**Week 12**

**1.Create and execute stored procedures 2. Create and execute functions**

**Write a procedure to find sum of two numbers:**

CREATE PROCEDURE sum(a int , b int)

BEGIN

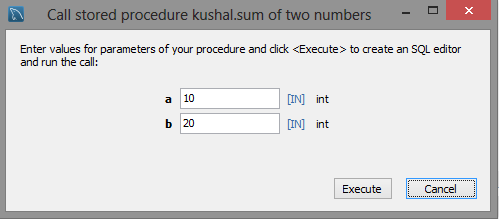
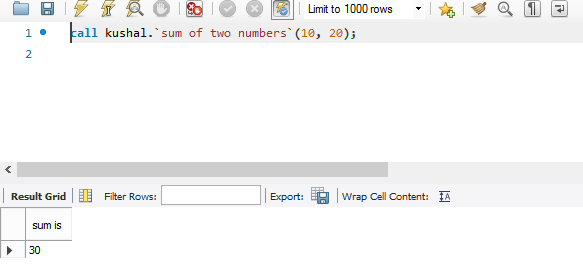
declare ans int;

set ans=a+b;

select ans as "sum is";

END

**Output:**



**Write a procedure to find whether the student is pass or fail**

CREATE PROCEDURE result(marks int)

BEGIN

if marks>=40 then

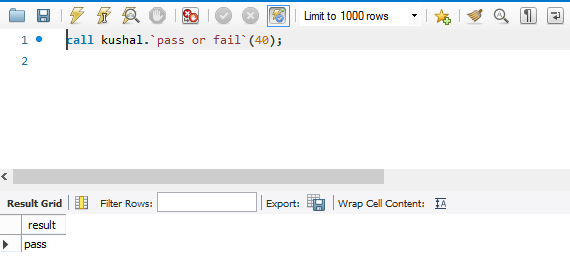
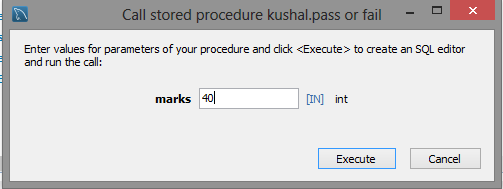
select "pass" as "result";

else

select "fail" as "result";

end if;

END

**Output:**

**Write a procedure to find whether the student has secured distinction, first class , second class or fail**

CREATE PROCEDURE DISPLAY(marks int)

BEGIN

if marks>85 then

select marks, 'distiction' as "result";

elseif marks between 85 and 60 then

select marks, 'first class' as "result";

elseif marks between 60 and 41 then

select marks, 'second class' as "result";

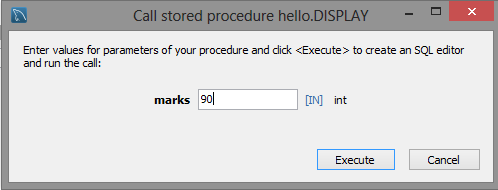
else

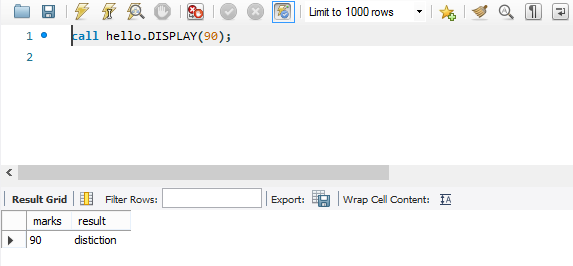
select marks, 'fail' as "result";

end if;

END

**Output:**

****

****

**Write a function to find sum of three numbers**

CREATE FUNCTION sum (a int, b int, c int) RETURNS int

DETERMINISTIC

BEGIN

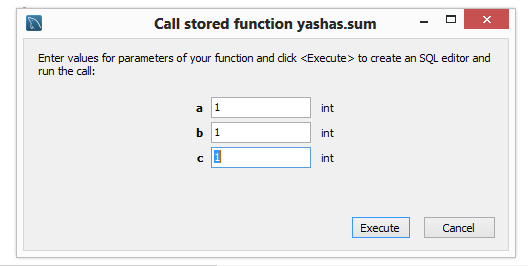
declare ans int;

set ans=a+b+c;

return ans;

END

**OUTPUT**

****

**Write a function to find greatest of two numbers.**

CREATE FUNCTION great (a int , b int) RETURNS int

DETERMINISTIC

BEGIN

if a>b then

return a;

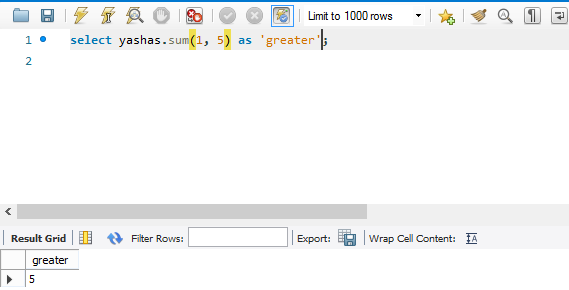
else

return b;

end if;

END

**Output:**



**Week 13**

# 1.Create and execute transactions 2.Call previously created store procedure or function in transaction

**TCL - Transaction Control Language**

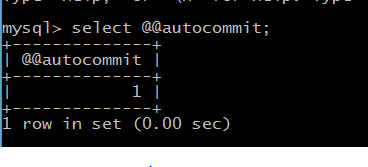
A transaction is a logical unit of work that contains one or more SQL statements. Transactions are atomic units of work that can be committed or rolled back. When a transaction makes multiple changes to the database, either all the changes succeed when the transaction is committed, or all the changes are undone when the transaction is rolled back.

**Commands to control transactions.**

* COMMIT − to save the changes caused by a transaction to the database.
* ROLLBACK − to roll back the changes. i.e undo transactions that have not already been saved to the database. This command can only be used to undo transactions since the last COMMIT or ROLLBACK command was executed.

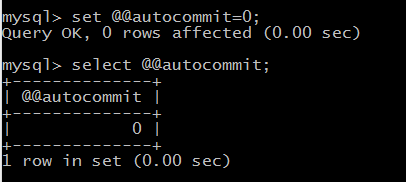
**To check value of autocommit varaiable;**

**mysql>** select @@autocommit;

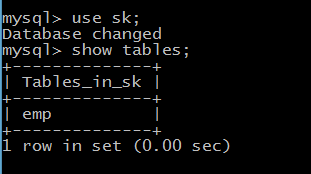


**Set autocommit variable to 0.**

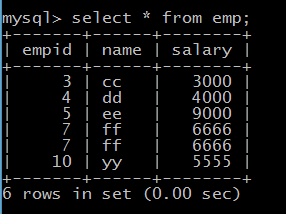
**mysql>** set @@autocommit=0;



mysql> use sk;



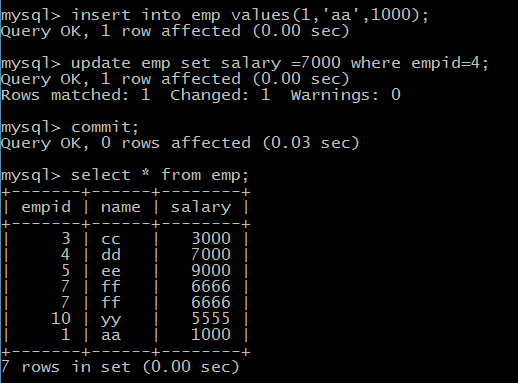
mysql> select \* from emp;



mysql> insert into emp values(1,'aa',1000);

mysql> update emp set salary =7000 where empid=4;

mysql> commit;

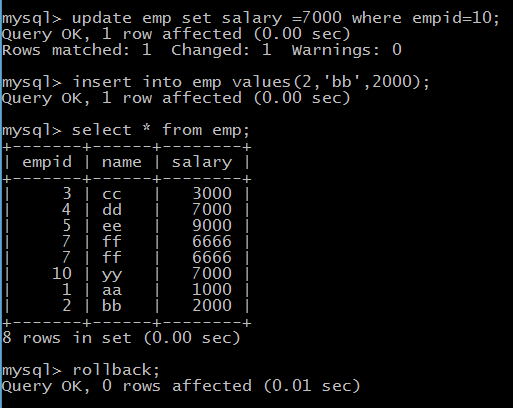


mysql> update emp set salary =7000 where empid=10;

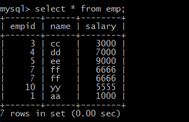
mysql> insert into emp values(2,'bb',2000);

mysql> select \* from emp;

mysql> rollback;



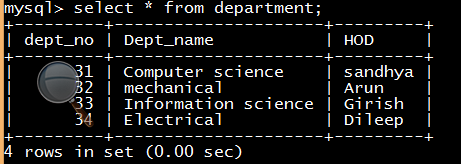
mysql> select \* from emp;

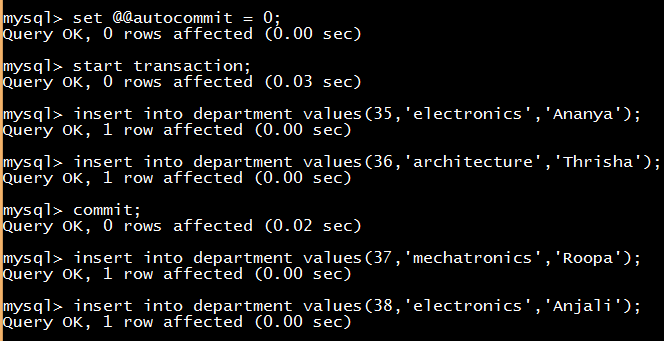


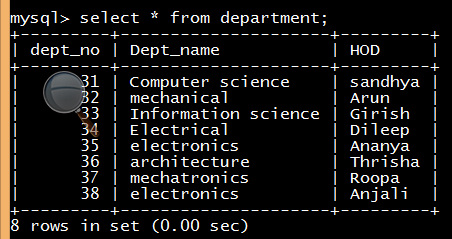
**Example to show savepoint**

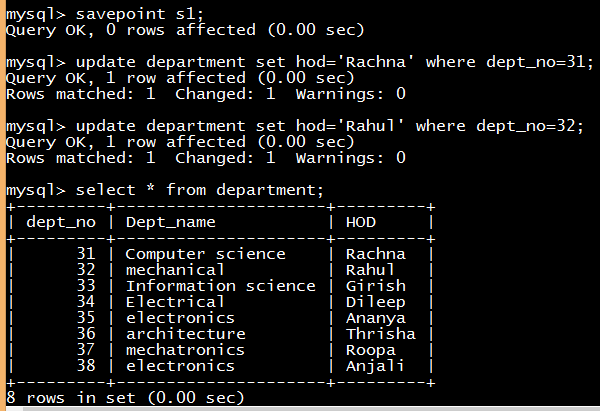
**SAVEPOINT**

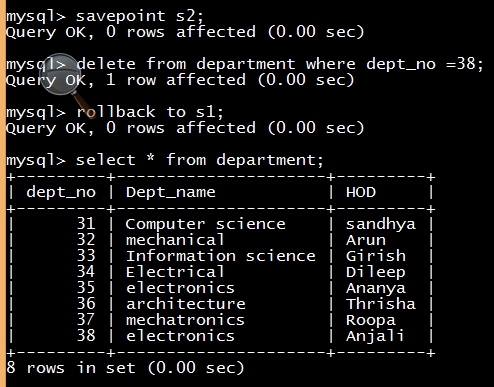
The SAVEPOINT in MySQL is used for dividing (or) breaking a transaction into multiple units so that the user has a chance of roll backing the transaction up to a specified point. That means using Save Point we can roll back a part of a transaction instead of the entire transaction.











Not changed (roll back)

**To call an already existing procedure from a transaction**

CREATE PROCEDURE sum(a int , b int)

BEGIN

declare ans int;

set ans=a+b;

select ans as "sum is";

END

**Output:**

