1. creating database

create database lec46;

1. using database

use lec46;

1. creating table

create table employee(

id int auto\_increment primary key,

name varchar(20),

salary float

);

1. checking table structure

desc employee;

1. inserting some records inside table

insert into employee(name,salary) values

("raj",25000),

("rani",30000);

1. Getting all employees

select \* from employee;

1. creating SP to get all employees from table
   1. step 1 : create SP

CREATE PROCEDURE `showEmployees`()

BEGIN

select \* from employee;

END

* 1. step 2 : call SP

call showEmployees();

1. getting list of all available SP

show procedure status where db="lec46";

1. creating variable inside SP

SQL QUERY : select count(\*) from employee;

CREATE PROCEDURE `countEmployees`()

BEGIN

declare empCount int default 0;

select count(\*) into empCount from employee;

select empCount;

END

call countEmployees();

1. creating SP with input parameter

SQL QUERY : select \* from employee where name="raj";

CREATE PROCEDURE `getEmployeeByName`(in empName varchar(20))

BEGIN

select \* from employee where name=empName;

END

call getEmployeeByName("rani");

1. creating SP with out parameter

CREATE PROCEDURE `countEmp`(out emp\_count int)

BEGIN

select count(\*) into emp\_count from employee;

END

call countEmp(@emp\_count);

select @emp\_count;

1. creating SP with in and out

CREATE PROCEDURE `inOutTogather`(in startWith varchar(5), out empCount int)

BEGIN

select count(\*) into empCount from employee where name like startWith;

END

call inOutTogather('r%', @empCount);

select @empCount;

1. creating SP with conditional statement

CREATE PROCEDURE `greaterNumber`()

BEGIN

DECLARE x int; DECLARE y int; DECLARE z int;

DECLARE result varchar(20);

set x = 100, y=20, z=30;

if x>y and x>z then

set result="x is greater";

elseif y>x and y>z then

set result="y is greater";

else

set result="z is greater";

end if;

select result;

END

call greaterNumber();

CREATE PROCEDURE `skillIndiaEligible`(out eligibleForSkillIndia varchar(20))

BEGIN

declare empcount int;

select count(\*) into empcount from employee;

if empcount>3 then

set eligibleForSkillIndia="YES";

else

set eligibleForSkillIndia="NO";

end if;

END

call skillIndiaEligible(@eligibleForSkillIndia);

select @eligibleForSkillIndia;

insert into employee(name,salary) values("amit",18000);

call skillIndiaEligible(@eligibleForSkillIndia);

select @eligibleForSkillIndia;

insert into employee(name,salary) values("nita",28000);

call skillIndiaEligible(@eligibleForSkillIndia);

select @eligibleForSkillIndia;

1. creating while loop in SP

CREATE PROCEDURE `whileLoop`()

BEGIN

declare i int;

declare result varchar(20) default '';

set i = 1;

while i<=5 do

set result = concat(result,i,",");

set i = i+1;

end while;

select result;

END

call whileLoop();

1. creating function

CREATE DEFINER=`root`@`localhost` FUNCTION `function\_1\_to\_5\_sum`() RETURNS int

READS SQL DATA

DETERMINISTIC

BEGIN

declare i int;

declare sum int default 0;

set i = 1;

while i<=5 do

set sum = sum+i;

set i = i+1;

end while;

RETURN sum;

END

select function\_1\_to\_5\_sum();

1. creating cursor
   1. steps to create cursor

-- 1. DECLARE cursor\_name CURSOR FOR select\_query

-- 2. DECLARE CONTINUE HANDLER FOR NOT FOUND termination\_condition;

-- 3. OPEN cursor\_name;

-- 4. FETCH cursor\_name INTO variable\_list;

-- 5. CLOSE cursor\_name

* 1. create SP

CREATE PROCEDURE `new\_procedure`()

BEGIN

declare is\_finished int default 0;

declare name\_list varchar(500) default '';

declare user\_name varchar(50) default '';

DECLARE employee\_cursor CURSOR FOR select name from employee;

DECLARE CONTINUE HANDLER FOR NOT FOUND set is\_finished=1;

OPEN employee\_cursor;

get\_user\_names\_loop : LOOP

FETCH employee\_cursor INTO user\_name;

IF is\_finished=1

THEN LEAVE get\_user\_names\_loop;

END IF;

SET name\_list = concat(name\_list,",",user\_name);

END LOOP get\_user\_names\_loop;

CLOSE employee\_cursor;

select name\_list;

END

* 1. call SP

use lec46;

call new\_procedure();

1. Triggers

create database triggerss;

use triggerss;

create table employee(name varchar(20),age int);

insert into employee (name,age) values("raj",25),("rani",28);

select \* from employee;

select avg(age) from employee;

create table avg\_emp\_age(avg\_age double);

insert into avg\_emp\_age select avg(age) from employee;

select \* from avg\_emp\_age;

-- create trigger that going to execute as soon as we insert record in

-- employee table

CREATE TRIGGER avg\_emp\_age\_after\_insert

AFTER INSERT ON employee

FOR EACH ROW

update avg\_emp\_age set avg\_age = (select avg(age) from employee);

select \* from employee;

select \* from avg\_emp\_age;

insert into employee (name,age) values("aniket",35);

select \* from employee;

select \* from avg\_emp\_age;

insert into employee (name,age) values("puja",29);

select \* from employee;

select \* from avg\_emp\_age;

-- create trigger that going to execute before adding record in

-- employee table

DELIMITER $$

CREATE TRIGGER age\_checking\_before\_insert

BEFORE INSERT ON employee

FOR EACH ROW

if new.age<18 then set new.age=0;

end if $$

DELIMITER ;

insert into employee (name,age) values("nita",12);

select \* from employee;

-- create trigger that going to execute before updating record in

-- employee table

DELIMITER $$

CREATE TRIGGER age\_checking\_before\_update

BEFORE UPDATE ON employee

FOR EACH ROW

if new.age<18 then set new.age=0;

end if $$

DELIMITER ;

insert into employee (name,age) values("nita",12);

select \* from employee;

update employee set age=30 where name="raj";

select \* from employee;

update employee set age=10 where name="raj";

select \* from employee;

-- create trigger that going to execute as soon as we update record in

-- employee table

CREATE TRIGGER avg\_emp\_age\_after\_update

AFTER UPDATE ON employee

FOR EACH ROW

update avg\_emp\_age set avg\_age = (select avg(age) from employee);

select \* from employee;

select \* from avg\_emp\_age;

update employee set age=38 where name="raj";

select \* from employee;

select \* from avg\_emp\_age;

-- create trigger that going to execute before deleting record in

-- employee table

-- we are going to archive that record into employee\_archive table.

-- lets create employee\_archive table

create table employee\_archive(name varchar(20),age int);

CREATE TRIGGER archive\_employee\_before\_deleting

BEFORE DELETE ON employee

FOR EACH ROW

insert into employee\_archive (name,age) values (old.name,old.age);

select \* from employee;

select \* from employee\_archive;

delete from employee where name="puja";

select \* from employee;

select \* from employee\_archive;

-- create trigger that going to execute as soon as we delete record in

-- employee table

CREATE TRIGGER avg\_emp\_age\_after\_delete

AFTER DELETE ON employee

FOR EACH ROW

update avg\_emp\_age set avg\_age = (select avg(age) from employee);

select \* from employee;

select \* from avg\_emp\_age;

delete from employee where name="rani";

select \* from employee;

select \* from avg\_emp\_age;

1. Mongo db

|  |  |
| --- | --- |
| MYSQL | MONGODB |
| structured | unstructred |
| row | document |
| commands | functions |
|  |  |