Lab Manual

for

Database Management System Lab

3139

Diploma In Computer Engineering

3 rd Semester

By

SITTTR

Kalamassery

STATE INSTITUTE OF TECHNICAL TEACHERS TRAINING AND RESEARCH

GENERAL INSTRUCTIONS

Rough record and Fair record are needed to record the experiments conducted in the laboratory. Rough records are needed to be certified immediately on completion of the experiment. Fair records are due at the beginning of the next lab period. Fair records must be submitted as neat, legible, and complete.

INSTRUCTIONS TO STUDENTS FOR WRITING THE FAIR RECORD

In the fair record, the index page should be filled properly by writing the corresponding experiment number, experiment name , date on which it was done and the page number.

On the **right side** page of the record following has to be written:

1. **Title**: The title of the experiment should be written in the page in capital letters.

2. In the left top margin, experiment number and date should be written.

3. **Aim**: The purpose of the experiment should be written clearly.

4.**Apparatus/Tools/Equipments/Components used:** A list of the Apparatus/Tools /Equipments /Components used for doing the experiment should be entered.

5. **Principle**: Simple working of the circuit/experimental set up/algorithm should be written.

6. **Procedure:** steps for doing the experiment and recording the readings should be briefly described(flow chart/programs in the case of computer/processor related experiments)

7. **Results**: The results of the experiment must be summarized in writing and should be fulfilling the aim.

8. **Inference** : Inference from the results is to be mentioned.

On the **Left side** page of the record following has to be recorded:

1. **Circuit/Program**: Neatly drawn circuit diagrams/experimental set up.

2. **Design**: The design of the circuit/experimental set up for selecting the components

should be clearly shown if necessary.

3. **Observations:** i) Data should be clearly recorded using Tabular Columns.

ii) Unit of the observed data should be clearly mentioned

iii) Relevant calculations should be shown. If repetitive calculations are needed, only show a sample calculation and summarize the others in a table.

4. **Graphs :** Graphs can used to present data in a form that show the results obtained, as one or more of the parameters are varied. A graph has the advantage of presenting large

amounts of data in a concise visual form. Graph should be in a square format.

GENERAL RULES FOR PERSONAL SAFETY

1. Always wear tight shirt/lab coat , pants and shoes inside workshops.
2. REMOVE ALL METAL JEWELLERY since rings, wrist watches or bands, necklaces, etc. make excellent electrodes in the event of accidental contact with electric power sources.
3. DO NOT MAKE CIRCUIT CHANGES without turning off the power.
4. Make sure that equipment working on electrical power are grounded properly.
5. Avoid standing on metal surfaces or wet concrete. Keep your shoes dry.
6. Never handle electrical equipment with wet skin.
7. Hot soldering irons should be rested in its holder. Never leave a hot iron unattended.
8. Avoid use of loose clothing and hair near machines and avoid running around inside lab .

TO PROTECT EQUIPMENT AND MINIMIZE MAINTENANCE:

DO: 1. SET MULTIRANGE METERS to highest range before connecting to an unknown source.

2. INFORM YOUR INSTRUCTOR about faulty equipment so that it can be sent for repair.

DO NOT: 1. Do not MOVE EQUIPMENT around the room except under the supervision of an instructor.

INDEX

EXPERIMENT NO: 1

DATA DEFINITION LANGUAGE (DDL) COMMANDS

AIM:

To execute and verify the Data Definition Language commands.

OBJECTIVES

To understand DDL commands.

THEORY

The commands used are:

● CREATE - It is used to create a table.

● ALTER – The structure of a table can be modified by using the ALTER TABLE command. This command is used to add a new column, modify the existing column definition and to include or drop integrity constraint.

● DROP - It will delete the table structure provided the table should be empty.

● TRUNCATE - If there is no further use of records stored in a table and the structure has to be retained, and then the records alone can be deleted.

● DESC - This is used to view the structure of the table

PROCEDURE

CREATION OF TABLE:

SYNTAX:

```
create table<table name>(column1 datatype,column2  datatype...);
```

EXAMPLE:

```
SQL>CREATE TABLE Employee ( EmpNo number(5), EName VarChar(15),
Job Char(10) , DeptNo  number(3));
```

ALTER TABLE

(a) To Add column to existing Table

Syntax:

alter table table-name add(**column-name** datatype);

EXAMPLE:

ALTER TABLE Employee ADD (phone_no char (20));

(b)To Add Multiple columns to existing Table

Syntax:

alter table table-name add(**column-name1** datatype1, **column-name2** datatype2, **column-name3** datatype3);

EXAMPLE:

alter table Employee add(salary number(7), age(5));

(c) Dropping a Column from a Table

Syntax:

ALTER TABLE <Table Name>DROP COLUMN <CoumnName>;

EXAMPLE:

ALTER TABLE Employee DROP COLUMN phone_no ;

(d) Modifying Existing Columns

Syntax:

ALTER TABLE <Table Name>MODIFY (<CoumnName><Newdata type>(<size>));

EXAMPLE:

ALTER TABLE Employee MODIFY (EName VarChar(25));

(e) **To Rename a column**

Using alter command we can rename an existing column

Syntax:

alter table *table-name* **rename** old-column-name to column-name;

EXAMPLE:

alter table Employee rename address to Location;

RENAMING TABLES

Syntax:

Rename <oldtable> to <new table>;

EXAMPLE:
rename Employee to Employee 1;

TRUNCATE TABLE

.
Syntax:

 TRUNCATE TABLE <TABLE NAME>;

Example:

 Truncate table Employee;

DESTROYING TABLES

Syntax:

 DROP TABLE <TABLE NAME>;

Example:

 DROP TABLE Employee;

DESCRIBE TABLES

. Syntax:

```
DESC <TABLE NAME>;
```

Example:

```
desc employee;
```

RESULT:

The DDL commands have been executed successfully

Problems

1.Create the tables described below

Table Name : PRODUCT_MASTER

Description : used to store product information

Column name	Data type	size
PRODUCTNO	Varchar2	6
DESCRIPTION	Varchar2	15
PROFITPERCENT	Varchar2	4,2
UNITMEASURE	Varchar2	10
QTYONHAND	Number	8
REORDERLVL	Number	8
SELLPRICE	Number	8,2
COSTPRICE	Number	8,2

Table Name : CLIENT_MASTER

Description : used to store client information

Column name	Data type	size
CLIENTNO	Varchar2	6
NAME	Varchar2	20
ADDRESS1	Varchar2	30
ADDRESS2	Varchar2	30
CITY	Varchar2	15
PINCODE	Number	8
STATE	Varchar2	15
BALDUE	Number	10,2

Table Name : SALESMAN_MASTER

Description : used to store salesman information working for the company

Column name	Data type	size
SALESMANNO	Varchar2	6
SALESMANNAME	Varchar2	20
ADDRESS1	Varchar2	30
ADDRESS2	Varchar2	30
CITY	Varchar2	15
PINCODE	Number	8
STATE	Varchar2	15

Table Name : STUDENT

Description : used to store student information

Column name	Data type	size
SNO	Number	5
SNAME	Varchar2	20
AGE	Number	5
SDOB	Date	
SMARK1	Number	4,2
SMARK2	Number	4,2
SMARK3	Number	4,4

2. Exercise on altering the table structure

(a) Add a column called 'telephone' of data type 'number' and size ='10' to the Client _Master table.

(b)Change the size of Sellprice column in Product_Master to 10,2

3. **Exercise on deleting the table structure along with the data**

(a)Destroy the table Client_Master along with its data

4. **Exercise on renaming the table**

(a)Change the name of the Salesman_Master table to sman_mast

EXPERIMENT NO:2
CONSTRAINTS

AIM:

To implement Data Constraints.

THEORY

Constraints are the business Rules which are enforced on the data being stored in a table are called *Constraints*

TYPES OF CONSTRAINTS:
1) Primary key
2) Foreign key/references
3) Check
4) Unique
5) Not null
6) Null
7) Default

PROCEDURE

(a) **The PRIMARY KEY**

The PRIMARY KEY defined at column level

Syntax:

CREATE TABLE tablename (Columnname1 DATATYPE CONSTRAINT <constraintname1> PRIMARY KEY,Columnname2 DATATYPE, columnname3 DATATYPE,.....);

EXAMPLE

SQL>create table Employee(empno number(4) **primary key,**ename varchar2(10),job varchar2(6),sal number(5),deptno number(7));

The PRIMARY KEY defined at table level

Syntax:

CREATE TABLE tablename (Columnname1 DATATYPE, columnname2 DATATYPE, columnname3 DATATYPE, PRIMARY KEY (columnname1, columnname2));

EXAMPLE

(b)CHECK CONSTRAINT

The CHECK Constraint defined at column level

Syntax:

CREATE TABLE tablename
(Columnname1 DATATYPE CHECK (logical expression), columnname2 DATATYPE, columnname3 DATATYPE,...);

EXAMPLE

CREATE TABLE Employee(empno number(3),ename varchar2(20),design varchar2(15),sal number(5) CHECK(sal>500 and sal<10001),deptno number(2));

The CHECK Constraint defined at table level

Syntax:

CREATE TABLE tablename
(Columnname1 DATATYPE, columnname2 DATATYPE, columnname3 DATATYPE, CHECK (logical expression1), CHECK (logical expression2));

EXAMPLE

CREATE TABLE Employee(empno number(3),ename varchar2(20), design varchar2(15),sal number(5),deptno number(2), CHECK(sal>500 and sal<1000));

(c) UNIQUE CONSTRAINT

The UNIQUE Constraint defined at the column level

Syntax

CREATE TABLE tablename (Columnname1 DATATYPE UNIQUE, columnname2 DATATYPE UNIQUE, columnname3 DATATYPE ...);

EXAMPLE

sql>CREATE TABLE Employee(empno number(3),ename varchar2(20), design varchar2(15),sal number(5), UNIQUE(design));

The UNIQUE Constraint defined at the the table level

Syntax

CREATE TABLE tablename (Columnname1 DATATYPE, columnname2 DATATYPE, columnname3 DATATYPE, UNIQUE (columnname1));

EXAMPLE

sql>create table Employee(empno number(3),ename varchar2(20), design varchar2(15),sal number(5), UNIQUE(design));

(d) Not Null

Syntax

CREATE TABLE tablename(Columnname1 DATATYPE NOT NULL, columnname2 DATATYPE NOT NULL,columnname3 DATATYPE,...);

EXAMPLE

sql>CREATE TABLE Employee(empno number(4),ename varchar2(20) NOT NULL,design varchar2(20),sal number(3));

<u>Problems</u>

1.Create the tables described below

Table Name : PRODUCT_MASTER

Description : used to store product information

Column name	Data type	size	Attributes
PRODUCTNO	Varchar2	6	Primary key/first letter must start with ‘p’
DESCRIPTION	Varchar2	15	Not Null
PROFITPERCENT	Varchar2	4,2	Not Null
UNITMEASURE	Varchar2	10	Not Null
QTYONHAND	Number	8	Not Null
REORDERLVL	Number	8	Not Null
SELLPRICE	Number	8,2	Not Null,cannot be 0
COSTPRICE	Number	8,2	Not Null,cannot be 0

Table Name : CLIENT_MASTER

Description : used to store client information

Column name	Data type	size	Attributes
CLIENTNO	Varchar2	6	Primary key/first letter must start with ‘C’

NAME	Varchar2	20	Not Null
ADDRESS1	Varchar2	30	
ADDRESS2	Varchar2	30	
CITY	Varchar2	15	
PINCODE	Number	8	
STATE	Varchar2	15	
BALDUE	Number	10,2	

Table Name : SALESMAN_MASTER

Description : used to store salesman information working for the company

Column name	Data type	size	Attributes
SALESMANNO	Varchar2	6	Primary key/first letter must start with 'S'
SALESMANNAME	Varchar2	20	Not Null
ADDRESS1	Varchar2	30	Not Null
ADDRESS2	Varchar2	30	
CITY	Varchar2	15	
PINCODE	Number	8	
STATE	Varchar2	15	

EXPERIMENT NO 3
DATA MANIPULATION LANGUAGE

AIM:

To execute the Data Manipulation Language (DML) commands in RDBMS.

OBJECTIVES

To understand Data Manipulation Language (DML) commands

THEORY

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

1. **INSERT**
   This is used to add one or more rows to a table. The values are separated by commas and the data types char and date are enclosed in apostrophes. The values must be entered in the same order as they are defined.
2. **SELECT**
   It is used to retrieve information from the table.it is generally referred to as querying the table. We can either display all columns in a table or only specify column from the table.
3. **UPDATE**
   It is used to alter the column values in a table. A single column may be updated or more than one column could be updated.
4. **DELETE**
   After inserting row in a table we can also delete them if required. The delete command consists of a from clause followed by an optional where clause

PROCEDURE

INSERT COMMAND

(a) Inserting a single row into a table:

Syntax:

```
insert into <table name> values (<expression1>,<expression2>)
```

Example:

SQL>INSERT INTO EMPLOYEE VALUES(101,'MANU','LECTURER',15000);

(b) Inserting more than one record using a single insert commands:

Syntax:

insert into <table name> values (&col1, &col2, ….)

Example:

SQL> INSERT INTO EMPLOYEE
VALUES(&EMPNO,'&ENAME','&DESIGNATIN','&SALARY');

(c) **Skipping the fields while inserting:**

Insert into <tablename>(<column name1>,<column name3>)>values
(<expression1>,<expression3>);
Other way is to give null while passing the values.

SELECT COMMAND

(a) **view all rows and all columns**

Syntax:

Select * from tablename;

Example:

Select * from Employee;

(b)**Selected Columns And All Rows**

Syntax:

Select <column1>,<column2> from tablename;

Example:

Select empno, empname from Employee;

(c)**Selected Columns And selected Rows**

Syntax:

SELECt <column1>, <column2> FROM <tablename> WHERE <condition> ;

Example:

Select empno, empname from Employee where designation='lecturer';

(c)**Eliminating duplicate rows**

Syntax:

SELECT DISTINCT <column1>, <column2> FROM <tablename>

Example:

Select distinct empname from Employee;

UPDATE COMMAND

(b)updating all rows

Syntax:

update tablename set columnname1>=<exprssion1>,<columnname2>=<exprssion2>;

Example:

Update Employee set Designation = 'lecturer';

(b)updating records conditionally

Syntax:

update tablename set field=values where condition;

Example:

Update Employeeemp set sal = 10000 where empno=135;

DELETE COMMAND

(b)Removal of all rows

Syntax:

Delete from <table name> ;

Example:

Delete from emp;

(b)removal of specific rows

Syntax:

Delete from <table name> where <condition>;

Example:

delete from emp where empno=135;

RESULT

The DML commands are executed successfully.

Problems

1. Insert the following data into their respective tables.

   Data for CLIENT_MASTER table

ClientNo	Name	City	Pincode	State	BalDue
C00001	Ivan	Mumbai	400054	Maharashtra	15000
C00002	Ashwini	Chennai	780001	TamilNadu	0
C00003	Joshi	Mangalore	560001	Karnataka	5000
C00004	Deepak	Chennai	780001	TamilNadu	0
C00005	Sharma	Mumbai	400054	Maharashtra	2000

Product No	Description	Profitpercent	unitmeasure	qtyonhand	sellprice	Cost price
P00001	Tshirt	5	piece	200	350	250
P00065	Shirt	6	piece	150	500	350
P00032	Jeans	5	piece	100	600	450
P00324	Skirts	4	piece	120	750	500
P02345	CottonJeans	3	piece	80	850	550

2. Data for PRODUCT_MASTER table

SalesmanNo	Name	Address1	Address2	city	Pincode	State
S00001	Aman	A/4	Worli	Mumbai	400002	Maharashtra
S00002	Omkar	65	Nariman	Mumbai	400001	Maharashtra
S00003	Raj	P-7	Bandra	Mumbai	400032	Maharashtra
S00004	Ashish	A/5	Juhu	Mumbai	400044	Maharashtra

3. Data for SALESMAN_MASTER table

4. **Exercise on retrieving records from a table**

a. Find out the names of all clients

b. Retrieve the entire contents of the Client _master table

c. Retrieve the list of names,city and the state of all the clients

d. List the various products available from the Product _Master table

e. List all the clients who are located in Mumbai

f. Find the names of salesmen who have a salary equal to Rs.3000

5**. Exercise on updating the records on a table**

a. Change the city of ClientNo'C00005' to 'Bangaluru'.

b. Change the cBalDue of ClientNo'C00001' to Rs.1000.

c. Change the costprice of 'Shirt ' to Rs.450.

d. Change the city of salesman to Pune.

6. **Exercise on deleting the records in a table**

a. Delete all salesman from the Salesman_master whose salaries are equal to Rs.3500.

b. Delete all sproducts from the Product_master where quantity on hand is equal to 100

c. Delete from the Client_master where the column state holds the value 'Tamilnadu'.

EXPERIMENT NO:3
DATA CONTROL LANGUAGE

AIM:

To implement DCL statements.

OBJECTIVES

To understand DCL commands

THEORY:

Data Control Language (DCL) consists of various commands which are related to data sharing and security of data in database.

They are

GRANT

REVOKE

Granting Privileges:

Objects that are created by a user are owned and controlled by that user. If user wishes to access any of the objects belonging to another user, the owner of the object will have to give permissions for such access. This is called Granting of Privileges.

Granting privileges using the GRANT statements:

The GRANT statements provide various types of access to database objects such as tables, views.

Syntax:

```
GRANT {object privileges}
ON object name
TO username;
```

Object Privileges:

each object privilege that is granted authorizes the grantee to perform some operation on the object. The user can grant all the privileges or grant only specific object privileges.

The list of object privileges is as follows:

• **ALTER:** allows the grantee to change the table definitions with the ALTER table command.

• **DELETE:** allows the grantee to remove the records from the table with the DELETE command.

• **INDEX:** allows the grantee to create an index on the table with the CREATE INDEX command.

• **INSERT:** allows the grantee to add records to the table with the INSERT command.

• **SELECT:** allows the grantee to query the table with SELECT command.

• **UPDATE:** allows the grantee to modify the records in the table with the UPDATE command.

Revoking privileges given:

Privileges once given can be denied to a user using the REVOKE command. The object owner can revoke privileges granted to another user. A user of an object who is not owner, but has been granted the GRANT privilege, has the power to REVOKE the privileges from the grantee.

Revoking permission using the REVOKE statement:

The REVOKE statement is used to deny the grant given on an object.

Syntax:

```
REVOKE {object privileges}
ON object name
FROM username;
```

The REVOKE command is used to revoke object privileges that the user previously granted to the Revoke.

The REVOKE command cannot be used to revoke the privileges granted through operating system.

<u>RESULT</u>:

Familiarised   DCL statements.

EXPERIMENT NO:5
COMPUTATIONS ON TABLE DATA WITH BUILT IN FUCTIONS

AIM:

To implement computations done on data of the given table

OBJECTIVES

To understand computations done on data of the given table with built in functions

THEORY

Group Functions/Aggregate functions

A group function returns a result based on group of rows.

1. avg

Example: select avg (total) from student;

2.max

Example: select max (percentagel) from student;

2.min

Example: select min (marksl) from student;

4. sum

Example: select sum(price) from product

Count Function

In order to count the number of rows, count function is used.

1. count(*) – It counts all, inclusive of duplicates and nulls.

Example: select count(*) from student;

2. count(col_name)– It avoids null value.

Example: select count(total) from order;

2. count(distinct col_name) – It avoids the repeated and null values.

Example: select count(distinct ordid) from order;

Special Clauses:

Group by clause

This allows us to use simultaneous column name and group functions.

Example: Select max(percentage), deptname from student group by deptname;

Having clause
This is used to specify conditions on rows retrieved by using group by clause.
Example: Select max(percentage), deptname from student group by deptname having count(*)>=50;
In / not in – used to select a equi from a specific set of values
Any - used to compare with a specific set of values
Between / not between – used to find between the ranges
Like / not like – used to do the pattern matching
<u>PROCEDURE</u>

<u>OUTPUT</u>

<u>RESULT</u>

<u>PROGRAMS</u>

1.generate SQL statements to perform the following computations on table data.

a.list the names of all clients having ‘a’ as the second letter in their names.

b.listing of clients who stay in a city whose first letter is ‘M’

c.list all clients who stay in ‘Bangaluru’ or ‘Mangalore’

d.list all clients whose BalDue is greater than 10000

e.display the order information of clientno ‘C00001’ and’C00002’ f.list products whose selling price is greater than 500 and less than or equal to 750

g.listing of names,city and state of clients who are not in the state of ‘maharashtra’.

h.count the total number of orders

i.calculating the average price of all products.

j.determining the maximum and minimum price for the product prices.

k.count the number of products having the price greater than or equal to 500

2.SQL statements for using having and group by clauses.

a. printing the description and total quantity sold for each product.

b. Finding the value of each product sold

c. find out the total of all the billed orders for the month of june.

EXPERIMENT NO:6
NESTED QUERIES/SUB QUERIES AND JOINS

AIM:

To implement e nested queries and joins on the given table

OBJECTIVES

To understand nested queries and joins.

THEORY

a)NESTED QUERIES:

A sub query is a query within a query. In Oracle, we can create sub queries within your SQL statements. These sub queries can reside in the WHERE clause, the FROM clause, or the SELECT clause.

b)JOINS:

Join is a query in which data is returned from two or more tables.

Natural join:
It returns the matching rows from the table that are being joined
.
Syntax:
>select <attribute> from TN where TN1.attribute=TN2.attribute.
Inner join:
It returns the matching rows from the table that are being joined.

Syntax:
>select <attribute> from TN1 innerjoin TN2 on TN1.attribute=TN2.attribute.
Left outer join:
It returns all the rows from the table1 even when they are unmatched.

Syntax:
5. select <attribute> from TN1 left outer join TN2 on TN1.attribute=TN2.attribute.
2. select <attribute> from TN where TN1.attribute(+)=TN2.attribute.

Right outer join:
It returns all the rows from the table2 even when they are unmatched.

Syntax:
4. select <attribute> from TN1 right outer join TN2 on TN1.attribute=TN2.attribute.
2. select <attribute> from TN where TN1.attribute=(+)TN2.attribute.

Full join:

It is the combination of both left outer and right outer join.
Syntax:
>select <attribute> from TN1 full join TN2 on TN1.attribute=TN2.attribute.

PROCEDURE

NESTED QUERIES -

```
SQL> desc  emp_det;
Name          Null?                Type
--------------------------------------- -------- ---------------------------
ENO            NOT NULL         NUMBER(3)
ENAME                          VARCHAR2(25)
ADDRESS                         VARCHAR2(30)
BASIC_SAL                      NUMBER(12,2)
JOB_STATUS                      VARCHAR2(15)
DNO                            NUMBER(3)
```

```
SQL> desc pro_det;
Name                Null?                Type

--------------------------------------- -------- ---------------------------
PNO                  NOT NULL          NUMBER(3)
PNAME                                   VARCHAR2(30)
NO_OF_STAFF                            NUMBER(3)
```

```
SQL> desc work_in;

Name         Null?          Type
--------------------------------------- -------- ---------------------------
PNO                          NUMBER(3)
ENO                         NUMBER(3)
```

PJOB CHAR(12)

SQL> select * from emp_det;

ENO	ENAME	ADDRESS	BASIC_SAL	JOB_STATUS	DNO
1	SaravanaKumar	GandhiNagar	8000	Manager	10
2	Mahendran	RainbowColony	5000	Supervisor	10
3	RajKumar	EastCoastRoad	10000	Professor	2
4	Shirley	KKnagar	8000	AsstManager	3

SQL> select * from Pro_det;

PNO	PNAME	NO_OF_STAFF
1	DBMS	2
2	COMPILER	3
3	C1	1

SQL> select * from work_in;

PNO	ENO	PJOB
1	1	Programmer
2	1	Analyst
1	2	Analyst
2	2	Programmer

NESTED QUERIES

(i) SQL> select ename from emp_det where dno not in(select dno from emp_det where ename ='SaravanaKumar');

ENAME

```
---------------
RajKumar
Shirley
```

(ii)SQL> select ename, dno from emp_det where dno = (select dno from emp_det where ename ='RajKumar');

```
ENAME DNO
--------------- ----------
RajKumar 2
```

(iii)SQL> select ename from emp_det where eno in(select eno from work_in where pno = (select pno from pro_det where pname = 'DBMS')) order by ename;

```
ENAME
---------------
Mahendran
SaravanaKumar
```

(iv)SQL> select ename, basic_sal from emp_det where dno = 2 and basic_sal>(select max(basic_sal) from emp_det where dno = 10) order by ename;

```
ENAME BASIC_SAL
--------------- ----------
RajKumar 10000
```

(v)SQL> select pno,pname from pro_det where exists(select pno from work_in where work_in.pno =pro_det.pno);

```
PNO PNAME
------ ------------------------------
1 DBMS
2 COMPILER
```

(vi)SQL>select ename, job_status,basic_sal from emp_det where (dno,basic_sal) in (select dno,basic_sal from emp_det where ename ='RajKumar');

```
ENAME JOB_STATUS BASIC_SAL
--------------- --------------- ----------
RajKumar Professor 10000
```

```
(vii)SQL>select * from emp_det where basic_sal=(select max(basic_sal) from
emp_det);
ENO ENAME ADDRESS BASIC_SAL JOB_STATUS DNO
------ -------------- -------------- ---------- -------------- ----------
3 RajKumar EastCoastRoad 10000 Professor 2

(viii)SQL>select max(basic_sal) from emp_det where basic_sal< (select
max(basic_sal) from emp_det);

MAX(BASIC_SAL)
--------------
8000

(ix)SQL> select * from emp_det where basic_sal < (select avg(basic_sal) from
emp_det);

ENO ENAME ADDRESS BASIC_SAL JOB_STATUS DNO
---- -------------- -------------- ---------- -------------- ----------
2 Mahendran RainbowColony 5000 Supervisor 10
```

JOINS

```
SQL> create table emp(name varchar2(20),salary number(10));
Table created.
SQL> select * from emp;
NAME SALARY
-------------------- ----------
ashu 10000
asma 1200
asif 2000
arif 1000
niyas 3000
SQL> create table emp1(name varchar2(20),empid number(10));
Table created.
.
SQL> select * from emp1;
NAME EMPID
-------------------- ----------
fathi 12
sumi 32
priya 11
wahab 10
sweety 9
asma 1200
```

```
6 rows selected.
```

NATURAL JOIN
****

```
SQL>select emp.name,salary from emp,emp1 where emp.name=emp1.name
NAME SALARY
------------------- ----------
asma 1200
```

LEFT OUTER JOIN

```
SQL>select emp.name,salary from emp left outer join emp1 on
emp.name=emp1.name
NAME SALARY
------------------- ----------
asma 1200
asif 2000
arif 1000
niyas 3000
ashu 10000
```

RIGHT OUTER JOIN

```
SQL>select emp1.name,empid from emp right outer join emp1 on
emp.name=emp1.name
NAME EMPID
------------------- ----------
asma 1200
sweety 9
sumi 32
wahab 10
fathi 12
priya 11
6 rows selected.
```

FULL JOIN

```
SQL>select emp1.name,emp.name,emp1.empid,salary from emp full join emp1
on
emp.name=emp1.name
NAME NAME EMPID SALARY
------------------- ------------------- ---------- ----------
asma asma 1200 1200
asif 2000
arif 1000
niyas 3000
ashu 10000
sweety 9
sumi 32
```

```
wahab 10
fathi 12
priya 11
10 rows selected.
```

RESULT:
Thus the nested queries and join operations are executed and verifiedin DBMS.

Programs

1. Exercises on sub-queries

a) find the non moving products.ie products not being sold.

b) Find the name and complete address for the customer who has placed order number ‘o19001’

c) find the clients who have placed orders before the month of may ‘02

d) find the names of clients who have placed orders worth Rs.10000 or more.

EXPERIMENT NO:7
VIEWS

AIM:

To create and drop View on the given table.

OBJECTIVES

To implement views

THEORY

A view is the tailored presentation of data contained in one or more table and can also be said as restricted view to the data‟s in the tables. A view is a "virtual table" or a "stored query" which takes the output of a query and treats it as a table. The table upon which a view is created is called as base table . 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. The tables on which a view is based are called base tables. The view is stored as a SELECT statement in the data dictionary .

Advantages of a view:

a. Additional level of table security.
b. Hides data complexity.
c. Simplifies the usage by combining multiple tables into a single table

Creating and dropping view:

Syntax:

```
Create or replace view   view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition
```

```
Drop view <view name>;
```

Example

```
Create or replace view empview as select * from emp;
```

```
Drop view empview;
```

PROCEDURE
1)create a table aa

```
`SQL> create table aa(name varchar2(20),book number(10),edition
number(20),price number(20), ISBN number(20));
```

2)describe the table aa

```
SQL> select * from aa;
NAME         BOOK        EDITION    PRICE       ISBN
------------------- ---------- ---------- ---------- ----------
Bb           23           2001        12          23435
Cc            55          342         76          687478
dd           2            1233        123         53616578
ee           21           1111        111         12435798
```

3)create table qq

```
SQL> create table qq(name varchar2(20),book number(10),author
varchar(20),publisher varchar2(20),ISBN number(20));
```

4) describe table qq

```
SQL> select * from qq;
NAME        BOOK         AUTHOR   PUBLISHER       ISBN
------------------- ---------- ------------------- -----------------------------
bb           21          23          dfd          573568
cc            43          55          fg          65839
ee            44          21         dfd          1235798
oo           87          34           gfh         6358379
```

5)create a view on qq

```
SQL>create view ww as select book,name,publisher from qq where
ISBN=573568

View created.
```

6)display the view

```
SQL> select * from ww;
BOOK         NAME        PUBLISHER
- --------- ------------------- --------------------
```

```
21              bb              dfd
```

7)UPDATE VIEW STATEMENT

```
SQL> update ww set publisher='qwa'where book=21;
1 row updated.
SQL> select * from ww;
BOOK NAME PUBLISHER
---------- -------------------- --------------------
21 bb qwa

SQL> create view wq as select name,ISBN,publisher from qq where book>21
View created.
SQL> select * from wq;

NAME ISBN PUBLISHER
-------------------- ---------- --------------------
cc 65839 fg
ee 1235798 dfd
oo 6358379 gfh

SQL> create view ss as select name,book from aa union select name,book from
qq;
View created.
SQL> select * from ss;
NAME BOOK
-------------------- ----------
bb 21
bb 23
cc 43
cc 55
dd 2
ee 21
ee 44
oo 87
8 rows selected.
```

Result
Thus the view creation commands are executed successfully

Problems

1)Create the following table and insert rows

Table:Hosp_doc

Column name	Data type and size
Doc_code	Varchar2(4)
Doc_name	Varchar2(4)
Specialization	Varchar2(4)
Department	Varchar2(4)
Date_of_join	Date
Exper	Number(2)

1)create a view vw_doctor on Hosp_doc table

2)create another view that contains doctor codes and doctor names of 'orthology' department

3)delete the view vw_doctor

EXPERIMENT NO:8
FUNCTIONS AND PROCEDURE

AIM:
To find factorial of a number using function

OBJECTIVES

To write PL/SQL(Functions)and to understand stored procedures in SQL.

THEORY

FUNCTION:

A function is a subprogram that computes a value.

```
  syntax
Create or replace function<function_name>[argument]
Return datatype is
(local declaration)
begin
(executable statements)
[Exception]
(exception handlers)
End
```

PROCEDURE:

```
create [or replace]  procedure  procedurename
[parameter[in/out/in/in out] datatype
[:=/default expression]
[(parameter)]
is/as
declaration
begin
pl/sql codes
[exception]
end
```

PROGRAM

```
SQL> create or replace function fact(a number) return number as
 i number;
 f number;
 begin
 f:=1;
 i:=1;
 while (i<=a)
 loop
 f:=f*i;
 i:=i+1;
 end loop;
return f;
end fact; /Function created.
SQL>begin
2 dbms_output.put_line('the factorial='||fact(&a));
3* end;
```

OUTPUT

```
SQL> /
Enter value for a: 4
old 2: dbms_output.put_line('the factorial='||fact(&a))
new 2: dbms_output.put_line('the factorial='||fact(4));
the factorial=24
PL/SQL procedure successfully completed.
```

RESULT:
Thus the functions and stored procedures are executed in SQL.

Problems;

1) procedure to find whether a given number is odd or even
2) procedure to display 1-10 using while
3) Procedure to display some numbers lesser than given number

EXPERIMENT NO: 9
CURSOR

AIM

To retrieve all students who have registered for Diploma and store their details into another table called diploma (id,name) using cursors.

OBJECTIVES

To implement cursor

PROCEDURE

1) TABLE CREATION

```
SQL>create table student(id number,name varchar2(25),programme
varchar2(25));

SQL>create table diploma(id number,name varchar2(25));
SQL>insert into students values(1,’rohan’,’diploma’);
SQL>insert into students values(2,’anu’,’MA’);

SQL>insert into students values(3,’robert’,’diploma’);

SQL>insert into students values(4,’tom’,’btech’);

SQL>insert into students values(5,’sunny’,’diploma’);

SQL>select * from students;

Id      name  programme
1       rohan           diploma
2        anu            MA
3       robert          diploma
4       tom             btech
5       sunny           diploma

SQL>declare
2       cursor stud is select * from students where programme =”diploma’;
```

```
id  students.id%type;
name students.name%type
prog students.programme%type
begin
open stud;
loop
fetch stud into id,name,prog;
exit when stud%notfound;
insert into diploma values(id,name);
endloop;
end;
/
```

OUTPUT

RESULT

PROGRAMS

1.A HRD manager has decided to raise the salary of all employees in department number 20 by 0.05.Whenever such raise is given to the employees,the employee number ,the date when raise was given and raise amunt are maintained in the emp_raise table.Write a PL/SQL block using cursors to update the salary of each employee of dept no 20 and insert a record in the emp_raise table as well

EXPERIMENT NO: 10
TRIGGER

AIM

Create a Trigger for EMP table it will update another table SALARY while inserting values

OBJECTIVES

To develop and execute a Trigger for Before and After update/Delete/Insert operations on a table

THEORY.

PROCEDURE

step 1: start
step 2: initialize the trigger with specific table id.
step 3:specify the operations (update, delete, insert) for which the trigger has to be executed.
step 4: execute the trigger procedure for both before and after sequences
step 5: carryout the operation on the table to check for trigger execution.
step 6: stop

PROGRAM

```
sql> create table emp(iname varchar2(10),iid number(5),salary number(10));

table created.

sql> create table sal(iname varchar2(10),totalemp number(5),totalsal
number(10));

table created.

sql> create or replace trigger emptrigr  after insert on emp
for each row
declare
a varchar2(10);
begin
a:=:new.iname;
update sal set
totalsal=totalsal+:new.salary,totalemp=totalemp+1 where
iname=a;
```

```
end;
/
trigger created.

sql> insert into emp values('vec',100,1000);
1 row created.

sql> insert into sal values('vec',0,0);
1 row created.
sql> insert into  sal values('srm',0,0);
1 row created.

sql> select * from sal;
iname          totalemp       totalsal
---------- ---------- ----------
vec              1                  1000
srm               0                 0

sql> insert into emp  values('srm',200,3000);
1 row created.

sql> select * from sal;
iname          totalemp       totalsal
---------- ---------- ----------
Vec               1               1000
srm              1               3000
sql> insert into emp values('vec',100,5000);
1 row created.

sql> select * from sal;
iname  totalemp  totalsal
---------- ---------- ----------
vec              2       6000
srm               1       3000
sql> insert into emp values('vec',100,2000);
1 row created.

sql> select * from sal;
iname totalemp totalsal
---------- ---------- ----------
vec     3                 8000
srm     1                 3000
sql> insert into emp values('srm',200,8000);
```

```
1 row created.

sql> select * from sal;
iname  totalemp  totalsal
---------- ---------- ----------
Vec    3              8000
Srm    2              11000
```

RESULT:

The trigger procedure has been executed successfully for both before and after sequences.

Problems

1.Write a trigger that stores the details of students changing their program from CT to CHM.
2.Write an update trigger on CLIENT_MASTER table.The system should keep track of the records that are being updated.The old values of the updated record should be added in the AUDIT_TRAIL table

Column name	Data type	size
Client_no	Varchar2	6
name	Varchar2	20
Bal_due	Number	10,2
Operation	Varchar2	8
userid	Varchar2	20
olddate	date	

Experiment No:11

CONCEPTS OF NORMALIZATION

AIM: Checking Normalization of a database table (First Normal form)

Problem Statement:
An exercise to check whether the given database table is normalized or not. If yes find out the status of normalization and reasoning.
Objective:
To study the concept of various levels of normalization and understand how to convert into normalized forms.

Requirements: Mysql database software
Design/Theory
Create a database table in SQL with a few no of rows and columns. Analyze the table and determine to which normal form it belongs to according to the rules and regulations of each normal forms.
Procedure:
Consider a student table as given below

Social Security Number	FirstName	LastName	Major
123-45-6789	Jack	Jones	Library and Information Science
222-33-4444	Lynn	Lee	Library and Information Science
987-65-4321	Mary	Ruiz	Pre-Medicine
123-54-3210	Lynn	Smith	Pre-Law

We can easily verify that this table satisfies the definition of 1NF: viz., it has no duplicated rows; each cell is single-valued (i.e., there are no repeating groups or arrays); and all the entries in a given column are of the same kind. In this table we can see that the key, SSN, functionally determines the other attributes; i.e.,FirstName, LastName, and Major. .

Experiment No:12

Checking Normalization of a database table(Third normal form).

Problem Statement:

An excercise to check whether the given database table is normalized or not. If yes find out the status of normalization and reasoning.

Objective:

To study the concept of various levels of normalization and understand how to convert into normalized forms.

Requirements: Mysql database software

Design/Theory

Create a database table in SQL with a few no of rows and columns.Analyze the table and determine to which normal form it beongs to according to the rules and regulations of each normal forms.

Procedure:

Consider a book database table as given below.

Author Last Name	Author First Name	Book Title	Subject	Collection or Library	Building
Berdahl	Robert	The Politics of the Prussian Nobility	History	PCL General Stacks	Perry-Castañeda Library
Yudof	Mark	Child Abuse and Neglect	Legal Procedures	Law Library	Townes Hall
Harmon	Glynn	Human Memory and Knowledge	Cognitive Psychology	PCL General Stacks	Perry-Castañeda Library
Graves	Robert	The Golden Fleece	Greek Literature	Classics Library	Waggener Hall
Miksa	Francis	Charles Ammi Cutter	Library	Library and	Perry-

			Biography	Information Science Collection	Castañeda Library
Hunter	David	Music Publishing and Collecting	Music Literature	Fine Arts Library	Fine Arts Building
Graves	Robert	English and Scottish Ballads	Folksong	PCL General Stacks	Perry-Castañeda Library

By examining the table, we can infer that books dealing with history, cognitive psychology, and folksong are assigned to the PCL General Stacks collection; that books dealing with legal procedures are assigned to the Law Library; that books dealing with Greek literature are assigned to the Classics Library; that books dealing with library biography are assigned to the Library and Information Science Collection (LISC);and that books dealing with music literature are assigned to the Fine Arts Library.

Moreover, we can infer that the PCL General Stacks collection and the LISC are both housed in the Perry-Castañeda Library (PCL) building; that the Classics Library is housed in Waggener Hall; and that the Law Library and Fine Arts Library are housed, respectively, in Townes Hall and the Fine Arts Building.

Thus we can see that a transitive dependency exists in the above table : any book that deals with history, cognitive psychology, or library biography will be physically housed in the PCL building (unless it is temporarily checked out to a borrower); any book dealing with legal procedures will be housed in Townes Hall; and so on. In short, if we know what subject a book deals with, we also know not only what library or collection it will be assigned to but also what building it is physically housed in.

A problem with transitive dependency is that, there is duplicated information: from three different rows we can see that the PCL General Stacks are in the PCL building. For another thing, we have possible deletion anomalies: if the Yudof book were lost and its row removed from table, we would lose the information that books on legal procedures are assigned to the Law Library and also the information the Law Library is in Townes Hall. As a third problem, we have possible insertion anomalies: if we wanted to add a chemistry book to the table, we would find that the above table nowhere contains the fact that the

Chemistry Library is in Robert A.Welch Hall. As a fourth problem, we have the chance of making errors in updating: a careless data-entry clerk might add a book to the LISC but mistakenly enter Townes Hall in the building column.

To solve this problem decompose the above table into three different tables as follows

Table A

Author Last Name	Author First Name	Book Title
Berdahl	Robert	The Politics of the Prussian Nobility
Yudof	Mark	Child Abuse and Neglect
Harmon	Glynn	Human Memory and Knowledge
Graves	Robert	The Golden Fleece
Miksa	Francis	Charles Ammi Cutter
Hunter	David	Music Publishing and Collecting
Graves	Robert	English and Scottish Ballads

Table B

Book Title	Subject
The Politics of the Prussian Nobility	History

Child Abuse and Neglect	Legal Procedures
Human Memory and Knowledge	Cognitive Psychology
The Golden Fleece	Greek Literature
Charles Ammi Cutter	Library Biography
Music Publishing and Collecting	Music Literature
English and Scottish Ballads	Folksong

Table C

Subject	Collection or Library
History	PCL General Stacks
Legal Procedures	Law Library
Cognitive Psychology	PCL General Stacks
Greek Literature	Classics Library
Library Biography	Library and Information Science Collection
Music Literature	Fine Arts Library
Folksong	PCL General Stacks

Table D

Collection or Library	Building
PCL General Stacks	Perry-Castañeda Library

Law Library	Townes Hall
Classics Library	Waggener Hall
Library and Information Science Collection	Perry-Castañeda Library
Fine Arts Library	Fine Arts Building

Experiment No: 13

Database Connectivity

Java Database Connectivity

Aim: To understand the java database connevtivity

Problem Statement:

A program which connects to an online book database table and select the tuple and display it.

Objective:

To understand the connectivity to a database table using java. In java database is accessed through java database connectivity(JDBC).

Requirements: Mysql database software, Java software.

Design/Theory:

The four main components required for implementation of JDBC are application,driver manager, data source specific drivers and corresponding data sources. The application program establishes/terminates connection with data. Driver manager will load JDBC drivers and pass JDBC function calls from the application to the driver.The data source processes commands from the driver and return the results.

1. Drivers for the database are loaded by using Class.for Name.
2. The getConnection() function of DriverManager class of JDBC is used to create the connection object. The URL specifies the machine name(db.onlinebook.edu)

```
public static void Sample(String DB_id, String U-id, String Pword)
{
 String URL =”jdbc:oracle:oci8:@db.onlinebook.edu:100:onbook_db”;
Class.forName(“oracle.jdbc.driver.OracleDriver”);
Connection cn=DriverManager.getConnection(URL,U_id, Pword);
Statement st = Cn.createStatement();
try
{
 st.executeUpdate(“INSERT INTO AUTHOR
VALUES('A010','SMITH',JONES','TEXAS')”);
}
catch(SQLException se)
{
 System.out.println(“Tuple cannot be inserted.”+se);
}
ResultSet rs = st.execute Query(“SELECT Aname,State from AUTHOR
WHERE City = 'Seatle'”);
while(rs.next())
{
System.out.println(rs.getString(1) + “ “+rs.getString(2));
}
st.close();
Cn.close();
}
```

Experiment No:14

Database Connectivity

AIM:

To create a PHP-Mysql Program to enter data into Mysql

Problem Statement:

A php program that connects to a database table and insert values into the table.

Objective:

To understand the connectivity to a database table using php and how to insert new values into that table.

Requirements:

Mysql database software, LAMP Server and Html

Design/Theory:

In an existing or a newly created database an employee table is formed with attributes firstname, lastname and email. A php-html program is used to insert values into the table.

Php code:

```
<?php

$host="localhost"; // Host name
$username=""; // Mysql username
$password=""; // Mysql password
$db_name="test"; // Database name
$tbl_name="test_mysql"; // Table name

// Connect to server and select database.
mysql_connect("$host", "$username", "$password")or die("cannot connect");
mysql_select_db("$db_name")or die("cannot select DB");

// Get values from form
$name=$_POST['name'];
$lastname=$_POST['lastname'];
$email=$_POST['email'];

// Insert data into mysql
```

```
$sql="INSERT INTO $tbl_name(name, lastname, email)VALUES('$name',
'$lastname', '$email')";
$result=mysql_query($sql);

// if successfully insert data into database, displays message "Successful".
if($result){
echo "Successful";
echo "<BR>";
echo "<a href='insert.php'>Back to main page</a>";
}
else {
echo "ERROR";
}
?>
<?php
// close connection
mysql_close();
?>
<html>
<body>
<form action= "<?php $_php_self ?>" method = "GET">
        Empid: Empname: Empage: Dept: City
        <input type="text" name="Empid" />
        <input type="text" name="Empname" />
        <input type="text" name="Empage" />
        <input type="text" name="Dept" />
        <input type="text" name="City" />
        <input type="submit" value="Show Result" />
        </form>
</body></html>
```

EXPERIMENT NO: 15

DATABASE CONNECTIVITY

AIM

Create a php program that allows to enter the employee details into a database.

To create an application program that process a query which returns the grade result of a student after processing the marks table.

OBJECTIVES

To understand PHP mysql database connectivity.

To study the concept of connecting database using an application program (PHP) and process results after manipulating the database table

Requirements:

Mysql database software, LAMP Server and html.

PROCEDURE

Create a database in Mysql. Create a 'marks' table with fields regno, name, batch, mark1, mark2 and mark3. Write a function which finds the total marks and if the total marks is greater than 225 then the result is categorized as "Distinction", if the total marks is less than 225 , the result is "firstclass", it the total marks is less than 180, the result is "second class", if the total marks is less than 150 the result is "passed". If the marks in any one of the subject is less than 40 the result is "Failed".

1. create a form in PHP/JAVA

2. create a database in mysql

3. provide the database connectivity

4. retrieve the details of employee through forms

php code:

```
        <?php
    $user_name = "root";
    $password = "";
    $server = "localhost";
    $database = "mysql";
    $regno = $_POST['regno'];
  //$regno = 1;
    $db_handle = mysql_connect($server, $user_name, $password);
    if (!$db_handle)
    {
        die('Could not connect: ' . mysql_error());
    }
    //echo 'Connected successfully';
    $db_found = mysql_select_db($database);
    if ($db_found)
    {
        //echo "Database found";
        $SQL = "SELECT * FROM marks WHERE regno = '".$regno."'";
        $result_set = mysql_query($SQL);
        $record = mysql_fetch_array($result_set);
        echo "<BR>MARKLIST";
        echo "<BR>Reg No..:".$record['regno'];
        echo "<BR>Name....:".$record['name'];
        echo "<BR>Group...:".$record['batch'];
        echo "<BR>Mark1...:".$record['mark1'];
        echo "<BR>Mark2...:".$record['mark2'];
```

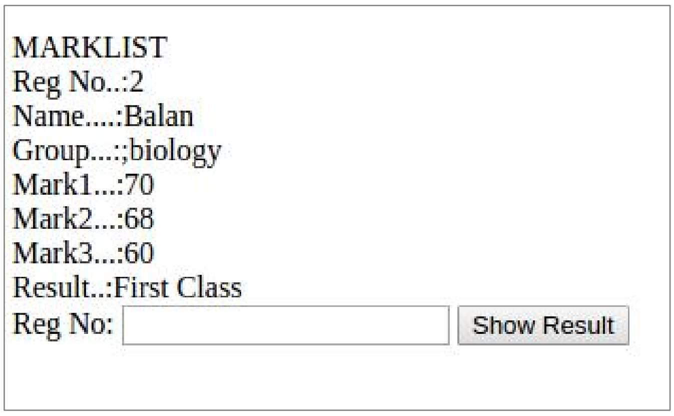
```
            echo "<BR>Mark3...:".$record['mark3'];
            echo                                                  "<BR>Result..:".
compute_result($record['mark1'],$record['mark2'],$record['mark3']);


        }
        else
        {
            echo "<BR>Database not Found";
        }
        mysql_close($db_handle);
        function compute_result($m1, $m2, $m3)
        {
            $tmarks = $m1 + $m2 + $m3;
            if (($m1<40) || ($m2<40) || ($m3<40))
                $result = "Failed";
            elseif ($tmarks < 150)
                $result = "Passed";
            elseif ($tmarks<180)
                $result = "Second Class";
            elseif ($tmarks<225)
                $result = "First Class";
            else
                $result = "Distinction";
        return $result;
        }
?>
```

Html code:

```
<html>
<body>
        <FORM Method = "post" Action="connect1.php">
        Reg No:
        <INPUT Type = "text" name = "regno"> <BR>
        <INPUT type = "submit" value = "Show Result">
        </FORM>
</body>
</html>
```

<u>OUTPUT</u>

<u>RESULT</u>



```
MARKLIST
Reg No..:2
Name....:Balan
Group...:;biology
Mark1...:70
Mark2...:68
Mark3...:60
Result..:First Class
Reg No:
```

VIVA VOCE QUESTIONS (DBMS)

1. **What is a database?**
2. **What is DBMS?**
3. **Give an example for an RDBMS.**
4. **List the benefits of DBMS.**
5. **Disadvantage in File Processing System**
6. **What is a key? what are different keys in database?**
7. **What is a primary key?**
8. **What is a secondary key?**
9. **What is a candidate key?**
10. **What is an alternate key?**
11. **What is a super key?**
12. **What is a composite key?**
13. **What is a relation?**
14. **What is a table?**
15. **What is an attribute?**
16. **What is a domain?**
17. **What is a tuple?**
18. **What is a selection?**
19. **what is a join operation?**
20. **What are base operations in relational algebra?**
21. **What are different DBMS facilities? How many types of facilities are provided by a DBMS?**
22. **What is Data Definition Language?**
23. What is Data Dictionary?
24. **What is a DML?**
25. **What is a query?**
26. **What is a query language?**
27. **What are the advantages of DBMS?**
28. **What is a SQL?**
29. **What are the features of SQL?**
30. **How SQL organizes the data?**
31. **What is data definition?**
32. **What is data retrieval?**
33. **What is data sharing?**
34. **What is a view?**
35. **What is normalization?**
36. **What is a first normal form?**
37. **What is a second normal form?**
38. **What is a third normal form?**
39. **What is BCNF?**
40. **What is fifth normal form?**
41. **What is Functional Dependency?**
42. **What is Lossless join property?**
43. **What are the commands to delete, modify and insert a record in the table?**
44. **What is time stamping?**
45. **What is data base schema?**
46. **What is a self join?**

47. **What are the different aggregate functions in SQL?**
48. **What is data integrity?**
49. **What is data independence?**
50. **What is dead locking?**
51. **What is decryption?**
52. **What is a distributed database?**
53. **What is an entity?**
54. **What is a conceptual data model?**
55. **What is two phase locking?**
56. **What is projection?**
57. **What are the different phases of transaction?**
58. **What is Relational Algebra?**
59. **What is Relational Calculus?**