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

Output:

+------------+---------+

| orderdate | status |

+------------+---------+

| 2003-02-11 | Shipped |

| 2003-02-17 | Shipped |

| 2003-02-24 | Shipped |

+------------+---------+

3 rows in set (0.001 sec)

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. 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.

Ans:

CREATE TABLE cancellations (id int AUTO\_INCREMENT, Ordernumber int NOT NULL, COMMENTS text, customernumber int NOT NULL, PRIMARY KEY (ID), FOREIGN KEY (customernumber) REFERENCES customers (customernumber),FOREIGN KEY (ordernumber) REFERENCES orders (ordernumber));

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

Ans:

create procedure sp\_cancellations\_record()

begin

insert into cancellations (ordernumber,comments,customernumber) select ordernumber,comments,customernumber 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

CREATE FUNCTION GetPurchaseStatus (IN customernumber INT)

RETURNS VARCHAR(50)

BEGIN

DECLARE total\_purchase\_amount DECIMAL(10, 2);

DECLARE status VARCHAR(50);

SELECT SUM(amount) INTO total\_purchase\_amount

FROM Payments

WHERE customer\_number = customernumber;

IF total\_purchase\_amount < 25000 THEN

SET status = 'Silver';

ELSEIF total\_purchase\_amount >= 25000 AND total\_purchase\_amount <= 50000 THEN

SET status = 'Gold';

ELSE

SET status = 'Platinum';

END IF;

RETURN status;

END;

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

Ans:

select p.customernumber,customername,(case when amount<25000 then 'Silver' when amount<=50000 then 'Gold' when amount>50000 then 'Platinum' end) as purchase\_status from payments as p inner join customers as c on c.customernumber=p.customernumber;

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 PROCEDURE check\_limit()

BEGIN

declare finished,cnum integer default 0;

declare crlimit,uplimit decimal (10,2) default 0.0;

declare pstatus varchar(10) default " ";

declare credit\_cur cursor for

select customernumber,creditlimit,purchage\_status(customernumber) from customers

where purchase\_status(customernumber) in ("platinum","gold");

declare exit handler for NOT FOUND SET finished = 1;

declare continue handler for SQLSTATE "45000"

BEGIN

select concat ('updating',cnum,'>',uplimit) as message;

update customers set creditlimit = uplimit where customernumber =cnum;

end;

open credit\_cur;

creditloop: Repeat fetch credit\_cur into cnum,crlimit,pstate;

if pstate= "Platinum" and crlimit < 40000 then set uplimit=100000;

signal sqlstate "45000";

elseif pstate ="Gold" and crlimit > 60000 then set uplimit=60000;

signal sqlstate "45000";

else

until finished =1

end Repeat creditloop;

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.

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