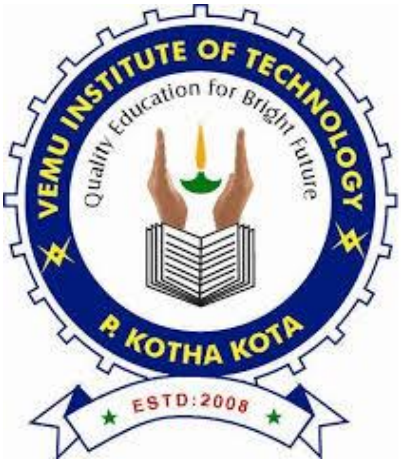
VEMU INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

LAB MANUAL



15A05303- DATABASE MANAGEMENT SYSTEMS LABORATORY

Regulation – R15

Year / Semester: II / I

INDEX

6	9. The percentage of hra and da are to be stored separately. 10. When the da becomes more than 100%, a message has to be generated and with user permission da has to be merged with basic. 11. Emp no should be unique and has to be generated automatically. 12. If the employee is going to retire in a particular month, automatically a message has to be generated. 13. The default value for date-of-birth is 1 jan, 1970.	18
7	14. 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. 15. Display the information of the employees and departments with description of the fields. 16. Display the average salary of all the departments. 17. Display the average salary department wise. 18. Display the maximum salary of each department and also all departments put together.	19

8	19. Commit the changes whenever required and rollback if necessary. 8. Use substitution variables to insert values repeatedly. 9. Assume some of the employees have given wrong information about date-of birth. Update the corresponding tables to change the value. 10. Find the employees whose salary is between 5000 and 10000 but not exactly 7500. 11. Find the employees whose name contains 'en'. 12. Try to delete a particular deptno. What happens if there are employees in it and if there are no employees?	21
9	13. Create alias for columns and use them in queries. 14. List the employees according to ascending order of salary. 15. List the employees according to ascending order of salary in each department. 16. Use '&&' wherever necessary 17. 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.	23

10	18. The retirement age is 60 years. Display the retirement day of all the employees. 19. If salary of all the employees is increased by 10% every year, what is the salary of all the employees at retirement time. 20. Find the employees who are born in leap year. 21. Find the employees who are born on feb 29. 22. Find the departments where the salary of atleast one employee is more than 20000.	25
11	23. Find the departments where the salary of all the employees is less than 20000. 24. 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. 25. As a designer identify the views that may have to be supported and create views. 26. As a designer identify the PL/SQL procedures necessary and create them using cursors. 27. Use appropriate Visual programming tools like oracle forms and reports, visual basic etc to create user interface screens and generate reports.	26
12	Practice of Views.	
13	Gather the required information, draw ER diagrams, map them to tables, normalize, create tables, triggers, procedures, execute queries, create user interfaces, and generate reports to better understand the following DBMS concepts. 1.Student information system 2.APSRTC reservation system	
14	3. Hostel management 4.Library management 5.Indian Railways reservation	
15	6.Super market management: 7.Postal system	

16	8. Banking system 9. Courier system 10.Publishing house system	
17	i)Practice of Triggers & Cursors ii)Practice of PL/SQL Programmes	30

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.

solution:

- DDL is Data Definition Language
- CREATE - to create objects in the database
- ALTER - alters the structure of the database
- DROP - delete objects from the database
- TRUNCATE - remove all records from a table, including all spaces allocated for therecords are removed.
- COMMENT - add comments to the data dictionary
- GRANT - gives user's access privileges to database
- REVOKE - withdraw access privileges given with the GRANT command
- DML is Data Manipulation Language statements. Some examples:
- SELECT - retrieve data from the a database
- INSERT - insert data into a table
- UPDATE - updates existing data within a table
- DELETE - deletes all records from a table, the space for the records remain
- CALL - call a PL/SQL or Java subprogram
- EXPLAIN PLAN - explain access path to data
- LOCK TABLE - control concurrency
- DCL is Data Control Language statements. Some examples:
- COMMIT - save work done
- SAVEPOINT -identify a point in a transaction to which you can later roll back
- ROLLBACK - restore database to original since the last COMMIT
- SET TRANSACTION - Change transaction options like what rollback segment to use.

Basic SQL DDL Commands.

To practice basic SQL DDL Commands such as CREATE, DROP, etc.

1. SQL - CREATE TABLE

Syntax:

Create table tablename

(columnnamedatatype(size), columnnamedatatype(size));

Syntax-

CREATE TABLE TABLENAME

[(columnname, columnname, ……)]

AS SELECT columnname, columnname……..FROM tablename;

Example: CREATE TABLE Persons (
PersonIDint,
LastNamevarchar(255),
FirstNamevarchar(255),
Address varchar(255),
City varchar(255));

INPUT:

SQL>

RESULT: Table created;

2. SQL - ALTER TABLE

(a) Add New column

Syntax: ALTER TABLE *table_name*ADD *column_namedatatype*;

INPUT:

SQL> ALTER TABLE Persons ADD DateOfBirth date;

ID	LastName	FirstName	Address	City	DateOfBirth
1	Hansen	Ola	Timoteivn 10	Sandnes	

2	Svendson	Tove	Borgvn 23	Sandnes	
3	Pettersen	Kari	Storgt 20	Stavanger	

RESULT:

(b) Modify Column name,size and data type

Syntax:ALTER TABLE *table_name*MODIFY COLUMN *column_namedatatype*;

INPUT:

SQL> ALTER TABLE Persons ALTER COLUMN DateOfBirth year;

RESULT: table altered;

(c) Delete Column

INPUT:

SQL> ALTER TABLE Persons DROP COLUMN DateOfBirth;

RESULT: Table COLUMNdroped;

3. SQL - DROP TABLE

Syntax: DROP TABLE *table_name*;

INPUT:

SQL> DROP TABLE Shippers;

RESULT:

Table droped;

4**. TRUNCATE:**

The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself.

Syntax

TRUNCATE TABLE *table_name*;

INPUT:

SQL> TRUNCATE TABLE *dept*;

RESULT:

Basic SQL DML Commands.

To practice basic SQL DML Commands such as INSERT, DELETE, etc.

INSERT: It is possible to write the INSERT INTO statement in two ways.

Syntax:

The first way specifies both the column names and the values to be inserted:

INSERT INTO *table_name* (*column1, column2, column3, ...*)
VALUES (*value1, value2, value3, ...*);

The second way specifies the values to be inserted

INSERT INTO *table_name*

VALUES (*value1, value2, value3, ...*);

SQL> INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode,Country)VALUES ('Cardinal', 'TomB.Erichsen', 'Skagen21', 'Stavanger', '4006', 'Norway');

RESULT:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	MattiKarttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland

92	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway

UPDATE: The UPDATE statement is used to modify the existing records in a table.

Syntax:

UPDATE *table_name*
SET *column1 = value1, column2 = value2, ...*
WHERE *condition*;

Note: Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!

SQL>

The following SQL statement updates the first customer (CustomerID = 1) with a new contact person *and* a new city.

Example

UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;

RESULT:

You have made changes to the database. Rows affected: 1

DELETE:

The DELETE statement is used to delete existing records in a table.

Syntax

DELETE FROM *table_name*
WHERE *condition*;

Note: Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) that should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

SQL> DELETE FROM Customers

WHERE CustomerName='AlfredsFutterkiste';

Basic SQL DCL Commands.

To practice basic SQL DCL Commands such as COMMIT, ROLLBACK etc.

1. **COMMIT:**

The COMMIT command is the transactional command used to save changes invoked by a transaction to the database.

The COMMIT command is the transactional command used to save changes invoked by a transaction to the database. The COMMIT command saves all the transactions to the database since the last COMMIT or ROLLBACK command.

Syntax:

Commit;

INPUT:

SQL> Consider the CUSTOMERS table having the following records –

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

Following is an example which would delete those records from the table which have age = 25 and then COMMIT the changes in the database.

```
SQL> DELETE FROM CUSTOMERS
  WHERE AGE = 25;
SQL> COMMIT;
```

RESULT:

Thus, two rows from the table would be deleted and the SELECT statement would produce the following result.

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

2. ROLLBACK:

The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database. This command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.

The syntax for a ROLLBACK command is as follows −

```
ROLLBACK;
```

INPUT:

Consider the CUSTOMERS table having the following records −

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
```

```
| 4 | Chaitali |  25 | Mumbai    |  6500.00 |
| 5 | Hardik   |  27 | Bhopal    |  8500.00 |
| 6 | Komal    |  22 | MP        |  4500.00 |
| 7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

Following is an example, which would delete those records from the table which have the age = 25 and then ROLLBACK the changes in the database.

```
SQL> DELETE FROM CUSTOMERS
  WHERE AGE = 25;
SQL> ROLLBACK;
```

RESULT:

Thus, the delete operation would not impact the table and the SELECT statement would produce the following result.

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
| 1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
| 2 | Khilan   |  25 | Delhi     |  1500.00 |
| 3 | kaushik  |  23 | Kota      |  2000.00 |
| 4 | Chaitali |  25 | Mumbai    |  6500.00 |
| 5 | Hardik   |  27 | Bhopal    |  8500.00 |
| 6 | Komal    |  22 | MP        |  4500.00 |
| 7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

3. SAVEPOINT:

A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction.

The syntax for a SAVEPOINT command is as shown below.

```
SAVEPOINT SAVEPOINT_NAME;
```

This command serves only in the creation of a SAVEPOINT among all the transactional statements. The ROLLBACK command is used to undo a group of transactions.

The syntax for rolling back to a SAVEPOINT is as shown below.

```
ROLLBACK TO SAVEPOINT_NAME;
```

Following is an example where you plan to delete the three different records from the CUSTOMERS table. You want to create a SAVEPOINT before each delete, so that you can ROLLBACK to any SAVEPOINT at any time to return the appropriate data to its original state.

INPUT: Consider the CUSTOMERS table having the following records.

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

The following code block contains the series of operations.

SQL>

```
SQL> SAVEPOINT SP1;
Savepoint created.
SQL> DELETE FROM CUSTOMERS WHERE ID=1;
1 row deleted.
SQL> SAVEPOINT SP2;
```

```
Savepoint created.

SQL> DELETE FROM CUSTOMERS WHERE ID=2;

1 row deleted.

SQL> SAVEPOINT SP3;

Savepoint created.

SQL> DELETE FROM CUSTOMERS WHERE ID=3;

1 row deleted.
```

Now that the three deletions have taken place, let us assume that you have changed your mind and decided to ROLLBACK to the SAVEPOINT that you identified as SP2. Because SP2 was created after the first deletion, the last two deletions are undone .

RESULT:

```
SQL> ROLLBACK TO SP2;Rollback complete.
```

Notice that only the first deletion took place since you rolled back to SP2.

```
SQL> SELECT * FROM CUSTOMERS;

+----+----------+-----+-----------+----------+

| ID | NAME     | AGE | ADDRESS   | SALARY   |

+----+----------+-----+-----------+----------+

|  2 | Khilan   |  25 | Delhi     |  1500.00 |

|  3 | kaushik  |  23 | Kota      |  2000.00 |

|  4 | Chaitali |  25 | Mumbai    |  6500.00 |

|  5 | Hardik   |  27 | Bhopal    |  8500.00 |

|  6 | Komal    |  22 | MP        |  4500.00 |

|  7 | Muffy    |  24 | Indore    | 10000.00 |

+----+----------+-----+-----------+----------+

6 rows selected.
```

Basic SQL SELECT Command.

To practice basic SQL SELECT Command.

1. **SELECT with * Keyword:**

   SELECT column1, column2, columnN FROM table_name;

   OR

   SELECT * FROM table_name;

   INPUT:

   SQL> SELECT * FROM CUSTOMERS;

RESULT:

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

2. **SELECT with WHERE Keyword:**

```
SELECT column1, column2, columnN
FROM table_name
WHERE [condition]
```

INPUT:

```
SQL>  SELECT ID, NAME, SALARY
      FROM CUSTOMERS
      WHERE SALARY > 2000;
```

RESULT:

This would produce the following result −

```
+----+----------+----------+
| ID | NAME     | SALARY   |
+----+----------+----------+
|  4 | Chaitali |  6500.00 |
|  5 | Hardik   |  8500.00 |
|  6 | Komal    |  4500.00 |
|  7 | Muffy    | 10000.00 |
+----+----------+----------+
```

3. **SELECT with SubQueries:**

Syntax:

```
SELECT column_name [, column_name ]
   FROM   table1 [, table2 ]
WHERE  column_name OPERATOR
  (SELECT column_name [, column_name ]
  FROM table1 [, table2 ]
  [WHERE])
```

INPUT:

SQL>

```
  SELECT *FROM CUSTOMERS
  WHERE ID IN (SELECT ID
       FROM CUSTOMERS
       WHERE SALARY > 4500) ;
```

RESULT: This would produce the following result.

```
+----+----------+-----+---------+----------+
| ID | NAME     | AGE | ADDRESS | SALARY   |
+----+----------+-----+---------+----------+
|  4 | Chaitali |  25 | Mumbai  |  6500.00 |
|  5 | Hardik   |  27 | Bhopal  |  8500.00 |
```

```
| 7 | Muffy    | 24 | Indore  | 10000.00 |
+----+----------+-----+---------+----------+
```

4. SELECT with Like keyword (% and _ symbols):

The SQL **LIKE** clause is used to compare a value to similar values using wildcard operators. There are two wildcards used in conjunction with the LIKE operator.

- The percent sign (%)
- The underscore (_)

The percent sign represents zero, one or multiple characters. The underscore represents a single number or character. These symbols can be used in combinations.

The basic syntax of % OR * and _ is as follows −

```
SELECT FROM table_name
WHERE column LIKE 'XXXX%'

or

SELECT FROM table_name
WHERE column LIKE '%XXXX%'

or

SELECT FROM table_name
WHERE column LIKE 'XXXX_'
```

INPUT:

```
SQL> SQL> SELECT * FROM CUSTOMERS

WHERE SALARY LIKE '200%';
```

RESULT:

This would produce the following result −

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
+----+----------+-----+-----------+----------+
```

Writing Queries using GROUP BY and other clauses.

The SQL **GROUP BY** clause is used in collaboration with the SELECT statement to arrange identical data into groups. This GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.

The **HAVING Clause** enables you to specify conditions that filter which group results appear in the results.

The WHERE clause places conditions on the selected columns, whereas the HAVING clause places conditions on groups created by the GROUP BY clause.

To write queries using clauses such as GROUP BY, ORDER BY, etc. and retrieving information by joining tables.

Source tables: emp, dept, programmer, software, study.

Order by : The order by clause is used to display the results in sorted order.

The **HAVING Clause** enables you to specify conditions that filter which group results appear in the results.

The WHERE clause places conditions on the selected columns, whereas the HAVING clause places conditions on groups created by the GROUP BY clause.

Syntax

The following code block shows the position of the HAVING Clause in a query.

```
SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY
```

The HAVING clause must follow the GROUP BY clause in a query and must also precede the ORDER BY clause if used. The following code block has the syntax of the SELECT statement including the HAVING clause −

```
SELECT column1, column2
FROM table1, table2
WHERE [ conditions ]
GROUP BY column1, column2
```

```
HAVING [ conditions ]
ORDER BY column1, column2
```

Group by : The attribute or attributes given in the clauses are used to form groups. Tupleswith the same value on all attributes in the group by clause are placed in one group.

```
SELECT NAME, SUM(SALARY) FROM EMP
  GROUP BY NAME;
```

Having: SQL applies predicates (conditions) in the having clause after groups have beenformed, so aggregate function be used.

```
SQL>SELECT * FROM EMP
  ORDER BY NAME, SALARY;
SQL > SELECT ID, NAME, AGE, ADDRESS, SALARY
FROM CUSTOMERS
GROUP BY age
HAVING COUNT(age) >= 2;
```

Write and Practice of Simple Queries.

To writesimple queries and practice them.

1. **Create tables department and employee with required constraints.**

SQL>create table emp(empnonumber(5) primary key,

Enamevarchar(15),basic number(5),hra number(5),da number(5));

SQL>create table dept(deptnonumber(5) primary key,

deptnamevarchar(15),description number(5));

Result:

Table created;

2. Initially only the few columns (essential) are to be added. Add the remaining columns separately by using appropriate SQL command

SQL>alter table emp add (deduct number(5));

alter table emp add (gross number(5));

alter table emp add (net number(5));

alter table emp add (dob date);

Result:

Table altered;

3.Basic column should not be null

SQL> create table emp (empnonumber(5) primary key,

Enamevarchar(15),basic number(5) not null);

Result:

Table created;

4.Add constraint that basic should not be less than 5000.

SQL>alter table empadd(check(basic>5000));

RESULT:

Table altered;

5. Calculate hra,da,gross and net by using PL/SQL program.

```
SQL>SQL>
Declare
Bs number;
Hra number;
Da number;
Gs number;
Dt number;
Ns number;

Begin

BS: = &bs;

Dt: = 200;

Hra: = bs* 0.8;

Da=bs *0.15;

Gs:=bs+hra +da;

Ns:= da – dt;

Dbms-output. Put – line('Basic salary='11bs);

Dbms-output. Put –line ('HRA' = 11 hra);
```

```
Dbms – output put – line('OD' –' 11da);

Dbms – output put –line ('Gross salary = '11gs);

Dbms – output put-line (' professional fax = '11 PA);

Dbms – output put –line ('NC+ salary = '11ns);

End;
```

RESULT : The Execution have successfully completed

6. Whenever salary is updated and its value becomes less than 5000 a trigger has tobe raised preventing the operation.

SQL>create trigger emp 5000

```
Before insert on emp

For each row

begin

If(:new.sal<5000) then

Dbms_output.put_line('sal shouldn't be less than 5000');

Else

Dbms_output.put_line('sal can beentered into employee table ');

End if;

End;

/
```

RESULT:

Trigger created;

7. The assertions are: hra should not be less than 10% of basic and da should not beless than 50% of basic.

SQL>create trigger emphrada

```
Before insert on emp

for each row

Begin

If (:new.hra<((basic*10)/100)) then
```

```
Dbms_output.put_line('HRA shouldn't be less than 10% of basic');
      Else if (:new.da<((basic*50)/100)) then
Dbms_output.put_line('Da shouldn't be less than 50% of basic');
      Else
Dbms_output.put_line('HRA,DA can be entered intoemptable');
     End if;
    End if;
    End;
/
```

RESULT:

Trigger created;

8. The percentage of hra and da are to be stored separately.

SQL>

```
Create table hrada1 as select hra,da from empt;
```

RESULT:

Table created;

9. When the da becomes more than 100%, a message has to be generated and with user permission da has to be merged with basic.

SQL> create trigger dabasic

```
      Before insert on emp
     For each row
begin
If(:new.da>=:old.basic)then
Dbms_output.put_line('da should be merged with basic');
End if;
End;
/
```

RESULT:

Trigger created;

10.Empno should be unique and has to be generated automatically.

SQL>

```
create table empt(empno number(5) unique);
```

RESULT:

Table created;

11.If the employee is going to retire in a particular month, automatically a message has to be generated.

SQL>

```
declare
Empno number:=7782
Hiredate date:= '23-jan-1999';
C varchar(15);
Begin
C:=extract(month from add_months(hiredate,720));
Dbms_output.put_line('empno'||empno ||'retiers in c'||c);
End;
/
```

RESULT:

Empno 7782 retires in c 2

12. The default value for date-of-birth is 1 jan, 1970.

SQL>

```
create table empt (dob date default '1-jan-1970');
```

(or)

```
Alter table emptadd(dob date default '1-jan-1970');
```

RESULT:

Table created;

13. 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.

SQL>

```
Declare
```

```
Jobtypevarchar(20);
Sal number(7);
Begin
Jobtype:='dailywager';
Sal:=10000;
If(jobtype='dailywager') then
If(sal>=5000) then
Dbms_output.put_line('employee can't be added');
else
Dbms_output.put_line('employee can be added');
End if;
End if;
End;
/
```

RESULT:

employee can't be added

14. Display the information of the employees and departments with description of the fields.
SQL>

descempt;

Descdept;

RESULT:

Empt:

Name	**null?**	**Type**
Empno		number(5)
Ename		varchar2(15)
Job		varchar2(15)
Mgr		number(5)
Hiredate		date
Sal		number(7,2)
Comm		number(7,2)

Dept:

Name	null?	Type
deptno		number(5)
deptname		varchar2(15)
sal		number(7,2)
manager		number(5)

15. Display the average salary of all the departments.

SQL>

Select avg(sal) from empt;

RESULT:

AVG(SAL)

2073.21429

16. Display the average salary department wise.

SQL>select deptno ,avg(sal) from emptgroupbydeptno;

RESULT:

Deptnoavg(sal)	
30	1566.6666
20	2175
10	2916.6667

17.Display the maximum salary of each department and also all departments put together.
SQL>

select max(sal) from empgroupbydeptno union select max(sal) from emp;

RESULT :

max(sal)

2850

3000

5000

18. Commit the changes whenever required and rollback if necessary.

SQL>

Commit;

Rollback to r1;

RESULT:

Commit completed;

Rollback completed;

19. Use substitution variables to insert values repeatedly.

SQL>

insert into empt values(&empno,'&empname',&basic,&hra,&da,&gross);

RESULT:

1 row inserted;

20.Assume some of the employees have given wrong information about date-of birth. Update the corresponding tables to change the value.

SQL>up dateempt set dob= '1-JAN-1976' where empno=7369

RESULT I row update

21. Find the employees whose salary is between 5000 and 10000 but not exactly 7500.

SQL>select * from empt where (sal between 1600 and 3000) and (sal ! =2975);

RESULT

Eno	Ename	Job	MGR	Hiredate	SAL	COMM	DEPT NO	DOB
7499	Allen	salesman	7698	20-2-81	1600	300	30	12-12-81
7698	blake	Manager	7839	1-5-81	2850		30	29-2-81
7782	clark	Mnager	7839	9-6-81	2450		10	15-8-86
7788	scott	analyst	7566	9-12-82	3000		20	16-9-84
7902	ford	analyst	7566	3-12-81	3000		20	1-1-72

22. Find the employees whose name contains 'en'.

SQL>selectempname from empt where empname like ‘ % en’**;**

RESULT: Emp name

ALLEN

23. Try to delete a particular deptno. What happens if there are employees in it and if there are no employees.

SQL>delete fromempt where dept no=20;

5 rows deleted

RESULT delete from empt where dept no=50;

Row deleted

24. Create alias for columns and use them in queries.

SQL>select empno n from empt where empno =7429

Select empnosals fromempt where sals 5000

RESULT:

select ename,empno en from empt where empno= 7369

Ename En

Smith 7369

25. List the employees according to ascending order of salary.

SQL> select from empt order by sal ;

RESULT:

enamesalenamesal

Smith 800 scott 3000

James 950 Ford 3000

Adams 1100 King 5000

Ward 1250

Martin 1250

Miller 1300

Turer 1500

Allen 1600

Clark 2450

Blake 2850

Jones 2975

26. List the employees according to ascending order of salary in each department.

SQL>select emptname fromempt group by dept no order by sal ;

Select deptno min (sal) from empt group by deptno order by min (sal);

RESULT:

Dept no	min (sal)
20	800
30	950
10	1300

27. Use '&&' wherever necessary

SQL>

RESULT

28. 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

SQL> declare

Fund number (5): =6000;

Sal number (7): = 10000;

Month vocher (15) = 'JAN'

Number (5) ;

begin

if (month = 'JAN') then

If (sal,< '0') then

Sal = 1000

else

x: = sal – 6000;

end if ;

end ;

Result:

29. The retirement age is 60 years. Display the retirement day of all the employees.

SQL>select exetract(day from add – months(hire - date , 720)) from emp;

RESULT:

Extract (day from add – months (hire- date , 720))

17
20
22
2
29
1
9
9
17
8
12
3
3
23

30. If salary of all the employees is increased by 10% every year, what is the salary of all the employees at retirement time.

SQL>

decare

Sal number (n): =20000

Dob date : = ‘ 12 –MAR – 1980 ‘

C :=extract (year from dob);

Pyearnumber(5):=2017;

X:=pyear-c;

Begin

While(x<=60)

```
loop
sal:=sal+((sal*10)/100);
x:=x+1;
end loop;
dbms_output.put_line('salary is '||sal);
end;
                /
```

31. Find the employees who are born in leap year.

SQL>select * from empt where mod(extract(year from dob),4)=0;

32. Find the employees who are born on feb 29

SQL>select * from emp where (extract(day from dob)=28) and(extract(month from dob)=2);

RESULT:

Empno	ename	job	mgr	hiredate	sal	comm	dob
7934	miller	clerk	7782	23-jan-99	1300	10	28-feb-1979

33. Find the departments where the salary of atleast one employee is more than 20000.

SQL>select distinct(deptno) from empt where deptno in (select deptno from empt where sal>20000);

RESULT:

Deptno

30

20

10

34. Find the departments where the salary of all the employees is less than 20000.

SQL>

select distinct(deptno) from empt where deptno in (select deptno from empt where sal<20000);

RESULT:

Deptno

30

20

10

35. 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.

SQL>

```
Declare
Basic number(5):=10000;
Hiredate date:='10-jan-70';
C:= Extract(month from hiredate);
Begin
Basic:=basic+((basic*10)/100);
C:=c+1;
While(c>7)
Loop
Basic:=basic-((basic*2)/100);
C:=c+1;
End loop;
End;
/
```

36. As a designer identify the views that may have to be supported and create views.

SQL>

RESULT

VIVA QUESTIONS-1

1. What is DBMS?

2. What is DBA?
3. What is DDL?
4. What is DML?
5. What is Query?
6. What is Atomicity?
7. What is consistency?
8. What is constraints?
9. What is primary key?
10. What is foreign key?
11. Difference between delete and drop command?
12. .Difference between commit and rollback?
13. Applications of DBMS?
14. What is file systems?
15. What is SQL?
16. Advantages of DBMS?
17. Difference between primary key and foreign key?
18. what is a row?
19. what is a column?
20. what is the another name of rows and columns?

Practice of SELECT Query with various options

prctice of Views.

Other Practice Queries

VIVA QUESTIONS-2

1. What is the syntax of select ?
2. What is the groupby clause?
3. Why we go for where clause?
4. What is the usage of create?
5. What is the diiference between alter and update?
6. What is the syntax of del?
7. Define join ?
8. Types of joins?

9. Why we go for IN ?
10. Why we go for BETWEEN?
11. Difference between IN and NOTIN?
12. Syntax of DATE?
13. Difference between COUNT and DISTINCT COUNT?
14. What is relational algebra?
15. What is relational Calculus?
16. What is election?
17. What is projection?
18. Difference between DBMS and RDBMS?
19. What is Normalization?
20. What is Multivalued Dependencies?

3.forbetter understand the DBMS concepts. Students should gather the requiredinformation, draw ER diagrams, map them to tables, normalize, create tables, triggers,procedures, execute queries, create user interfaces, and generate reports.

1. Student information system:

2. APSRTC reservation system:

3. Hostel management:

4. Library management:

5. Indian Railways reservation:

6.Super market management:

7. Postal system:

8. Banking system:

9. Courier system:

10. Publishing house system:

VIVA QUESTIONS-3

1. What is the difference between DBMS and RDBMS?
2. What is mean by DDL Queries?
3. What are DML Queries?
4. What are the constrains are available in DBMS?

5. What is Entity?
6. What is relation?
7. What is weak entity?
8. What is strong entity?
9. Difference between weak and strong entity?
10. What is composition?
11. What is the usage of ER-diagram?
12. What is Trigger?
13. What is aggregation?
14. What is generalization?
15. What is dependency?
16. Types of normal forms?
17. What is the usage of integrity constraints?
18. What is Event?
19. What is Optimization?
20. What is Transaction?
21. What is ACID?
22. What is Decomposition?
23. What is Redundancy?
24. Syntax of OrderBy Clause?
25. What is relation? 26. What are the symbols used for E-R diagrams?

Practice of Triggers& Cursors

TRIGGERS

Triggers :

A data base trigger is a named PL/SQL block diagram in a data base and executed implicity when a triggering event occurs the act of executing a trigger the is called firing the trigger. A triggering can be one of the following

*A DML statement (such as insert, update, or DELETE) executed against a database table. Such a trigger can be fire after a triggering exent for example if you have defined a triggering to

fire before an insert statement of the statement the students table this trigger fired each time before you insert a row in the student table

*ADDL statement (such as CREATE or ALTER) executed eighter by a particular user against a schema or by any user such trigger are aften used for auditing purposes and age specifically helpful to oracle.

DBMS They are age specified record various schema changes when they were made, and by which user.

*A system event such as start up or shutdown of the data base.
A user event such as 10g on or 100 off example you can define a trigger that fixed after database. 10g on that record the username time to 10g on the general syntax for creating a trigger is a follower (the reserved word and phrases in braket are optional):

Create (or replace) trigger- Tigger – name {before e/after}
Triggering – event on table name for each row

(follows another- trigger)
(enable/ disable)
(where condition)

Declare
Declaration statement

Practice of PL/SQL Programmes

1. Write a PL/SQL Programme to calculate addition

SQL>

Delare

i number;

j number ;

k number ;

begin

i: = &I;

j: = &j;

k: = i+j;

dbms – output – line (‘sum’1) ill and 11= (11k);

end;

Output:

Enter value for j= 7 Enter the value of i= 6

Old 7:j; = &j; old 6: i: = &i;

New 7: j; = 7 new 6: I; =6;

Sum of 6 and 7= 12

2. Write a PL /SQL program to insert roll no from 15BF1A1245 to 15BF1A1249 (or) start number to ending number

s number;
e number;
i number;

For i in s..e

loop

Insert into student (sid) value (concoct (15BF1A…)):

End for ;

end

3. Write PL/ SQL Programme to print multiplication table

```
SQL > Declare
N= &n;
For I in 1 .. 10 100p
Dbms –output put –line (n11'*'11'11'='11 ixn);
End 100p;
End;
```

output:

```
enter the value for n=8
8x1=8
8x2=16
8x3=24
8x4=32
8x5=40
8x6=48
8x7=56
8x8=64
8x9=72
8x10=80
```

4. Write a PL/SQL Program for calculate the salary of the employee

```
SQL>Dec;are
Bs number;
Hra number;
Da number;
Gs number;
Dt number;
Ns number;
Begin
BS: = &bs;
Dt: = 200:
Hra: = bs* 0.8:
Da=bs *0.15;
Gs:=bs+hra +da;
Ns: da – dt;
Dbms-output. Put – line(‘Basic salary=’11bs);
Dbms-output. Put –line (‘HRA’ = 11 hra);
Dbms – output put – line(‘OD’ –‘ 11da);
Dbms – output put –line (‘Gross salary = ‘11gs);
Dbms – output put-line (‘ professional fax = ’11 PA);
Dbms – output put –line (‘NC+ salary = ‘11ns);
End;
```

Result:

The Execution have successfully completed.