Name:Ashish g patni

Course:Msc-cs-III

Sub:DBMS(P)

Batch:II

Roll:92

Assignmnet-1 page:1

Assignmnet-2 page:12

Assignmnet-3 page:61

Assignmnet-4 page:78

Assignment -1

```
use db1;
show tables;
#1. maximum department
create table tb1(did int,dcount int);
insert into tb1 value(1,15),(2,100),(3,21);
#create procedure for find max
delimiter //
create procedure find_max()
begin
declare big int;
select max(dcount) as find_max into big  from tb1;
select concat("Max department is:",big);
end
```

```
//
delimiter ;
drop procedure find_max;
call find_max();
# concat("Max department is:",big)
-- Max department is:110
—-------------------------------------------------------------------------
#2 find the maximum of deparatment and +10  and create one other department and which is create by user
delimiter //
create procedure create_department()
begin
 DECLARE          max_new int;
 declare max_did int;
 select max(did)+1 into max_did from tb1;
 select max(dcount)+10 into max_new from tb1;
 insert into tb1 value(max_did,max_new);
 end
 //
 delimiter ;
call create_department();
select *from tb1;
# 08:45:50        call create_department()          1 row(s) affected          0.062 sec
-- ------------------------------------------------------------------------------------------------------
#3 update  department id and department count

delimiter //
create procedure update_dep(in dep_id int,in dep_cout int)
begin
```

```
update tb1
set dcount=dep_cout,did=dep_id
where did=4;
end
//
delimiter ;
drop procedure update_dep;
call update_dep(5,300);
select * from tb1;

# did, dcount
-- 2, 100
-- 3, 21
-- 5, 300
-- 5, 120
-- 6, 130
-- ------------------------------------------------------------------------------
#4.delete the deparment
delimiter //
create procedure del_dep(in dep_id int)
begin
delete from tb1
where did=dep_id;
end
//
delimiter ;
call del_dep(5);
select * from tb1;
```

```
# did, dcount
-- 1, 15
-- 2, 100
-- 3, 21
-- 6, 130

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

 #5.accept employy name ,b_salary and other filds
create table employ (
name varchar(20),
b_salary float,
HRA float,
DA float,
NET_SALARY float,
PF float
);
delimiter //
create procedure data_insert(in input_name varchar(20),in input_salary float)
begin

        declare input_hra float;
        declare input_da float;
        declare input_netsalary float;
   declare input_pf float;
   set input_hra = (31*input_salary)/100;
   set input_da =(15*input_salary)/100;
   set input_netsalary=input_salary+input_hra+input_da;
```

```
    if input_salary<3000 then
    set input_pf =(5*input_salary)/100;

    elseif input_salary >=3000 and input_salary<=5000 then
            set input_pf=(7*input_salary)/100;

    elseif input_salary >=5000 and input_salary<=8000 then
    set input_pf=(8*input_salary)/100;

    else
    set input_pf=00;

    end if;

    insert into employ values(input_name,input_salary,input_hra,input_da,input_netsalary,input_pf);

end;
//
delimiter ;
drop procedure data_insert;
call data_insert('yoges',2000);
select * from employ;

# name, b_salary, HRA, DA, NET_SALARY, PF, salary_grade
-- ashish, 200000, 62000, 30000, 292000, ,
-- akash, 20000, 6200, 3000, 29200, 0,
-- yoges, 2000, 620, 300, 2920, 100,
—-------------------------------------------------------------------------------
#7.
```

```
create table employ3(id int auto_increment primary key, salary float,commission float,bonus float);
#procedure
delimiter //
create procedure bonus_count(in input_salary float , in input_commission float)
begin
declare input_bonus float;
        if input_commission is null then
        select "Your employ not eligible for bonus";
        else
        set input_bonus = (15*input_commission)/100;
        end if;
   insert into employ3(salary,commission,bonus)value(input_salary,input_commission,input_bonus);
end
//
delimiter ;
drop procedure bonus_count;
call bonus_count(12000,NULL);
select * from employ3;

-- 'Your employ not eligible for bonus'
—-------------------------------------------------------------------------------
```

#8 create table deparmnet_id,dep_name,no_emp,

```
create table employ4(dep_id int primary key auto_increment ,dep_name varchar(20),no_employ int ,
everage_salary float);
alter table employ4 add total_salary float;
```

```
alter table employ4 drop column everage_salary;
select * from employ4;
delimiter //
create procedure avg_salary()
DELIMITER //

CREATE PROCEDURE avg_salary()
BEGIN
   DECLARE in_dep_id INT;
   DECLARE in_dep_name VARCHAR(10);
   DECLARE in_dep_nemploy INT;
   DECLARE in_dep_tsalary INT;
   DECLARE in_dep_esalary FLOAT;

   SET in_dep_id = 10;

   WHILE in_dep_id <= 40 DO
      SELECT no_employ INTO in_dep_nemploy FROM employ4 WHERE dep_id = in_dep_id;

      IF in_dep_nemploy > 0 THEN
         SELECT dep_name INTO in_dep_name FROM employ4 WHERE dep_id = in_dep_id;
         SELECT total_salary INTO in_dep_tsalary FROM employ4 WHERE dep_id = in_dep_id;
         SET in_dep_esalary = in_dep_tsalary / in_dep_nemploy;

         -- Display results
         SELECT CONCAT('Department ', in_dep_id, ': ', in_dep_name) AS Department;
         SELECT CONCAT('Total Employees: ', in_dep_nemploy) AS Total_Employees;
         SELECT CONCAT('Average Salary: ', in_dep_esalary) AS Average_Salary;
      ELSE
```

```
        -- Handle the case where there are no employees in the department
        SELECT CONCAT('Department ', in_dep_id, ': No employees in this department') AS Department;
    END IF;

    SET in_dep_id = in_dep_id + 10;
  END WHILE;
END;
//
DELIMITER ;
delimiter ;
select * from employ4;
insert into employ4 value(11,'cs',12,1000000),(20,'it',11,11000000),(30,'nikon',null,null);
call avg_salary();

# Department
-- Department 40: No employees in this department

-- ------------------------------------------------------------------------------------------------
-- 9. Write a PL/SQL block which accepts employee number and finds the average salary of the
-- employees working in the department where that employee works.
-- If his salary is more than the average salary of his department, then display message that
-- ‘employee’s salary is more than average salary’ else display ‘employee’s salary is less than
-- average salary’

select * from employ;
INSERT INTO employ (name, b_salary, HRA, DA, NET_SALARY, PF)
VALUES
 ('Sarah Johnson', 62000.00, 12400.00, 9300.00, 65100.00, 7440.00),
 ('Matthew Williams', 58000.00, 11600.00, 8700.00, 60900.00, 6960.00),
```

```
  ('Olivia Davis', 53000.00, 10600.00, 7950.00, 55650.00, 6360.00),
  ('Ethan Anderson', 72000.00, 14400.00, 10800.00, 75600.00, 8640.00),
  ('Ava Jackson', 55000.00, 11000.00, 8250.00, 57750.00, 6600.00);

  INSERT INTO employ (name, b_salary, HRA, DA, NET_SALARY, PF)
VALUES
  ('Noah Garcia', 61000.00, 12200.00, 9150.00, 64150.00, 7320.00),
  ('Mia Miller', 49000.00, 9800.00, 7350.00, 51450.00, 5880.00),
  ('Liam Martinez', 67000.00, 13400.00, 10050.00, 70500.00, 8040.00),
  ('Emma Johnson', 54000.00, 10800.00, 8100.00, 56700.00, 6480.00),
  ('James Wilson', 63000.00, 12600.00, 9450.00, 66150.00, 7560.00);

INSERT INTO employ (name, b_salary, HRA, DA, NET_SALARY, PF)
VALUES
  ('Oliver Smith', 59000.00, 11800.00, 8850.00, 61950.00, 7080.00),
  ('Isabella Davis', 52000.00, 10400.00, 7800.00, 54600.00, 6240.00),
  ('Benjamin Lee', 71000.00, 14200.00, 10650.00, 74700.00, 8520.00),
  ('Charlotte Wilson', 50000.00, 10000.00, 7500.00, 52500.00, 6000.00),
  ('Luna Johnson', 59000.00, 11800.00, 8850.00, 61950.00, 7080.00);

INSERT INTO employ (name, b_salary, HRA, DA, NET_SALARY, PF)
VALUES
  ('Logan Davis', 54000.00, 10800.00, 8100.00, 56700.00, 6480.00),
  ('Harper Smith', 64000.00, 12800.00, 9600.00, 67200.00, 7680.00),
  ('Elijah Johnson', 56000.00, 11200.00, 8400.00, 58800.00, 6720.00),
  ('Amelia Wilson', 59000.00, 11800.00, 8850.00, 61950.00, 7080.00),
  ('Mason Lee', 73000.00, 14600.00, 10950.00, 76650.00, 8760.00);

DELIMITER //
```

```
CREATE PROCEDURE find_avg_emp_and_store()
BEGIN
 DECLARE avg_Salary FLOAT;
 DECLARE emp_id INT;
 DECLARE emp_salary FLOAT;
 DECLARE emp_name VARCHAR(50);
 DECLARE emp_result_message VARCHAR(100);

 -- Create a temporary table to store the results
 CREATE TEMPORARY TABLE temp_results (
  employee_name VARCHAR(50),
  result_message VARCHAR(100)
 );

 -- Calculate the average salary
 SELECT AVG(b_salary) INTO avg_Salary FROM employ;

 -- Initialize the employee_id
 SET emp_id = 1;

 WHILE emp_id <= (SELECT MAX(ROWID) FROM employ) DO
  -- Fetch employee data based on ROWID
  SELECT name, b_salary INTO emp_name, emp_salary
  FROM employ
  WHERE ROWID = emp_id;

  IF emp_salary IS NULL THEN
   SET emp_result_message = 'This employee has no salary information';
  ELSE
```

```
    IF emp_salary = avg_Salary THEN
      SET emp_result_message = 'This employee has an average salary';
    ELSEIF emp_salary > avg_Salary THEN
      SET emp_result_message = 'This employee has more than the average salary';
    ELSE
      SET emp_result_message = 'This employee has less than the average salary';
    END IF;
  END IF;

  -- Insert the result into the temporary table
  INSERT INTO temp_results (employee_name, result_message)
  VALUES (emp_name, emp_result_message);

  -- Increment the employee_id
  SET emp_id = emp_id + 1;
 END WHILE;

 -- Select and display the results
 SELECT * FROM temp_results;

 -- Drop the temporary table when done
 DROP TEMPORARY TABLE temp_results;
END;
//
DELIMITER ;


drop procedure find_avg_emp_and_store;
call find_avg_emp_and_store();
```

select * from employ

-

—----------

Assignment-2

---

1. Write a PL/SQL block that selects the maximum department number in the department table and store it in a SQL*PLUS variable. And print the results to screen.

```
create database sem3;
use sem3;

create table  department
(
        department_id int,
   department_num int,
        department_name varchar(50),
        location_de varchar(30)
);

insert into department value
("1","89","rollwala","ahmedabad"),
("2","56","jinal","ahmedabad"),
```

```
("3","34","sk","mahesana"),
("4","12","gls","ahmedabad"),
("5","43","silvar","ahmedabad");

call dep_number();



CREATE DEFINER=`root`@`localhost` PROCEDURE `dep_number`()
BEGIN
        declare max_num int;

  set max_num =  (select max(department_num) from department);

  select max_num;
END
```

output:-

max_num
89

---

2. Create a PL/SQL block to insert a new department number into the Departments table. Use maximum dept number fetched from above and adds 10 to it. Use SQL*PLUS substitution variable for department name. Leave the location AS null.

```
create database sem3;
use sem3;

create table  department
(
        department_id int,
   department_num int,
        department_name varchar(50),
        location_de varchar(30)
);

insert into department value
("1","89","rollwala","ahmedabad"),
("2","56","nirma","ahmedabad"),
("3","34","sk","mahesana"),
("4","12","gls","ahmedabad"),
("5","43","silvar","ahmedabad");

select * from department;

call dep_number();

CREATE DEFINER=`root`@`localhost` PROCEDURE `dep_number`()
```

```
BEGIN
        declare max_num int;

  set max_num =  (select max(department_num) from department);

  select max_num;

  set max_num = max_num+10;
  set @department_name ="new deparment";

  insert into department(department_num,department_name) value (max_num,@department_name);
END
```

output:-

1	89	rollwala	ahmedabad
2	56	nirma	ahmedabad
3	34	sk	mahesana
4	12	gls	ahmedabad
5	43	silvar	ahmedabad
	99	new deparment	null

---

3. Create a PL/SQL block to update the location for an existing department. Use substitution variable for dept no. and dept location.

```
create database sem3;
use sem3;

create table  department
(
        department_id int,
   department_num int,
        department_name varchar(50),
        location_de varchar(30)
);

insert into department value
("1","89","rollwala","ahmedabad"),
("2","56","nirma","ahmedabad"),
("3","34","sk","mahesana"),
("4","12","gls","ahmedabad"),
("5","43","silvar","ahmedabad");

select * from department;

call dep_number();


CREATE DEFINER=`root`@`localhost` PROCEDURE `dep_change_info`()
BEGIN
        declare dep_no int;
   declare dep_loc varchar(10);
   set dep_no =10;
   set dep_loc ="surat";
```

```
    update department set location_de = dep_loc where department_num = dep_no;
END
```

output:-

```
1       89      rollwalaahmedabad
2       56      nirma   ahmedabad
3       34      sk      mahesana
4       12      gls     ahmedabad
5       43      silvar  ahmedabad
        99      new deparment
```

---

4. Create a PL/SQL Block to delete the department created in exercise 2. Print to the screen the number of rows affected.

```
create database sem3;
use sem3;

create table  department
(
        department_id int,
    department_num int,
        department_name varchar(50),
        location_de varchar(30)
```

```
);

insert into department value
("1","89","rollwala","ahmedabad"),
("2","56","nirma","ahmedabad"),
("3","34","sk","mahesana"),
("4","12","gls","ahmedabad"),
("5","43","silvar","ahmedabad");

select * from department;

call dep_number();
call dep_change_info();
call dep_del();



CREATE DEFINER=`root`@`localhost` PROCEDURE `dep_del`()
BEGIN
	set @max_no = (select max(department_num) from department);

	delete from department where department_num = @max_no;

	select concat('Number of rows affected: ', row_count()) as result;
END
```

output:-

```
Number of rows affected: 1
```

---

5. Write a PL/SQL block which accepts employee name, basic and should display Employee name, PF and net salary.

HRA=31% of basic salary

DA=15% of basic salary

Net salary=basic+HRA+DA-PF

If the basic is less than 3000 PF is 5% of basic salary.

If the basic is between 3000 and 5000 PF is 7% of basic salary.

If the basic is between 5000 and 8000 PF is 8% of basic salary.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `emp_sal`()
BEGIN
        set @emp_name = 'sagar';
        set @basic_salary = 5000;
        set @pf_rate = case
          when @basic_salary < 3000 then 0.05
          when @basic_salary between 3000 and 5000 then 0.07
          when @basic_salary between 5000 and 8000 then 0.08
         end;

set @pf = @basic_salary * @pf_rate;
set @hra = @basic_salary * 0.31;
set @da = @basic_salary * 0.15;
```

```
set @net_salary = @basic_salary + @hra + @da - @pf;

SELECT CONCAT('Employee Name: ', @emp_name," PF : ",@pf," Net Salary: ",@net_salary) AS result ;

END

call emp_sal();
```

output:-

result

Employee Name: sagar PF : 350.00 Net Salary: 6950.00

---

6. Write a PL/SQL block to find the salary grade of the specified employee.

If grade is 1 display ‘the employee is junior engineer’

If grade is 2 display ‘the employee is engineer’

If grade is 3 display ‘the employee is lead engineer’

If grade is 4 display ‘the employee is Manager’

If grade is 5 display ‘the employee is Project manager’

(Use case expression)

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `employee_grade`()
BEGIN
        set @emp_grade = 1;
         set @grade = case
                when @emp_grade = 1 then "the employee is junior engineer"
         when @emp_grade = 2 then "the employee is engineer"
         when @emp_grade = 3 then "the employee is lead engineer"
         when @emp_grade = 4 then "the employee is Manager"
         when @emp_grade = 5 then "the employee is Project manager"
         end;

select concat("Employee grade is ",@emp_grade," specified is ",@grade);
END


call employee_grade();
```

output:-

Employee grade is 1 specified is the employee is junior engineer

---

7. Wrtie a PL/SQL block to award an employee with the bonus. Bonus is 15% of commission drawn by the employee. If the employee does not earn any commission then

display a message that ‘employee does not earn any commission’. Otherwise add bonus to the salary of the employee. The block should accept an input for the

employee number.

```
create database dw;

use dw;


create table employees (
   employee_id int primary key,
   employee_name varchar(50),
   commission decimal(10, 2),
   salary decimal(10, 2)
);

-- insert sample data into the employee table
insert into employees (employee_id, employee_name, commission, salary)
values
   (1, 'Ashish', 500.00, 3000.00),
   (2, 'jinal', 0.00, 2500.00),
   (3, 'jay', 750.00, 3500.00),
   (4, 'dev', 0.00, 2800.00),
   (5, 'tej', 200.00, 3200.00);

select*from employees;

CALL award_bonus(3);
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `award_bonus`(in emp_id int)
begin
   declare commission_amount decimal(10, 2);
   declare bonus decimal(10, 2);


   select commission into commission_amount from employees where employee_id = emp_id;


   if commission_amount > 0 then

      set bonus = commission_amount * 0.15;


      update employees set salary = salary + bonus where employee_id = emp_id;

      select concat('bonus of $', bonus, ' awarded to employee ', emp_id) as message;
   else
      select 'employee does not earn any commission' as message;
   end if;
end
```

output:-

message

bonus of $112.50 awarded to employee 3

---

8. Write a PL/SQL block which displays the department name, total no of employees in the department, avg salary of the employees in the department for all the

departments from department 10 to department 40 in the Dept table.If no employees are working in the department ,then display a message that no employees are

working in that department.

```
use dw;


-- create a "dept" table for testing purposes
create table dept (
   deptno int primary key,
   dname varchar(50),
   loc varchar(50),
   total_emp int,
   salary int
);


insert into dept (deptno, dname, loc,total_emp,salary)
values
   (10, 'Accounting', 'New York',45,60000),
   (20, 'Research', 'Dallas',' 0',7654),
   (30, 'Sales', 'Chicago','23',98765),
   (40, 'Marketing', 'Los Angeles',90,100000);
```

```
select*from dept;

call info(10);
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `info`(in depid int)
begin
   declare deptn varchar(20);
   declare ab int;
   declare cd int;

   set ab=depid;
   select dname into deptn from dept where deptno=ab;
   select total_emp into cd from dept where deptno=ab;

   if cd=0 then
   select 'NO EMPLOYEES ARE WORKING IN THIS DEPARTMENT' as messege;
   else
   select concat('IN',deptn,' DEPARTMENT',cd,'EMPLOYEES ARE WORKING') as messege;
   end if;
end
```

output:-

call info(10);

INAccounting DEPARTMENT10EMPLOYEES ARE WORKING

call info(20);

NO EMPLOYEES ARE WORKING IN THIS DEPARTMENT

---

9. Write a PL/SQL block which accepts employee number and finds the average salary of the employees working in the department where that employee works.

If his salary is more than the average salary of his department, then display message that ‘employee’s salary is more than average salary’ else display

‘employee’s salary is less than average salary’

---

use dw;

create table emp1

(

eid int,

ename varchar(20),

dep varchar(20),

salary int,

avgsalary int

);

```
insert into emp1 values
(12,'jay','Accounting',5000,2000),
(15,'dev','Marketing',25000,20000),
(18,'tej','Research',5900,10000),
(29,'Om','Sales',8000,2290),
(38,'Ashish','production',5480,9640);

select *from emp1;

call emp(15);

CREATE DEFINER=`root`@`localhost` PROCEDURE `emp`(in eid1 int)
BEGIN
        declare av int;
  declare sal int;
  declare tt int;
  set tt=eid1;


  select avgsalary into av from emp1 where eid=tt;
  select salary into sal from emp1 where eid=tt;

  if av>sal then
  select 'Employee salary is less than average salary of department'as text;
  else
  select 'Employee salary is more than average salary of department'as text;
  end if;

END
```

output:-

call emp(15);

Employee salary is more than average salary of department

call emp(18);

Employee salary is less than average salary of department

---

10. Create a procedure that deletes rows from the emp table. It should accept 1 parameter, job; only delete the employee's with that job. Display how many

employees were deleted.

---

use dw;

```
create table emp2 (
   employee_id int auto_increment primary key,
   first_name varchar(50),
   last_name varchar(50),
   job_title varchar(100),
```

```
    hire_date date,
    salary decimal(10, 2),
    department_id int
);

-- Insert data into the employees table
insert into emp2 (first_name, last_name, job_title, hire_date, salary, department_id)
values
    ('Ashish', 'Patni', 'Manager', '2022-01-15', 60000.00, 1),
    ('Jinal', 'patel', 'Developer', '2022-02-20', 55000.00, 2),
    ('Darshan', 'virugama', 'Salesperson', '2022-03-10', 48000.00, 3),
    ('Jay', 'prajapati', 'Designer', '2022-04-05', 52000.00, 1),
    ('Hiren', 'Chaudhari', 'Analyst', '2022-05-15', 58000.00, 2);

select*from emp2;


CREATE DEFINER=`root`@`localhost` PROCEDURE `deleterow`(in jobtodelete varchar(255))
begin
    declare row_count int;


    delete from emp2 where job_title = jobtodelete;


    select row_count() into row_count;


    select concat(row_count, ' employees deleted.') as result;
```

end

output:-

1	John	Doe	Manager	2022-01-15	60000.00	1
2	Jane	Smith	Developer	2022-02-20	55000.00	2
3	Bob	Johnson	Salesperson	2022-03-10	48000.00	3
4	Alice	Brown	Designer	2022-04-05	52000.00	1
5	Charlie	Davis	Analyst	2022-05-15	58000.00	2

After delete row:-

1 employees deleted.

1	Ashish	Patni	Manager	2022-01-15	60000.00	1
2	Jinal	patel	Developer	2022-02-20	55000.00	2
3	Darshan	virugama	Salesperson	2022-03-10	48000.00	3
5	Hiren	Chaudhari	Analyst	2022-05-15	58000.00	2

---

11. Change the above procedure so that it returns the number of employees removed via an OUT parameter.

```
use dw;
create table emp2 (
   employee_id int auto_increment primary key,
   first_name varchar(50),
   last_name varchar(50),
   job_title varchar(100),
   hire_date date,
   salary decimal(10, 2),
   department_id int
);

-- Insert data into the employees table
insert into emp2 (first_name, last_name, job_title, hire_date, salary, department_id)
values
   ('Ashish', 'Patni', 'Manager', '2022-01-15', 60000.00, 1),
   ('Jinal', 'patel', 'Developer', '2022-02-20', 55000.00, 2),
   ('Darshan', 'virugama', 'Salesperson', '2022-03-10', 48000.00, 3),
   ('Jay', 'prajapati', 'Designer', '2022-04-05', 52000.00, 1),
   ('Hiren', 'Chaudhari', 'Analyst', '2022-05-15', 58000.00, 2);

select*from emp2;
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `deleterow`(in jobtodelete varchar(255))
begin
   declare row_count int;

   delete from emp2 where job_title = jobtodelete;

   select row_count() into row_count;

   select concat(row_count, ' employees deleted.') as result;
end
```

output:-

1	John	Doe	Manager	2022-01-15	60000.00	1
2	Jane	Smith	Developer	2022-02-20	55000.00	2
3	Bob	Johnson	Salesperson	2022-03-10	48000.00	3
4	Alice	Brown	Designer	2022-04-05	52000.00	1
5	Charlie	Davis	Analyst	2022-05-15	58000.00	2

After delete row:-

1 employees deleted.

1	Ashish	Patni	Manager	2022-01-15	60000.00	1	
2	Jinal	patel	Developer	2022-02-20	55000.00	2	
3	Darshan		virugama	Salesperson	2022-03-10	48000.00	3
5	Hiren	Chaudhari		Analyst	2022-05-15	58000.00	2

---

12. Convert the above program to a function. Instead of using an OUT parameter for the number of employees deleted, use the functions return value and display how many

employees were deleted.

---

cursor:-

14.14. Write a PL/SQL block to accept an employee number. and use a record variable to store the

record of that employee. and insert it into retired_employee table.

Retired_employee table has the following structure

Retired_employee (empno, ename, hiredate, leaveDate, salary, mgr_id, deptno).

Set the leavedate to the current date.

```
use dw;


CREATE TABLE employes (
  empno INT PRIMARY KEY,
  ename VARCHAR(50),
  hiredate DATE,
  salary DECIMAL(10, 2),
  mgr_id INT,
  deptno INT
);


INSERT INTO employes (empno, ename, hiredate, salary, mgr_id, deptno)
VALUES
  (1, 'Ashish', '2020-01-15', 50000.00, 0, 10),
  (2, 'Jay', '2019-03-22', 60000.00, 1, 20),
  (3, 'yusuf', '2020-07-10', 55000.00, 1, 20),
  (4, 'jannat', '2018-11-05', 48000.00, 2, 30),
  (5, 'jinal', '2021-05-18', 52000.00, 2, 30);


CREATE TABLE retired_employee (
  empno INT PRIMARY KEY,
  ename VARCHAR(50),
  hiredate DATE,
  leaveDate DATE,
  salary DECIMAL(10, 2),
  mgr_id INT,
```

```
  deptno INT
);



select*from employes;
select*from retired_employee;
CALL retire_employees_with_condition(55000.00);

CREATE DEFINER=`root`@`localhost` PROCEDURE `retire_employees_with_condition`(IN
salary_threshold DECIMAL(10, 2))
BEGIN
  DECLARE done INT DEFAULT FALSE;
  DECLARE v_empno INT;
  DECLARE v_ename VARCHAR(50);
  DECLARE v_hiredate DATE;
  DECLARE v_salary DECIMAL(10, 2);
  DECLARE v_mgr_id INT;
  DECLARE v_deptno INT;
  DECLARE v_leaveDate DATE;

  DECLARE employee_cursor CURSOR FOR
    SELECT empno, ename, hiredate, salary, mgr_id, deptno
    FROM employes
    WHERE salary <= salary_threshold;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

SET v_leaveDate = CURDATE();
```

```
OPEN employee_cursor;

 read_loop: LOOP
  FETCH employee_cursor INTO v_empno, v_ename, v_hiredate, v_salary, v_mgr_id, v_deptno;

  IF done THEN
    LEAVE read_loop;
  END IF;


  INSERT INTO retired_employee (empno, ename, hiredate, leaveDate, salary, mgr_id, deptno)
  VALUES (v_empno, v_ename, v_hiredate, v_leaveDate, v_salary, v_mgr_id, v_deptno);
 END LOOP;

 CLOSE employee_cursor;

 -- Commit the transaction
 COMMIT;
END
```

---

output:-

1	Ashish	2020-01-15	2023-10-02	50000.00	0	10
3	yusuf	2020-07-10	2023-10-02	55000.00	1	20
4	jannat	2018-11-05	2023-10-02	48000.00	2	30
5	jinal	2021-05-18	2023-10-02	52000.00	2	30

15.Write a PL/SQL Block to create a PL/SQL table which can store grade and on of employees with that grade. Get the information about the grade and number of employees with that grade and store it in the PL/SQL table. Then retrieve the information from the PL/SQL table and display it on the screen in the following way.

No of employees with the grade 1 are 3

No of employees with the grade 2 are 2

No of employees with the grade 3 are 1

No of employees with the grade 4 are 2

No of employees with the grade 5 are 5

```
use dw;

CREATE DEFINER=`root`@`localhost` PROCEDURE `calculate_employee_grades`()
BEGIN
  DECLARE v_grade INT;
  DECLARE v_employee_count INT;
  DECLARE v_done INT DEFAULT FALSE;


  DECLARE grade_cursor CURSOR FOR
    SELECT grade, employee_count FROM temp_employee_grades2;


  DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_done = TRUE;


  CREATE TEMPORARY TABLE temp_employee_grades2 (
    grade INT,
    employee_count INT
```

```
);

INSERT INTO temp_employee_grades2 (grade, employee_count)
VALUES
 (1, 3),
 (2, 2),
 (3, 1),
 (4, 2),
 (5, 5);

OPEN grade_cursor;

display_loop: LOOP
 FETCH grade_cursor INTO v_grade, v_employee_count;

 IF v_done THEN
  LEAVE display_loop;
 END IF;

 SELECT CONCAT('No of employees with the grade ', v_grade, ' are ', v_employee_count) AS message;

END LOOP;

CLOSE grade_cursor;

DROP TEMPORARY TABLE temp_employee_grades2;
```

END

---

output:-

No of employees with the grade 1 are 3

No of employees with the grade 2 are 2

No of employees with the grade 3 are 1

No of employees with the grade 4 are 2

No of employees with the grade 5 are 5

---

16. Write a program that gives all employees in department 10 a 15% pay increase. Display a message displaying how many Employees were awarded the increase.

```
use dw;


CREATE TABLE empls (
   employee_id INT PRIMARY KEY,
   first_name VARCHAR(50),
   last_name VARCHAR(50),
   department_id INT,
   salary DECIMAL(10, 2)
);


INSERT INTO empls (employee_id, first_name, last_name, department_id, salary)
```

```
VALUES
   (1, 'Ashish', 'patni', 10, 50000.00),
   (2, 'jannat', 'naikh', 10, 55000.00),
   (3, 'Aman', 'gupta', 20, 60000.00),
   (4, 'gautam', 'chauhan', 10, 48000.00),
   (5, 'Drshti', 'patoliya', 10, 52000.00);

select*from empls;
CALL IncreaseSalariesForDept10();



CREATE DEFINER=`root`@`localhost` PROCEDURE `IncreaseSalariesForDept10`()
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE emp_id INT;
   DECLARE cur CURSOR FOR
      SELECT employee_id FROM empls WHERE department_id = 10;


   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

   OPEN cur;


   employee_loop: LOOP
      FETCH cur INTO emp_id;

      IF done THEN
```

```
            LEAVE employee_loop;
        END IF;


        UPDATE empls
        SET salary = salary * 1.15
        WHERE employee_id = emp_id;
    END LOOP;

    CLOSE cur;


    SELECT COUNT(*) AS num_employees_awarded
    FROM empls
    WHERE department_id = 10;
END
```

OUTPUT:-

1	Ashish	patni	10	57500.00
2	jannat	naikh	10	63250.00
3	Aman	gupta	20	60000.00
4	gautam	chauhan	10	55200.00

5 Drshti patoliya10 59800.00

---

17. Write a PL/SQL block and use cursor to retrieve the details of the employees with grade 5.and then display employee no,job_id ,max_sal and min_sal and grade for all these employees.

```
use dw;


CREATE TABLE empl (
   employee_id INT PRIMARY KEY,
   job_id VARCHAR(50),
   salary DECIMAL(10, 2),
   grade INT
);


INSERT INTO empl (employee_id, job_id, salary, grade)
VALUES
   (101, 'Manager', 75000.00, 5),
   (102, 'Analyst', 60000.00, 5),
   (103, 'Engineer', 65000.00, 4),
   (104, 'Clerk', 45000.00, 3),
   (105, 'Analyst', 62000.00, 5);


select*from empl;
```

```
CALL GetEmployeeDetailsWithGrade5();



CREATE DEFINER=`root`@`localhost` PROCEDURE `GetEmployeeDetailsWithGrade5`()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE emp_id INT;
    DECLARE job_id VARCHAR(50);
    DECLARE max_sal DECIMAL(10, 2);
    DECLARE min_sal DECIMAL(10, 2);
    DECLARE emp_grade INT;



    DECLARE cur CURSOR FOR
        SELECT e.employee_id, e.job_id, MAX(e.salary) AS max_sal, MIN(e.salary) AS min_sal, e.grade
        FROM empl e
        WHERE e.grade = 5
        GROUP BY e.employee_id, e.job_id, e.grade;



    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;



    CREATE TEMPORARY TABLE temp_employee_details (
        employee_id INT,
        job_id VARCHAR(50),
        max_sal DECIMAL(10, 2),
        min_sal DECIMAL(10, 2),
        grade INT
```

```
    );

    OPEN cur;

    employee_loop: LOOP
        FETCH cur INTO emp_id, job_id, max_sal, min_sal, emp_grade;

        IF done THEN
            LEAVE employee_loop;
        END IF;

        INSERT INTO temp_employee_details (employee_id, job_id, max_sal, min_sal, grade)
        VALUES (emp_id, job_id, max_sal, min_sal, emp_grade);
    END LOOP;

    CLOSE cur;

    SELECT * FROM temp_employee_details;

    DROP TEMPORARY TABLE IF EXISTS temp_employee_details;

END
```

---

output:-

101 Manager 75000.00 75000.00 5

102 Analyst 60000.00 60000.00 5

105 Analyst 62000.00 62000.00 5

---

18. Write a PL/SQL block that copies all departments to a table called old_dept. Do not use a cursor FOR loop. Display how many rows were copied.

```
use dw;


CREATE TABLE departments (
   department_id INT PRIMARY KEY,
   department_name VARCHAR(50)
);


INSERT INTO departments (department_id, department_name)
VALUES
   (10, 'HR'),
   (20, 'Finance'),
   (30, 'Sales'),
   (40, 'Marketing'),
```

```
    (50, 'IT');

select*from departments;
CALL CopyDepartmentsToOldDept();

CREATE DEFINER=`root`@`localhost` PROCEDURE `CopyDepartmentsToOldDept`()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE dept_id INT;
    DECLARE dept_name VARCHAR(50);


    DECLARE cur CURSOR FOR
        SELECT department_id, department_name FROM departments;


    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;


    CREATE TABLE IF NOT EXISTS old_dept (
        department_id INT PRIMARY KEY,
        department_name VARCHAR(50)
    );

    OPEN cur;


    department_loop: LOOP
        FETCH cur INTO dept_id, dept_name;
```

```
        IF done THEN
            LEAVE department_loop;
        END IF;


        INSERT INTO old_dept (department_id, department_name)
        VALUES (dept_id, dept_name);
    END LOOP;

    CLOSE cur;

    -- Get the count of rows copied
    SELECT COUNT(*) AS num_rows_copied FROM old_dept;

END
```

---

output:-

```
5
10      HR
20      Finance
30      Sales
40      Marketing
```

50 IT

---

19. Display the names of employees who are working for Department 30.

```
use dw;



CREATE TABLE employees (
   employee_id INT PRIMARY KEY,
   first_name VARCHAR(50),
   last_name VARCHAR(50),
   department_id INT
);



INSERT INTO employees (employee_id, first_name, last_name, department_id)
VALUES
   (101, 'John', 'Doe', 30),
   (102, 'Jane', 'Smith', 30),
   (103, 'Alice', 'Johnson', 20),
   (104, 'Bob', 'Williams', 30),
   (105, 'Eva', 'Brown', 40);

select*from employees;
CALL GetEmployeesInDept30();
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `GetEmployeesInDept30`()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE first_name VARCHAR(50);
    DECLARE last_name VARCHAR(50);


    DECLARE cur CURSOR FOR
        SELECT first_name, last_name
        FROM employees
        WHERE department_id = 30;


    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;


    employee_loop: LOOP
        FETCH cur INTO first_name, last_name;

        IF done THEN
            LEAVE employee_loop;
        END IF;


        SELECT CONCAT(first_name, ' ', last_name) AS employee_name;
    END LOOP;
```

```
    CLOSE cur;

END
```

---

20. Write a PL/SQL Block that mimics selecting all columns and rows from the dept table. There is no need to format the output, just select all columns and all rows. Use a cursor FOR loop.

```
use dw;



CREATE TABLE dept (
    dept_id INT PRIMARY KEY,
    dept_name VARCHAR(50)
);



INSERT INTO dept (dept_id, dept_name)
VALUES
    (1, 'HR'),
    (2, 'Finance'),
    (3, 'Sales'),
    (4, 'Marketing'),
    (5, 'IT');
```

```
select*from dept;
CALL SelectAllFromDept();



CREATE DEFINER=`root`@`localhost` PROCEDURE `SelectAllFromDept`()
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE dept_id INT;
   DECLARE dept_name VARCHAR(50);



   DECLARE cur CURSOR FOR
      SELECT *
      FROM dept;



   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;


   OPEN cur;



   dept_loop: LOOP
      FETCH cur INTO dept_id, dept_name;


      IF done THEN
         LEAVE dept_loop;
      END IF;
```

```
    SELECT dept_id, dept_name;
  END LOOP;

  CLOSE cur;

END
```

---

output:-

1 HR

2 Finance

3 Sales

4 Marketing

5 IT

---

21. Write a PL/SQL block to display the top 6 employees with respect to salaries using cursors.

```
use dw;

CREATE TABLE employees (
  employee_id INT PRIMARY KEY,
  employee_name VARCHAR(255),
  salary DECIMAL(10, 2)
);
```

```
INSERT INTO employees (employee_id, employee_name, salary)
VALUES
   (1, 'Ashish', 60000.00),
   (2, ''yusuf', 65000.00),
   (3, 'nirmal', 75000.00),
   (4, 'jannat', 70000.00),
   (5, 'hiren', 80000.00);

select*from employees;
CALL GetTopEmployees();



CREATE DEFINER=`root`@`localhost` PROCEDURE `GetTopEmployees`()
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE employee_id INT;
   DECLARE employee_name VARCHAR(255);
   DECLARE salary DECIMAL(10, 2);


   DECLARE cur CURSOR FOR
      SELECT employee_id, employee_name, salary
      FROM employees
      ORDER BY salary DESC
      LIMIT 6;


   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
```

```
CREATE TEMPORARY TABLE IF NOT EXISTS top_employees (
   employee_id INT,
   employee_name VARCHAR(255),
   salary DECIMAL(10, 2)
);

OPEN cur;

read_loop: LOOP
   FETCH cur INTO employee_id, employee_name, salary;
   IF done THEN
      LEAVE read_loop;
   END IF;


   INSERT INTO top_employees (employee_id, employee_name, salary)
   VALUES (employee_id, employee_name, salary);
END LOOP;

CLOSE cur;


SELECT * FROM top_employees;


DROP TEMPORARY TABLE IF EXISTS top_employees;
```

END

---

output:-

1 Ashish 60000.00

2 yusuf 65000.00

3 nirmal 75000.00

4 jannat 70000.00

5 hiren 80000.00

---

22. Use a cursor to retrieve the department number and the department name from the dept table. Pass the department number to another cursor to retrieve from the emp table the details of employee name, job, hiredate and salary of all the employees who work in that department.

```
use dw;

CREATE TABLE dept (
   deptno INT PRIMARY KEY,
   dname VARCHAR(50)
);

CREATE TABLE emp (
   empno INT PRIMARY KEY,
   ename VARCHAR(50),
   job VARCHAR(50),
```

```
    hiredate DATE,
    salary DECIMAL(10, 2),
    deptno INT,
    FOREIGN KEY (deptno) REFERENCES dept(deptno)
);



INSERT INTO dept (deptno, dname)
VALUES
    (10, 'HR'),
    (20, 'IT'),
    (30, 'Finance');

INSERT INTO emp (empno, ename, job, hiredate, salary, deptno)
VALUES
    (1, 'Ashish, 'Manager', '2023-01-10', 70000.00, 10),
    (2, 'Jayesh', 'Developer', '2022-11-15', 60000.00, 20),
    (3, 'yusuf', 'Accountant', '2022-09-20', 55000.00, 30),
    (4, 'nilesh', 'Developer', '2022-08-05', 62000.00, 20),
    (5, 'Jayesh', 'Analyst', '2023-02-28', 58000.00, 30);

select*from dept;
select*from emp;

CALL GetDepartmentAndEmployeeDetails();



CREATE DEFINER=`root`@`localhost` PROCEDURE `GetDepartmentAndEmployeeDetails`()
```

```
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE dept_number INT;
    DECLARE dept_name VARCHAR(50);

    DECLARE employee_name VARCHAR(50);
    DECLARE employee_job VARCHAR(50);
    DECLARE employee_hiredate DATE;
    DECLARE employee_salary int;



    DECLARE dept_cursor CURSOR FOR
        SELECT deptno, dname
        FROM dept;



    DECLARE emp_cursor CURSOR FOR
        SELECT ename, job, hiredate, salary
        FROM emp
        WHERE deptno = dept_number;



    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN dept_cursor;

    read_dept_loop: LOOP
        FETCH dept_cursor INTO dept_number, dept_name;
        IF done THEN
```

```
            LEAVE read_dept_loop;
        END IF;


        SELECT CONCAT('Department Number: ', dept_number, ', Department Name: ', dept_name) AS
Department_Info;


        OPEN emp_cursor;
        FETCH emp_cursor INTO employee_name, employee_job, employee_hiredate, employee_salary;


        SELECT CONCAT('Employee Name: ', employee_name, ', Job: ', employee_job, ', Hire Date: ',
employee_hiredate, ', Salary: ', employee_salary) AS Employee_Details;


        CLOSE emp_cursor;
    END LOOP;


    CLOSE dept_cursor;
END
```

---

output:-

```
Department Number: 10, Department Name: HR
Employee Name: Ashish, Job: Manager, Hire Date: 2023-01-10, Salary: 70000
Department Number: 20, Department Name: IT
Employee Name: Jayesh, Job: Developer, Hire Date: 2022-11-15, Salary: 60000
Department Number: 30, Department Name: Finance
Employee Name: Aman, Job: Accountant, Hire Date: 2022-09-20, Salary: 55000
```

---

23. Write a procedure Raise_salary which gives a specified hike to all the employees working in a specified department.The procedure should take department number and percemtage of hike as input.(Use for update and where current of)

```
use dw;
CREATE TABLE emp (
   empno INT PRIMARY KEY,
   ename VARCHAR(50),
   job VARCHAR(50),
   hiredate DATE,
   salary DECIMAL(10, 2),
   deptno INT,
   FOREIGN KEY (deptno) REFERENCES dept(deptno)
);

INSERT INTO emp (empno, ename, job, hiredate, salary, deptno)
VALUES
   (1, 'Ashish', 'Manager', '2023-01-10', 70000.00, 10),
   (2, 'yusuf', 'Developer', '2022-11-15', 60000.00, 20),
   (3, 'NIrmal', 'Accountant', '2022-09-20', 55000.00, 30),
   (4, 'jannat', 'Developer', '2022-08-05', 62000.00, 20),
   (5, 'Drshti', 'Analyst', '2023-02-28', 58000.00, 30);
```

```
select*from emp;

CALL Raise_salary(20, 10);



CREATE DEFINER=`root`@`localhost` PROCEDURE `Raise_salary`(IN p_deptno INT, IN p_percentage
DECIMAL(5,2))

BEGIN

   DECLARE done INT DEFAULT FALSE;

   DECLARE emp_id INT;

   DECLARE emp_salary DECIMAL(10,2);



   DECLARE emp_cursor CURSOR FOR

      SELECT empno, salary

      FROM emp

      WHERE deptno = p_deptno

      FOR UPDATE;



   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;


   OPEN emp_cursor;


   update_loop: LOOP

      FETCH emp_cursor INTO emp_id, emp_salary;

      IF done THEN

         LEAVE update_loop;

      END IF;
```

```
        SET @new_salary = emp_salary * (1 + (p_percentage / 100));


        UPDATE emp
        SET salary = @new_salary;

    END LOOP;

    CLOSE emp_cursor;
END
```

---

output:-

1	Ashish	Manager	2023-01-10	75020.00	10
2	yusuf	Developer	2022-11-15	75020.00	20
3	NIrmal	Accountant	2022-09-20	75020.00	30
4	jannat	Developer	2022-08-05	75020.00	20
5	Drshti	Analyst	2023-02-28	75020.00	30

—----------

Assignment-3

```
use db8;
```

```
create table emp(
ename varchar(10));
truncate emp;
insert into emp values('bhoomi');
insert into emp values('heer');
insert into emp values('deep');
insert into emp values('dev');

select * from emp;
-------------------------------------------------------------------------------
CREATE DEFINER=`root`@`localhost` TRIGGER `emp_BEFORE_INSERT` BEFORE INSERT ON `emp` FOR EACH ROW BEGIN
set new.ename=upper(new.ename);
END
-----------------------------------------------------------------
create database db8;
use db8;
CREATE TABLE stu_backup (
   id INT,
   sname VARCHAR(20),
   address VARCHAR(30),
   contact INT,
   operation_date timestamp DEFAULT now()
);
desc stu_backup;

create table student(
s_id int,
```

```
s_name varchar(20),
s_address varchar(30),
contact_no int);
desc student;

insert into student values (1,'bhoomi','gurukul',12324545);
insert into student values (2,'heer','bhadaj',45324545);
insert into student values (3,'nirali','maninagar',9874545);

update student
set s_address = "memnagar"
where s_name="bhoomi";
select * from student;
select * from stu_backup;
--------------------------------------------------
CREATE DEFINER=`root`@`localhost` TRIGGER `student_BEFORE_UPDATE` BEFORE UPDATE ON `student`
FOR EACH ROW BEGIN
insert into stu_backup(id,sname,address,contact)
values(old.s_id,old.s_name,old.s_address,old.contact_no);
END
-------------------------------------------------------------------------
use  db8;

create table info(
ename varchar(15),
age int
);
insert into info values ('bhoomi',19);
insert into info values ('heer',-18);
```

```
insert into info values ('dev',18);
insert into info values ('deep',-18);

select * from info;
-------------------------------------------------
CREATE DEFINER=`root`@`localhost` TRIGGER `info_BEFORE_INSERT` BEFORE INSERT ON `info` FOR EACH ROW BEGIN
if new.age<0 then
set new.age = "not valid";
end if;
END
-------------------------------------------------------------------------------
use db8;

create table employee(
  empno int,
  empname varchar(20),
  empsalary int);

insert into employee values (101,"bhoomi",20000),
(102,"heer",30000),
(103,"nirali",40000),
(104,"dev",55000),
                    (105,"deep",67000);

select * from employee;
drop table employee;
create table employee_backup(
    empno int,
```

```
    empname varchar(20),
    empsalary int,
    date_of_operation timestamp default now(),
    type_of_operation varchar(10));

select * from employee_backup;



update employee
 set empsalary=empsalary+10000
 where empno=103;

update employee
 set empname="abc"
 where empno=101;

delete from employee
  where empno=105;

delete from employee
  where empname="dev";
--------------------------------------------------------------------------------
CREATE DEFINER=`root`@`localhost` TRIGGER `employee_BEFORE_UPDATE` BEFORE UPDATE ON
`employee` FOR EACH ROW BEGIN
 insert into employee_backup(empno,empname,empsalary,type_of_operation)
 values (old.empno,old.empname,old.empsalary,"Update");
END
--------------------------------------------------------------------------------
```

```
CREATE DEFINER=`root`@`localhost` TRIGGER `employee_BEFORE_DELETE` BEFORE DELETE ON
`employee` FOR EACH ROW BEGIN

 insert into employee_backup(empno,empname,empsalary,type_of_operation)

 values (old.empno,old.empname,old.empsalary,"Delete");

END

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

use db8;

create table emp1(

id int,

ename varchar(15));

insert into emp1 values (101,"bhoomi");

insert into emp1 values (102,"heer");

insert into emp1 values (103,"nirali");

insert into emp1 values (104,"dev");

insert into emp1 values (105,"deep");

                    select * from emp1;

update emp1

set ename = "abc"

where id = 101;


delete from emp1

where id = 104;

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

CREATE DEFINER=`root`@`localhost` TRIGGER `emp1_BEFORE_UPDATE` BEFORE UPDATE ON `emp1` FOR
EACH ROW BEGIN

SIGNAL SQLSTATE "45000" set MESSAGE_TEXT="updating";

END

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

CREATE DEFINER=`root`@`localhost` TRIGGER `emp1_BEFORE_DELETE` BEFORE DELETE ON `emp1` FOR
EACH ROW BEGIN
```

```
SIGNAL SQLSTATE "45000" set MESSAGE_TEXT = "deleting";
END
-------------------------------------------------------------------------------
use db8;
CREATE TABLE product_master (
   pid INT PRIMARY KEY,
   pname VARCHAR(255),
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

insert into product_master(pid,pname)
values(1,"abc"),
(2,"def"),
(3,"ghi"),
(4,"jkl");
drop table product_master;
insert into product_master
values
(5,"mno",'2023-11-11');

select *from product_master;

delete from product_master
where pid=5;
delete from product_master
where pid=2;
-----------------------------------------------------------------
CREATE DEFINER=`root`@`localhost` TRIGGER `product_master_BEFORE_DELETE` BEFORE DELETE ON
`product_master` FOR EACH ROW BEGIN
```

```
declare week_day int;


set week_day=dayofweek(old.created_at);
if week_day = 1 or week_day= 7
then
signal sqlstate "45000"
set message_text='deleting from weekends is not allowed';
end if;
END
--------------------------------------------------------------------------------
create database db9;
use db9;
call pro1(40,20);
call pro2(1,25,35,45);
call pro3();
call pro5(10,30,1);
-------------------------------------------------
CREATE DEFINER=`root`@`localhost` PROCEDURE `pro1`(in num INT,in num1 int)
BEGIN
DECLARE total int;
declare sub int;
declare mul int;
declare division int;
set total = num + num1;
set sub =num - num1;
set mul = num * num1;
set division = num / num1;
SELECT total,sub,mul,division;
END
```

```
-----------------------------------------------------------------
CREATE DEFINER=`root`@`localhost` PROCEDURE `pro2`(in roll_no int,in s1 int,in s2 int,in s3 int)
BEGIN
declare total int;
declare percent int;
declare result varchar(10);
 set total = s1 + s2 + s3;
 set percent = (total/150) * 100;
 if percent < 50 then
  set result = "fail";
  else
  set result = "pass";
  end if;
 select roll_no,total,percent,result;
END
-----------------------------------------------------------------
CREATE DEFINER=`root`@`localhost` PROCEDURE `pro3`()
BEGIN
declare i int DEFAULT 0;
loop1 : loop
 set i = i+1;
if i >=20 then
   leave loop1;
   ELSE if mod(i,2)=0 then
   iterate loop1;
   end if;
   select i;
   end if;
   end loop;
```

```
END
-------------------------------------------------------------------------
CREATE DEFINER=`root`@`localhost` PROCEDURE `pro5`(in n1 int,in n2 int,in n3 int)
BEGIN
DECLARE max_num INT;
   DECLARE min_num INT;

   IF n1 >= n2 AND n1 >= n3 THEN
      SET max_num = n1;
   ELSEIF n2 >= n1 AND n2 >= n3 THEN
      SET max_num = n2;
   ELSE
      SET max_num = n3;
   END IF;

   IF n1 <= n2 AND n1 <= n3 THEN
      SET min_num = n1;
   ELSEIF n2 <= n1 AND n2 <= n3 THEN
      SET min_num = n2;
   ELSE
      SET min_num = n3;
   END if;
   select min_num,max_num;
END

—-------------------------------------------------------------------------

create database fun;
use fun;
```

```
 create table emp(
 empno int,
 deptno int,
 salary int);

 insert into emp values(1,11,56000);
 insert into emp values(2,10,55000);
 insert into emp values(3,11,79000);

select new1('bhoomi') as length_of_string;
select prime(10);
 select function3(11);
 select function4(2);
 set @bhoomi = 8;
 call function5(@bhoomi);
 select @bhoomi;
--------------------------------------------------------------------------------
CREATE DEFINER=`root`@`localhost` FUNCTION `new1`(input_name varchar(20)) RETURNS int
    DETERMINISTIC
BEGIN
declare name_len int;
set name_len=length(input_name);
return name_len;
RETURN 1;
END
--------------------------------------------------------------------------------------------
CREATE DEFINER=`root`@`localhost` FUNCTION `prime`(num INT) RETURNS tinyint(1)
    DETERMINISTIC
BEGIN
```

```
DECLARE i INT DEFAULT 2;
   DECLARE prime BOOLEAN DEFAULT TRUE;


   IF num <= 1 THEN
      SET prime = FALSE;
   ELSE
      WHILE i <= SQRT(num) DO
         IF num % i = 0 THEN
            SET prime = FALSE;
         END IF;
         SET i = i + 1;
      END WHILE;
   END IF;
RETURN prime;
END
 ----------------------------------------------------------------------------------
CREATE DEFINER=`root`@`localhost` FUNCTION `function3`(dno int ) RETURNS int
   DETERMINISTIC
BEGIN
declare max_sal int;
select max(salary) into max_sal from emp
where deptno = dno;
if max_sal is null then
signal sqlstate "45000" set  message_text = "dept not found";
end if;
return max_sal;
RETURN 1;
END
-------------------------------------------------------------------------
```

```
CREATE DEFINER=`root`@`localhost` FUNCTION `function4`(emno int) RETURNS varchar(20) CHARSET utf8mb4
    DETERMINISTIC
BEGIN
declare ans varchar(20) default "does not";
declare emp_count int;
select count(*) into emp_count from emp
where empno=emno;
if emp_count > 0 then
set ans = "exists";
end if;
return ans;
RETURN 1;
END
```

```
create database db10;
use db10;

create table emp(
id int,
ename varchar(20));

insert into emp values (101,"bhoomi");
insert into emp values (102,"heer");
insert into emp values (103,"deep");
insert into emp values (104,"dev");
insert into emp values (105,"nirali");
select * from emp;
call pro1(110);
```

```
 ------------------------------------------------
CREATE DEFINER=`root`@`localhost` PROCEDURE `pro1`(in empno int)
BEGIN
declare eid int;
select id into eid from emp where empno = id;
select ename from emp where eid = id;
if eid is null then
select "employee number does not exist";
end if;
END
-------------------------------------------------------------------------------
use db10;
create table student(
stu_id int,
stu_name varchar(20),
address varchar(15),
m1 int,
m2 int,
m3 int
);

insert into student values(1,"bhoomi","gurukul",34,43,33);
insert into student values(2,"heer","bhadaj",21,41,23);
insert into student values(3,"nirali","maninagar",22,33,44);
call pro2(2);
call pro3("bhoomi");
call pro3("BHOOMI");
call pro3("BhooMI");
-----------------------------------------------------------------------------------------
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pro2`(in id int)
BEGIN
select stu_name,address,m1,m2,m3 from student
where id = stu_id;
END
--------------------------------------------------------------------------------
CREATE DEFINER=`root`@`localhost` PROCEDURE `pro3`(in str varchar(15))
BEGIN
declare ans varchar(25);
if binary str = binary upper(str) then
set ans = "uppercase";
elseif binary str = binary lower(str) then
set ans = "lowercase";
else
set ans = "mixedcase";
end if;
select ans;
END
------------------------------------------------------------------------
use db10;

create table stu(
id int,
sname varchar(20),
percent int);

insert into stu values (1,"abc",88);
insert into stu values (2,"def",77);
insert into stu values (3,"xyz",66);
```

```
insert into stu values (4,"lmn",100);


select * from stu;
call pro4();
-------------------------------------------------------------------------------
CREATE DEFINER=`root`@`localhost` PROCEDURE `pro4`()
BEGIN
declare max_per int;
select max(percent) into max_per from stu;
select id,sname,percent from stu
where max_per = percent;
END
-----------------------------------------------------------------------
use db10;


create table employee(
id int,
ename varchar(20),
DOJ date);


insert into employee values (101,"bhoomi","2020-08-08");
insert into employee values (102,"heer","2021-10-09");
insert into employee values (103,"nirali","2022-11-10");
insert into employee values (104,"devanshi","2000-11-14");
drop table employee;
call pro5(104);
-----------------------------------------------------------------
CREATE DEFINER=`root`@`localhost` PROCEDURE `pro5`(in eid int)
BEGIN
```

```
DECLARE joining_date DATE;
DECLARE e_year int;
DECLARE emp_name varchar(20);
  SELECT ename, DOJ
   INTO emp_name, joining_date
   FROM employee
   WHERE eid = id;
   SET e_year = YEAR(CURDATE()) - YEAR(joining_date);
   select e_year,emp_name ;
END
-----------------------------------------------------------------------
use db10;

create table stu_6(
id int,
sname varchar(20),
m1 int,
m2 int,
m3 int);

insert into stu_6 values (101,"abc",44,45,46);
insert into stu_6 values (102,"def",31,32,33);
insert into stu_6 values (107,"lmn",24,25,26);
insert into stu_6 values (104,"xyz",14,15,16);
call pro6(107);
-----------------------------------------------------------------------------
CREATE DEFINER=`root`@`localhost` PROCEDURE `pro6`(in s_id int)
BEGIN
declare s1 int;
```

```
declare s2 int;
declare s3 int;
declare total int;
declare percent int;
declare result varchar(30);
select m1,m2,m3 into s1,s2,s3 from stu_6
where id = s_id;
set total = s1 + s2 + s3;
set percent = (total/150) * 100;
if percent < 40 then
set result = "fail";
ELSEIF percent >=40 and percent <= 55 then
set result = "third class";
ELSEIF percent >=56 and percent <=70 then
set result = "second class";
elseif percent >=71 and percent <= 90 then
set result = "first class";
ELSE
set result = "distinction";
end if;
select s_id,percent,result;
END
```

Assignment-4

Book Exercises

Q2. Write a program to create trigger signal to restrict entering negative value in balance.

Query:-

```
create table Account(
  a_id int primary key,
  account_holder_name varchar(50),
  balance float
);


insert into account values(
  101,
  'Ram Shah',
  56000
);

insert into account values(
  105,
  'Mohan Pandit',
100
  );

update account set balance = -100 where a_id = 101;
```

Output :-

23:12:21 update account set balance = -100 where a_id = 101 Error Code: 1644. Account balance cannot be less than 0 0.000 sec

Trigger:-

```
CREATE DEFINER=`root`@`localhost` TRIGGER `account_BEFORE_UPDATE` BEFORE UPDATE ON `account` FOR EACH ROW BEGIN
	if (new.balance < 0) then
		signal sqlstate '80000'
			set message_text = 'Account balance cannot be less than 0';
	end if;
END
```

Q3. Write a program to perform data validation using select statement.

Query :-

```
	update account set balance = -678 where a_id = 105;
```

Output:-

23:16:53 update account set balance = -678 where a_id = 105 Error Code: 1054. Unknown column 'Account amount invalid !! less than 0' in 'field list' 0.000 sec

Trigger:-

```
CREATE DEFINER=`root`@`localhost` TRIGGER `account_BEFORE_UPDATE` BEFORE UPDATE ON `account` FOR EACH ROW BEGIN
	DECLARE dummy INT;
	IF NEW.balance<0 THEN
		SELECT `Account amount invalid !! less than 0` INTO dummy
		FROM account
		WHERE a_id=NEW.a_id;
	END IF;
END
```

Q4. Write a example to create sales table which provides free shipping on orders above 500.

Query :-

```
create table sales(
  customer_id int primary key,
  product_id int,
sale_date date,
quantity int,
sale_value float,
department_id int,
sales_rep_id varchar(10),
free_shipping char(1),
discount float
  );

insert into
sales(customer_id,product_id,sale_date,quantity,sale_value,department_id,sales_rep_id) values
  (1,123,'2023-03-21',100, 600, 456, 'SR345'),
  (5,152,'2023-07-12',450, 1600, 256, 'SR245'),
  (7,166,'2021-02-22',600, 456, 777, 'SR385'),
  (9,177,'2020-11-09',10, 100, 456, 'SR349'),
  (10,189,'2019-07-02',5, 789, 856, 'SR145'),
  (28,198,'2023-08-07',2, 99, 999, 'SR545'),
```

```
  (35,200,'2015-01-05',10, 1000, 231, 'SR745'
);
```

Output:-

```
23:40:55 insert into sales(customer_id,product_id,sale_date,quantity,sale_value,department_id,sales_rep_id) values (1,123,'2023-03-21',100, 600, 456, 'SR345'), (5,152,'2023-07-12',450, 1600, 256, 'SR245'), (7,166,'2021-02-22',600, 456, 777, 'SR385'), (9,177,'2020-11-09',10, 100, 456, 'SR349'), (10,189,'2019-07-02',5, 789, 856, 'SR145'), (28,198,'2023-08-07',2, 99, 999, 'SR545'), (35,200,'2015-01-05',10, 1000, 231, 'SR745')
7 row(s) affected Records: 7  Duplicates: 0  Warnings: 0   0.016 sec
```

```
SELECT sale_value,free_shipping,discount FROM sales;
```

Output:-

600	Y	0
1600	Y	240
456	N	0
100	N	0
789	Y	0
99	N	0
1000	Y	0

Trigger:-

```
CREATE DEFINER=`root`@`localhost` TRIGGER `sales_BEFORE_INSERT` BEFORE INSERT ON `sales` FOR EACH ROW BEGIN
	if (new.sale_value > 500 ) then
		set new.free_shipping='Y';
	else
		set new.free_shipping='N';
	end if;
    if (new.sale_value > 1000) then
		set new.discount = new.sale_value*.15;
	else
		set new.discount=0;
	end if;

END
```

Transaction

Q5. Create a procedure to commence a transaction using auto commit.

Query :-

```
set@from_account=101;
set@to_account=105;
set@tfer=2400;
call tfer_funds(@from_account, @to_account, @tfer);
```

Output:-

```
00:07:29    call tfer_funds(@from_account, @to_account, @tfer)    0 row(s) affected
0.000 sec
```

Procedure :-

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `tfer_funds`(in from_account int, in to_account
int,in tfer_amount numeric(10,2))
BEGIN
    SET autocommit=0;
    UPDATE account
    SET balance=balance-tfer_amount
    WHERE a_id=from_account;
```

```
    UPDATE account
    SET balance=balance+tfer_amount
    WHERE a_id=to_account;
    COMMIT;


END
```

Q6. Create a procedure to commence a transaction using start transaction.

Query :-

```
    set@from_account=105;
    set@to_account=101;
    set@tfer=440;

    call tfer_funds(@from_account, @to_account, @tfer);
```

Output:-

00:12:20 call tfer_funds(@from_account, @to_account, @tfer) 0 row(s) affected 0.015 sec

Procedure :-

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `tfer_funds`(in from_account int, in to_account int,in tfer_amount numeric(10,2))
BEGIN

    START TRANSACTION;
    UPDATE account
    SET balance=balance+tfer_amount
    WHERE a_id=from_account;
    UPDATE account
    SET balance=balance+tfer_amount
    WHERE a_id=to_account;
    COMMIT;

END
```

Q7. Create a procedure which displays use of savpoint with transaction.

Query :-

```
call Savepoints();
```

Output:-

00:24:50	call Savepoints()	0 row(s) affected	0.016 sec

Procedure:-

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `Savepoints`()
BEGIN
    BEGIN
        INSERT INTO sales (customer_id, product_id, quantity, discount) VALUES (24, 112, 167, 5.00);

        SAVEPOINT my_savepoint;

        UPDATE sales SET quantity = quantity + 2 WHERE customer_id = 1;
```

```
            ROLLBACK TO my_savepoint;

        END;


    COMMIT;

END
```