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);

🡪

DELIMITER //

CREATE PROCEDURE order\_status(month1 varchar(20), year1 int)

BEGIN

select orderNumber,orderDate,status from orders

where year(orderdate)=year1 and month(orderDate)=month1;

END //

DELIMITER ;

call order\_status(01,2004); /\* Month 01-Jan , Year 2004 \*/

call order\_status(04,2005); /\* Month 04-April , Year 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.

🡪

DELIMITER //

CREATE PROCEDURE cancel()

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 //

DELIMITER ;

call cancel();

select\*from cancellations;

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

🡪

DELIMITER $$

CREATE FUNCTION get\_data(customernumber int) RETURNS varchar(200) CHARSET utf8mb4

BEGIN

declare P\_status varchar(20);

select

case

when sum(amount) < 25000 then 'Silver'

when sum(amount) between 25000 and 50000 then 'Gold'

when sum(amount) > 50000 then 'Platinum'

end as purchasestatus

into P\_status from payments where customerNumber=customerNumber group by customerNumber;

RETURN P\_status;

END $$

DELIMITER ;

select get\_data(customerNumber) as total\_purchase\_status

from customers;

select customerNumber,customerName,get\_data(customerNumber) as total\_purchase\_status

from customers;

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

🡪

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;

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.

🡪

select \*from movies;

select \*from rentals;

/\* Update Trigger \*/

DELIMITER //

Create trigger movren\_trg\_update

AFTER UPDATE on movies for each row

Begin

UPDATE rentals SET rentals.movieid=NEW.id

where rentals.movieid=OLD.id;

END;

//

select \*from movies;

select \*from rentals;

update movies set id=90 where title like 'Marley and me' and category like 'Romance';

select \*from movies;

select \*from rentals;

/\* Delete Trigger \*/

DELIMITER //

Create trigger movren\_trg\_delete

AFTER DELETE on movies for each row

Begin

DELETE from rentals where rentals.movieid=OLD.id;

END;

//

delete from movies where id=90;

select \*from movies;

select \*from rentals;

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

🡪

select \* from(

select fname, salary, dense\_rank()

over(order by salary desc)r from employee)a

where a.r=3;

#IF we have duplicate salary values then rank will skip the next row\_num this might create some data issue while fetching the record

select \* from(

select fname, salary, rank()

over(order by salary desc)r from employee)a

where a.r=3;

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

🡪

select \* from( select \*, dense\_rank() over(order by salary desc)r from employee)a;