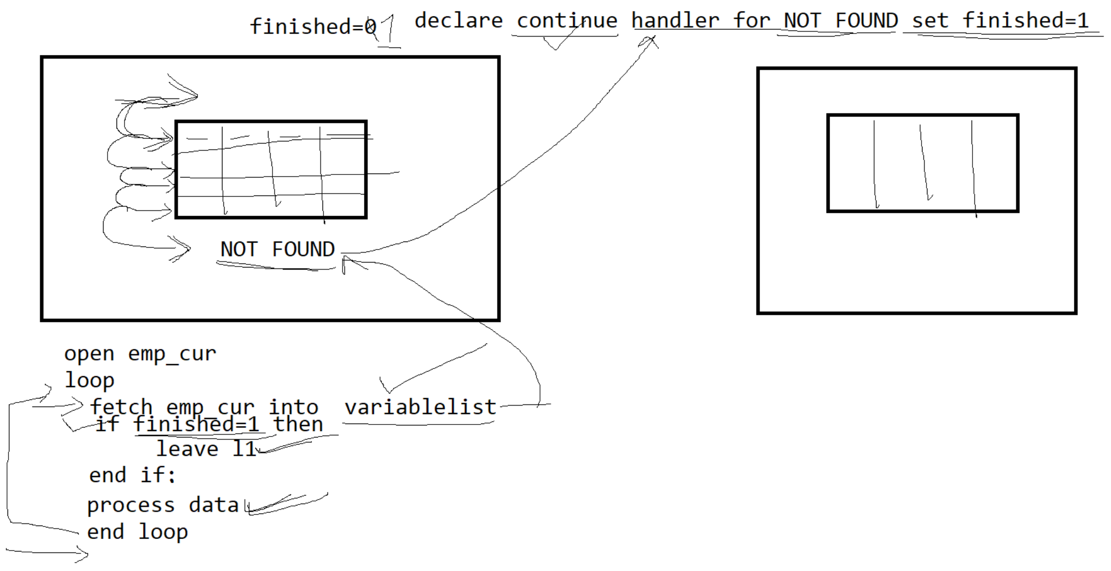


```
open emp_cur
loop
 fetch emp_cur into  variablelist
  if finished=1 then
      leave l1
 end if;
 process data
 end loop
```

Step by step

1. Set the variable finished to 0;
   Declare finished int default 0;
   Declare variables for cursor
2. Declare cursor

    Declare emp_cur CURSOR for select * from emp;

3. Declare continue handler for NOT FOUND
   Declare continue handler for NOT FOUND set finished=1;
4. Open cursor
   Open emp_cur;
5. Fetch the cursor one row at a time
   Fetch emp_cur into vempvo,venm,vjob,vhiredate,vmgr,vsal,vcomm,vdeptno
6. Check value of finished

```
If  finished=1 then

    Leave l1;

End if;
```

7. Process data
8. Repeat steps 5 to 7 till we do not leave the loop
9. Once come out of the loop then close the cursor
   Close emp_cur

Example

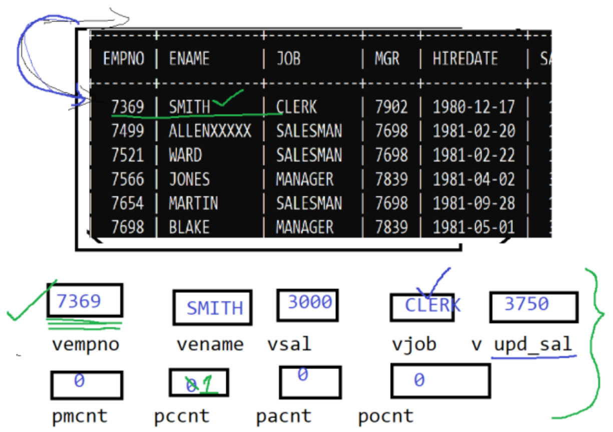
1. Write a procedure to display employee details row by row using cursor

```
delimiter //
create procedure display_emp_cur()
begin
  declare finished int default 0;
  declare vempno,vmgr,vdeptno int;
```

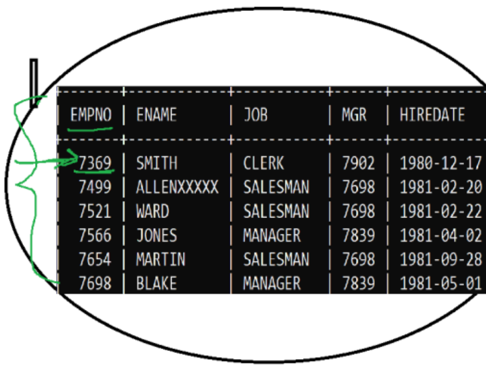
```
    declare vename,vjob varchar(20);
    declare vhiredate date;
    declare vsal,vcomm double(9,2);
    declare emp_cur cursor for select * from emp;
    declare continue handler for NOT FOUND set finished=1;

    open emp_cur;
    l1:loop
      fetch emp_cur into
  vempno,vename,vjob,vmgr,vhiredate,vsal,vcomm,vdeptno;
      if finished=1 then
        leave l1;
        end if;
      select vempno,vename,vjob,vsal,vcomm;
     end loop;
     close emp_cur;
  end//
  delimiter ;
```

2. Update sal of employee, also give cnt of each type as output.
   If manager, then increase it by 10%
   If analyst, then increase by 20%
   If CLERK, the increase by 25%
   Otherwise increase by 8%



```
+-------+------------+----------+------+------------+--
| EMPNO | ENAME      | JOB      | MGR  | HIREDATE   | SA
+-------+------------+----------+------+------------+--
|  7369 | SMITH      | CLERK    | 7902 | 1980-12-17 |
|  7499 | ALLENXXXXX | SALESMAN | 7698 | 1981-02-20 |
|  7521 | WARD       | SALESMAN | 7698 | 1981-02-22 |
|  7566 | JONES      | MANAGER  | 7839 | 1981-04-02 |
|  7654 | MARTIN     | SALESMAN | 7698 | 1981-09-28 |
|  7698 | BLAKE      | MANAGER  | 7839 | 1981-05-01 |
```



```
+-------+------------+----------+------+------------
| EMPNO | ENAME      | JOB      | MGR  | HIREDATE
+-------+------------+----------+------+------------
|  7369 | SMITH      | CLERK    | 7902 | 1980-12-17
|  7499 | ALLENXXXXX | SALESMAN | 7698 | 1981-02-20
|  7521 | WARD       | SALESMAN | 7698 | 1981-02-22
|  7566 | JONES      | MANAGER  | 7839 | 1981-04-02
|  7654 | MARTIN     | SALESMAN | 7698 | 1981-09-28
|  7698 | BLAKE      | MANAGER  | 7839 | 1981-05-01
```

```
delimiter //
create procedure update_emp_sal(out pmcnt  int,out pacnt  int,out pccnt  int,out pocnt  int)
begin
  declare finished int default 0;
  declare vempno,vmgr,vdeptno int;
  declare vename,vjob varchar(20);
  declare vhiredate date;
  declare vsal,vcomm,vupd_sal double(9,2);
  declare emp_cur cursor for select * from emp;
  declare continue handler for NOT FOUND set finished=1;
  set pmcnt=0;
```

```
 set pacnt=0;
 set pccnt=0;
 set pocnt=0;
 open emp_cur;
 l1:loop
   fetch emp_cur into vempno,vename,vjob,vmgr,vhiredate,vsal,vcomm,vdeptno;
   if finished=1 then
     leave l1;
end if;
if vjob='manager' then
  set vupd_sal=1.1*vsal;
  update emp
  set sal=1.1*sal
 where empno=vempno;
 set pmcnt=pmcnt+1;
elseif vjob='analyst' then
  set vupd_sal=1.2*vsal;
  update emp
  set sal=1.2*sal
 where empno=vempno;
 set pacnt=pacnt+1;
elseif vjob='clerk' then
  set vupd_sal=1.25*vsal;
  update emp
  set sal=1.25*sal
 where empno=vempno;
 set pccnt=pccnt+1;
else
  set vupd_sal=1.08*vsal;
  update emp
  set sal=1.08*sal
 where empno=vempno;
 set pocnt=pocnt+1;
end if;
select vempno,vename,vjob,vsal,vcomm,vupd_sal;
 end loop;
 select pmcnt,pacnt,pccnt,pocnt;
 close emp_cur;
end//
delimiter ;
```

3. Write a procedure to update price of product using cursor.
   If the category is chips then increase the price by 10%
   If the category is cold drink then increase the price by 20%
   Else increase by 8 %

```
delimiter //
create procedure changeprice()
begin
  declare finished int default 0;
  declare vpid,vcid,vchipscatid,vdrinkcatid,vqty int;
  declare vprice double(9,2);
  declare vname varchar(20);
  declare prod_cur cursor for select * from product;
  declare continue handler for NOT FOUND set finished=1;

  open prod_cur;
  select cid into vchipscatid
  from category
  where cname='chips';

  select cid into vdrinkcatid
  from category
  where cname='cold drink';
  l1:loop
    fetch prod_cur into vpid,vname,vqty,vprice,vcid;
      if finished=1 then
        leave l1;
      end if;
      if vcid = vchipscatid then
         update product
                set price=1.1*ifnull(price,1)
                where pid=vpid;
      elseif vcid = vdrinkcatid then
         update product
                set price=1.2*ifnull(price,1)
                where pid=vpid;
      else
         update product
                set price=1.08*ifnull(price,1)
                where pid=vpid;
      end if;

  end loop;
end//
```

4. Write a procedure to find comma separated list of emails.

```
delimiter //
create procedure generate_email()
begin
  declare str varchar(1000) default '';
  declare vemail,vename,vjob varchar(50);
  declare finished int default 0;
  declare emp_cur cursor for select ename,job from emp;
  declare continue handler for NOT found set finished=1;

  open emp_cur;
  l1:loop
    fetch emp_cur into vename,vjob;
      if finished=1 then
```

```
          leave l1;
        end if;
        if vjob is not null then
                  set
vemail=concat(substr(vename,1,3),'.',substr(vjob,1,3),'@mycompany.com');
                  set str=concat(str,vemail,',');
        end if;

   end loop;
   close emp_cur;
   select str;

end//
delimiter ;
```

To see the list of all procedures and functions

```
SELECT ROUTINE_DEFINITION
FROM information_schema.ROUTINES WHERE
SPECIFIC_NAME='procedurename'

SHOW FUNCTION STATUS WHERE Db = 'db_name';
SHOW procedure STATUS WHERE Db = 'db_name';
```

difference between functions and procedure

procedure	function
1.it doesnot return any value	it returns a single value
2. use call statement to call a procedure, you cannot use it in select statement	we can call functions in select statement

to allow create functions

```
SET GLOBAL log_bin_trust_function_creators = 1;
```

Function

When you want to return one value as output then write functions

1. Write a function to generate email

```
Delimiter //
Create function get_email(name varchar(20),jb varchar(20)) returns varchar(50)
Begin
    Declare email varchar(50)
    Set email=concat((substr(name,1,3),'.',substr(jb,1,3),'@mycompany.com');
    Return email;
End//
Delimiter ;
```

2. Write a function to find exp

```
delimiter //
create function get_exp(hdate date) returns int
begin
  declare exp int;
  set exp=floor(datediff(curdate(),hdate)/365);
  return exp;
end//
```

3. Write a function which accepts price and qty as i/p and returns discounted price.

   If qty < 20 then apply 10% discount on price
   Else if qty >= 20 and <=30 discount 20%
   Otherwise 30% discount

```
Delimiter //
Create function get_discount(pr double(9,2), qty int) returns double(9,2)
Begin
   Declare dis_price double(9,2) default 0;
If qty!=0 then
    If qty<20 then
        Set dis_price=0.9*pr;
    Elseif qty<30 then
       Set dis_price=0.8*pr;
    Else
       Set dis_price=0.7*pr;
   End if;

End if;

   Return dis_price;

End//
```

Tiggers
Tiggers are procedures which are automatically called
Used for monitoring DML activities on tables by all users.

Tiggers can be executed either before or after the dml statement,
There are 2 types of triggers

1. Row level trigger
2. Statement level trigger –this trigger does not work in mysql

In mysql we can use row level trigger on all DML operation

Timings

1. Before---- before trigger gets executed before the actual statement
2. After----- after trigger gets executed after execution of the actual statement

3. Insteadof ----these triggers are used only on views, but mysql doe not support it

Before creation of trigger we need to create a table to store the required information needed for monitoring the table

```
Create table empsecurity(

Empno int,

Ename varchar(20),

Action varchar(20),

Oldsal double(9,2),

Newsal double(9,2),

Uname varchar(20),

Act_date date);
```

```
Create trigger emp_update before update on emp

For each row

Begin

   Insert into empsecurity values(OLD.empno,OLD.ename,’update’,OLD.sal,NEW.sal,user(),curdate())

End//
```