TPCT's

College of Engineering, Osmanabad.

Laboratory Manual

Of

Database Management System

For

Third Year Students

Dept. of Computer Science & Engineering

Manual Prepared by

Prof. Bhosale A.U.

Author

COE ,Osmanabad

TPCT's

College of Engineering
Solapur Road, Osmanabad
Department of Computer Science & Engg.

<u>Vision of the Department</u>:

To achieve and evolve as a center of academic excellence and research center in the field of Computer Science & Engineering. To develop computer engineers with necessary analytical ability & human values who can creatively design, implement a wide spectrum of computer system for welfare of the society.

<u>Mission of the Department:</u>

The department strives to continuously engage in providing the student with in-depth understanding of fundamentals and practical training related to professional skills & their application through effective Teaching-Learning Process and state of the art laboratories pertaining to CSE and interdisciplinary areas. Preparing students in developing research, design, entrepreneurial skills and employability capabilities.

College of Engineering

Technical Document

This technical document is a series of Laboratory manuals of Computer Science & Engg. Department and is a certified document of College of Engineering, Osmanabad. The care has been taken to make the document error-free. But still if any error is found. Kindly bring it to the notice of subject teacher and HOD

Recommended by,

HOD

Approved by,

Principal

FOREWORD

It is my great pleasure to present this laboratory manual for Third year engineering students for the subject of DBMS database concepts keeping in view the vast coverage required for understanding the concept of DMBS.

As a student, many of you may be wondering with some of the questions in your mind regarding the subject and exactly what has been tried is to answer through this manual.

Faculty members are also advised that covering these aspects in initial stage itself, will greatly relived them in future as much of the load will be taken care by the enthusiasm energies of the students once they are conceptually clear.

HOD
CSE DEPT

LABORATORY MANUAL CONTENTS

This manual is intended for the Third year students of Computer Science & Engineering in the subject of Database Management System. This manual typically contains practical/Lab Sessions related DBMS concept covering various aspects related the subject to enhanced understanding.

It is my great pleasure to present this laboratory manual for Third year engineering students for the subject of Database Management System. As a student, many of you may be wondering with some of the questions in your mind regarding the subject and exactly what has been tried is to answer through this manual.

Students are advised to thoroughly go through this manual rather than only topics mentioned in the syllabus as practical aspects are the key to understanding and conceptual visualization of theoretical aspects covered in the books.

Prof. A.U.Bhosale
Subject In-charge

SUBJECT INDEX

1. Do’s & Don’ts in Laboratory.
2. Lab Exercises
3. Quiz
4. Conduction of viva voce examination
5. Evaluation & marking scheme

1. Dos and Don’ts in Laboratory:

1. Make entry in the Log Book as soon as you enter the Laboratory. All the students should sit according to their roll numbers starting from their left to right.

2. Read carefully the power rating of the equipment before it is switched ON ,whether rating 230 V/ 50 HZ or 115 V/60 HZ. For Indian equipment the power ratings are normally 230 V/50HZ. If you have equipment with 115/60HZ ratings, do not insert power plug, which will damage the equipment.

3. Do not change the terminal on which you are working.

4. All the students are expected to get at least the algorithm of the program/concept to be implemented.

5. Strictly observe the instructions given by the teacher/Lab Instructor.

Instruction for Laboratory Teachers:-

1. Submission related to whatever lab work has been completed should be done during the next lab session.

2. Students should be instructed to switch on the power supply after getting the checked by the lab assistant/teacher.

3. The promptness of submission should be encouraged by way of marking and evaluation patterns that will benefit the sincere students.

2. Lab Exercises

1. Implementation of DDL commands of SQL with suitable examples
   - Create table
   - Alter table
   - Drop Table
2. Implementation of DML commands of SQL with suitable examples
   - Insert
   - Update
   - Delete

3. Implementation of different types of function with suitable examples
   - Number function
   - Aggregate Function
   - Character Function
   - Conversion Function
   - Date Function

4. Implementation of different types of operators in SQL
   - Arithmetic Operators
   - Logical Operators
   - Comparison Operator
   - Special Operator
   - Set Operation

5. Implementation of different types of Joins
   - Inner Join
   - Outer Join
   - Natural Join etc..

6. Study and Implementation of
   - Group By & having clause
   - Order by clause
   - Indexing

7. Study & Implementation of
   - Sub queries
   - Views

8. Study & Implementation of different types of constraints.
9. Study & Implementation of Database Backup & Recovery commands, Rollback, Commit, Savepoint.

10. Creating Database/ Table Space

EXPERIMENT NO : 01

AIM : IMPLEMENTATION OF DDL COMMANDS OF SQL.

1)CREATE TABLE

2)ALTER TABLE

3)DROP TABLE

Objective:

- To understand the different issues involved in the design and implementation of a database system.
- To understand and use data definition language to write query for a database

Theory:

Oracle has many tools such as SQL * PLUS, Oracle Forms, Oracle Report Writer, Oracle Graphics etc.

- **SQL * PLUS**: The SQL * PLUS tool is made up of two distinct parts.
- **Interactive SQL:** Interactive SQL is designed for create, access and manipulate data structures like tables and indexes.
- **PL/SQL:** PL/SQL can be used to developed programs for different applications.
- **Oracle Forms:** This tool allows you to create a data entry screen along with the suitable menu objects. Thus it is the oracle forms tool that handles data gathering and data validation in a commercial application.
- **Report Writer:** Report writer allows programmers to prepare innovative report using data from the oracle structures like tables, views etc. It is the report writer tool that handles the reporting section of commercial application.
- **Oracle Graphics:** Some of the data can be better represented in the form of pictures.

The oracle graphics tool allows programmers to prepare graphs using data from oracle structures like tables, views etc.

SQL (Structured Query Language):

Structured Query Language is a database computer 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's relational model and became the most widely used language for relational databases.

- IBM developed SQL in mid of 1970’s.
- Oracle incorporated in the year 1979.
- SQL used by IBM/DB2 and DS Database Systems.
- SQL adopted as standard language for RDBS by ASNI in 1989.

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 255 characters.

2. VARCHAR (Size) / VARCHAR2 (Size): This data type is used to store variable length alphanumeric data. The maximum character can hold is 2000 character.

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 38 digits.

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 current month.

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 be applied.

6. RAW: The RAW data type is used to store binary data, such as digitized picture or image. Data loaded into columns of these data types are stored without any further conversion. RAW data type can have a maximum length of 255 bytes. LONG RAW data type can contain up to 2GB.

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)

```
SQL>create table cust(Inamevarchar(20),fnamevarchar(20),address char(10));
```

```
insert into cust3 values('Kumbhar','Radha','Osmanabad');
```

```
SQL>desccust;
```

```
Name        Null?  Type
-------------------------------------------------------------------------------------------
LNAME                                      VARCHAR2(20)
FNAME                                      VARCHAR2(20)
ADDRESS                                     CHAR(10)
SQL>alter table cust
2add cellnovarchar(10)
3;
Table alerted.
SQL>desccust;
2 modify Inamechar(25);
Table alerted.
SQL>desccust;
Name                          Null?  Type
------------------------------ ------ -------------------------------------------------
LNAME                                      CHAR2(25)
FNAME                                      VARCHAR2(20)
ADDRESS                                     CHAR(10)
CELLNO                                      VARCHAR2(10)
SQL>ed
wrote file afiedt.buf
1 alter table cust
2 *rename column Iname to last_name
SQL>/
Table alerted.
SQL>desccust;
```

```
Name                    Null?  Type
-------------------------------------------------------------------
LNAME           CHAR2(25)
FNAME           VARCHAR2(20)
ADDRESS          CHAR(10)
CELLNO          VARCHAR2(10)
SQL>alter table cust
2 drop column cellno;
Table alerted.
SQL>desccust;
Name      Null?      Type
------------------------------------------------------------------------
LNAME                                    CHAR2(25)
FNAME                                    VARCHAR2(20)
ADDRESS                                   CHAR(10)
CELLNO                                    VARCHAR2(10)
SQL>desccust;
Name       Null?  Type
----------------------------------------------------------------------
LNAME                                    CHAR2(25)
FNAME                                    VARCHAR2(20)
ADDRESS                                   CHAR(10)
CELLNO                                    VARCHAR2(10)
SQL>drop table cust
2;
```

```
Table dropped.
```

RESULT: The above SQL queries was successfully executed and verified.

EXPERIMENT NO : 02

AIM : IMPLEMENTATION OF DML COMMANDS OF SQL.

1)INSERT

2)UPDATE

3)DELETE

Objective :

- To understand the different issues involved in the design and implementation of a database system
- To understand and use data manipulation language to query, update, and manage a database

Theory :

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

a) Inserting a single record

Syntax: INSERT INTO < relation/table name> (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’);

b) Inserting a single record

Syntax: INSERT INTO < relation/table name>VALUES (data_1,data_2,........data_n);

Example: SQL>INSERT INTO student VALUES (1,’Ravi’,’M.Tech’,’Palakol’);

c) Inserting all records from another relation

Syntax: INSERT INTO relation_name_1 SELECT Field_1,field_2,field_n FROM relation_name_2 WHERE field_x=data;

Example: SQL>INSERT INTO std SELECT sno,sname FROM student WHERE name = ‘Ramu‘;

d) Inserting multiple records

Syntax: INSERT INTO relation_name field_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: Pooja
Enter value for class: B.Tech
Enter value for address: Osmanabad
```

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

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

Example: SQL>UPDATE student SET sname = ‘kumar’ WHERE sno=1;

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

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

Syntax: SQL>DELETE FROM relation_name;

Example: SQL>DELETE FROM std;

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

Syntax: SQL>DELETE FROM relation_name WHERE condition;

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

```
SQL> create table cust2 (lnamevarchar(20),fnamevarchar(20),address char(20));

Table created.

SQL>desccust;

Name                    Null?      Type

-------------------------------------------------------------------------------------------

LNAME                              VARCHAR (20)

FNAME                              VARCHAR (20)

ADDRESS                             CHAR (20)
```

```
SQL> insert into cust2 values('Kumbhar','Radha','AT TULIAPUR');

insert into cust2 values('Kumbhar','Radha','AT TULIAPUR')

SQL> ED

Wrote filebafiedt.buf

1* insert into cust2 values('Kumbhar','Radha','AT TULIAPUR');

SQL>/

1 row created.

SQL> ED

Wrote filebafiedt.buf

1* insert into cust2 values('Shelke','D','KALLAM');

SQL>/

1 row created.

SQL> ED

Wrote filebafiedt.buf

1* insert into cust2 values('Gourkar','G','OSMANABAD');

SQL>/

1 row created.

SQL> ED

Wrote filebafiedt.buf

1* insert into cust2 values('Patil','R','SOLAPUR');

SQL>/

1 row created.

SQL> ED

Wrote filebafiedt.buf

1*insert into cust2 values('Gore','Rahul','MOHAL');
```

```
SQL>/

1 row created.

SQL> select * from cust2;

LNAME              FNAME       ADDRESS

------------------------------------------------------------------

KumbharRadha                   TULIPUR

Shelke              D                KALLAM

Gourkar            G                OSMANABAD

Patil                R                SOLAPUR

Gore                Rahul          MOHAL

SQL> UPDATE cust2 set lname='kumbhar' where fname='G';

1 row updated.

SQL> select * from cust2;

LNAME               FNAME                ADDRESS

------------------------------------------------------------------

Kumbhar              Radha                       TULIPUR

Shelke                 D                             KALLAM

Kumbhar              G                             OSMANABAD

Patil                   R                             SOLAPUR

Gore                 Rahul                         MOHAL


SQL> DELET from cust2 where address='MOHAL';

1 row deleted.

SQL> select * from cust2;

LNAME                    FNAME            ADDRESS
```

```
----------------------------------------------------------------
Kumbhar                 Radha             TULIPUR
Shelke                   D                 KALLAM
Kumbhar                 G                 OSMANABAD
Patil                    R                 SOLAPUR
```

RESULT:

The above SQL queries was successfully executed and verified.

EXPERIMENT NO : 03

AIM :- IMPLEMENTATION OF DIFFERENT TYPE OF FUNCTION.

1)NUMBER FUNCTION

2)AGGREGATE FUNCTION

3)CHARACTER FUNCTION

Objective:

- To understand and implement various types of function in SQL.

NUMBER FUNCTION:

- Abs(n) :Select abs(-17) from dual;
- Exp(n): Select exp(6) from dual;
- Power(m,n): Select power(6,2) from dual;
- Mod(m,n): Select mod(10,3) from dual;
- Round(m,n): Select round(100.256,2) from dual;
- Trunc(m,n): ;Select trunc(100.256,2) from dual;
- Sqrt(m,n);Select sqrt(64) from dual;

AGGREGATE FUNCTION*:* 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 and SUM.

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 the column.
2. **SUM:** SUM followed by a column name returns the sum of all the values in that column.
3. **AVG:** AVG followed by a column name returns the average value of that column values

4. **MAX:** MAX followed by a column name returns the maximum value of that column.
5. **MIN:** MIN followed by column name returns the minimum value of that column.

STRING FUNCTIONS:

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

Ltrim: Returns a character expression after removing leading blanks.

```
SQL>SELECT LTRIM('SSMITHSS','S')FROM DUAL;
MITHSS
```

Rtrim: Returns a character string after truncating all trailing blanks

```
SQL>SELECT RTRIM('SSMITHSS','S')FROM DUAL;
SSMITH
```

Lower: Returns a character expression after converting uppercase character data to lowercase.

```
SQL>SELECT LOWER('DBMS')FROM DUAL;
dbms
```

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

```
SQL>SELECT UPPER('dbms')FROM DUAL;
DBMS
```

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;
```

```
SQL>create table TECSE(roll int,namevarchar(20),DBMS int,JAVAint);

Table created;

SQL>insert into TECSE values(1,'Kumbhar',15,25);

1 row inserted.

SQL>ED

Wrote file afiedt.buf
```

```
insert into TECSE values(2,'Mali',40,50);

SQL>/

1 row inserted.

SQL>ED

Wrote file afiedt.buf

insert into TECSE values(3,'Shelke',35,22);

SQL>/

1 row inserted.

SQL>ED

Wrote file afiedt.buf

insert into TECSE values(4,'Thorat',50,40);SQL>/

1 row inserted.

SQL>ED

Wrote file afiedt.buf

insert into TECSE values(5,'More',50,48);SQL>/

1 row inserted.

SQL>ED

Wrote file afiedt.buf

select * from TECSE;

ROLL        NAME                DBMS              JAVA

---------    ----------        ------------      -----------

1           KUNBHAR               15                  25

2           MALI                  40                  45

3          SHELKE                 35                  22
```

```
4          THORAT                    50                     42

5          MORE                      45                     48

SQL>select count(*) from TECSE;

COUNT(*)

--------------

5

SQL>select  min(DBMS) from TECSE;

MIN(DBMS)

--------------

15

SQL>select  max(DBMS) from TECSE;

MAX(DBMS)

--------------

50

SQL>select  avg(JAVA) from TECSE;

AVG(JAVA)

--------------

36.4

SQL>select  sum(JAVA) from TECSE;

SUM(JAVA)

--------------

182
```

RESULT:

The above SQL queries was successfully executed and verified.

EXPERIMENT NO : 04

AIM : IMPLEMENTATION OF DIFFERENT TYPES OF OPERATORS IN SQL.

1) AIRTHMETIC OPERATORS
2) LOGICAL OPERATORS
3) COMPARISION OPERATORS

Objective :

- To learn different types of operator.

Theory :

ARIHMETIC OPERATORS:

(+) : Addition - Adds values on either side of the operator .

(-):Subtraction - Subtracts right hand operand from left hand operand .

(*):Multiplication - Multiplies values on either side of the operator .

(/):Division - Divides left hand operand by right hand operand .

(^):Power- raise to power of .

(%):Modulus - Divides left hand operand by right hand operand and returns remainder.

LOGICAL OPERATORS:

AND : The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.

OR: The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.

COMPARISION OPERATORS:

(=): Checks if the values of two operands are equal or not, if yes then condition becomes true.

(!=): Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.

(< >): Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.

(>): Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true

(<): Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.

(>=): Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.

(<=): Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.

```
SQL>SELECT * FROM TECSE;

ROLL        NAME               DBMS        JAVA

---------   -----------        -----------  ---------

1           KUMBHAR            15           25

2           MALI               40           45

3           SHELKE             35           22

4           THORAT             50           42

5           MORE               45           48

SQL>SELCET ROLL,NAME,(DBMS+5) FROM TECSE

ROLL        NAME               ( DBMS +5)

---------   ------------       ----------------

1           KUMBHAR                 20

2           MALI                    45

3           SHELKE                  40

4           THORAT                  55

5           MORE                    50

SQL>SELECT * FROM TECSE;

ROLL          NAME               DBMS          JAVA

----------    --------           -----------   -----------

1             KUMBHAR            15            25
```

```
2                MALI                          40                 45

3                SHELKE                          35                22

4                THORAT                         50                42

5                MORE                            45                48

SQL>SELCET ROLL,NAME,(DBMS+JAVA)/2 AS PER FROM TECSE

ROLL             NAME                  PER

-----------          ------------                -----------

1                  KUMBHAR                20

2                  MALI                        42.5

3                  SHELKE                    28.5

4                  THORAT                     46

5                   MORE                       46.5

SQL>SELCET ROLL,NAME FROM TECSE WHERE DBMS>30 OR JAVA>35;

ROLL              NAME

----------          -------------

1                  KUMBHAR

2                 MALI

3                 SHELKE

4                 THORAT

5                 MORE

SQL>SELCET ROLL,NAME FROM TECSE WHERE DBMS>30 AND JAVA>32;

ROLL             NAME

----------          -------------

2                     MALI

4                    THORAT
```

```
SQL>SELCET ROLL,NAME FROM TECSE WHERE DBMS>30 ;

ROLL                    NAME

---------               --------------

2                         MALI

3                         SHELKE

4                         THORAT
```

RESULT:

The above SQL queries was successfully executed and verified.

EXPERIMENT NO : 05

AIM : IMPLEMENTATION DIFFERENT TYPES OF JOIN LIKE INNER JOIN OUTER JOIN,RIGHT JOIN,LEFT JOIN.

Objective :

- To implement different types of joins

Theory :

The SQL **Joins** clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each. The join is actually performed by the 'where' clause which combines specified rows of tables.

Syntax:

SELECT column 1, column 2, column 3...

FROM table_name1, table_name2

WHERE table_name1.column name = table_name2.columnname;

Types of Joins :

1. Simple Join
2. Self Join
3. Outer Join

Simple Join:

It is the most common type of join. It retrieves the rows from 2 tables having a common column and is further classified into

Equi-join :

A join, which is based on equalities, is called equi-join.

Example:

Select * from item, cust where item.id=cust.id;

In the above statement, item-id = cust-id performs the join statement. It retrieves rows from both the tables provided they both have the same id as specified by the where clause. Since the where clause uses the comparison operator (=) to perform a join, it is said to be equijoin. It combines the matched rows of tables. It can be used as follows:

- To insert records in the target table.
- To create tables and insert records in this table.
- To update records in the target table.
- To create views.
- To insert records in the target

```
SQL>select * from TECSE1

2 ;

ROLL             DIP              SDL

---------        --------------   ----------

1                45               23

2                49               48

3                40               41

5                43               41

7                48               43

6                42               42


6 rows selected.

SQL> select * from TECSE

2;

ROLLNO          DBMS         JAVA

--------------   -------------   ------------------

1 KUMBHAR      15            25

2 MALI           40           45

3 SHELKE         35            22

4 THORAT        50            42

5 MORE           45            48
```

```
SQL> select TECSE.ROLL,NAME,DBMS,JAVA,DIP,SDL from TECSE,TECSE1;

ROLL NO      NAME          DBMS        JAVA        DIP       SDL

----------        --------          ------------    ------------    ----------   ------------

1            KUMBHAR        15          25          45        23

1            KUMBHAR        15          25          49        48

1            KUMBHAR        15          25          40        41

1            KUMBHAR        15          25          43        41

1            KUMBHAR        15          25          48        43

1            KUMBHAR        15          25          42        42

2            MALI           40          45          45        23

2            MALI           40          45          49        48

2            MALI           40          45          40        41

2            MALI           40          45          43        41

2            MALI           40          45          48        41

2            MALI           40          45          42        42

3            SHELKE         35          22          45        23

3            SHELKE         35          22          49        48

3            SHELKE         35          22          40        41

3            SHELKE         35          22          43        41

3            SHELKE         35          22          48        43

3            SHELKE         35          22          42        42

4            THORAT         50          42          45        23

4            THORAT         50          42          49        48

4            THORAT         50          42          40        41

4            THORAT         50          42          43        41
```

```
ROLL NO        NAME          DBMS        JAVA        DIP       SDL
---------              --------          -----------       ------------    ----------   ------------
4                    THORAT         50                42          48          43
4                    THORAT         50                42          42           42
5                    MORE             45                 48          45           23
5                    MORE             45                 48          49           48
5                     MORE             45                48          40          41
5                     MORE             45                48          43           41
5                      MORE             45               48         48            43
5                      MORE             45              48         42             42
```

SQL> select TECSE.ROLL,NAME,DBMS,JAVA,DIP,SDL from TECSE,TECSE1 WHERE TECSE.ROLL=TECSE1.ROLL;

```
ROLLNO         NAME          DBMS        JAVA        DIP       SDL
--------------       -------------    ------------      ----------      ------------     ---------
1                     KUMBHAR      15            25             45           23
2                      MALI             40           45             49          48
3                      SHEKLE         35           22              40          41
5                       MORE           45           48              43          41
```

SQL> select TECSE.ROLL,NAME,DBMS,JAVA,DIP,SDL from TECSE LEFT JOIN TECSE.ROLL=TECSE1.ROLL;

```
ROLLNO        NAME           DBMS        JAVA        DIP       SDL
--------------      -------------         ------------    ----------   ------------     ---------
1                   KUMBHAR         15            25           45           23
2                    MALI                 40             45          49          48
```

```
3              SHEKLE              35             22          40           41

5                MORE                 45             48          43           41

4               THORAT             50             42         48            43
```

```
Wrote file afiedt.buf

1 * SELECT  TECSE1.ROLL,NAME,DBMS,JAVA,DIP,SDL from TECSE RIGHT JOIN
TECSE1 ON TECSE.ROLL=TECSE1.ROLL;

SQL>/

ROLL NO                NAME          DBMS        JAVA        DIP       SDL

--------------              -------------      ------------      ----------      ---------     -----------

1                           KUMBHAR          15             25          45          23

2                           MALI                 40             45          49          48

3                          SHEKLE             35            22           40         41

5                          MORE                 45            48           43         41

6

7

4 rows selected.
```

RESULT:

The above SQL queries was successfully executed and verified.

EXPERIMENT NO: 06

AIM : STUDY AND IMPLEMENTATION OF GROUP BY AND HAVING CLAUSE, ORDER BY CLAUSE.

Objective:

- To learn the concept of group functions.

Theory:

• **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 the relation.

GROUP BY-HAVING : The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions. The HAVING clause must follow the GROUP BY clause in a query and must also precede the ORDER BY clause if used.

ORDER BY: This query is used to display a selected set of fields from a relation in an ordered manner base on some field.

```
SQL>SELECT ROLL,NAME,DBMS,JAVA FROM TECSE ORDER BY NAME;

ROLL    NAME              DBMS            JAVA
-------------------------------------------------------------------------------------

1       KUMBHAR           15              25

2       MALI              40              45

3       MORE              45              48

 4      SHELKE            35              22

 5      THORAT           30               42

SQL>ED

Wrote file afiedt.buf

1*SELECT ROLL,NAME,DBMS,JAVA FROM TECSE ORDER BY DBMS;
```

```
SQL>/

ROLL     NAME                      DBMS            JAVA

------------------------------------------------------------------------------------------------------

1           KUMBHAR                 15                  25

2            SHELKE                    35                   22

3            MALI                        40                  45

4            MORE                       45                  48

SQL>ED

Wrote file afiedt.buf

1*SELECT ROLL,NAME,DBMS,JAVA FROM TECSE ORDER BY ROLL;

ROLL     NAME                        DBMS            JAVA

----------------------------------------------------------------------------------------------------------------

1           KUMBHAR                     15                25

2            MALI                            40                45

3            SHEKKE                        35                22

4            THORAT                        50                 42

5            MORE                           45                48

SQL>SELECT*FROM COEOS;

ID      NAME                           DEPT                    SALARY

--------------------------------------------------------------------------------------------------------------

1       KUMBHAR                     CSE                    5000

2       MALI                            MECH                  15000

3       MORE                           MECH                 17000

4       PATIL                            CSE                    12000

5       WADGAONKAR              CSE                    11000
```

```
6      SHELE                          CSE                    20000

7      KHIRSAGAR                   CSE                    2000

7 rows selected

SQL>SELECT DEPT,AVG(SALARY)AS AVG_SALARY FROM COEOS GROUP BY
DEPT;

DEPT                          AVG_SALARY

------------------------------------------------------------------------------------------

CSE                             10000

MECH                              16000

SQL>ED

Wrote file afiedt.buf

1*SELECT DEPT,SUM(SALARY)AS AVG_SALARY FROM COEOS GROUP BY DEPT;

SQL>/

DEPT                          AVG_SALARY

-------------------------------------------------------------------------------

CSE                        50000

MECH                        32000

SQL>ED


Wrote file afiedt.buf

1*SELECT DEPT,SUM(SALARY)AS AVG_SALARY FROM COEOS GROUP BY DEPT
HAVING AVG(SALARY)>10000;

SQL>/

DEPT                          AVG_SALARY

------------------------------------------------------------------------------------

MECH                        32000
```

```
SQL>SELECT*FROM COEOS;

2 WHERE DEPT='CSE';

ID          NAME                        DEPT          SALARY

-------------------------------------------------------------------------------------------------------------

1           KUMBHAR                 CSE                 5000

2           PATIL                        CSE             12000

3           WADGAONKAR          CSE             11000

4          SHELE                         CSE             20000

5          KHIRSAGAMR             CSE             2000

SQL>ED

Wrote file afiedt.buf

1 SELECT*FROM COEOS;

2WHERE DEPT='CSE';

3*AND SALARY =5000;

SQL>/

ID        NAME                      DEPT                SALARY

-------------------------------------------------------------------------------------------------------------

1         KHIRSAGAR                  CSE                 5000
```

RESULT:

The above SQL queries was successfully executed and verified.

EXPERIMENT NO : 07

AIM: STUDY AND IMPLEMENTATION OF SUBQUERIES AND VIEWS.

Objective:

- ✓ To perform nested Queries and joining Queries using DML command
- ✓ To understand the implementation of views.

Theory:

SUBQUERIES: The query within another is known as a sub query. A statement containing sub query is called parent statement. The rows returned by sub query are used by the parent statement or in other words A subquery is a SELECT statement that is embedded in a clause of another SELECT statement You can place the subquery in a number of SQL clauses:

- WHERE clause
- HAVING clause
- FROM clause
- OPERATORS(IN.ANY,ALL,<,>,>=,<= etc..)

Types

1. Sub queries that return several values

Sub queries can also return more than one value. Such results should be made use along with the operators in and any.

2. Multiple queries

Here more than one sub query is used. These multiple sub queries are combined by means of ‘and’ & ‘or’ keywords.

3. Correlated sub query

A sub query is evaluated once for the entire parent statement whereas a correlated Sub query is evaluated once per row processed by the parent statement.

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 does not store in the database. View is a query stored as an object.

```
SQL>SELECT *FROM COEOS;

 ID    NAME                        DEPT          SALARY

------------------------------------------------------------------------------------

 1      KUMBHAR                CSE              20000

5      WADGAONKAR       CSE              2000

2      KSHIRSAGAR          MECH         20000

7      PATIL                       MECH         21000

3      PAWAR                     MECH         23000

4      MORE                       MECH         24000

4      MORE                       MECH         24000

6      SHELKE                    MECH         22000

8 rows selected.

SQL>ED

Wrote file afiedt.buf

1*SELECT ID,NAME,DEPT,SALARY FROM COEOS WHERE SALARY>(SALARY)
FROM COEOS)

SQL>/

ID NAME                 DEPT            SALARY

-----------------------------------------------------------------------------

1 KUMBHAR             CSE              20000

2 KSHIRSAGAR      MECH            20000

7 PATIL                  MECH            21000

3 PAWAR                MECH             23000

4 MORE                MECH            24000

4 MORE                MECH             24000

6 SHELKE              MECH            22000

7 rows selected.
```

```
SQL>/ED

Wrote file afiedt.buf

1*SELECT ID,NAME,DEPT,SALARY FROM COEOS WHERE SALARY<(SELECT
AVG(SALARY)FROM COEOS)

SQL>/

ID NAME                    DEPT              SALARY

------------------------------------------------------------------------------------------

5 WADGAONKAR          CSE                  2000

SQL>SELECT*FROM COEOS;

ID NAME                       DEPT             SALARY

---------------------------------------------------------------------------------

1 KUMBHAR                   CSE                20000

5 WADGAONKAR          CSE                2000

2 KSHIRSAGAR             MECH            20000

7 PATIL                         MECH            21000

3 PAWAR                       MECH            23000

4 BHAKTE                     MECH            24000

4 BHAKTE                     MECH            24000

6 SHELKE                      MECH            22000

8 rows selected.

SQL>/CREATE VIEW COE AS SELCT ID,NAME,SALARY FROM COEOS;

view created.

SQL>/SELECT*FROM COE;

ID NAME                          SALARY

--------------------------------------------------------------------

1 KUMBHAR                    20000

5 WADGAONKAR           2000

2 KSHIRSAGAR              20000
```

```
7 PATIL                  21000

3 PAWAR                 23000

4 BHAKTE               24000

4 BHAKTE               24000

6 SHELKE                22000

8 rows selected.
```

RESULT:

The above SQL queries was successfully executed and verified.

EXPERIMENT NO : 08

AIM: STUDY AND IMPLEMENTATION OF DIFFURENT TYPE OF CONSTRAINTS LIKE PRIMARY KEY, NOT NULL, DEFAULT,CHECK,UNIQUE.

Objective:

- ✓ To practice and implement constraints

Theory:

CONSTRAINTS:

Constraints are used to specify rules for the data in a table. If there is any violation between the constraint and the data action, the action is aborted by the constraint. It can be specified when the table is created (using CREATE TABLE statement) or after the table is created (using ALTER TABLE statement).

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 the table.

2. UNIQUE: The purpose of a unique key is to ensure that information in the column(s) is unique 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 unique keys.

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 a null).

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 Master Table.
It must uniquely identify each record in a table.
It must contain unique values.
It cannot be a null field.
It cannot be multi port field.
It should contain a minimum no. of fields necessary to be called unique.

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 **master table**.

```
SQL>CREATE TABLE CSE1(ROLL INT PRIMARY KEY,NAME VARCHAR(20),CITY
VARCHAR(20));
```

```
TABLE CREATED.

SQL>DESC CSE1

NAME                NULL?                          TYPE
---------------------------------------------------------------------------------
ROLL                NOT NULL                 NUMBER(38)

NAME                                         VARCHAR(20)

CITY                                         VARCHAR(20)

SQL>SQL>ED

Wrote file afiedt.buf

1* CREATE TABLAE CSE2(ROLL INT PRIMARY KEY,NAME VARCHAR(20) NOT
NULL, CITY VARCHAR(20))

SQL>/

TABLE CREATED

SQL>DESC CSE2

NAME                      NULL                           TYPE
---------------------------------------------------------------------------------
ROLL                      NOT  NULL                      NUMBER(38)

NAME                      NOT NULL                       VARCHAR(20)

CITY                                                     VARCHAR(20)

SQL>ED

      Wrote file afiedt.buf

1* CREATE TABLAE STUDENT(ROLL INT UNIQUE, NAME VARCHAR(20) , CITY
VARCHAR(20))

SQL>/

TABLE CREATED

SQL>DESC  STUDENT

NAME                      NULL                 TYPE
---------------------------------------------------------------------------------
ROLL                NOT                  NUMBER(38)
```

```
NAME                                    VARCHAR(20)

CITY                                    VARCHAR(20)

1* CREATE TABLAE STUDENT1(ROLL INT UNIQUE ,NAME VARCHAR(20) , CITY
VARCHAR(20),MARK INT DEFAULT 40)

TABLE CREATED

SQL>DESC STUDENT

NAME                NULL                             TYPE
----------------------------------------------------------------------------------
ROLL                NOT                 NUMBER(38)

NAME                                    VARCHAR(20)

CITY                                    VARCHAR(20)

SQL>ED

      Wrote file afiedt.buf

1* CREATE TABLAE STUDENT1 (ROLL INT UNIQUE,NAME VARCHAR(20), CITY
VARCHAR(20),MARK INT DEFAULT 40)

SQL>/

TABLE CREATED

SQL>DESC STUDENT1

NAME                      NULL                 TYPE
----------------------------------------------------------------------------------
ROLL                    NOT                   NUMBER(38)

NAME                                           VARCHAR(20)

CITY                                          VARCHAR(20)

SQL>INSERT INTO STUDENT1 VALUES(1,’PRAGATI’,’AUSA’,84);

1 row created.

SQL>ED

Wrote file afiedt.buf

1* INSERT INTO STUDENT1 VALUES(2,’SWARUPA’,’LOHARA’,64);
```

```
SQL>/

INSERT INTO STUDENT1 VALUES(2,’SWARUPA’,’LOHARA’,64);

ERROR AT LINE 1

ORA-00947:not enough values
```

RESULT:

The above SQL queries was successfully executed and verified.

EXPERIMENT NO : 09

AIM: STUDYAND IMPLIMENTATION OF DATABAS BACKUP,ROLLBACK.

Objective:

- ✓ To understand the concept of administrative commands

***Theory*:**

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 made permanent to the database only if they are committed a transaction begins with an executable SQL statement & ends explicitly with either rollback 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 the database.

Syntax: SQL> COMMIT;

Example: SQL> COMMIT;

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 role back.

Syntax: SQL> SAVE POINT ID;

Example: SQL> SAVE POINT xyz;

3. ROLLBACK: A role back command is used to undo the current transactions. We can role 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 role back.

Syntax: ROLLBACK (current transaction can be role back)
ROLLBACK to save point ID;

Example: SQL> ROLLBACK;
SQL> ROLLBACK TO SAVE POINT xyz;

RESULT:

The above SQL queries was successfully executed and verified.

EXPERIMENT NO:10

Aim *:* CREATING DATABASE/ TABLE SPACE

Objective:

- ✓ To understand the concept of administrative commands

Theory:

DATABASE is collection of coherent data.

To create database we have :

Syntax: CREATE DATABASE <database_name>

Example : CREATE DATABASE my_db;

TABLESPACE:

The oracle database consists of one or more logical storage units called *tablespaces.* Each tablespace in an Oracle database consists of one or more files called *datafiles,* which are physical structures that confirm to the operating system in which Oracle is running.

Syntax:

CREATE<tablespace name> DATAFILE'C:\oraclexe\app\oracle\product\10.2.0\server \<file name.dbf 'SIZE 50M;

Example:

Create tablespace te_cs DATAFILE 'C:\oraclexe\app\oracle\product\10.2.0\server\usr.dbf 'SIZE 50M;

CREATE USER:

The DBA creates user by executing CREATE USER statement. The user is someone who connects to the database if enough privilege is granted.

Syntax:

SQL> CREATE USER < username> -- (name of user to be created) IDENTIFIED BY <password> -- (specifies that the user must login with this password)

SQL> user created

Eg: create user *James* identified by *bob*;

(The user does not have privilege at this time, it has to be granted.These privileges determine what user can do at database level.)

PRIVILEGES:

A privilege is a right to execute an SQL statement or to access another user's object In Oracle, there are two types of privileges

- ✓ System Privileges
- ✓ Object Privileges

▪ **System Privileges *:*** are those through which the user can manage the performance of database actions. It is normally granted by DBA to users.

Eg: Create Session,Create Table,Create user etc..

▪ **Object Privileges *:*** allow access to objects or privileges on object, i.e. tables, table

columns. tables,views etc..It includes alter,delete,insert,select update etc.
(After creating the user, DBA grant specific system privileges to user)

GRANT:

The DBA uses the GRANT statement to allocate system privileges to other user.

Syntax:

SQL> GRANT privilege [privilege…. …] TO USER ;

SQL> Grant succeeded

Eg: Grant create session, create table, create view to James;

Object privileges vary from object to object.An owner has all privilege or specific privileges on object.

SQL> GRANT object_priv [(column)] ON object TO user;

SQL>GRANT select, insert ON emp TO James;

SQL>GRANT select ,update (e_name,e_address) ON emp TO James;

CHANGE PASSWORD:

The DBA creates an account and initializes a password for every user.You can change password by using ALTER USER statement.

Syntax:

Alter USER <*some user name*> IDENTIFIED BY<*New password*>

Eg: ALTER USER James IDENTIFIED BY sam

REVOKE:

REVOKE statement is used to remove privileges granted to other users.The privileges you specify are revoked from the users.

Syntax:

REVOKE [privilege.. …] ON *object* FROM *user*

Eg:

REVOKE create session,create table from James;

REVOKE select ,insert ON emp FROM James

ROLE:

A role is a named group of related privileges that can be granted to user. In other words, role is a predefined collection of previleges that are grouped together, thus privileges are easier to assign user.

SQL> Create role custom;

SQL> Grant create table, create view TO custom;

SQL> Grant select, insert ON emp TO custom;

Eg: Grant custom to James, Steve;

RESULT:

The above SQL queries was successfully executed and verified.

3. QUIZ

1. Define DCL?

The DCL language is used for controlling the access to the table and hence securing the database. DCL is used to provide certain privileges to a particular user. Privileges are rights to be allocated.

2. **List the DCL commands used in data bases**

The privilege commands are namely, Grant and Revoke

3. **Write the syntax for grant command**

Grant < database_priv [database_priv…..] > to <user_name> identified by <password> [,<pass word…..];

Grant <object_priv> | All on <object> to <user | public> [With Grant Option];

4. **What are TCL commands?**

*Commit *Rollback *save point

5. What are single row functions?

A single row function or scalar function returns only one value for every row queries in table. Single row function can appear in a select command and can also be included in a where clause. The single row function can be broadly classified as,

* Date Function * Conversion Function

* Numeric Function * Miscellaneous Function

*Character Function

6. List some character functions

initcap(char);

lower (char);

upper (char);

ltrim (char,[set]); rtrim (char,[set]);

7. What is a view?

A view is a logical table based on a table or another view. A view contains no data of its own but is like a window through which data from tables can be viewed or changed.

8. List any two advantages of view?

1. Hides data complexity.

2. Simplifies the usage by combining multiple tables into a single table

9. List the set operations of SQL?

1) Union 2) Intersect operation 3) The except operation (minus)

10. What is the use of sub Queries?

A sub Queries is a select-from-where expression that is nested with in another Queries. A common use of sub Queries is to perform tests for set membership, make set comparisons and determine set cardinality.

11. Define the terms DDL:

Data base schema is specified by a set of definitions expressed by a special language
called a data definition language.

12.What are the categories of SQL command?

SQL commands are divided in to the following categories:

Data Delimitation language

Data manipulation language

Data control language

Transaction Control Language

13. What is integrity constraint?

An integrity constraint is a mechanism used by oracle to prevent invalid data entry into
the table. It has enforcing the rules for the columns in a table.

14. List the types of constraint.

a) Domain Integrity

b) Entity Integrity

c) Referential Integrity

15. Primary Key Constraint

A primary key avoids duplication of rows and does not allow null values. It can be defined on one or more columns in a table and is used to uniquely identify each row in a table. These values should never be changed and should never be null.

16. Referential Integrity

It enforces relationship between tables. To establish parent-child relationship between 2 tables having a common column definition, we make use of this constraint. To implement this, we should define the column in the parent table as primary key and same column in the child table as foreign key referring to the corresponding parent entry.

17. What is DML?

DML commands are the most frequently used SQL commands and is used to query and manipulate the existing database objects.

18. What are DML command?

Some of the commands are Insert, Select, Update, Delet

19. Give the general form of SQL Queries? Select

A1, A2…………., An

From R,1R2……………, R m Where P

20. What is the use of rename operation?

Rename operation is used to rename both relations and an attributes. It uses the as clause, taking the form: Old-name as new-name

21. Define tuple variable?

Tuple variables are used for comparing two tuples in the same relation. The tuple variables are defined in the from clause by way of the as clause.

22. Write the syntax to retrieve specific columns from a table:

Syntax: Select column_name1, …..,column_name from table name;

4. Conduction of Viva –Voce Examinnations

Teacher should conduct oral exams of the students with full preparation. Normally, the objective questions with guess are to be avoided. To make it meaningful, the questions should be such that depth of the students in the subject is tested. Oral examinations are to be conducted in cordial environment amongst the teachers taking the examination. Teachers taking such examinations should not have ill thoughts about each other and courtesies should be offered to each other in case of difference of opinion, which should be critically suppressed in front of the students.

5. Evaluation and marking system:

Basic honesty in the evaluation and marking system is absolutely essential and in the process impartial nature of the evaluator is required in the examination system to become. It is a primary responsibility of the teacher to see that right students who are really putting up lot of hard work with right kind of intelligence are correctly awarded.

The marking patterns should be justifiable to the students without any ambiguity and teacher should see that students are faced with just circumstances.