Experiment 12
MySQL Stored Procedure Programming II

Submitted By,
Niranjan V Ram
39, S5 CSE

Aim: Practise the use Non-SELECT SQL statements and SELECT-INTO clause within stored procedures.

1. Create a table temp with two fields,
TEMP01(num:INTEGER, message TEXT)
Insert values into this table using a stored procedure such that the num field is having values from 1 to 10 and corresponding message is either even or odd.

Code:

```
delimiter $$

drop procedure if exists inserttemp$$

create table temp (num INTEGER,message VARCHAR(5));

create procedure inserttemp()
begin
  insert into temp (num, message) VALUES (1, 'Odd');
  insert into temp (num, message) VALUES (2, 'Even');
  insert into temp (num, message) VALUES (3, 'Odd');
  insert into temp (num, message) VALUES (4, 'Even');
  insert into temp (num, message) VALUES (5, 'Odd');
  insert into temp (num, message) VALUES (6, 'Even');
  insert into temp (num, message) VALUES (7, 'Odd');
  insert into temp (num, message) VALUES (8, 'Even');
  insert into temp (num, message) VALUES (9, 'Odd');
  insert into temp (num, message) VALUES (10, 'Even');

end$$

delimiter ;
```

Output:

```
call inserttemp();

+------+---------+
| num  | message |
+------+---------+
|    1 | Odd     |
|    2 | Even    |
|    3 | Odd     |
|    4 | Even    |
|    5 | Odd     |
|    6 | Even    |
|    7 | Odd     |
|    8 | Even    |
|    9 | Odd     |
|   10 | Even    |
+------+---------+
```

2. Create an employee table and insert 5 rows. Write a procedure to calculate income tax of a specified employee. [Give the employee SSN as input parameter]
Employee(SSN,Name,Designation,Basic_pay,DA,HRA,Gender,Years_of_exp)
Note: You can create and insert values outside the procedure as usual. Insert meaningful values to all fields and use original way of calculating tax for a person.

Code:
```
delimiter $$

drop procedure if exists insertemployee$$

drop table if exists employee$$

create table employee(SSN INT,Name VARCHAR(30),Designation
VARCHAR(30),Basic_pay INT,DA INT,HRA INT,Gender VARCHAR(1),Years_of_exp
INT);

insert into employee values(1, 'John Smith', 'Developer', 45000, 10000,
5000, 'M', 3);
insert into employee values(2, 'Jane Doe', 'Project Manager', 15000,
7000, 800, 'F', 5);
insert into employee values(3, 'Jack Johnson', 'Tester', 35000, 8000,
4000, 'M', 2);
insert into employee values(4, 'Jill Anderson', 'Analyst', 45000, 10000,
5000, 'F', 4);
insert into employee values(5, 'Jeff Williams', 'Architect', 60000, 2000,
10000, 'M', 6);

create procedure insertemployee(ssn INT)
BEGIN
      DECLARE ts INT;
      DECLARE it INT;

      select Basic_pay+DA+HRA into ts from employee where SSN=ssn LIMIT
1;

      IF ts<=25000 THEN
            SET it=0;
      ELSEIF ts<=50000 THEN
            SET it=(ts-25000)*5/100;
      ELSEIF ts<=100000 THEN
            SET it=1250+(ts-50000)*20/100;
      ELSE
            SET it=11250+(ts-100000)*30/100;
      END IF;

      select it as "Income_tax";
END$$

delimiter ;
```

Output:

```
call insertemployee(5);

+------------+
| Income_tax |
+------------+
|       3250 |
+------------+
```

3. Write a procedure to Display Salary of a specified employee (as input argument) increased by 500 if his/her salary is more than 30000. [Use above table]

Code:
```
delimiter $$

drop procedure if exists dispsalary$$

create procedure dispsalary(ssn INT)
begin
     select Name,(Basic_pay+DA+HRA+IF(Basic_pay>30000,500,0)) as Salary
from employee where SSN=ssn LIMIT 1;

end$$

delimiter ;
```

Output:

```
call dispsalary(3);

+------------+--------+
| Name       | Salary |
+------------+--------+
| John Smith |  60500 |
+------------+--------+
```

4. Create a procedure to calculate the bonus of an employee whose SSN is given as input, based on experience and store it into the bonus table:
Bonus(SSN, Name, Bonus)
If exp < 5 years then bonus is 1 month salary
If exp between 5 and 9 years then bonus is 20% of annual salary
If exp more than 9 years then bonus is 1 month salary plus 25% of annual salary

Code:
```
delimiter $$

drop procedure if exists bonuscalc$$
drop table if exists bonus;

create table bonus(SSN INTEGER,Name VARCHAR(30),Bonus INTEGER);

CREATE PROCEDURE bonuscalc(ssn VARCHAR(50))
BEGIN
      DECLARE b DECIMAL(18,2);
      DECLARE salary DECIMAL(18,2);
      DECLARE expe INT;
      DECLARE n VARCHAR(30);

      select Years_of_exp into expe from employee where SSN=ssn LIMIT 1;

      select Basic_pay into salary from employee where SSN=ssn LIMIT 1;

      select Name into n from employee where SSN=ssn LIMIT 1;

      IF expe < 5 THEN
            SET b = salary;
      ELSEIF expe < 9 THEN
            SET b = (salary * 0.2 * 12);
      ELSE
            SET b = (salary + (salary * 0.25 * 12));
      END IF;

      insert into bonus values(ssn,n,b);
      select * from bonus LIMIT 1;

END$$

delimiter ;
```

Output:

```
call bonuscalc(3);

+------+------------+-------+
| SSN  | Name       | Bonus |
+------+------------+-------+
|    3 | John Smith | 45000 |
+------+------------+-------+
```

5. Create a table
account_master (acct_no :int, customer_name: text, balance:decimal).
Write a stored procedure to accept the account number and the amount to withdraw. Do proper updation on the table only if there is sufficient amount, otherwise display proper message.

Code:

```
delimiter $$

drop procedure if exists withdraw$$
drop table if exists account_master;

create table account_master(acct_no INTEGER,customer_name
VARCHAR(30),balance DECIMAL(10,2));

insert into account_master values(1,"John Doe",2000.00),(2,"Jane
Jacob",7000.00),(3,"Jack Smith",3000.00);

create procedure withdraw(accno INT,amount DECIMAL(10,2))
BEGIN
      DECLARE curbal DECIMAL(10,2);
      SET curbal = (select balance from account_master where
acct_no=accno);

      IF curbal >= amount THEN
      BEGIN
            select 'Transaction success!!' as Message;
            update account_master set balance=balance-amount where
acct_no=accno;
            select * from account_master;
      END;
      ELSE
            select 'Insufficient balance!!' as Message;
      END IF;
END$$

delimiter ;
```

Output:

```
call withdraw(2,4000.00);

+-----------------------+
| Message               |
+-----------------------+
| Transaction success!! |
+-----------------------+
1 row in set (0.001 sec)

+---------+---------------+---------+
| acct_no | customer_name | balance |
+---------+---------------+---------+
|       1 | John Doe      | 2000.00 |
|       2 | Jane Jacob    | 3000.00 |
|       3 | Jack Smith    | 3000.00 |
+---------+---------------+---------+
```

Submitted By,
Niranjan V Ram
39, S5 CSE