**Assignment 3**

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)

**CREATE DEFINER=`root`@`localhost` PROCEDURE `new\_procedure`(mon varchar(3),yr varchar(4))**

**BEGIN**

**select orderdate , status from orders where orddate=concat(mon,yr);**

**END**

FOR ADDING ADDITONAL COLUMN TO PAYMENTS TABLE (ORDATE—>2003—JAN)

**select customernumber,purchase (customernumber) from payments where customernumber=114;**

**alter table orders add column orddate date;**

**insert into orders(orddate) values(date\_format(orderdate,"%M %Y"));**

**select concat( left(monthname(orderdate),3)," " ,year(orderdate)) as orddate from orders;**

**alter table orders modify column orddate varchar(50);**

**update orders set orddate=(concat( left(monthname(orderdate),3),year(orderdate)));**

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.

**Create table cancellations (id integer primary key auto\_increment,custnumber integer,ordnumber integer,foreign key(custnumber,ordnumber) references orders(customerNumber,orderNumber)**

**on delete cascade**

**on update cascade);**

**select ordernumber,customernumber,status from orders;**

**select customernumber,customername,purchase\_status(customernumber) from customers;**

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

**BEGIN**

**declare id integer;**

**declare lcl\_ord integer;**

**declare lcl\_cust integer;**

**declare lcl\_st varchar(20);**

**declare can\_cur cursor for select ordernumber,customernumber,status from orders;**

**open can\_cur;**

**orderloop:loop**

**fetch can\_cur into lcl\_ord,lcl\_cust,lcl\_st;**

**if lcl\_st="cancelled" then**

**insert into cancellations values(id,lcl\_cust,lcl\_ord);**

**end if;**

**end loop orderloop;**

**close can\_cur;**

**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 DEFINER=`root`@`localhost` FUNCTION `purchase\_status`(cstno integer) RETURNS varchar(20) CHARSET utf8mb4**

**BEGIN**

**declare status varchar(20);**

**declare amt integer;**

**select sum(amount) into amt from payments where customernumber=cstno;**

**if (amt<25000) then**

**set status="silver";**

**elseif (amt between 25000 and 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

**Select customernumber,customername,purchase\_status(customernumber) from customers;**

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.

**CREATE PROCEDURE `check\_credit`()**

**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, purchase\_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, pstatus;**

**if pstatus = 'platinum' and crlimit < 40000 then**

**set uplimit = 100000;**

**signal sqlstate '45000';**

**elseif pstatus = 'gold' and crlimit > 60000 then**

**set uplimit= 60000;**

**signal sqlstate '45000';**

**else**

**iterate creditloop;**

**end if ;**

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 `movies\_BEFORE\_DELETE` BEFORE DELETE ON `movies` FOR EACH ROW BEGIN**

**delete from rentals where movieid = old.id;**

**END**

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

**BEGIN**

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

**END**