JAWAHARLAL COLLEGE OF ENGINEERING AND TECHNOLOGY

JAWAHAR GARDENS, LAKKIDI, MANGALAM, PALAKKAD DT.

V SEMESTER B.TECH.

CSL331 – DATABASE MANAGEMENT SYSTEMS LAB

INSTITUTE VISION AND MISSION

Vision

Emerge as a center of excellence for professional education to produce high quality engineers and entrepreneurs for the development of the region and the Nation.

Mission

MI1: To become an ultimate destination for acquiring latest and advanced knowledge in the multidisciplinary domains.

MI2: To provide high quality education in engineering and technology through innovative teaching-learning practices, research and consultancy, embedded with professional ethics.

MI3: To promote intellectual curiosity and thirst for acquiring knowledge through outcome-based education.

MI4: To have partnership with industry and reputed institutions to enhance the employability skills of the students and pedagogical pursuits.

MI5: To leverage technologies to solve the real-life societal problems through community services.

.

VISION OF THE DEPARTMENT

To produce competent professionals with research and innovative skills, by providing them with the most conducive environment for quality academic and research oriented undergraduate education along with moral values committed to building a vibrant nation

MISSION OF THE DEPARTMENT

M1: Provide a learning environment to develop creativity and problem-solving skills in a professional manner.

M2: Expose to the latest technologies and tools used in the field of computer science.

M3: Provide a platform to explore the industries to understand the work culture and expectations of an organization.

M4: Enhance Industry Institute Interaction program to develop entrepreneurship skills.

M5: Develop research interest among students which will impart a better life for the society and the nation.

PROGRAMME OUTCOMES

PO1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	**Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	**The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	**Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12	**Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

PSO1: Use fundamental knowledge of mathematics to solve problems using suitable analysis methods, data structure and algorithms.
PSO2: Interpret the basic concepts and methods of computer systems and technical specifications to provide accurate solutions.
PSO3: Apply theoretical and practical proficiency with a wide area of programming knowledge and design new ideas and innovations towards research.

Programme Educational Objectives (PEOs)

Graduates of Computer Science Engineering Program shall

Graduates of Computer Science Engineering will:

PEO1: Provide a high-quality knowledge in Computer Science and Engineering required for a computer professional to identify and solve problems in various application domains.

PEO2: Persist with the ability in innovative ideas in computer support systems and transmit the knowledge and skills for research and advanced learning.

PEO3: Manifest the motivational capabilities, and turn on a social and economic commitment to community services.

COURSE OBJECTIVE

To give hands-on experience for Learners on creating databases by using Structured Query Language.

COURSE OUTCOME(COs)

After the completion of the course the student will be able to

CO	DESCRIPTION
CO1	Design database schema for a given real world problem-domain using standard design and modeling approaches. (Cognitive Knowledge Level: Apply)
CO2	Construct queries using SQL for database creation, interaction, modification, and updation. (Cognitive Knowledge Level: Apply)
CO3	Design and implement triggers and cursors. (Cognitive Knowledge Level: Apply)
CO4	Implement procedures, functions, and control structures using PL/SQL. (Cognitive Knowledge Level: Apply)
CO5	Perform CRUD operations in NoSQL Databases. (Cognitive Knowledge Level: Apply)
CO6	Develop database applications using front-end tools and back-end DBMS. (Cognitive Knowledge Level: Create)

GENERAL INSTRUCTIONS

Do's

1. Come with completed observation and record
2. Wear apron and ID card before entering into the lab.
3. Know the location of the fire extinguisher and the first aid box and how to use them in case of an emergency.
4. Read and understand how to carry out an activity thoroughly before coming to the laboratory.
5. Report any broken plugs or exposed electrical wires to your lecturer/laboratory technician immediately.
6. Write in time, out time and system details in the login register.

Don'ts

1. Do not eat or drink in the laboratory.
2. Do not operate mobile phones in the lab. Keep mobile phones either in silent or switched off mode.
3. Do not change system settings.
4. Do not disturb your neighbouring students. They may be busy in completing tasks.
5. Do not remove anything from the computer laboratory without permission.
6. Do not use pen drives.
7. Do not misbehave.

CSL333 - DATABASE MANAGEMENT SYSTEMS LAB

EXPERIMENT.NO 1

ER DIAGRAM

EXP:1a) UNIVERSITY MANAGEMENT SYSTEM ER DIAGRAM

Aim:A university registrar's office maintains data about the following entities: (a) courses, including number, title, credits, syllabus, and prerequisites; (b) course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom; (c) students, including student-id, name, and program; and (d) instructors, including identification number, name, department, and title. Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled. Construct an E-R diagram for the registrar's office. Document all assumptions that you make about the mapping constraints.

Output

The main entity sets are student, course, course-offering, and instructor. The entity set course-offering is a weak entity set dependent on course. The assumptions made are :

- A class meets only at one particular place and time. This E-R diagram cannot model a class meeting at different places at different times.
- There is no guarantee that the database does not have two classes meeting at the same place and time

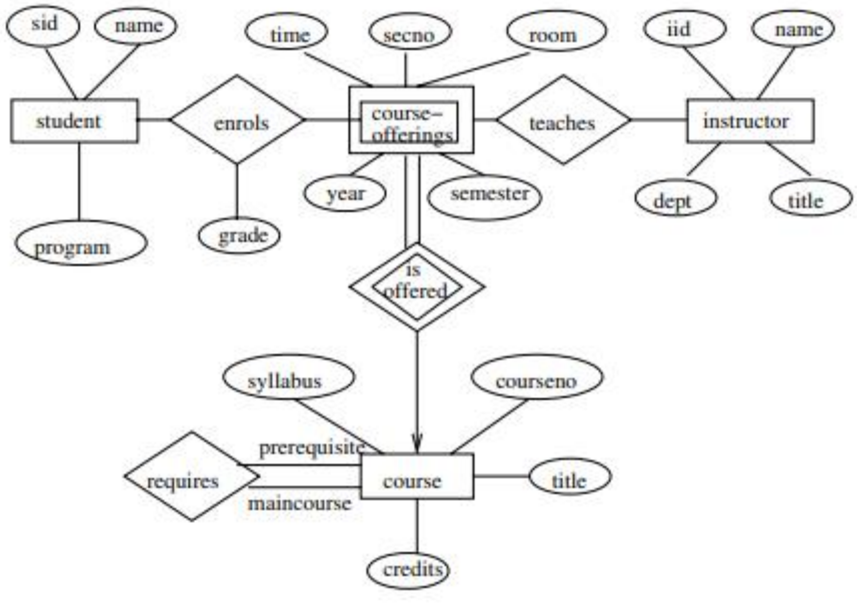


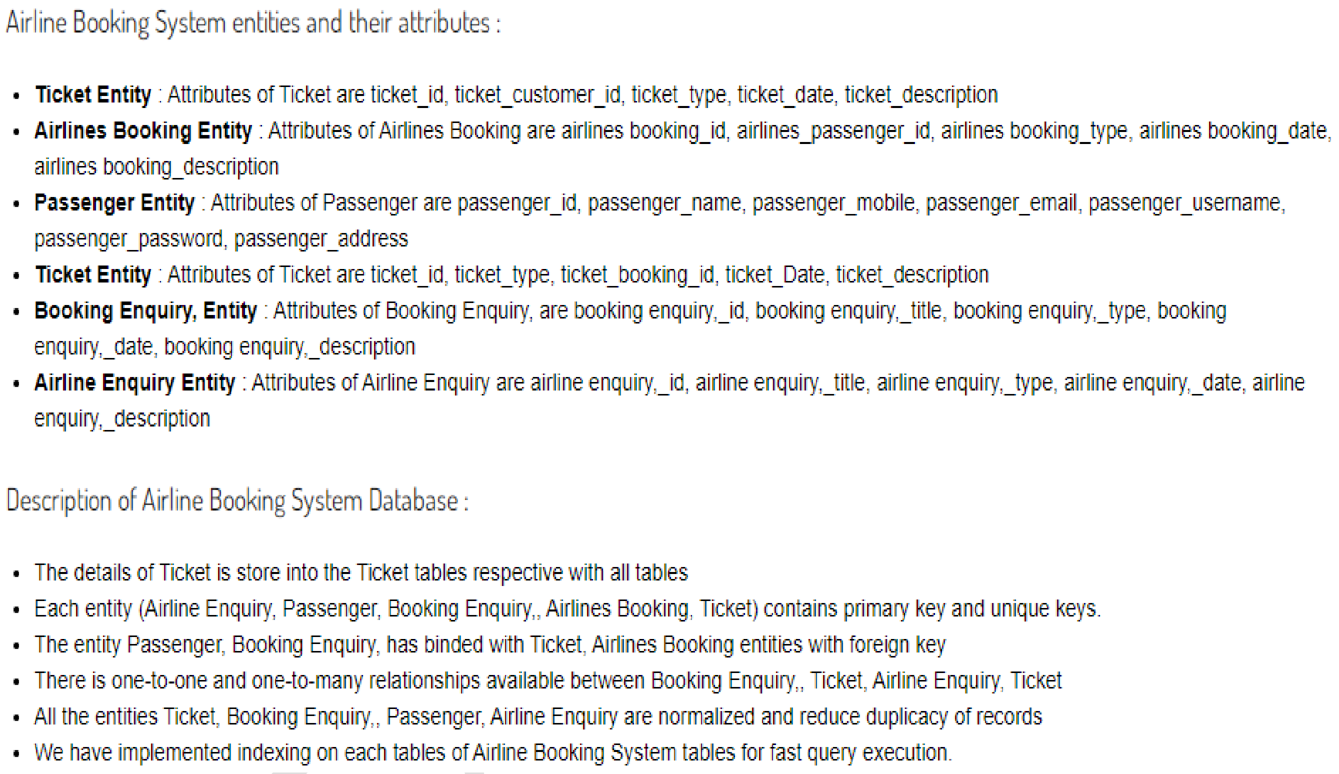
Figure 2.3 E-R diagram for a university.

EXP:1b) AIRLINE BOOKING SYSTEM ER DIAGRAM

Aim:

Construct an ER Diagram to represent a model an Airline Booking System. The ER diagram should show all relations between Airline booking, Ticket, Airline Enquiry. The main entities of this system are Ticket, Airline Booking, Passenger, Ticket, Booking Enquiry, and Airline

Enquiry.



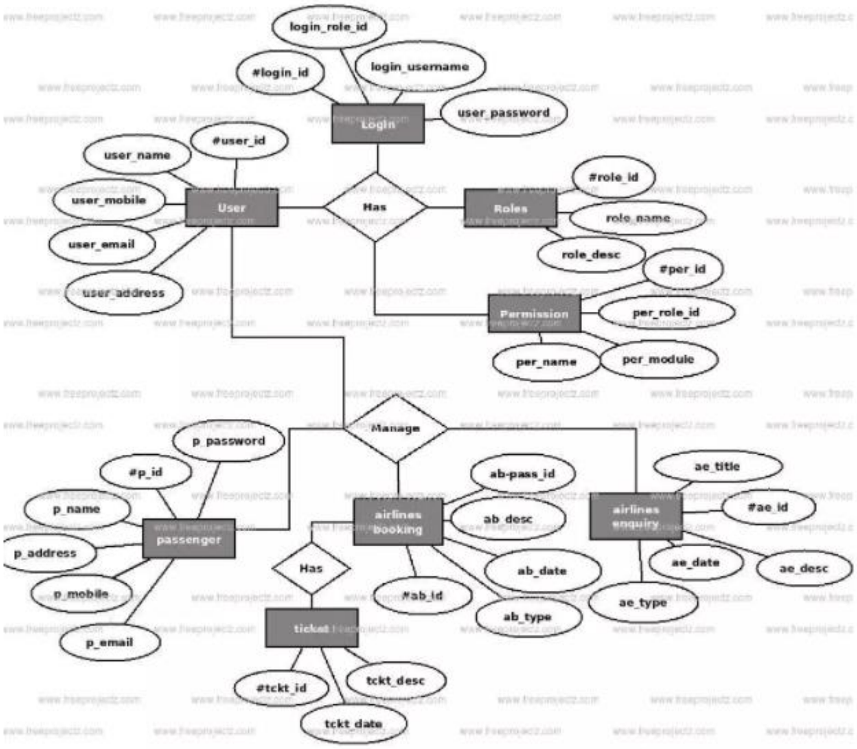
Airline Booking System entities and their attributes :

- **Ticket Entity** : Attributes of Ticket are ticket_id, ticket_customer_id, ticket_type, ticket_date, ticket_description
- **Airlines Booking Entity** : Attributes of Airlines Booking are airlines booking_id, airlines_passenger_id, airlines booking_type, airlines booking_date, airlines booking_description
- **Passenger Entity** : Attributes of Passenger are passenger_id, passenger_name, passenger_mobile, passenger_email, passenger_username, passenger_password, passenger_address
- **Ticket Entity** : Attributes of Ticket are ticket_id, ticket_type, ticket_booking_id, ticket_Date, ticket_description
- **Booking Enquiry, Entity** : Attributes of Booking Enquiry, are booking enquiry,_id, booking enquiry,_title, booking enquiry,_type, booking enquiry,_date, booking enquiry,_description
- **Airline Enquiry Entity** : Attributes of Airline Enquiry are airline enquiry,_id, airline enquiry,_title, airline enquiry,_type, airline enquiry,_date, airline enquiry,_description

Description of Airline Booking System Database :

- The details of Ticket is store into the Ticket tables respective with all tables
- Each entity (Airline Enquiry, Passenger, Booking Enquiry,, Airlines Booking, Ticket) contains primary key and unique keys.
- The entity Passenger, Booking Enquiry, has binded with Ticket, Airlines Booking entities with foreign key
- There is one-to-one and one-to-many relationships available between Booking Enquiry,, Ticket, Airline Enquiry, Ticket
- All the entities Ticket, Booking Enquiry,, Passenger, Airline Enquiry are normalized and reduce duplicacy of records
- We have implemented indexing on each tables of Airline Booking System tables for fast query execution.

Output


login_role_id
#login_id
login_username
user_password
Login
#user_id
user_name
user_mobile
user_email
user_address
User
Has
Roles
#role_id
role_name
role_desc
Permission
#per_id
per_role_id
per_name
per_module
Manage
p_password
#p_id
p_name
p_address
p_mobile
p_email
passenger
airlines booking
ab-pass_id
ab_desc
ab_date
#ab_id
ab_type
airlines enquiry
ae_title
#ae_id
ae_date
ae_desc
ae_type
Has
ticket
#tckt_id
tckt_desc
tckt_date

EXPERIMENT.NO 2

Creation, Modification, Configuration, And Deletion Of Databases Using UI And SQL Commands

STUDENT DATABASE

Create a student table with the following fields Name, Roll no, Age, Branch and insert the following data into the table.

Name	Roll no	Age	Branch
Anil Kumar	201	18	CS
Ramesh V.	202	19	ME
John Paul	103	17	EC
Reema Dev	111	16	CS
Sachin Gaur	301	18	ME

Create table called ‘distributor’ with columns Dcode, Codename, Amount, Limit and insert the following data.

Dcode	Codename	Amount	Limit
207	BlueStar Ltd	78,000	1,00,000
202	HCL Ltd	80,000	1,00,000
150	Microsystems	60,000	80,000
160	PHI Systems	-	90,000
203	Soft Agency	80,000	90,000

Create a table called ‘Agencies’ with columns Agcode, Agname, Amount, Aglimit and insert the following data into the table.

Agcode	Agname	Amount	Aglimit

401	Nath & Co	2500	10,000
40	Ram Sons	3600	10,000
403	Krishna Stores	4000	15,000
409	Kantt Mart	3279	10,000
407	Paico	-	10,000

1. Alter the structure of table student by adding a column called Totmarks and insert values into the added field
2. Update the ‘Agencies table, set amount field to 5000 corresponding to Agcode = 403.
3. Delete records from distributor table, whose amount is less than 70,000.
4. Display the Agcode and Agname from table Agencies in reverse order of their amount.
5. Display all Agency names for amount less than 4000 from table ‘Agencies’.
6. Display the student info from student table renaming the fields name as studentname, roll no as student_rollno.
7. Display the student info only in CS.
8. Update the limit field and amount field of distributor table to 1, 50,000 and 10,000 respectively whose limit is 1,00,000.
9. Display the average of total marks of the students

EXPERIMENT.NO. 3

Creation Of Database Schema - DDL

Aim:

To create the given tables database with the given attributes and to retrieve the necessary attribute values and to do table manipulations.

Algorithm:

Start

Create the tables using “Create table” command with the attributes of type character, varchar, number with precision or date wherever necessary.

Insert multiple data into the table using “insert into” command.

Display the employee details using the “Select …AS” command for all the various queries given below.

Use the necessary commands to do the table alterations and retrievals.

```
SQL> create table student(name varchar2(15),roll_no number(5),age number(3),branch varchar2(5));

Table created.

SQL> /

Enter value for name: Anil Kumar

Enter value for roll_no: 201

Enter value for age: 18

Enter value for branch: CS

new   1: insert into student values('Anil Kumar',201,18,'CS')

1 row created.

SQL> select * from student;

NAME              ROLL_NO        AGE BRANC

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

Anil Kumar             201         18 CS

Ramesh                 202         19 ME
```

```
John Paul           103        17 EC

Reema Dev             111        16 CS

Sachin Gaur          301        18 ME


SQL> create table distributor (dcode number(4), code_name varchar2(15), amount number(6),limit
number(6));

Table created.

SQL> insert into distributor  values (&dcode,'&code_name',&amount,&limit);

Enter value for dcode: 207

Enter value for code_name: BlueStar Ltd

Enter value for amount: 78000

Enter value for limit: 100000

old   1: insert into distributor values (&dcode,'&code_name',&amount,&limit)

new   1: insert into distributor values (207,'BlueStar Ltd',78000,100000)

1 row created.

SQL> select * from distributor ;

    DCODE CODE_NAME           AMOUNT      LIMIT

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

      207 BlueStar Ltd         78000     100000

      202 HCL Ltd              80000     100000

      150 Microsystems         60000      80000

      160 PHI Systems                     90000

      203 Soft Agency          80000      90000


SQL> Create table agencies (Agcode number(4), Agname varchar2(15), Amount number(5),Aglimit
number(6));

Table created.
```

```
SQL>  insert into agencie values(&Agcode,'&Agname',&Amount,&Aglimit);

Enter value for agcode: 401

Enter value for agname: Nath & Co

Enter value for amount: 2500

Enter value for aglimit: 10000

old   1:  insert into agencies values(&Agcode,'&Agname',&Amount,&Aglimit)

new   1:  insert into agencies values(401,'Nath & Co',2500,10000)

1 row created.

SQL> select * from agencies;

   AGCODE AGNAME            AMOUNT    AGLIMIT

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

     401 Nath & Co           2500      10000

     402 Ram Sons            3600      10000

     403 Krishna Stores       4000      15000

     409 Kantt Mart          3279      10000

     407 Paico                    10000
```

```
(4) SQL> alter table student add(totmarks number(4));

Table altered

(5)SQL> update agencies set amount=5000 where agcode=403;

(6)SQL> delete  from distributor where amount<70000;

    1 row deleted.

   SQL> select * from distributor;

    DCODE CODE_NAME          AMOUNT      LIMIT

     207 BlueStar Ltd        78000     150000
```

```
      202 HCL Ltd          80000     150000

      160 PHI Systems                90000

      203 Soft Agency        80000      90000
```

```
(7)SQL> select agname,agcode,amount from agencies order by amount desc;

AGNAME             AGCODE     AMOUNT

Paico                  407

Krishna Stores        403       4000

Ram Sons             402       3600

Kantt Mart           409       3279

Nath & Co            401       2500
```

```
(8)SQL> select agname from agencies where amount<4000;

AGNAME

Nath & Co

Ram Sons

Kantt Mart
```

```
 (9)SQL> select name studentname,roll_no student_rollno,age,branch,totmarks from student;


STUDENTNAME     STUDENT_ROLLNO        AGE BRANC   TOTMARKS

Anil Kumar              201         18 CS           46

Ramesh                 202        19 ME           89

John Paul              103        17 EC           65

Reema Dev               111         16 CS           56

Sachin Gaur             301         18 ME           65
```

```
(10)SQL> select  * from student where branch='CS';
```

```
NAME            ROLL_NO      AGE BRANC   TOTMARKS

Anil Kumar          201       18 CS          46

Reema Dev           111       16 CS          56
```

```
(11)SQL> update distributor set limit=150000,amount=10000 where limit=100000;

2 rows updated.

SQL> select * from distributor;

   DCODE CODE_NAME          AMOUNT      LIMIT

     207 BlueStar Ltd        10000     150000

     202 HCL Ltd             10000     150000

     160 PHI Systems          90000

     203 Soft Agency         80000      90000
```

```
(12)SQL> select avg(totmarks) from student;

AVG(TOTMARKS)

      64.2
```

EXPERIMENT NO 4

SQL Commands For DML

AIM:

To create a database and familiarize the DML commands.

Create the following table and insert the data and find the result for question given below using SQL.

EMPLOYEE

EID	LNAME	FNAME	MNAME	JOBID	DOJ	SALARY	DEPID
1000	John	Smith	b	111	12-jun-12	25000	1
1001	Larry	Becker	T	222	12-jun-12	30000	2
1002	Paul	Mathew	p	333	10-mar-12	20000	3
1003	George	Thomas	v	222	13-jun-11	35000	1
1004	Jacob	Mathew	v	111	10-jun-11	40000	2

1 List all the employees details where salary greater than 25000.

1. Alter the structure of the table by adding a field called commission and insert values into the added field.

2. Give 10% increase in the salary to the employee of department 1.

3. Delete the details of Employee working in department 3.

4. CREATE a view of Employee table with attributes EID,LNAME,FNAME,JOBID and DEPID

5. List out the employee id ,Lname, salary in descending order based on salary.

6. How many employees are working in each department in the organization?

EXPERIMENT NO 5

Creation Of Database Using Different Constraints

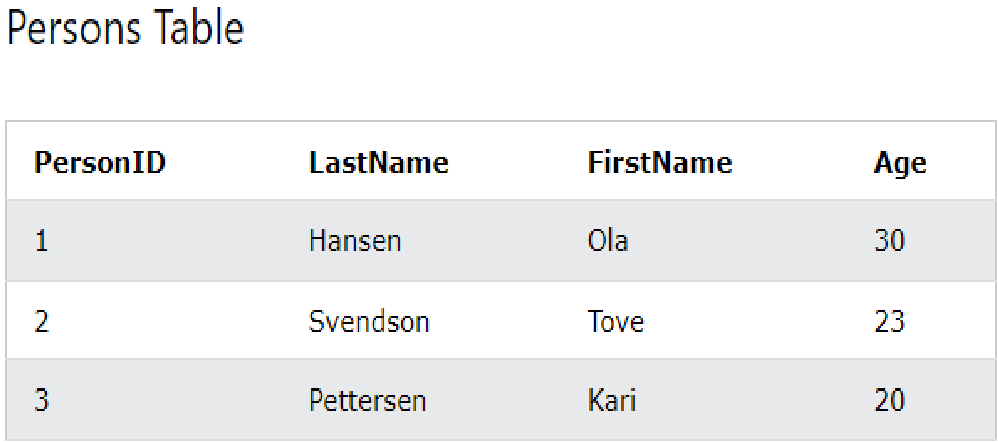
AIM

To create the given tables with different types of constraints in SQL and do the manipulations.

Create a table PERSON (personID,last_name,first_name,age city)

Constraints

- Set all the attributes as NOT NULL
- Set personID as UNIQUE constraint
- Set personID as PRIMARY KEY



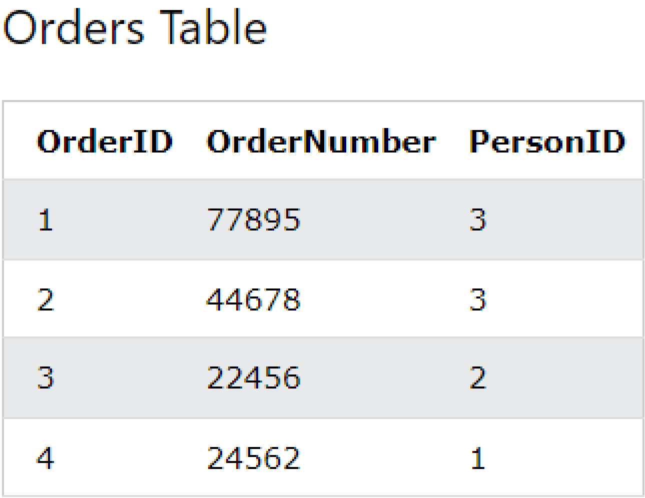
Persons Table

PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

Create a table ORDERS(orderID,order_number)

Constraints

- Set all the attributes as NOT NULL
- Set orderID as PRIMARY KEY
- Set personID as FOREIGN KEY



Orders Table

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

Notice that the "PersonID" column in the "Orders" table points to the "PersonID" column in the "Persons" table.

The "PersonID" column in the "Persons" table is the PRIMARY KEY in the "Persons" table.

The "PersonID" column in the "Orders" table is a FOREIGN KEY in the "Orders" table.

The FOREIGN KEY constraint prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the parent tabl

QUERY

```
CREATE TABLE Persons(
   personID                                                         int NOT NULL,
    LastName                                                varchar(255) NOT NULL,
                        FirstName                           varchar(255) NOT NULL,
    Age int NOT NULL,

   City                varchar(200)                  NOT                     NULL
);
```

```
CREATE TABLE Persons                                                            (
                               personID                                  int NOT NULL,
                           LastName                             varchar(255) NOT NULL,
                               FirstName                                 varchar(255),
    Age int NOT NULL,

   City                    varchar(200)                  NOT                    NULL
    UNIQUE (ID)
);



CREATE TABLE Persons                                                            (
                               personID                                  int NOT NULL,
                           LastName                             varchar(255) NOT NULL,
                               FirstName                                 varchar(255),
    Age int NOT NULL,

   City varchar(200) NOT NULL

    PRIMARY KEY (ID)
);
CREATE TABLE Orders                                                             (
                                OrderID                                  int NOT NULL,
                              OrderNumber                                int NOT NULL,
                                    PersonID                                      int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);



CREATE TABLE Persons                                                            (
                               personID                                  int NOT NULL,
                           LastName                             varchar(255) NOT NULL,
                               FirstName                                 varchar(255),
                        Age                      int                         NOT NULL,
                         City                         varchar(255) DEFAULT 'Sandnes'
);
```

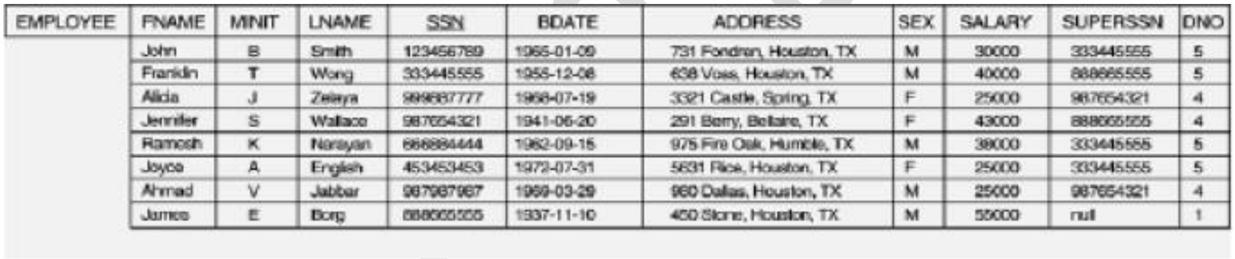
Experiment No: 6

To create views

AIM

To create views in database.

A database view is a virtual table or logical table which is defined as a SQL SELECT query with joins. Because a database view is similar to a database table, which consists of rows and columns, so you can query data against it. Most database management systems, including MySQL, allow you to update data in the underlying tables through the database view with some prerequisites.



EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

For a given EMPLOYEE tables

Perform the Following

1. Creating Views (With and Without Check Option),
2. Selecting from a View
3. Dropping Views

```
SQL> CREATE TABLE EMPLOYEE (

        SSN VARCHAR2 (20) PRIMARY KEY,
        FNAME VARCHAR2 (20),

        LNAME VARCHAR2 (20),

        ADDRESS VARCHAR2 (20),

        SEX CHAR (1),
        SALARY INTEGER,
        SUPERSSN REFERENCES EMPLOYEE (SSN),
        DNO REFERENCES DEPARTMENT (DNO));
```

```
SQL> DESC EMPLOYEE;
    Name                            Null?        Type
    ------------------------------- --------     --------------------
    SSN                             NOT NULL     VARCHAR2(20)
    FNAME                                        VARCHAR2(20)
    LNAME                                        VARCHAR2(20)
    ADDRESS                                      VARCHAR2(20)
    SEX                                          CHAR(1)
    SALARY                                       NUMBER(38)
    SUPERSSN                                     VARCHAR2(20)
    DNO                                          NUMBER(38)
```

```
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (‘RNSECE01‘,‘JOHN‘,‘SCOTT‘,‘BANGALORE‘,‘M‘, 450000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (‘RNSCSE01‘,‘JAMES‘,‘SMITH‘,‘BANGALORE‘,‘M‘, 500000);


INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (‘RNSCSE02‘,‘HEARN‘,‘BAKER‘,‘BANGALORE‘,‘M‘, 700000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (‘RNSCSE03‘,‘EDWARD‘,‘SCOTT‘,‘MYSORE‘,‘M‘, 500000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (‘RNSCSE04‘,‘PAVAN‘,‘HEGDE‘,‘MANGALORE‘,‘M‘, 650000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (‘RNSCSE05‘,‘GIRISH‘,‘MALYA‘,‘MYSORE‘,‘M‘, 450000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (‘RNSCSE06‘,‘NEHA‘,‘SN‘,‘BANGALORE‘,‘F‘, 800000);
```

```
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (‘RNSACC01‘,‘AHANA‘,‘K‘,‘MANGALORE‘,‘F‘, 350000);
```

```
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (‘RNSACC02‘,‘SANTHOSH‘,‘KUMAR‘,‘MANGALORE‘,‘M‘, 300000);
```

```
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (‘RNSISE01‘,‘VEENA‘,‘M‘,‘MYSORE‘,‘M‘, 600000);
```

```
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (‘RNSIT01‘,‘NAGESH‘,‘HR‘,‘BANGALORE‘,‘M‘, 500000);
```

1. Creating Views

```
SQL> CREATE VIEW sales_staff  AS

        2     SELECT fname, ssn, dno
        3     FROM employee
        4     WHERE dno =5
        5    WITH CHECK OPTION CONSTRAINT sales_staff_cnst;

View created.
```

2. Selecting from a View

```
SQL> select * from sales_staff;
```

3. Drop View

```
SQL>DROP VIEW sales_staff;
```

Experiment.No 7

Implementation of aggregate functions in SQL

Aim

To implement various aggregate functions in SQL

Consider Employee table

EMPNO	EMP_NAME	DEPT	SALARY	DOJ	BRANCH
E101	Amit	oduction	45000	12-Mar-00	Bangalore
E102	Amit	HR	70000	03-Jul-02	Bangalore
E103	sunita	anagemen	120000	11-Jan-01	mysore
E105	sunita	IT	67000	01-Aug-01	mysore
E106	mahesh	Civil	145000	20-Sep-03	Mumbai

Perform the following

1. Display all the fields of employee table
2. Retrieve employee number and their salary
3. Retrieve average salary of all employee
4. Retrieve number of employee
5. Retrieve distinct number of employee
6. Retrieve total salary of employee group by employee name and count similar names
7. Retrieve total salary of employee which is greater than >120000
8. Display name of employee in descending order
9. **Display details of employee whose name is AMIT and salary greater than 50000;**

1. Display all the fields of employee table

```
SQL> select * from employee;


EMPN  EMP_NAM   DEPT          SALARY     DOJ          BRANCH
O     E

E101   Amit     Production    45000      12-MAR-00    Bangalore

E102  Amit      HR            70000      03-JUL-02    Bangalore

E103  sunita     Management   120000      11-JAN-01   mysore

E105   sunita    IT           67000      01-AUG-01    mysore

E106  mahesh     Civil        145000     20-SEP-03    Mumbai
```

2. Retrieve employee number and their salary

```
SQL> select empno, salary from
employee;EMPNO   SALARY


------------------------
E101      45000

E102      70000

E103     120000

E105      67000

E106     145000
```

3. Retrieve average salary of all employee

```
SQL> select  avg(salary) from employee;

AVG(SALARY)

-------

     89400
```

4. Retrieve number of employee

```
SQL> select count(*) from

  employee;COUNT(*)
-------

        5
```

5. Retrieve distinct number of employee

```
SQL> select count(DISTINCT emp_name) from employee;
COUNT(DISTINCTEMP_NAME)

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

            3
```

6. Retrieve total salary of employee group by employee name and count similar names

```
SQL> SELECT EMP_NAME, SUM(SALARY),COUNT(*) FROM
  EMPLOYEE2  GROUP BY(EMP_NAME);
```

```
EMP_NAME                SUM(SALARY) COUNT(*)
-----------------------------------------------------------------------------------
mahesh                  145000          1

sunita                  187000          2

Amit                    115000          2
```

7. Retrieve total salary of employee which is greater than >120000

```
SQL> SELECT EMP_NAME, SUM(SALARY) FROM
  EMPLOYEE2  GROUP BY(EMP_NAME)

  3  HAVING SUM(SALARY)>120000;


EMP_NAME                SUM(SALARY)

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

mahesh                  145000

sunita                  187000
```

8. Display name of employee in descending order

```
SQL> select emp_name from employee
  2  order by emp_name desc;

EMP_NAME

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

sunita
sunita
mahesh
Amit
Amit
```

9. Display details of employee whose name is AMIT and salary greater than 50000;

```
SQL> select * from employee

  2  where emp_name='Amit' and salary>50000;
```

```
EMPNO       EMP_NAME             DEPT      SALARY     DOJ          BRANCH
E102         Amit                HR       70000      03-JUL-02     Bangalore
```

Experiment No. 8

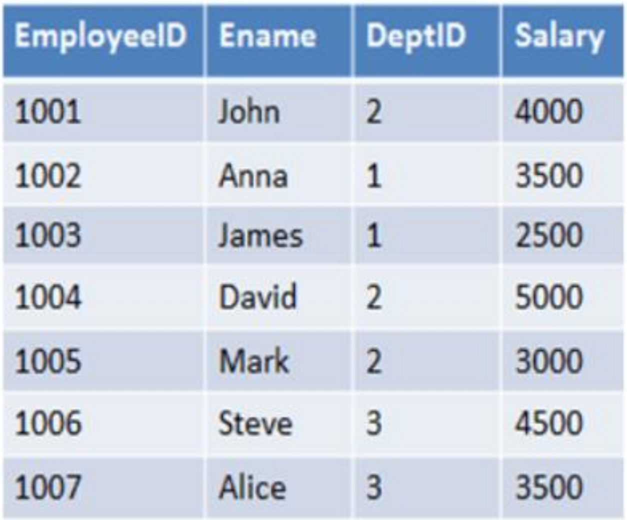
Implementation of ORDERBY,GROUPBY and HAVING clause

AIM

To implement ORDERBY,GROUPBY and HAVING clause in SQL

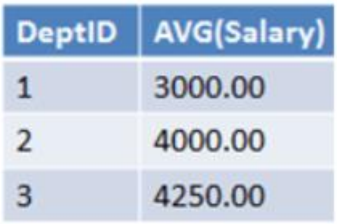
GROUPBY

Consider the EMPLOYEE table



EmployeeID	Ename	DeptID	Salary
1001	John	2	4000
1002	Anna	1	3500
1003	James	1	2500
1004	David	2	5000
1005	Mark	2	3000
1006	Steve	3	4500
1007	Alice	3	3500

```
SELECT DeptID,AVG(salary)  FROM employee GROUPBY DeptID;
```

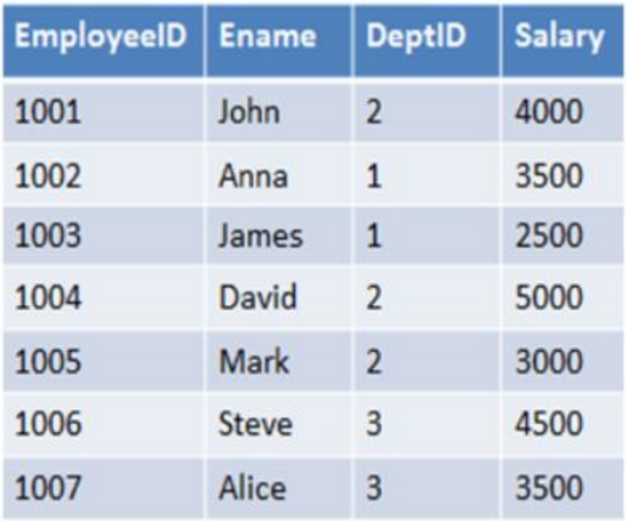


DeptID	AVG(Salary)
1	3000.00
2	4000.00
3	4250.00

Table is grouped based on the DeptID column and Salary is aggregated department-wise.

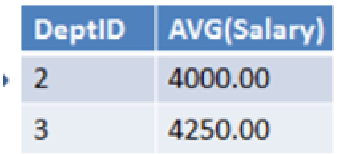
HAVING CLAUSE

Consider the EMPLOYEE table



EmployeeID	Ename	DeptID	Salary
1001	John	2	4000
1002	Anna	1	3500
1003	James	1	2500
1004	David	2	5000
1005	Mark	2	3000
1006	Steve	3	4500
1007	Alice	3	3500

SELECT DeptID,AVG(salary) FROM employee GROUPBY DeptID JHAVING AVG(salary)>3000;

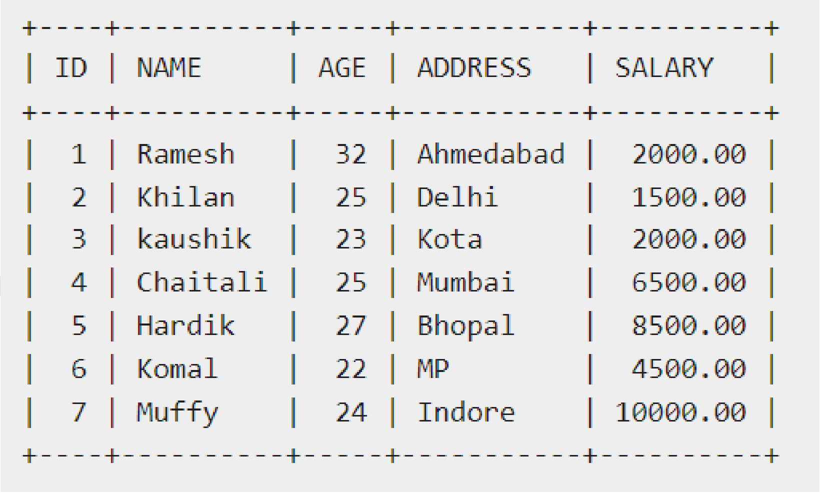


DeptID	AVG(Salary)
2	4000.00
3	4250.00

Table is grouped based on DeptID column and these grouped rows filtered using HAVING Clause with condition AVG(Salary) > 3000.

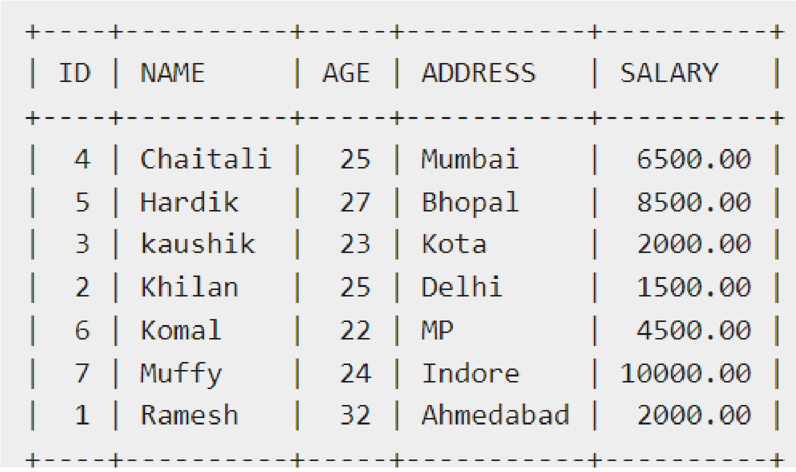
ORDER BY

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 |
+----+----------+-----+-----------+----------+
```

```
SQL> SELECT * FROM CUSTOMERS
   ORDER BY NAME, SALARY
```



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

Experiment . No : 9

Performing TCL commands COMMIT and ROLLBACK

AIM

To manipulate database using TCL commands.

These statements provide control over use of transactions: START TRANSACTION or BEGIN start a new transaction.

- COMMIT commits the current transaction, making its changes permanent.
- ROLLBACK rolls back the current transaction, canceling its changes.

SET autocommit disables or enables the default autocommit mode for the current session.

By default, MySQL runs with autocommit mode enabled. This means that as soon as you execute a statement that updates (modifies) a table, MySQL stores the update on disk to make it permanent. The change cannot be rolled back.

```
create database cserebels;

create table stud (id int primary key auto_increment,name char(50),dob date,age int);

start transaction;

insert into students (name,dob,age) values ('naveen',"1996-08-20",20);

insert into students (name,dob,age) values ('sathya',"1996-06-10",21);

SELECT * FROM students;

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

| id | name | dob | age|

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

| 1 | naveen | 1997-08-20 | 20 |

| 2 | sathya | 1996-09-10 | 21 |

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

ROLLBACK;

SELECT * FROM students;
```

```
+----+-------------+-------------------+------+

| id | name | dob | age|

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

| 1 | naveen | 1997-08-20 | 20 |

| 2 | sathya | 1996-09-10 | 21 |

+----+------------+--------------------+------+
```

```
SET autocommit=0 ;

savepoint s1;

update students SET dob="1997-09-10" where id=2;

SELECT * FROM students;

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

| id | name | dob | age|

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

| 1 | naveen | 1997-08-20 | 20 |

| 2 | sathya | 1997-09-10 | 21 |

+----+------------+--------------------+------+
```

```
ROLLBACK to s1;

SELECT * FROM students;

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

| id | name | dob | age|

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

| 1 | naveen | 1997-08-20 | 20 |

| 2 | sathya | 1996-09-10 | 21 |

+----+------------+--------------------+------+
update students SET dob="1997-09-10" and age=20 where id=2;

SELECT * FROM students;

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

| id | name | dob | age|

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

| 1 | naveen | 1997-08-20 | 20 |

| 2 | sathya | 1997-09-10 | 20 |

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

commit;

insert into students (name,dob,age) values ('kathir',"1995-06-15",22);

SELECT * FROM students;

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

| id | name | dob | age |

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

| 1 | naveen | 1997-08-20 | 20 |

| 2 | sathya | 1997-09-10 | 20 |

| 3 | kathir | 1995-06-15 | 22 |

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


ROLLBACK;

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

| id | name | dob | age|

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

| 1 | naveen | 1997-08-20 | 20 |

| 2 | sathya | 1997-09-10 | 20 |

+----+------------+--------------------+------+
```

Experiment . No : 10

Performing DCL commands GRANT and REVOKE

AIM

To manipulate database using DCL commands.

COMMANDS

Grant Privileges on Table

You can grant users various privileges to tables. These permissions can be any combination of SELECT, INSERT, UPDATE, DELETE, INDEX, CREATE, ALTER, DROP, GRANT OPTION or ALL.

Syntax

```
GRANT privileges ON object TO user;
```

Revoke Privileges on Table

Once you have granted privileges, you may need to revoke some or all of these privileges. To do this, you can run a revoke command. You can revoke any combination of SELECT, INSERT, UPDATE, DELETE, REFERENCES, ALTER, or ALL.

Syntax

```
REVOKE privileges ON object FROM user;
```

```
mysql -u root -p

create user 'UI@localhost' identified by 'jecc@1234';

create database class;

use class;

create table stud (id int primary key auto_increment,name char(20),age int);

mysql -u UI@localhost -p

use class;

create table students (id int primary key auto_increment,name char(50),dob date, age int);
```

```
CREATE command denied to user 'UI@localhost'@'localhost' for table 'students'
insert into stud (name,age) values ('naveen',20);


select * from stud;
+----+--------+------+
| id | name | age |
+----+--------+------+
| 1 | naveen | 20 |
+----+--------+------+
delete from stud;
DELETE command denied to user 'UI@localhost'@'localhost' for table 'stud'
mysql -u root -p
use class;
select * from stud;
+----+--------+------+
| id | name | age |
+----+--------+------+
| 1 | naveen | 20 |
+----+--------+------+


GRANT ALL ON class.* to 'UI@localhost' WITH GRANT option ;
create user 'GUI@localhost' identified by 'jecc@1234';
mysql -u UI@localhost -p
use class;
create table student (id int primary key auto_increment,name char(50),dob date, age int);
show tables;
+-----------------+
| Tables_in_class |
+-----------------+
| stud |
| student |
+-----------------+
insert into stud (name,age) values ('nas',20);
select * from stud;
+----+--------+------+
| id | name | age |
+----+--------+------+
| 1 | naveen | 20 |
| 2 | nas | 20 |
+----+--------+------+
delete from stud where id=2 ;
select * from stud;
+----+--------+------+
| id | name | age |
+----+--------+------+
| 1 | naveen | 20 |
+----+--------+------+

GRANT INSERT,SELECT ON class.* to 'GUI@localhost';
mysql -u GUI@localhost -p
use class;
create table student (id int primary key auto_increment,name char(50),dob date, age int);
CREATE command denied to user 'UI@localhost'@'localhost' for table 'students'
insert into stud (name,age) values ('nas',20);
```

```
select * from stud;

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

| id | name | age |

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

| 1 | naveen | 20 |

| 3 | nas | 20 |

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

mysql -u root -p

REVOKE ALL ON class.* from 'UI@localhost';

mysql -u UI@localhost -p

use class;

insert into stud (name,age) values ('naveen',20);

INSERT command denied to user 'UI@localhost'@'localhost' for table 'stud'

mysql -u GUI@localhost -p
```

Experiment No : 11

Performing SET operations

AIM

To query database using SET operations, nested and join queries `UNION` is used to combine the result from multiple `SELECT` statements into a single result set.

```
mysql> select * from students;
+-------+--------+------------+------+
| id | sname | phno | age |
+-------+--------+------------+------+
| 12345 | | 1234123412 | 19 |
| 12346 | arjith | 1234123413 | 17 |
| 12347 | arjith | 1234123417 | 13 |
| 12348 | Adam| 1234123418 | 13 |
| 12349 | NULL | 1234123410 | 13 |
+-------+--------+------------+------+
```

```
mysql> select * from sub;
+------+-------+-------+------+
| code | sname | id | dept |
+------+-------+-------+------+
| 123 | dda | 12347 | ME |
| 123 | ddc | 12346 | CS |
| 123 | Adam | 12348 | CS |
+------+-------+-------+------+
2 rows in set (0.00 sec)
```

```
mysql> select * from students UNION select * from sub;
+-------+--------+-----------+------+
| id | name | phno | age |
+-------+--------+-----------+------+
| 12345 | | 1234123412 | 19 |
| 12346 | arjith | 1234123413 | 17 |
| 12347 | arjith | 1234123417 | 13 |
| 12348 |Adam | 1234123418 | 13 |
| 12349 | NULL | 1234123410 | 13 |
| 123 | dda | 12347 | ME |
| 123 | ddc | 12346 | CS |
+-------+--------+-----------+------+
7 rows in set (0.00 sec)
mysql> select * from students UNION ALL select * from sub;
+-------+--------+-----------+------+
| id | name | phno | age |
+-------+--------+-----------+------+
| 12345 | | 1234123412 | 19 |
| 12346 | arjith | 1234123413 | 17 |
| 12347 | arjith | 1234123417 | 13 |
| 12348 |Adam | 1234123418 | 13 |
| 12349 | NULL | 1234123410 | 13 |
| 123 | dda | 12347 | ME |
| 123 | ddc | 12346 | CS |
| 123 | Adam | 12348 | CS |
+-------+--------+-----------+------+
8rows in set (0.00 sec)
```

SELECT * FROM Students **INTERSECT** SELECT * FROM Sub;

```
| 12348 | Adam |

1rows in set (0.00 sec)
```

SELECT * FROM Students **MINUS** SELECT * FROM Sub;

```
+-------+--------+------------+------+

| id | sname | phno | age |

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

| 12345 | | 1234123412 | 19 |

| 12346 | arjith | 1234123413 | 17 |

| 12347 | arjith | 1234123417 | 13 |

| 12349 | NULL | 1234123410 | 13 |

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

4rows in set (0.00 sec
```