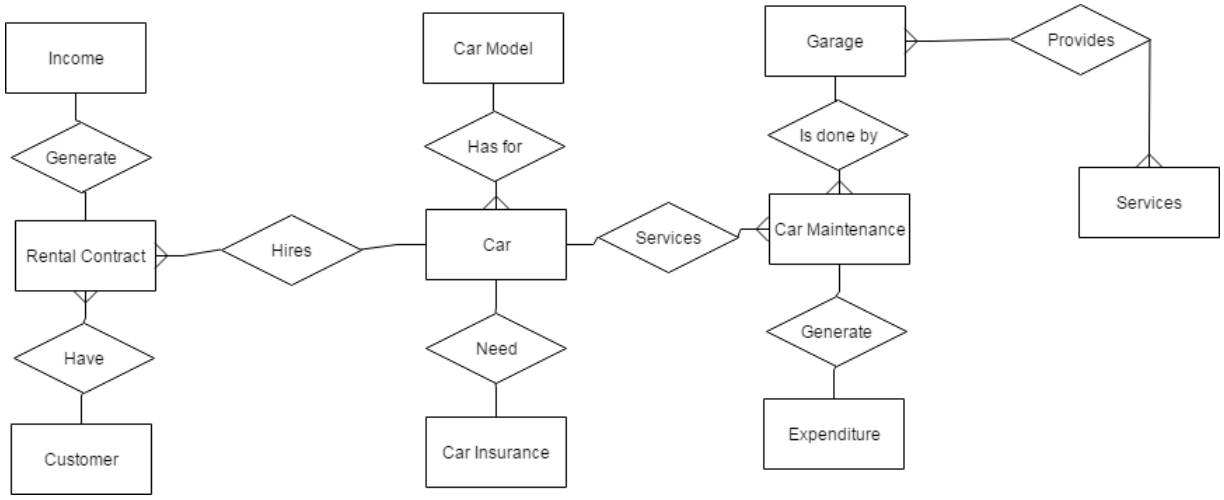
WEEK -11

CASE STUDY: CAR RENTAL COMPANY

AIM: A database is to be designed for a Car Rental Co. (CRC). The information required includes a description of cars, subcontractors (i.e. garages), company expenditures, company revenues and customers. Cars are to be described by such data as: make, model, year of production, engine size, and fuel type, number of passengers, registration number, purchase price, purchase date, rent price and insurance details. It is the company policy not to keep any car for a period exceeding oneyear.

All major repairs and maintenance are done by subcontractors (i.e. franchised garages), with whom CRC has long-term agreements. Therefore the data about garages to be kept in the database includes garage names, addresses, range of services and the like. Some garages require payments immediately after a repair has been made; with others CRC has made arrangements for credit facilities. Company expenditures are to be registered for all outgoings connected with purchases, repairs, maintenance, insurance etc.

Similarly the cash inflow coming from all sources - car hire, car sales, insurance claims - must be kept of file.CRCmaintains a reasonably stable client base. For this privileged category of customers special credit card facilities are provided. These customers may also book in advance a particular car. These reservations can be made for any period of time up to one month. Casual customers must pay a deposit for an estimated time of rental, unless they wish to pay by credit card. All major credit cards are accepted. Personal details (such as name, address, telephone number, driving license, number) about each customer are kept in thedatabase.



LAB ASSIGNMENT:

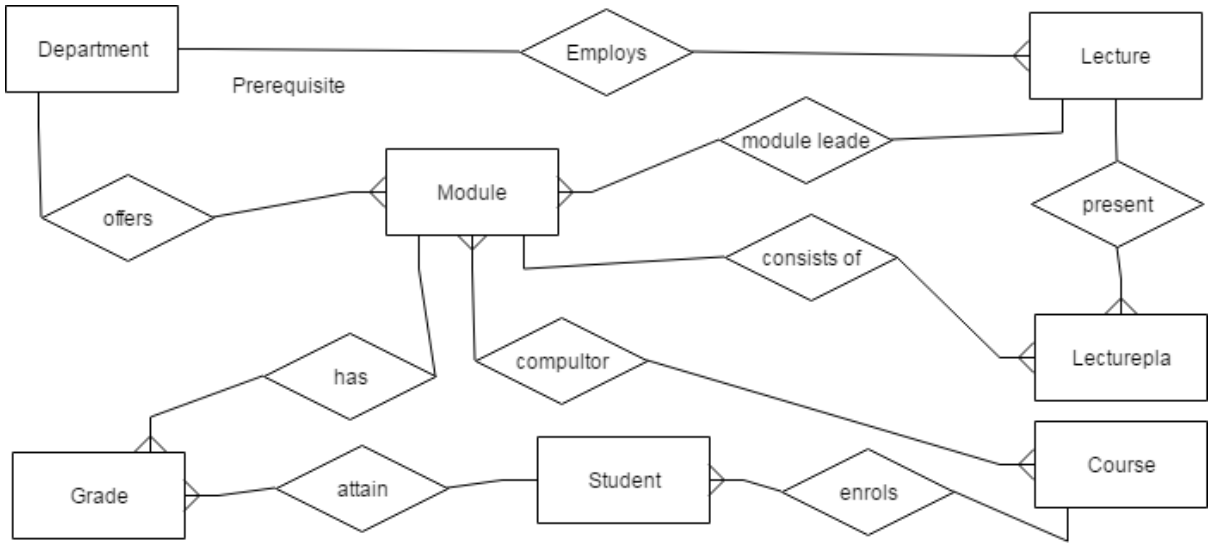
1. Analyze thedatarequired.
2. Normalizetheattributes.
3. CreatethelogicaldatamodelusingE-Rdiagrams

WEEK-12

CASE STUDY: STUDENT PROGRESS MONITORING SYSTEM

AIM: A database is to be designed for a college to monitor students' progress throughout their course of study. The students are reading for a degree (such as BA, BA(Hons) MSc, etc) within the framework of the modular system. The college provides a number of module, each being characterised by its code, title, credit value, module leader, teaching staff and the department they come from. A module is co- ordinated by a module leader who shares teaching duties with one or morelecturers.

A lecturer may teach (and be a module leader for) more than one module. Students are free to choose any module they wish but the following rules must be observed: some modules require pre- requisites modules and some degree programmes have compulsory modules. The database is also to contain some information about students including their numbers, names, addresses, degrees they read for, and their past performance (i.e. modules taken and examination results).



LAB ASSIGNMENT:

1. Analyze thedatarequired.
2. Normalizetheattributes.
3. CreatethelogicaldatamodelusingE-Rdiagrams