**MySQL Exercise 7**

1.

DROP TABLE IF EXISTS Ord\_dtl;

DROP TABLE IF EXISTS Ord\_mst;

DROP TABLE IF EXISTS Prod\_mst;

-- Create Prod\_mst table first

CREATE TABLE Prod\_mst (

Prod\_cd VARCHAR(10) PRIMARY KEY,

Prod\_name VARCHAR(50),

Qty\_in\_stock INT,

Booked\_qty INT

);

-- Create Ord\_mst table

CREATE TABLE Ord\_mst (

Ord\_no INT PRIMARY KEY,

Cust\_cd VARCHAR(10),

Status CHAR(1)

);

-- Create Ord\_dtl table with foreign key references

CREATE TABLE Ord\_dtl (

Ord\_no INT,

Prod\_cd VARCHAR(10),

Qty INT,

PRIMARY KEY (Ord\_no, Prod\_cd),

FOREIGN KEY (Ord\_no) REFERENCES Ord\_mst(Ord\_no),

FOREIGN KEY (Prod\_cd) REFERENCES Prod\_mst(Prod\_cd)

);

-- Insert sample data into Prod\_mst

INSERT INTO Prod\_mst (Prod\_cd, Prod\_name, Qty\_in\_stock, Booked\_qty) VALUES

('P1', 'Floppies', 10000, 1000),

('P2', 'Printers', 5000, 600),

('P3', 'Modems', 3000, 200);

-- Insert sample data into Ord\_mst

INSERT INTO Ord\_mst (Ord\_no, Cust\_cd, Status) VALUES

(1, 'C1', 'P');

-- Insert sample data into Ord\_dtl

INSERT INTO Ord\_dtl (Ord\_no, Prod\_cd, Qty) VALUES

(1, 'P1', 100),

(1, 'P2', 200);

DELIMITER //

CREATE TRIGGER before\_insert\_ord\_dtl

BEFORE INSERT ON Ord\_dtl

FOR EACH ROW

BEGIN

-- Increase the Booked\_qty in Prod\_mst by the quantity being ordered

UPDATE Prod\_mst

SET Booked\_qty = Booked\_qty + NEW.Qty

WHERE Prod\_cd = NEW.Prod\_cd;

END //

DELIMITER ;

2.

DELIMITER //

CREATE TRIGGER before\_delete\_ord\_dtl

BEFORE DELETE ON Ord\_dtl

FOR EACH ROW

BEGIN

-- Decrease the Booked\_qty in Prod\_mst by the quantity being deleted

UPDATE Prod\_mst

SET Booked\_qty = Booked\_qty - OLD.Qty

WHERE Prod\_cd = OLD.Prod\_cd;

END //

DELIMITER ;

3.

DELIMITER //

CREATE TRIGGER before\_update\_ord\_dtl

BEFORE UPDATE ON Ord\_dtl

FOR EACH ROW

BEGIN

-- Check if the Prod\_cd is changing

IF OLD.Prod\_cd != NEW.Prod\_cd THEN

-- Decrease the Booked\_qty for the old Prod\_cd

UPDATE Prod\_mst

SET Booked\_qty = Booked\_qty - OLD.Qty

WHERE Prod\_cd = OLD.Prod\_cd;

-- Increase the Booked\_qty for the new Prod\_cd

UPDATE Prod\_mst

SET Booked\_qty = Booked\_qty + NEW.Qty

WHERE Prod\_cd = NEW.Prod\_cd;

ELSE

-- If only the Qty is changing, adjust the Booked\_qty accordingly

UPDATE Prod\_mst

SET Booked\_qty = Booked\_qty + (NEW.Qty - OLD.Qty)

WHERE Prod\_cd = OLD.Prod\_cd;

END IF;

END //

DELIMITER ;

4.

DELIMITER //

CREATE TRIGGER before\_update\_ord\_mst\_status

BEFORE UPDATE ON Ord\_mst

FOR EACH ROW

BEGIN

-- Check if the Status is changing from P to D (Delivered)

IF OLD.Status = 'P' AND NEW.Status = 'D' THEN

-- Decrease the Booked\_qty and Qty\_in\_stock for each product in Ord\_dtl

DECLARE product\_qty INT;

-- Cursor to loop through all the products associated with this order

DECLARE product\_cursor CURSOR FOR

SELECT Prod\_cd, Qty

FROM Ord\_dtl

WHERE Ord\_no = OLD.Ord\_no;

-- Continue handler to handle the end of the cursor

DECLARE CONTINUE HANDLER FOR NOT FOUND SET product\_qty = NULL;

-- Open cursor to iterate through Ord\_dtl for the current order

OPEN product\_cursor;

read\_loop: LOOP

FETCH product\_cursor INTO product\_cd, product\_qty;

IF product\_qty IS NULL THEN

LEAVE read\_loop;

END IF;

-- Decrease Booked\_qty in Prod\_mst

UPDATE Prod\_mst

SET Booked\_qty = Booked\_qty - product\_qty,

Qty\_in\_stock = Qty\_in\_stock - product\_qty

WHERE Prod\_cd = product\_cd;

END LOOP;

CLOSE product\_cursor;

-- Check if the Status is changing from P to C (Cancelled)

ELSEIF OLD.Status = 'P' AND NEW.Status = 'C' THEN

-- Delete the corresponding order details from Ord\_dtl

DELETE FROM Ord\_dtl WHERE Ord\_no = OLD.Ord\_no;

-- No need for additional update on Booked\_qty here as it will be handled

-- by the Before Delete trigger on Ord\_dtl.

END IF;

END //

DELIMITER ;

DELIMITER //

CREATE TRIGGER before\_update\_ord\_mst\_status

BEFORE UPDATE ON Ord\_mst

FOR EACH ROW

BEGIN

-- Check if the Status is changing from P to D (Delivered)

IF OLD.Status = 'P' AND NEW.Status = 'D' THEN

-- Decrease the Booked\_qty and Qty\_in\_stock for each product in Ord\_dtl

DECLARE product\_cd VARCHAR(10);

DECLARE product\_qty INT;

-- Cursor to loop through all the products associated with this order

DECLARE product\_cursor CURSOR FOR

SELECT Prod\_cd, Qty

FROM Ord\_dtl

WHERE Ord\_no = OLD.Ord\_no;

-- Continue handler to handle the end of the cursor

DECLARE CONTINUE HANDLER FOR NOT FOUND SET product\_qty = NULL;

-- Open cursor to iterate through Ord\_dtl for the current order

OPEN product\_cursor;

read\_loop: LOOP

FETCH product\_cursor INTO product\_cd, product\_qty;

IF product\_qty IS NULL THEN

LEAVE read\_loop;

END IF;

-- Decrease Booked\_qty in Prod\_mst

UPDATE Prod\_mst

SET Booked\_qty = Booked\_qty - product\_qty,

Qty\_in\_stock = Qty\_in\_stock - product\_qty

WHERE Prod\_cd = product\_cd;

END LOOP;

CLOSE product\_cursor;

-- Check if the Status is changing from P to C (Cancelled)

ELSEIF OLD.Status = 'P' AND NEW.Status = 'C' THEN

-- Delete the corresponding order details from Ord\_dtl

DELETE FROM Ord\_dtl WHERE Ord\_no = OLD.Ord\_no;

-- No need for additional update on Booked\_qty here as it will be handled

-- by the Before Delete trigger on Ord\_dtl.

END IF;

END //

DELIMITER ;