**Review Questions**

**Using Stored Procedures**

**I. Multiple Choice Questions**

1. Which one is the statement that you use to declare variables inside a stored procedure?
   1. call
   2. execute
   3. **declare**
   4. var
2. Which one is the correct statement to declare a variable that holds hexadecimal value?
   1. declare x int = 0xA;
   2. declare x integer = 0xA;
   3. **declare x int default 0xA;**
   4. declare x int default ‘0xA’;
3. Which one is the correct statement to drop a stored procedure called ‘test’?
   1. drop test;
   2. drop stored procedure test;
   3. drop table test;
   4. **drop procedure test;**
4. Which one is the correct statement to declare a variable to store a date value?
   1. declare x datetime default 23/11/2012;
   2. **declare x date default ‘2012-11-23’;**
   3. declare x date = ‘ 23/11/2012’;
   4. declare x date set = ‘2012-11-23’;
5. A stored procedure cannot return a value.
   1. true
   2. **false**
6. In MySQL, the user variable can be considered as the global variable as in other languages.
   1. **true**
   2. false
7. If you declare a variable as a user variable, you cannot pass it to a stored procedure.
   1. true
   2. **false**
8. If you declare a variable as a local variable inside a stored procedure, its scope only exist inside that stored procedure.
   1. **true**
   2. false
9. A local variable is a variant data type the same a user variable.
   1. true
   2. **false**
10. Which one is not a comment in a stored procedure?
    1. -- comment
    2. # comment
    3. /\* comment \*/
    4. **// comment**
11. MySQL supports C style operators such as ++, -- and +=.
    1. true
    2. **false**
12. What is the output of the following code segment?  
    **select 10 div 3;**
    1. 3.3333
    2. **3**
    3. 3.0
    4. 3.1
13. What is the output of the following code segment?  
    **select if(3<=>null,'True','False') as Output;**
    1. True
    2. **False**
    3. NULL
    4. NaN
14. What is the output of the following code segment?  
    **set @a=null; select ifnull(@a,2);**
    1. null
    2. **2**
    3. 0
    4. NaN
15. What is the output of the following code segment?  
    **select least(3,1,4,6);**
    1. 3
    2. **1**
    3. 4
    4. 6
16. Which statement do you use to exit a loop?
    1. iterate
    2. **leave**
    3. exit
    4. end

**II. Exercises**

1. Create a simple stored procedure to calculate your age by input born year.

delimiter //

**create** **procedure** findAge(bornyear **int**)

**begin**

**select** year(now()) - bornyear **as** 'Your Age';

**end**//

call findAge(1990)//

1. Create a simple stored procedure to calculate the sum, difference, product, division, power and average of two floating point numbers dynamically.

delimiter //

**create** **procedure** cal2Numbers(n1 **float**, n2 **float**)

**begin**

**select** n1+n2 **as** 'Sum', n1\*n2 **as** 'Product',

n1-n2 **as** 'Minus', n1/n2 **as** 'Division',

pow(n1,n2) **as** 'Power', (n1+n2)/2 **as** 'Average';

**end**//

call cal2Numbers(12.12,1.1)//

1. Create a stored procedure to select all category from table category.

delimiter //

**create** **procedure** getCatList()

**begin**

**select** \* **from** category;

**end**//

call getCatList()//

1. Create a stored procedure to find the product that has minimum quantity.

delimiter //

**create** **procedure** getMinProduct()

**begin**

**select** \* **from** product **where** quantity **in**(

**select** min(quantity) **from** product

);

**end**//

call getMinProduct()//

1. Create a stored procedure to select all data from any table dynamically.

delimiter //

**create** **procedure** getData(tblname **varchar**(30))

**begin**

**set** @txt = concat('select \* from ', tblname);

prepare stmt **from** @txt;

execute stmt;

**end**//

call getData('product')//

1. Create a stored procedure to delete product dynamically by its id.

delimiter //

**create** **procedure** delProduct(pro\_id **int**)

**begin**

**delete** **from** product **where** pid = pro\_id;

**end**//

call delProduct(2)//

1. Create a stored procedure to show all products that don’t have category or have wrong category.

delimiter //

**create** **procedure** getWrongProducts()

**begin**

**select** \* **from** product **where** catid **not** **in**(

**select** catid **from** category

) **or** catid **is** **null**;

**end**//

call getWrongProducts()//

1. Create a stored procedure to insert data into table category dynamically. If the category already exist, don’t allow them to insert it.

delimiter //

**create** **procedure** addCategory(cat\_name **varchar**(30), cat\_desc **varchar**(200))

**begin**

if **exists**(**select** \* **from** category **where** catname = cat\_name) **then**

**select** concat(cat\_name, ' already exists!');

else

**insert** **into** category (catname, description) **values**

(cat\_name, cat\_desc);

**end** if;

**end**//

call addCategory('Monitor','Try to some new monitors')//

1. Create a stored procedure to search product by its name.

delimiter //

**create** **procedure** searchProduct(pro\_name **varchar**(30))

**begin**

**select** \* **from** product **where** pname **like** concat('%', pro\_name,'%');

**end**//

call searchProduct('mo')//

1. Create a stored procedure that creates a sale report by year. Your report should include this information: year, minimum sale, maximum sale, average sale and total sale.

delimiter //

**create** **procedure** saleReportByYear()

**begin**

**select** year(sales.salesdate) **as** 'Year',

concat('$ ', format(min(product.unitprice\*sales.quantity),3)) **as** 'Min Sale',

concat('$ ', format(max(product.unitprice\*sales.quantity),3)) **as** 'Max Sale',

concat('$ ', format(avg(product.unitprice\*sales.quantity),3)) **as** 'Avg Sale',

concat('$ ', format(sum(product.unitprice\*sales.quantity),3)) **as** 'Total'

**from** sales **inner** **join** product **on** sales.pid = product.pid

**group** **by** Year;

**end**//

call saleReportByYear()//

1. Create a stored procedure for a user login. If user logins success, display 1, otherwise display 0.

delimiter //

**create** **procedure** userLogin(username **varchar**(30), pass **varchar**(30))

**begin**

if **exists**(**select** \* **from** users **where** loginname = username **and** userpass = pass) **then**

**select** 1;

else

**select** 0;

**end** if;

**end**//

call userLogin('admin','123')//

1. Create a stored procedure to insert data into table employee. If the employee already exists, don’t allow to insert.

delimiter //

**create** **procedure** addEmployee(

fname **varchar**(30), lname **varchar**(30), emp\_title **varchar**(30),

emp\_age **int**, serviceyear **int**, emp\_salary **float**,

emp\_perks **float**, emp\_email **varchar**(60), depid **int**

)

**begin**

if **exists**(**select** \* **from** employee **where** firstname = fname **and** lastname=lname) **then**

**select** 'This employee already exist!';

else

**insert** **into** employee (firstname, lastname, title, age, yearofservice, salary, perks, email, departmentid)

**values**(fname, lname, emp\_title, emp\_age, serviceyear, emp\_salary, emp\_perks, emp\_email, depid);

**end** if;

**end**//

call addEmployee('HENG', 'Vongkol', 'Teacher', 50, 3, 1234, 432, 'hengvongkol@gmail.com', 12)//

1. Create a stored procedure to find who the best seller in 2009, 2010, 2011 and 2012.

-- First, create a view

-- I show you only for one year, 2009

-- for the other you can try it.

**create** **or** replace **view** v\_sale\_report **as**

**select** sales.seller **as** 'seller',

sum(product.unitprice \* sales.quantity) **as** 'total' **from** sales

**inner** **join** product **on** sales.pid = product.pid

**where** year(sales.salesdate)=2009 **group** **by** seller;

-- Second, create stored procedure

delimiter //

**create** **procedure** bestSellers()

**begin**

**select** seller **from** v\_sale\_report **where** total = (

**select** max(total) **from** v\_sale\_report

);

**end**//

call bestSellers()//

1. Create a stored procedure to export data from table product to a csv file that can be opened with MS Excel.

delimiter //

**create** **procedure** exportProduct()

**begin**

**select** \* **from** product **into** outfile 'e:/product.csv' fields terminated **by** ',';

**end**//

call exportProduct()//

1. Create a stored procedure to update discount (%) in table sales based on the following conditions:  
   - If quantity less than 50, discount 2%.  
   - If quantity less than 80, discount 3%.  
   - If quantity less than 150, discount 4%.  
   - If quantity less than 200, discount 5%.  
   - If quantity greater than or equal to 200, discount 10%.

**Method #1 (fast and easy)**

delimiter //

**create** **procedure** upDisc()

**begin**

**update** sales **set** discount = 2 **where** quantity>=0 **and** quantity<50;

**update** sales **set** discount = 3 **where** quantity>=50 **and** quantity<80;

**update** sales **set** discount = 4 **where** quantity>=80 **and** quantity<150;

**update** sales **set** discount = 5 **where** quantity>=150 **and** quantity<200;

**update** sales **set** discount = 10 **where** quantity>=200;

**end**//

call upDisc()//

**Method #2 (using curssor)**

delimiter //

**create** **procedure** upDisc()

**begin**

**declare** qty **int**;

**declare** id **int**;

**declare** cur **cursor** for **select** salesid, quantity **from** sales;

**declare** continue handler for **not** found **set** @a=**true**;

open cur;

**set** @a=**false**;

lb: loop

fetch cur **into** id, qty;

if qty<50 **then**

**update** sales **set** discount = 2 **where** salesid = id;

elseif qty<80 **then**

**update** sales **set** discount = 3 **where** salesid = id;

elseif qty<150 **then**

**update** sales **set** discount = 4 **where** salesid = id;

elseif qty<200 **then**

**update** sales **set** discount = 5 **where** salesid = id;

else

**update** sales **set** discount = 10 **where** salesid = id;

**end** if;

if @a **then**

leave lb;

**end** if;

**end** loop lb;

**set** @a=**true**;

close cur;

**end**//

call upDisc()//

1. Create a stored procedure to create a table called “delsales” if not exists and then move all sales in 2009 to that table.

delimiter //

**create** **procedure** moveData()

**begin**

**create** **table** if **not** **exists** delsales **like** sales;

if **exists**(**select** \* **from** information\_schema.tables **where** TABLE\_NAME = 'delsales') **then**

**insert** **into** delsales **select** \* **from** sales **where** year(salesdate) = 2009;

**delete** **from** sales **where** year(salesdate) = 2009;

**end** if;

**end**//

call moveData()//