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);

SET SQL\_SAFE\_UPDATES =0;

-- Q1.

Order\_status

Delimiter //

CREATE PROCEDURE Order\_status (IN in\_month INT, IN in\_year INT )

BEGIN

SELECT orderNumber, orderDate ,status FROM orders

WHERE in\_month = MONTH(shippeddate) AND in\_year = YEAR(shippeddate) ;

END //

CALL Order\_status ( 5 , 2005);

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

STEPS:

1. Create a table called cancellations with the following fields

id (primary key),

customernumber (foreign key - Table customers),

ordernumber (foreign key - Table Orders),

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.

---------🡪

-- Q2

CREATE TABLE cancellations(

id INT PRIMARY KEY AUTO\_INCREMENT,

customernumber INT,

ordernumber INT,

comments VARCHAR(250)

);

DELIMITER //

CREATE PROCEDURE cancel\_order()

BEGIN

INSERT INTO cancellations (customerNumber,ordernumber, comments)

SELECT customerNumber,ordernumber, comments FROM orders WHERE status ='cancelled' ;

END//

CALL cancel\_order();

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, stat = Gold

if amount > 50000, stat =Platinum

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

-- Q3 a.

DELIMITER //

CREATE FUNCTION purchase\_status(in\_customerNumber INT)

RETURNS varchar(10)

READS SQL DATA

DETERMINISTIC

BEGIN

DECLARE Stat VARCHAR(20);

DECLARE total FLOAT DEFAULT 0;

SET total =

(SELECT SUM(amount) FROM payments WHERE in\_customerNumber =customerNumber);

IF total<25000 THEN SET Stat = 'Silver';

ELSEIF total > 25000 AND amount <50000 THEN SET Stat = 'Gold';

ELSEIF total < 50000 THEN SET Stat ='Platinum';

END IF;

RETURN Stat;

END//

-- Q3b

DELIMITER //

CREATE PROCEDURE cust\_detail()

BEGIN

Select cutomerNumber, customerName,

if (creditLimit>50000,'Platinum',IF(50000>creditLimit>25000,'Gold','Silver'))

FROM customers;

END//

4. 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.

-- Q4

DELIMITER $$

CREATE TRIGGER trg\_movies\_update

AFTER DELETE ON movies

FOR EACH ROW

BEGIN

UPDATE rentals

SET movieid = id

WHERE movieid = OLD.id ;

END;

DELIMITER $$

CREATE TRIGGER trg\_movies\_delete

AFTER DELETE ON movies

FOR EACH ROW

BEGIN

DELETE FROM rentals

WHERE movieid

NOT IN (SELECT DISTINCT id FROM movies);

END;

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

-- Q5

SELECT fname FROM employee ORDER BY salary DESC LIMIT 2, 1;

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

-- Q6

SELECT \*,

RANK () OVER (

ORDER BY salary DESC

) AS Rank\_no

FROM employee;