Practical 1

Aim: Design any database with at least 3 entities and relationships between them. Draw suitable ER/EER diagram for the system.

```
CREATE TABLE Book (
    BookID INT PRIMARY KEY AUTO_INCREMENT,
    Title VARCHAR(255),
    Author VARCHAR(255),
    ISBN VARCHAR(20),
    Publisher VARCHAR(255),
    YearPublished INT
);

CREATE TABLE Member (
    MemberID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(100),
    Email VARCHAR(100),
    Phone VARCHAR(20),
    Address VARCHAR(255)
);

CREATE TABLE Loan (
    LoanID INT PRIMARY KEY AUTO_INCREMENT,
    BookID INT,
    MemberID INT,
    LoanDate DATE,
    ReturnDate DATE,
    DueDate DATE,
    FOREIGN KEY (BookID) REFERENCES Book(BookID),
    FOREIGN KEY (MemberID) REFERENCES Member(MemberID)
);
```

Practical 2

Aim: Design and implement a database using DDL statements

```
create database student_detail;
use student_detail;
create table stud_info (st_id int(3),stud_name varchar(20),stud_subject varchar(20));
desc stud_info;
alter table stud_info add Email_Id varchar(20);
desc stud_info;
insert into stud_info values(1,"Amaan","M3","as@gmail.com");
insert into stud_info values(2,"Mohit","CG","mh@gmail.com");
insert into stud_info values(3,"Raj","DBMS","rj@gmail.com");
insert into stud_info values(4,"Riya","M3","ri@gmail.com");
select* from stud_info;
truncate table stud_info;
desc stud_info;
drop table stud_info;
desc stud_info;
```

Practical 3

Aim: Create Student Table with primary key and foreign key constraints,

a. Alter table with add n modify b. Drop table .

```
CREATE TABLE Department (
   DeptID INT PRIMARY KEY,
   DeptName VARCHAR(100)
);

CREATE TABLE Student (
   StudentID INT PRIMARY KEY,
   Name VARCHAR(100),
   Age INT,
   DeptID INT,
   FOREIGN KEY (DeptID) REFERENCES Department(DeptID)
);

ALTER TABLE Student ADD Email VARCHAR(100);

ALTER TABLE Student MODIFY Name VARCHAR(150);

DROP TABLE Student;
```

Practical 4

Aim: Create Employee Table with primary key and foreign key constraints,

a. Alter table with add n modify b. Drop table .

```
CREATE TABLE Department (
   DeptID INT PRIMARY KEY,
   DeptName VARCHAR(100)
);

CREATE TABLE Employee (
   EmpID INT PRIMARY KEY,
   EmpName VARCHAR(100),
   Salary DECIMAL(10,2),
   DeptID INT,
   FOREIGN KEY (DeptID) REFERENCES Department(DeptID)
);

ALTER TABLE Employee ADD Email VARCHAR(100);

ALTER TABLE Employee MODIFY EmpName VARCHAR(150);

DROP TABLE Employee;
```

Practical 5

Aim: Implementation of relational operators in SQL

```
create database student_detail;
use student_detail;
create table stud_info (st_id int(3),stud_name varchar(20),stud_subject varchar(20),stud_email
varchar(20));
desc stud_info;
insert into stud_info values(1,"Amaan","M3","as@gmail.com");
insert into stud_info values(2,"Mohit","CG","mh@gmail.com");
insert into stud_info values(3,"Raj","DBMS","rj@gmail.com");
insert into stud_info values(4,"Riya","M3","ri@gmail.com");
select* from stud_info where st_id >= 2;
select* from stud_info where stud_name = "Amaan";
select* from stud_info where st_id < 2;
```

Practical 6

Aim: Implementation of Boolean operators and pattern matching.

```
create database student_detail;
use student_detail;
create table stud_info (st_id int(3),stud_name varchar(20),stud_subject varchar(20),stud_email
varchar(20));
desc stud_info;
insert into stud_info values(1,"Amaan","M3","as@gmail.com");
insert into stud_info values(2,"Mohit","CG","mh@gmail.com");
insert into stud_info values(3,"Raj","DBMS","rj@gmail.com");
insert into stud_info values(4,"Rehan","M3","ri@gmail.com");
select* from stud_info where NOT st_id >3 ;
select* from stud_info where stud_name LIKE "Am%" ;
select* from stud_info where stud_name LIKE "%n" ;
```

Practical 7

Aim: Implementation of Arithmetic operations and built in functions.

```
CREATE DATABASE student_db;
USE student_db;

CREATE TABLE info (
   stud_id INT(2),
   depart_id INT(3),
   name VARCHAR(20)
);

-- View table structure
DESC info;

INSERT INTO info VALUES (1, 111, "Amaan");
INSERT INTO info VALUES (2, 222, "Mohit");
INSERT INTO info VALUES (3, 333, "Krishna");
INSERT INTO info VALUES (4, 444, "Amit");
```

```
SELECT * FROM info;

SELECT * FROM info WHERE (stud_id + depart_id) > 100;
SELECT * FROM info WHERE (stud_id - depart_id) < 170;
SELECT * FROM info WHERE (stud_id * depart_id) > 200;
SELECT * FROM info WHERE (stud_id / depart_id) < 130;

SELECT
   stud_id,
   depart_id,
   ABS(stud_id - depart_id) AS Absolute_Diff,
   MOD(depart_id, stud_id) AS Modulus,
   POWER(stud_id, 2) AS StudID_Squared,
   ROUND(depart_id / stud_id, 2) AS Rounded_Division,
   GREATEST(stud_id, depart_id) AS Max_Value,
   LEAST(stud_id, depart_id) AS Min_Value
FROM info;
```

Practical 8

Aim: Implementation of Group functions.

```
create database student_db;
use student_db;
create table info (stud_id int(2),depart_id int(3),name varchar(20));
desc info;
insert into info values(1,111,"Amaan");
insert into info values(2,222,"Mohit");
insert into info values(3,333,"Krish");
insert into info values(4,444,"Amit");
select* from info;
select AVG(stud_id) from info;
select SUM(stud_id) from info;
select MIN(stud_id) from info;
select MAX(stud_id) from info;
```

Practical 9

Aim: Implementation of processing Date and Time functions.

```
create database Birth_info;
use Birth_info;
create table info (name varchar(20),DOB DATE,Last_login DATETIME);
desc info;
insert into info values("Amaan","2003-01-01",'2023-05-16 04:15:22');
insert into info values("Mohit","2003-10-03",'2023-05-16 05:15:22');
insert into info values("Sanjay","2004-10-03",'2023-10-16 06:15:22');
select* from info;
```

Practical 10

Aim: Implementation of Complex queries and set operators.

```
create database worker_details;
use worker_details;
create table first (id int(2) ,name varchar(20));
create table second (id int(2) ,name varchar(20));
insert into first values(1,"Amaan");
insert into first values(2,"Mohan");
insert into first values(3,"Sam");
insert into second values(3,"Sam");
insert into second values(4,"david");
insert into second values(5,"rock");
select* from first UNION select* from second;
select* from first UNION ALL select* from second;
select* from first INTERSECT select* from second;
```

Practical 11

Aim: Execute DDL/DML statements which demonstrate the use of views. Update the base table using its corresponding view.

```
CREATE DATABASE student_detail;
USE student_detail;

CREATE TABLE stud_info (
   st_id INT(3),
   stud_name VARCHAR(20),
   stud_subject VARCHAR(20),
   stud_email VARCHAR(20)
);

DESC stud_info;

INSERT INTO stud_info VALUES
(1, "Amaan", "M3", "as@gmail.com"),
(2, "Mohit", "CG", "mh@gmail.com"),
(3, "Raj", "DBMS", "rj@gmail.com"),
(4, "Riya", "M3", "ri@gmail.com"),
(5, "Rohan", "CG", "rh@gmail.com");

SELECT * FROM stud_info;

CREATE VIEW V1 AS
SELECT * FROM stud_info WHERE st_id IN (1, 3);

SELECT * FROM V1;

UPDATE V1
SET stud_subject = 'AI', stud_email = 'amaan.ai@gmail.com'
WHERE st_id = 1;
```

```
SELECT * FROM stud_info;
```

Practical 12

Aim: Write and execute PL/SQL stored procedure and function to perform a suitable task on the database. Demonstrate its use.

```
use dem2;

delimiter $
create procedure addp()
begin
   declare a,b,c int;
   set a=2;
   set b=3;
   set c=a+b;
   select concat('value',c);
end;
$

delimiter ;
call addp();

use dem1;

delimiter $
create procedure subp()
begin
   declare a,b,c int;
   set a=2;
   set b=3;
   set c=a-b;
   select concat('value',c);
end;
$

delimiter ;
call subp();
```

Practical 13

Aim: Write and execute suitable database triggers.

```
create database Company;
use Company;
create table Employee (emp_id int(20), emp_name varchar(10));
show tables;
desc Employee;
insert into Employee values (112,"Amaan");
insert into Employee values (100,"Sujeet");
insert into Employee values (102,"Amit");
```

```
Create Trigger sample_trigger before insert on Employee For Each Row set new.emp_id =
new.emp_id+100;
select *from Employee;
insert into Employee values (120,"suda");
select *from Employee;
```

Practical 14

Aim: Write a PL/SQL block to implement all types of cursor.

```
use db1;

delimiter $$
create procedure proce_emp()
begin
  declare v_name varchar(100);
  declare v_id int;
  declare v_finish integer default 0;
  declare c1 cursor for select emp_id, emp_name from employee;
  declare continue handler for NOT FOUND set v_finish=1;
  open c1;
  get_emp: LOOP
    fetch c1 into v_id,v_name;
    if v_finish=1 then
      leave get_emp;
    end if;
    select concat(v_id,v_name);
  END LOOP get_emp;
  close c1;
end $$


call proce_emp();
&&
```