SET 3

-----

1. Write a stored procedure that accepts the month and year as inputs and prints the ordernumber, orderdate and status of the orders placed in that month.

***Example***: call order\_status(2005, 11);

Ans:

create procedure sp\_order\_status(in order\_year INT(4),in order\_month varchar(3))

begin

select orderdate,status from orders where year(orderdate)=order\_year and

month(orderdate)=month(str\_to\_date(order\_month,%b’));

end

2. a. Write function that takes the customernumber as input and returns the purchase\_status based on the following criteria . [table:Payments]

if the total purchase amount for the customer is < 25000 status = Silver, amount between 25000 and 50000, status = Gold

if amount > 50000 Platinum

Ans:

create procedure sp\_payment\_status (in customerno int)

begin

select (case when amount<25000 then ‘Silver’ when amount<=50000 then ‘Gold’ when

amount>50000 then ‘Platinum’ end) as purchase\_status from payments where

customernumber=customerno;

end

b. Write a query that displays customerNumber, customername and purchase\_status from customers table.

Ans:

select p.customernumber,customername,(case when amount<25000 then ‘Silver’ when

amoun<=50000 ‘Gold’ when amount>50000 then ‘Platinum’ end) as purchase\_status from

payments as p inner join customers as c on c.customernumber=p.customernumber;

3. Replicate the functionality of 'on delete cascade' and 'on update cascade' using triggers on movies and rentals tables. Note: Both tables - movies and rentals - don't have primary or foreign keys. Use only triggers to implement the above.

Ans:

Update trigger for update cascade functionality

CREATE DEFINER=`root`@`localhost` TRIGGER `movies\_AFTER\_UPDATE` AFTER UPDATE

ON `movies` FOR EACH ROW BEGIN

update rentals

set movieid = new.id

where movieid = old.id;

END

delete trigger for update delete functionality

CREATE DEFINER=`root`@`localhost` TRIGGER `movies\_AFTER\_DELETE` AFTER DELETE

ON `movies` FOR EACH ROW BEGIN

delete from rentals

-- set movieid = new.id

where movieid = old.id;

END

4. Select the first name of the employee who gets the third highest salary. [table: employee]

Ans: select first name,MAX(salary) from employee

Group by first name

Order by MAX(salary) Limit2,1;

5. Assign a rank to each employee based on their salary. The person having the highest salary has rank 1. [table: employee]

Ans: select emp\_id,first name,MAX(salary) from employee

Group by emp\_id

Order by MAX(salary)