Stored Functions and Triggers

Stored Functions

A stored function is a special kind stored program that returns a single value. They are reusable among SQL statements or stored programs.

Different from a stored procedure, you can use a stored function in SQL statements wherever an expression is used.

- Syntax for creating Function

```
CREATE FUNCTION function_name(
    param1 datatype, param2 datatype,…

      )
RETURNS datatype;
BEGIN
 -- statements

 RETURN expression;

END $$
```

By default, all parameters are the IN parameters. **You cannot specify IN , OUT or INOUT modifiers** to parameter like stored procedure in Function

write the code in the body of the stored function in the BEGIN END block

- Call the function
  Function can be called by select statement or within a stored procedure also.

e.g Create function which work like pow function. Pass 2 arguments no1 and no2 and returns the multiplication of no1 and no2

- Create function

Step -1 change delimiter.

Mysql> delimiter $$

Step -2 define the function

```
  create function powfunct(b int,p int)
  returns int(4)
        DETERMINISTIC
    begin
            declare temp int;
            set temp = b*p;
          return (temp);
      end $$
```

Step 4 : change delimiter

Mysql> delimiter ;

Step 3 : call the function

Mysql> select powfunct(3,4) ;

```
 powfunct(3,4)
        12
```

e.g 2 create function which calculate the bonus of the employee based on salary

step 1 : mysql> delimiter $$

step 2 : mysql> create function calbonus(sal int)

```
 returns float(10,2)
DETERMINISTIC
begin

  declare bonus float(10,2);

  if sal > 15000 then
    set bonus = sal * 0.10 ;
  else
    set bonus = sal * 0.15;
  end if;

  return (bonus);
  end $$
```

step 3 : mysql> delimiter ;

step 4 **mysql> select name, salary , calbonus(salary) from employee;**

Trigger

In MySQL, a trigger is a stored program invoked automatically in response to an event such as insert, update, or delete that occurs in the associated table.
For example, you can define a trigger that is invoked automatically before a new row is inserted into a table.

MySQL supports triggers that are invoked in response to the INSERT, UPDATE or DELETE event.

MySQL supports only row-level triggers. That is trigger activated on each row level action.

Create Trigger

```
CREATE TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE| DELETE }
ON table_name FOR EACH ROW
Begin
  Statements
End
```

First, specify the name of the trigger that you want to create after the CREATE TRIGGER keywords

Second specify the **trigger action time** which can be either BEFORE or AFTER which indicates that the trigger is invoked before or after each row is modified/insert/delete.

Third specify the operation that **activates the trigger**, which can be INSERT, UPDATE, or DELETE

Fourth specify the **name of the table** to which the trigger belongs after the ON keyword.

Fifth, specify **the statement to execute when the trigger activates**.

o distinguish between the value of the columns BEFORE and AFTER the DML has fired, you use the NEW and OLD modifiers.

e.g if you update the column description, in the trigger body, you can access the value of the description before the update OLD.description and the new value NEW.description.

Trigger Event	OLD	NEW
INSERT	No	Yes
UPDATE	Yes	Yes
DELETE	Yes	No

1. create trigger which activate on before inserting records on post table and it inserts the record into category table.

Step 1 : mysql> delimiter $$

Step 2 : mysql>

```
create trigger firsttrg
  before insert
on post for each row
begin

  insert into category values(NEW.id,NULL);

end $$
```

step 4 : delimiter ;

step 5 : insert values on post table

```
insert into post values('dance','Various form of classical dances','2021-02-03',6);
```

it will automaticall add the cat_id 6 in category table. By the trigger.

2. Create trigger for after inserting record on temp table if inserted no is odd it will insert into odd table other wise in even table.

```
Create table temp(no int(2));
Create table odd(no int(2));
Create table even(no int(2));
```

Step 1 delimiter $$

Step 2 create trigger

```
create trigger trig2
after insert
on temp for each row
begin
  if mod(new.no,2)=0 then
```

```
    insert into even values(new.no);
  else
    insert into odd values(new.no);
  end if;
end $$
```

step 3 : delimiter ;

step 4 insert values in temp table so trigger will be activated
mysql> insert into temp values(25) ;   // trigger will be activated after record inserted into temp and it will add record in odd table.

3. Trigger for after updation .

Create a trigger after updation on book price in book table, if updated price less than old price then stores old record in book2 table if the updated price is greater than the old price then stores old book record in book1.

Step 1 : delimiter $$

Step 2 :

```
create trigger updatetrigger
after update
on book for each row
begin
  if old.price < new.price then
    insert into book1 values(old.bookid,old.bookname,old.price);
  else
   insert into book2 values(old.bookid,old.bookname,old.price);
end if;
end $$
```

Step 3 : delimiter ;

Step 4 : Fire update query on book table
Mysql> update book set price = 700 where bookid='B5';

It will fire trigger after updating row in book table and trigger can add old price record in book1.

Delete the Trigger

Mysql> drop trigger triggername

List out triggers form database

Syntax : show triggers from 'databasename';
e.g show triggers from 'testdb';

Exercise

1. Create a function func1 which takes the number as parameter and return the value "odd" or "even" .
2. Create a function func2()which take the age attribute of employee table, if age is <=25 status will be "young" , if age between 26 to 32 status "middle" if age > 32 status will be "old". Function returns the status. Write a select query which display the name ,age and status of every employee.

3. Create a function fun3() which takes a orderno as input and returns the name(description) of the product . hint(use product_master and sales_order_detail)
Use necessary select query to display function output.

4. Create Trigger Trig1 which execute after update on product_master. While updating value of any sellprice it should also update product_rate filed value of same prodcutno in sales_order_detail table.

5. Create Trigger Trig2 Which execute before delete on employee table. It copy the to deleted record to the new table deleted_emp.
(create a deleted_emp table first which contains same attribute and datatype as employee)

6. Create Trigger Trig3 Which execute before insert new record in employee table if new records joining date is after 2021-01-01- then insert new record in deleted_emp table also.