SQL ASSIGNMENT – 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. The month should be abbreviated to three characters.

Example:

Input: month -> 'Feb'

year -> 2003

ans:

CREATE DEFINER=`root`@`localhost` PROCEDURE `My\_procedure\_ans1`(month1 varchar(255), year1 integer)

BEGIN

select orderNumber,orderDate,status from orders where year(orderdate)=year1 and left(monthname(orderdate),3)=month1;

END

2. Write a stored procedure to insert a record into the cancellations table for all cancelled orders.

STEPS: a. Create a table called cancellations with the following fields

id (primary key), custumernumber (foreign key), ordernumber (foreign key), comments

All values except id should be taken from the order table.

b. Read through the orders table . If an order is cancelled, then put an entry in the cancellations table.

ans:

CREATE DEFINER=`root`@`localhost` PROCEDURE `New\_proc\_ans2`()

BEGIN

create table if not exists cancellations(id int PRIMARY KEY auto\_increment,

customernumber int,

FOREIGN KEY(customerNumber)REFERENCES customers(customerNumber) ,

orderNumber int, FOREIGN KEY(orderNumber)REFERENCES orders(orderNumber));

insert into cancellations(customernumber,orderNumber)

select customerNumber,orderNumber from orders where status='Cancelled' ;

END

3. 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 DEFINER=`root`@`localhost` FUNCTION `New\_function\_ans3a`(customernumber integer) RETURNS varchar(200) CHARSET utf8mb4

BEGIN

declare P\_status varchar(20);

select

case

when amount < 25000 then 'silver'

when amount between 25000 and 50000 then 'Gold'

when amount > 50000 then 'Platinum'

end as purchasestatus

into P\_status from payments where customerNumber=customerNumber;

RETURN (P\_status);

END

b. Write a query that displays customerid, customername and purchase\_status

ans:

CREATE DEFINER=`root`@`localhost` PROCEDURE `New\_proc\_ans3b`()

BEGIN

select customers.customerNumber ,

customerName ,

case

when amount < 25000 then 'silver'

when amount between 25000 and 50000 then 'Gold'

when amount > 50000 then 'Platinum'

end as purchasestatus

from payments

inner join customers ON payments.customerNumber=customers.customerNumber;

END

4. Write a stored procedure that checks the creditlimit and the purchase status of the customers.

If a platinum customer has crediltlimit less than 100,000 raise an exception. In the exception handler update the crediltlimit to 100000.

If a silver customer has creditlimit greater than 60,000 raise an exception. In the exception handler update the crediltlimit to 60000.

Ans:

CREATE DEFINER=`root`@`localhost` PROCEDURE `New\_proc\_ans4`(custNo integer)

BEGIN

declare custNo integer default 0;

DECLARE credit DECIMAL(10,2) DEFAULT 0;

DECLARE P\_Status varchar(10);

DECLARE Update\_condition CONDITION FOR SQLSTATE '22012';

if ( select 1

from (

select creditLimit,

case

when amount < 25000 then 'silver'

when amount between 25000 and 50000 then 'Gold'

when amount > 50000 then 'Platinum'

end as purchasestatus,a.customerNumber from payments a inner join customers b

on a.customerNumber=b.customerNumber) as a

where customerNumber=custNo and purchasestatus='Platinum' and creditLimit<100000

)=1 then

update customers

set creditlimit =100000

where customerNumber=custNo;

SIGNAL SQLSTATE '22012'

SET MESSAGE\_TEXT ='credit is less than 100000';

elseif

( select 1

from (

select creditLimit,

case

when amount < 25000 then 'silver'

when amount between 25000 and 50000 then 'Gold'

when amount > 50000 then 'Platinum'

end as purchasestatus,a.customerNumber from payments a inner join customers b

on a.customerNumber=b.customerNumber) as a

where customerNumber=custNo and purchasestatus='silver' and creditLimit>60000

)=1 then

update customers

set creditlimit =60000

where customerNumber=custNo;

SIGNAL SQLSTATE '22012'

SET MESSAGE\_TEXT ='credit is less than 60000';

end if;

commit;

END

5. 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.

CREATE DEFINER=`root`@`localhost` TRIGGER `UPDATE\_cascade` AFTER UPDATE ON `movies` FOR EACH ROW

BEGIN

update rentals set movieid = new.id where movieid = old.id;

END

CREATE DEFINER=`root`@`localhost` TRIGGER `DELETE\_cascade` BEFORE DELETE ON `movies` FOR EACH ROW

BEGIN

delete from rentals where movieid = old.id;

END