

Vietnam National University - Ho Chi Minh City
Ho Chi Minh City University of Technology
Faculty of Computer Science and Engineering



Database Systems

ASSIGNMENT 2

Instructor(s): Ms. Võ Thị Ngọc Châu
Ms. Nguyễn Thị Ái Thảo

Student(s): Lê Quốc Bảo - 2252065 (Leader-Team BK_Shop)
Ngô Phan Khải Tú - 2153951 (Team BK_Shop)
Lê Phan Nhật Minh - 2252478 (Team BK_Shop)
Tiêu Trí Bằng - 2252079 (Team BK_Shop)

Ho Chi Minh City, 10/2024



Contents

1 Database implementation (MySQL)	2
1.1 Create table and sample data	2
1.1.1 Statements to create table	2
1.1.2 Insert data	7
1.2 Implement application	8
1.2.1 Procedures to insert, update and delete data on Product table	8
1.2.2 Triggers control statements INSERT, UPDATE and DELETE data on certain table	10
1.2.3 Procedures contain statements SELECT together with WHERE and/or HAVING	11
1.2.4 Functions having the following requirements	13
1.2.5 The application (web) for our database system	15
1.2.5.a Interface to insert, delete and update data	15
1.2.5.b Interface to display the output of section 1.2.3	17
1.2.5.c Interface to demonstrate procedure in requirement 1.2.3/1.2.4 (pick one)	18

1 Database implementation (MySQL)

1.1 Create table and sample data

1.1.1 Statements to create table

First we create the database and every needed table including primary keys, foreign keys and some data constraints according to our relational database schema:

- Table related to shop's workforce. The Employee table contain primary key ID and the employee's detail. The table Store_Manager contain the ID (foreign key) of the employee who is the manager and will also be deleted if the employee is deleted, the table Warehouse_Manager is similar. The table Warehouse contain the warehouse details with it's address being the primary key and is linked to Warehouse_Manager through foreign key WM_ID, this key will be set to null if the manager is deleted.

```
1 CREATE TABLE BKShop.Employee (  
2     ID INT PRIMARY KEY,  
3     SSN VARCHAR(255) UNIQUE NOT NULL,  
4     Name VARCHAR(255) NOT NULL  
5 );  
6  
7 CREATE TABLE BKShop.Store_Manager (  
8     ID INT PRIMARY KEY,  
9     FOREIGN KEY (ID) REFERENCES Employee(ID) ON DELETE CASCADE ON UPDATE CASCADE  
10 );  
11  
12 CREATE TABLE BKShop.Warehouse_Manager (  
13     ID INT PRIMARY KEY,  
14     FOREIGN KEY (ID) REFERENCES Employee(ID) ON DELETE CASCADE ON UPDATE CASCADE  
15 );  
16  
17 CREATE TABLE BKShop.Warehouse (  
18     Address VARCHAR(255) PRIMARY KEY,  
19     Name VARCHAR(255),  
20     Area DECIMAL(10,2) CHECK (Area > 0),  
21     WM_ID INT,  
22     FOREIGN KEY (WM_ID) REFERENCES Warehouse_Manager(ID) ON DELETE SET NULL ON  
23     UPDATE CASCADE  
24 );
```

- Table related to discount programs that the shop run. Discount_Program table contain the program's ID, it's name and description, begin and end date together with percent off and a CHECK to make sure the discount does not end before it begin. Table Brand contain Name of all the brand that store is selling.

```
1 CREATE TABLE BKShop.Discount_Program (  
2     ID INT PRIMARY KEY,  
3     Name VARCHAR(255) NOT NULL,  
4     Description VARCHAR(255),  
5     FromDate DATE NOT NULL,  
6     ToDate DATE NOT NULL,  
7     Percent INT NOT NULL CHECK (Percent >= 20 AND Percent <= 80),  
8     CHECK (FromDate < ToDate)  
9 );  
10  
11 CREATE TABLE BKShop.Brand (  
12     Name VARCHAR(255) NOT NULL,  
13     Description VARCHAR(255),  
14     BrandImage VARCHAR(255)
```

```
12 Name VARCHAR(255) PRIMARY KEY
13 );
```

- About tables related to payment. First, table Transaction with primary key ID together with attribute for the detail of transaction. Next, table Customer also have primary key ID and attribute for customer's information. Finally, table Payment_Method to store the description of the transaction type. Each transactions has it's transaction type.

```
1 CREATE TABLE BKShop.Payment_Method (
2     Type VARCHAR(255) PRIMARY KEY,
3     Description VARCHAR(255)
4 );
5
6 CREATE TABLE BKShop.Transaction (
7     ID INT PRIMARY KEY,
8     Date DATE NOT NULL,
9     Time TIME NOT NULL,
10    Status VARCHAR(255) NOT NULL,
11    PM_Type VARCHAR(255),
12    Total INT NOT NULL DEFAULT 0 CHECK (Total >= 0),
13    FOREIGN KEY (PM_Type) REFERENCES Payment_Method(Type) ON DELETE SET NULL
14 );
15
16 CREATE TABLE BKShop.Customer (
17     ID INT PRIMARY KEY,
18     SSN VARCHAR(255) UNIQUE NOT NULL,
19     Name VARCHAR(255) NOT NULL,
20     Address VARCHAR(255) NOT NULL,
21     Phone VARCHAR(255) NOT NULL
22 );
```

- How we manage our product is there will be one big table named Product that store every product of our business and additional tables to store the product's detail linked over the Product's ID (primary key). Each products also linked with it's brand and the warehouse stores it, product will be deleted if one of those is deleted.

```
1 CREATE TABLE BKShop.Product (
2     ID INT PRIMARY KEY,
3     Name VARCHAR(255) NOT NULL,
4     Image VARCHAR(255) NOT NULL,
5     Price INT NOT NULL CHECK (Price >= 0),
6     M_Date DATE NOT NULL,
7     E_Date DATE NOT NULL,
8     Status VARCHAR(255) NOT NULL,
9     B_Name VARCHAR(255) NOT NULL,
10    W_Addr VARCHAR(255),
11    C_ID INT DEFAULT NULL,
12    T_ID INT DEFAULT NULL,
13    FOREIGN KEY (B_Name) REFERENCES Brand(Name) ON UPDATE CASCADE ON DELETE CASCADE
14    ,
15    FOREIGN KEY (W_Addr) REFERENCES Warehouse(Address) ON UPDATE CASCADE ON DELETE SET NULL,
16    FOREIGN KEY (C_ID) REFERENCES Customer(ID) ON UPDATE CASCADE ON DELETE SET NULL
17    ,
18    FOREIGN KEY (T_ID) REFERENCES Transaction(ID) ON UPDATE CASCADE ON DELETE SET NULL,
19    CHECK (M_Date < E_Date)
20 );
```

- These are tables that contain product's detail, support for the table Product. The products are split into two main category, Laptop and Electronic_Accessories. There are tables Mouse, Keyboard, Headphone carry additional data to support table Electronic_Accessories. All of them is connected via product's ID and the information will be deleted if the product it self is deleted.

```
1 CREATE TABLE BKShop.Laptop (  
2     ID INT PRIMARY KEY,  
3     RAM INT NOT NULL,  
4     CPU VARCHAR(255) NOT NULL,  
5     Graphic_Card VARCHAR(255) NOT NULL,  
6     Purpose VARCHAR(255),  
7     FOREIGN KEY (ID) REFERENCES Product(ID) ON UPDATE CASCADE ON DELETE CASCADE  
8 );  
9  
10 CREATE TABLE BKShop.Electronic_Accessories  
11 (  
12     ID INT PRIMARY KEY,  
13     Connection VARCHAR(255) NOT NULL,  
14     FOREIGN KEY (ID) REFERENCES Product(ID) ON UPDATE CASCADE ON DELETE CASCADE  
15 );  
16  
17 CREATE TABLE BKShop.Mouse (  
18     ID INT PRIMARY KEY,  
19     LED_Color VARCHAR(255) NOT NULL,  
20     DPI INT NOT NULL,  
21     FOREIGN KEY (ID) REFERENCES Electronic_Accessories(ID) ON UPDATE CASCADE ON  
22     DELETE CASCADE  
23 );  
24  
25 CREATE TABLE BKShop.Keyboard (  
26     ID INT PRIMARY KEY,  
27     Switch_Type VARCHAR(255) NOT NULL,  
28     Layout VARCHAR(255) NOT NULL,  
29     FOREIGN KEY (ID) REFERENCES Electronic_Accessories(ID) ON UPDATE CASCADE ON  
30     DELETE CASCADE  
31 );  
32  
33 CREATE TABLE BKShop.Headphone (  
34     ID INT PRIMARY KEY,  
35     Type VARCHAR(255) NOT NULL,  
36     FOREIGN KEY (ID) REFERENCES Electronic_Accessories(ID) ON UPDATE CASCADE ON  
37     DELETE CASCADE  
38 );
```

- Table to store all the requests if the customer submit. The request can be either a feedback (compliment on product or service) or a request (request to perform maintenance on or to replace product). Each requests links to the customer that submit and the store manager that process the request, it will be deleted if one of those is deleted.

```
1 CREATE TABLE BKShop.Request (  
2     No INT PRIMARY KEY,  
3     Requirement VARCHAR(255),  
4     Feedback VARCHAR(255),  
5     Date DATE NOT NULL,  
6     C_ID INT NOT NULL,  
7     P_ID INT NOT NULL,  
8     SM_ID INT,  
9     FOREIGN KEY (C_ID) REFERENCES Customer(ID) ON UPDATE CASCADE ON DELETE CASCADE,
```

```

10 FOREIGN KEY (P_ID) REFERENCES Product(ID) ON UPDATE CASCADE ON DELETE CASCADE,
11 FOREIGN KEY (SM_ID) REFERENCES Store_Manager(ID) ON UPDATE CASCADE ON DELETE
    SET NULL
12 );

```

- Tables to handle the application of discount program, include table applies and has. Applies table to store information about what discount is applied to what product. Has table to know if a brand is running some discount program for their product. We make these so that the store can make discount for each product independently and brands can run discount program for all of their products.

```

1 CREATE TABLE BKShop.applies (
2     P_ID INT NOT NULL,
3     D_ID INT NOT NULL,
4     Saving INT NOT NULL DEFAULT 0 CHECK (Saving >= 0),
5     PRIMARY KEY (P_ID, D_ID),
6     FOREIGN KEY (P_ID) REFERENCES Product(ID) ON UPDATE CASCADE ON DELETE CASCADE,
7     FOREIGN KEY (D_ID) REFERENCES Discount_Program(ID) ON UPDATE CASCADE ON DELETE
    CASCADE
8 );
9
10 CREATE TABLE BKShop.has (
11     B_Name VARCHAR(255) NOT NULL,
12     D_ID INT NOT NULL,
13     PRIMARY KEY (B_Name, D_ID),
14     FOREIGN KEY (B_Name) REFERENCES Brand(Name) ON UPDATE CASCADE ON DELETE CASCADE
    ,
15     FOREIGN KEY (D_ID) REFERENCES Discount_Program(ID) ON UPDATE CASCADE ON DELETE
    CASCADE
16 );

```

Then we create Triggers to further guard the designed constrain:

- An employee cannot be store manager and warehouse manager at the same time, business policy. Before insert into one of the tables Store_Manager and Warehouse_Manager, the ID will be check to make sure it is not appears on the other table already. Error message " Employee cannot be both a Store Manager and a Warehouse Manager ." will appear if this happen.

```

1 DELIMITER $$
2 CREATE TRIGGER BKShop.Disjoint_SM BEFORE INSERT ON BKShop.Store_Manager
3 FOR EACH ROW
4 BEGIN
5     IF EXISTS (SELECT 1 FROM BKShop.Warehouse_Manager WHERE ID = New.ID) THEN
6         SIGNAL SQLSTATE '45000'
7         SET MESSAGE_TEXT = 'Employee cannot be both a Store Manager and a Warehouse
            Manager.';
8     END IF;
9 END$$
10 DELIMITER ;
11
12 DELIMITER $$
13 CREATE TRIGGER BKShop.Disjoint_WM BEFORE INSERT ON BKShop.Warehouse_Manager
14 FOR EACH ROW
15 BEGIN
16     IF EXISTS (SELECT 1 FROM BKShop.Store_Manager WHERE ID = New.ID) THEN
17         SIGNAL SQLSTATE '45000'
18         SET MESSAGE_TEXT = 'Employee cannot be both a Store Manager and a Warehouse
            Manager.';

```

```
19 END IF;  
20 END$$  
21 DELIMITER ;
```

- According to our policy, if a product's status is 2nd (second handed) we will sell it at 95% price. When every a product is added, this trigger will check for the status and calculate that new price for us.

```
1 DELIMITER $$  
2 CREATE TRIGGER BKShop.Product_2Hand BEFORE INSERT ON BKShop.Product  
3 FOR EACH ROW  
4 BEGIN  
5     IF NEW.Status = '2nd' THEN  
6         SET NEW.Price = ROUND(NEW.Price*0.95, 0);  
7     END IF;  
8 END$$  
9 DELIMITER $$
```

- By design, a product's detail can only be store in one table of it's category, this trigger is to help protect that by checking if the product's id exited in any other tables than that it's about to be inserted.

```
1 DELIMITER $$  
2 CREATE TRIGGER BKShop.Disjoint_Laptop BEFORE INSERT ON BKShop.Laptop  
3 FOR EACH ROW  
4 BEGIN  
5     IF EXISTS (SELECT 1 FROM Mouse WHERE ID = New.ID)  
6         OR EXISTS (SELECT 1 FROM Keyboard WHERE ID = New.ID)  
7         OR EXISTS (SELECT 1 FROM Headphone WHERE ID = New.ID) THEN  
8         SIGNAL SQLSTATE '45000'  
9         SET MESSAGE_TEXT = 'A Product can only belong to one type category.';  
10    END IF;  
11 END$$  
12 DELIMITER ;  
13  
14 DELIMITER $$  
15 CREATE TRIGGER BKShop.Disjoint_Mouse BEFORE INSERT ON BKShop.Mouse  
16 FOR EACH ROW  
17 BEGIN  
18     IF EXISTS (SELECT 1 FROM Laptop WHERE ID = New.ID)  
19         OR EXISTS (SELECT 1 FROM Keyboard WHERE ID = New.ID)  
20         OR EXISTS (SELECT 1 FROM Headphone WHERE ID = New.ID) THEN  
21         SIGNAL SQLSTATE '45000'  
22         SET MESSAGE_TEXT = 'A Product can only belong to one type category.';  
23    END IF;  
24 END$$  
25 DELIMITER ;  
26  
27 DELIMITER $$  
28 CREATE TRIGGER BKShop.Disjoint_Keyboard BEFORE INSERT ON BKShop.Keyboard  
29 FOR EACH ROW  
30 BEGIN  
31     IF EXISTS (SELECT 1 FROM Laptop WHERE ID = New.ID)  
32         OR EXISTS (SELECT 1 FROM Mouse WHERE ID = New.ID)  
33         OR EXISTS (SELECT 1 FROM Headphone WHERE ID = New.ID) THEN  
34         SIGNAL SQLSTATE '45000'  
35         SET MESSAGE_TEXT = 'A Product can only belong to one type category.';  
36    END IF;
```

```

37 END$$
38 DELIMITER ;
39
40 DELIMITER $$
41 CREATE TRIGGER BKShop.Disjoint_Headphone BEFORE INSERT ON BKShop.Headphone
42 FOR EACH ROW
43 BEGIN
44     IF EXISTS (SELECT 1 FROM Laptop WHERE ID = New.ID)
45         OR EXISTS (SELECT 1 FROM Mouse WHERE ID = New.ID)
46         OR EXISTS (SELECT 1 FROM Keyboard WHERE ID = New.ID) THEN
47         SIGNAL SQLSTATE '45000'
48         SET MESSAGE_TEXT = 'A Product can only belong to one type category.';
49     END IF;
50 END$$
51 DELIMITER ;

```

1.1.2 Insert data

We insert some sample data into every table to test the functionality of the database. The sample data was created using excel and then converted to csv file to put in the code. For report's presentation, we will only give some example but in sort they follow the designed schema:

- Input Employee and Customer.

```

1 INSERT INTO BKShop.Employee (ID, SSN, Name) VALUES
2 (1, '1372', 'V Song Anh'),
3 (2, '1831', 'H C ng Gia'),
4 (3, '1019', 'N guyn c Gia'),
5 (4, '1297', 'L Q u c B nh'),
6 -----etc-----
7
8 INSERT INTO BKShop.Customer (ID, SSN, Name, Address, Phone) VALUES
9 (1, '1868', 'C t Nh Khang', '257 . Tr n Quang K h i , Ph Íng T n nh
, Q u n 1, H Ch Minh, V i t Nam', '0617658325'),
10 (2, '1992', 'Kh u Minh Ti ín ', 'CC 16 Tr Íng S n , Ph Íng 15, Q u n 10,
H Ch Minh, V i t Nam', '0840848353'),
11 (3, '1621', 'T u n Ng c H i n ', 'Y1B, H ng L nh , Ph Íng 15, Q u n 10,
H Ch Minh, V i t Nam', '0378662878'),
12 (4, '1867', 'N g c H i ng ', '334 . T H i ín Th nh , Ph Íng 14, Q u n
10, H Ch Minh 700000, V i t Nam', '0714400215'),
13 -----etc-----

```

- Input the products and their supportive details into their designed table.

```

1 INSERT INTO BKShop.Product (ID, Name, Price, M_Date, E_Date, Status, B_Name, W_Addr
, Image) VALUES
2 (1001, 'Laptop gaming Lenovo Legion 5 16IRX9 83DG0051VN', 42990000, '2024-11-15', '
2026-11-15', 'New', 'Lenovo', '268 L í Th Íng K i t ', '//product.hstatic.net
/200000722513/product/ava_48385ba1307849189dd774c9d489ddef_grande.png'),
3 (1031, 'Chut Pulsar X2 Red', 2490000, '2023-07-13', '2025-07-12', 'New', 'Pulsar
', '268 L í Th Íng K i t ', '//product.hstatic.net/200000722513/product/ezgif
-1-50
b65a0ec7_50e93b9ed6ae4a5bb444389471be493b_master_5308595cba9c4e6c89d8e532cf1aea22_grande
.png'),
4 (1042, 'B n ph m c AKK0 3098N Multi-modes Blue On White TTC Flame Red',
2290000, '2024-10-25', '2026-10-25', 'New', 'Akko', '268 L í Th Íng K i t ',
 '//product.hstatic.net/200000722513/product/
phim_9a02ee4b8aef40cfa7d79511fb029d37_7a1d7923d6e947ee82ee6b1c749b8a8d_da58e57f1a004b5b926a324
.png'),

```



```

5 (1051, 'Tai nghe Asus ROG Cetra II Core Moonlight', 1090000, '2023-05-17', '
    2025-05-16', 'New', 'Asus', '268 L í T h í n g K i t ', '//product.hstatic.net
    /200000722513/product/10249
    _rog-cetra-ii-core-moonlight-white_2_33df49bdb12244509291cae8c0ecf5a6_medium.jpg
    '),
6 -----etc-----
7
8 INSERT INTO BKShop.Laptop (ID, RAM, CPU, Graphic_Card, Purpose) VALUES
9 (1001, 16, 'Core i7', 'NVIDIA 4060', 'Gaming'),
10 (1002, 16, 'Core ultra', 'Intel Arc', 'Gaming'),
11 (1003, 16, 'Core i7', 'NVIDIA 4060', 'Gaming'),
12 (1004, 16, 'Core ultra', 'Intel Arc', 'Gaming'),
13 -----etc-----
14
15 INSERT INTO BKShop.Electronic_Accessories (Connection, ID) VALUES
16 ('Both', 1031),
17 ('Wireless', 1032),
18 ('Wired', 1033),
19 ('Wireless', 1034),
20 -----etc-----
21
22 INSERT INTO BKShop.Mouse (ID, LED_Color, DPI) VALUES
23 (1031, 'No', 26000),
24 (1032, 'RGB', 2400),
25 (1033, 'No', 2400),
26 (1034, 'No', 44000),
27 -----etc-----
28
29 INSERT INTO BKShop.Keyboard (ID, Switch_Type, Layout) VALUES
30 (1042, 'Linear', 'Fullsize'),
31 (1044, 'Linear', 'Tenkeyless'),
32 (1045, 'Tactile', 'Fullsize'),
33 (1046, 'Linear', 'Fullsize'),
34 -----etc-----
35
36 INSERT INTO BKShop.Headphone (Type, ID) VALUES
37 ('In-ear', 1051),
38 ('Earmuffs', 1052),
39 ('Earmuffs', 1053),
40 ('Earmuffs', 1054),
41 -----etc-----

```

- Input into the has table, indicate what brand have what discount program.

```

1 INSERT INTO BKShop.has (B_Name, D_ID) VALUES
2 ('Acer', 1),
3 ('Akko', 1),
4 ('Asus', 1),
5 ('Corsair', 2),
6 -----etc-----

```

1.2 Implement application

1.2.1 Procedures to insert, update and delete data on Product table

There are many procedure that can insert, update and delete on the tables of our database but for demonstration we choose the table Product.

- Procedure to insert a new laptop to the table product. The procedure take in all the detail of the laptop that we want to add and then insert them into the table Product store the product is self and the table Laptop store the details that come with that product.

```

1 DELIMITER $$
2 CREATE PROCEDURE BKShop.AddLaptop(
3     IN p_ID INT, IN p_Name VARCHAR(255), IN p_Image VARCHAR(255),
4     IN p_Price INT, IN p_M_Date DATE, IN p_E_Date DATE, IN p_Status VARCHAR(255),
5     IN p_B_Name VARCHAR(255), IN p_W_Addr VARCHAR(255), IN p_C_ID INT, IN p_T_ID
6     INT,
7     IN p_RAM INT, IN p_CPU VARCHAR(255), IN p_Graphic_Card VARCHAR(255), IN
8     p_Purpose VARCHAR(255)
9 )
10 BEGIN
11     INSERT INTO BKShop.Product (ID, Name, Image, Price, M_Date, E_Date, Status,
12     B_Name, W_Addr, C_ID, T_ID)
13     VALUES (p_ID, p_Name, p_Image, p_Price, p_M_Date, p_E_Date, p_Status, p_B_Name,
14     p_W_Addr, p_C_ID, p_T_ID);
15
16     INSERT INTO BKShop.Laptop (ID, RAM, CPU, Graphic_Card, Purpose)
17     VALUES (p_ID, p_RAM, p_CPU, p_Graphic_Card, p_Purpose);
18 END$$
19 DELIMITER ;

```

- Procedure to perform an update on the price of a product on the table product. This procedure take in the id of the product and the new price, then look for the product that have the id and update it's price. After that check the status and apply 5% off if is a 2nd product, then delete if existed the discount information for the old price of the product and check if there are one that currently running can be applied for the product.

```

1 DELIMITER $$
2 CREATE PROCEDURE BKShop.UpdatePrice(IN ProductID INT, IN NewPrice INT)
3 BEGIN
4     DECLARE ProductStatus VARCHAR(255);
5     DECLARE DiscountID INT;
6     DECLARE DiscountPercent INT;
7     DECLARE Done INT DEFAULT 0;
8
9     DECLARE DiscountCursor CURSOR FOR
10         SELECT D.ID, D.Percent
11         FROM BKShop.Discount_Program D JOIN BKShop.Product P ON CURRENT_DATE
12         BETWEEN D.FromDate AND D.ToDate
13         WHERE P.ID = ProductID;
14     DECLARE CONTINUE HANDLER FOR NOT FOUND SET Done = 1;
15
16     SELECT Status INTO ProductStatus FROM BKShop.Product WHERE ID = ProductID;
17     UPDATE BKShop.Product SET Price = NewPrice WHERE ID = ProductID;
18
19     IF ProductStatus = '2nd' THEN
20         UPDATE BKShop.Product SET Price = ROUND(Price*0.95, 2) WHERE ID = ProductID;
21     END IF;
22
23     DELETE FROM BKShop.applies WHERE P_ID = ProductID;
24     OPEN DiscountCursor;
25     DiscountLoop: LOOP
26         FETCH DiscountCursor INTO DiscountID, DiscountPercent;
27         IF Done = 1 THEN

```

```

27         LEAVE DiscountLoop;
28     END IF;
29
30     INSERT INTO BKShop.applies (P_ID, D_ID, Saving)
31     VALUES (ProductID, DiscountID, ROUND(NewPrice*DiscountPercent/100, 2));
32     SET NewPrice = NewPrice - ROUND(NewPrice*DiscountPercent/100, 2);
33 END LOOP;
34 CLOSE DiscountCursor;
35
36 UPDATE BKShop.Product P
37 SET P.Price = P.Price - (SELECT SUM(Saving) FROM BKShop.applies A WHERE A.P_ID
38 = ProductID)
39 WHERE P.ID = ProductID;
40 END$$$
41 DELIMITER ;

```

- Procedure to delete a product given it's ID, the related product information on other table is already handled using CASCADE, the related customer and transaction information if exist and are not appear on any other product are deleted. The procedure will let you know if it does not find the id

```

1 DELIMITER $$
2 CREATE PROCEDURE BKShop.DeleteProduct (IN ProductID INT)
3 BEGIN
4 IF EXISTS (SELECT FROM BKShop.Product WHERE ID = ProductID) THEN
5     DECLARE TransactionID INT;
6     DECLARE CustomerID INT;
7     IF EXISTS (SELECT C_ID, T_ID INTO CustomerID, TransactionID FROM BKShop.Product
8     WHERE ID = ProductID) THEN
9         IF NOT EXISTS (SELECT C_ID FROM BKShop.Product WHERE NOT ID = ProductID)
10        THEN DELETE FROM BKShop.Customer WHERE C_ID = CustomerID;
11        IF NOT EXISTS (SELECT T_ID FROM BKShop.Product WHERE NOT ID = ProductID)
12        THEN DELETE FROM BKShop.Customer WHERE T_ID = TransactionID;
13    END IF;
14    DELETE FROM BKShop.Product WHERE ID = ProductID;
15 ELSE
16     SIGNAL SQLSTATE '45000'
17     SET MESSAGE_TEXT = 'ID not found';
18 END IF;
19 END$$$
20 DELIMITER ;

```

1.2.2 Triggers control statements INSERT, UPDATE and DELETE data on certain table

Our database have a lot of triggers to help with the control of the data flows. For demonstration purpose we will give two trigger.

- Trigger auto update the data when a customer is deleted. Our design is that when a customer enter the shop, that customer's information will be add into the table customer and a new transaction with status is 'pending' will be added to table transaction. When customer add product to their card the product status will be update to 'ordering' so it cannot be sold to other customer in the mean time. If the customer just leave and buy nothing, they will be deleted off the customer table and this trigger is to change the table transaction and the product's status back to it's initial stage. The trigger belong to the table customer but perform update on table transaction and table product.

```

1 DELIMITER $$
2 CREATE TRIGGER BKShop.Delete_Customer BEFORE DELETE ON BKShop.Customer
3 FOR EACH ROW
4 BEGIN
5     UPDATE BKShop.Product P JOIN BKShop.Transaction T ON T.ID = P.T_ID
6     SET P.Status = CASE
7         WHEN P.Status = 'New_Ordering' THEN 'New'
8         WHEN P.Status = '2nd_Ordering' THEN '2nd'
9         ELSE P.Status
10    END
11    WHERE P.C_ID = OLD.ID AND T.Status = 'Pending';
12
13    DELETE FROM BKShop.Transaction WHERE ID = OLD.ID AND Status = 'Pending';
14 END $$
15 DELIMITER ;
16
17 DELIMITER $$
18 CREATE TRIGGER BKShop.Create_Transaction AFTER INSERT ON BKShop.Customer
19 FOR EACH ROW
20 BEGIN
21     INSERT INTO BKShop.Transaction (ID, Date, Time, Status)
22     VALUES (New.ID, CURRENT_DATE, CURRENT_TIME, 'Pending');
23 END $$
24 DELIMITER ;

```

- Trigger auto update the product price when a new discount program is active. When a new discount is running, a new tuple will be added into table has and activate the trigger. The trigger will calculate the saving amount base on the discount percent for each product and then add it into the table applies together with the discount information, the trigger will only perform this with product that is not sold or already in somebody card. The trigger belong to the table has but perform update on table applies.

```

1 DELIMITER $$
2 CREATE TRIGGER BKShop.applies_discount AFTER INSERT ON BKShop.has
3 FOR EACH ROW
4 BEGIN
5     INSERT INTO BKShop.applies(P_ID, D_ID, Saving)
6     SELECT P.ID, NEW.D_ID, ROUND((P.Price*D.Percent/100), 2) AS Saving
7     FROM BKShop.Product P JOIN BKShop.Discount_Program D ON D.ID = NEW.D_ID
8     WHERE P.B_Name = NEW.B_Name AND CURRENT_DATE BETWEEN D.FromDate AND D.ToDate
9     AND (P.Status = 'New' OR P.Status = '2nd');
10
11    UPDATE BKShop.Product P JOIN BKShop.Discount_Program D ON D.ID = NEW.D_ID
12    SET P.Price = P.Price - ROUND((P.Price*D.Percent/100), 2)
13    WHERE P.B_Name = NEW.B_Name AND CURRENT_DATE BETWEEN D.FromDate AND D.ToDate
14    AND (P.Status = 'New' OR P.Status = '2nd');
15 END $$
16 DELIMITER $$

```

1.2.3 Procedures contain statements SELECT together with WHERE and/or HAVING

Here is our two procedures that meet the requirement:

- Procedure to search for laptops that meet given specifications. The procedure take in the specs of the laptop and return all the laptop that have the specs that we want and order them by ascending price. The procedure retrieving data from table product (relate to requirement 1.2.1)

```

1 DELIMITER $$
2 CREATE PROCEDURE BKShop.SearchLaptop(
3     IN SearchRAM INT,
4     IN SearchCPU VARCHAR(255),
5     IN SearchGPU VARCHAR(255),
6     IN SearchPurpose VARCHAR(255),
7     IN SearchPrice INT
8 )
9 BEGIN
10     SELECT * FROM BKShop.Product P JOIN BKShop.Laptop L ON P.ID = L.ID
11     WHERE
12         (L.RAM = SearchRAM OR SearchRAM IS NULL)
13         AND (L.CPU = SearchCPU OR SearchCPU IS NULL)
14         AND (L.Graphic_Card = SearchGPU OR SearchGPU IS NULL)
15         AND (L.Purpose = SearchPurpose OR SearchPurpose IS NULL)
16         AND (P.Price <= SearchPrice OR SearchPrice IS NULL)
17     ORDER BY P.Price ASC;
18 END$$
19 DELIMITER ;

```

- This procedure is used when a customer put a product into their card. When this happen the procedure will select the product and update it's status to 'ordering' and put the customer id and transaction id into the product detail, after that it select the price and update the transaction total. The procedure will let you know if it does not find the product by an error message.

```

1 DELIMITER $$
2 CREATE PROCEDURE BKShop.Put2Transaction(IN ProductID INT, IN TransactionID INT)
3 BEGIN
4     DECLARE NewProductID INT;
5     DECLARE ProductName VARCHAR(255);
6     DECLARE ProductPrice INT;
7     DECLARE ProductStatus VARCHAR(255);
8     SELECT Name, Price, Status INTO ProductName, ProductPrice, ProductStatus FROM
9     BKShop.Product WHERE ID = ProductID;
10
11     IF EXISTS (SELECT 1 FROM BKShop.Product WHERE ID = ProductID AND (Status = '2nd
12     ' OR Status = 'New') AND T_ID IS NULL) THEN
13         UPDATE BKShop.Product
14         SET
15             C_ID = TransactionID, T_ID = TransactionID,
16             Status = CASE
17                 WHEN Status = 'New' THEN 'New_Ordering'
18                 WHEN Status = '2nd' THEN '2nd_Ordering'
19             END
20         WHERE ID = ProductID;
21         UPDATE BKShop.Transaction SET Total = Total + ProductPrice WHERE ID =
22         TransactionID;
23     ELSE
24         SELECT ID, Price, Status INTO NewProductID, ProductPrice, ProductStatus
25         FROM BKShop.Product WHERE Name = ProductName AND (Status = '2nd' OR Status
26         = 'New') AND T_ID IS NULL
27         LIMIT 1;
28
29         IF NewProductID IS NOT NULL THEN
30             UPDATE BKShop.Product
31             SET
32                 C_ID = TransactionID, T_ID = TransactionID,
33                 Status = CASE

```

```

30         WHEN Status = 'New' THEN 'New_Ordering'
31         WHEN Status = '2nd' THEN '2nd_Ordering'
32     END
33     WHERE ID = NewProductID;
34     UPDATE BKShop.Transaction SET Total = Total + ProductPrice WHERE ID =
TransactionID;
35     ELSE
36         SIGNAL SQLSTATE '45000'
37         SET MESSAGE_TEXT = 'No valid product available.';
38     END IF;
39 END IF;
40 END$$
41 DELIMITER ;

```

1.2.4 Functions having the following requirements

- Function to check the availability of a product, this function take in the product name and return if found one of that product which haven't been sold or currently in any customer card or not.

```

1 DELIMITER $$
2 CREATE FUNCTION BKShop.IsAvailable(p_Name VARCHAR(255))
3 RETURNS BOOLEAN
4 DETERMINISTIC
5 BEGIN
6     DECLARE isAvailable BOOLEAN DEFAULT FALSE;
7     IF EXISTS (SELECT 1 FROM BKShop.Product WHERE Name = p_Name AND (Status = 'New'
OR Status = '2nd')) THEN
8         SET isAvailable = TRUE;
9     ELSE
10        SET isAvailable = FALSE;
11    END IF;
12    RETURN isAvailable;
13 END$$
14 DELIMITER ;

```

- Procedure to perform an update on the price of a product on the table product. This procedure take in the id of the product and the new price, then look for the product that have the id and update it's price. After that check the status and apply 5% off if is a 2nd product, then delete if existed the discount information for the old price of the product and check if there are one that currently running can be applied for the product.

```

1 DELIMITER $$
2 CREATE PROCEDURE BKShop.UpdatePrice(IN ProductID INT, IN NewPrice INT)
3 BEGIN
4     DECLARE ProductStatus VARCHAR(255);
5     DECLARE DiscountID INT;
6     DECLARE DiscountPercent INT;
7     DECLARE Done INT DEFAULT 0;
8
9     DECLARE DiscountCursor CURSOR FOR
10        SELECT D.ID, D.Percent
11        FROM BKShop.Discount_Program D JOIN BKShop.Product P ON CURRENT_DATE
BETWEEN D.FromDate AND D.ToDate
12        WHERE P.ID = ProductID;
13     DECLARE CONTINUE HANDLER FOR NOT FOUND SET Done = 1;
14
15     SELECT Status INTO ProductStatus FROM BKShop.Product WHERE ID = ProductID;

```

```
16 UPDATE BKShop.Product SET Price = NewPrice WHERE ID = ProductID;
17
18 IF ProductStatus = '2nd' THEN
19     UPDATE BKShop.Product SET Price = ROUND(Price*0.95, 2) WHERE ID = ProductID
20 ;
21 END IF;
22
23 DELETE FROM BKShop.applies WHERE P_ID = ProductID;
24 OPEN DiscountCursor;
25 DiscountLoop: LOOP
26     FETCH DiscountCursor INTO DiscountID, DiscountPercent;
27     IF Done = 1 THEN
28         LEAVE DiscountLoop;
29     END IF;
30
31     INSERT INTO BKShop.applies (P_ID, D_ID, Saving)
32     VALUES (ProductID, DiscountID, ROUND(NewPrice*DiscountPercent/100, 2));
33     SET NewPrice = NewPrice - ROUND(NewPrice*DiscountPercent/100, 2);
34 END LOOP;
35 CLOSE DiscountCursor;
36
37 UPDATE BKShop.Product P
38 SET P.Price = P.Price - (SELECT SUM(Saving) FROM BKShop.applies A WHERE A.P_ID
39 = ProductID)
40 WHERE P.ID = ProductID;
41 END$$$
42 DELIMITER ;
```

1.2.5 The application (web) for our database system

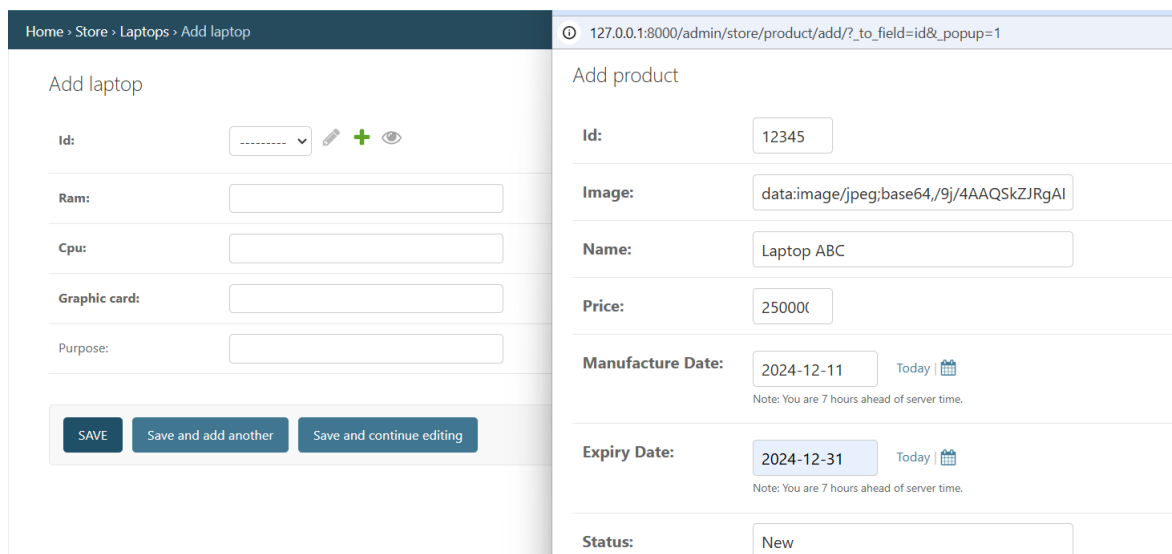
We created a web based application for the business. Here customer can come and query/search for the product that fit their need, see the price, see the price after discount, add product to card and confirm to buy the product in card. Here we can also come and manage, interact with the database, further will be demonstrate below.

1.2.5.a Interface to insert, delete and update data

Our application have the interface to perform insert, delete and update on the table product. To demonstrate we will give some instance with picture below.

Setting up a new laptop for selling:

- Filling up all the laptop's information.



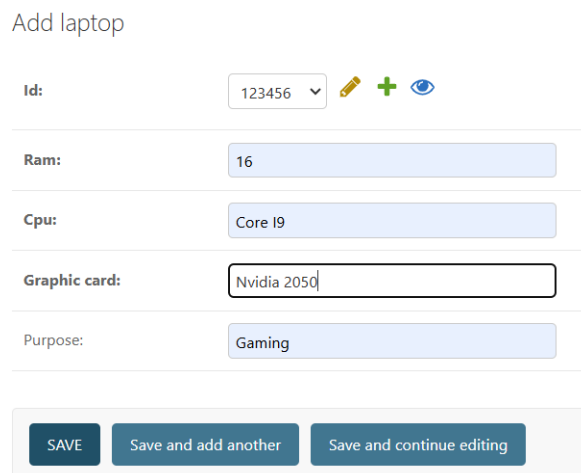
The screenshot shows a web application interface for adding a new laptop. The breadcrumb trail is 'Home > Store > Laptops > Add laptop'. The URL bar shows '127.0.0.1:8000/admin/store/product/add/?_to_field=id&_popup=1'. The form is titled 'Add product' and contains the following fields:

- Id:** A dropdown menu with a value of '12345'.
- Image:** A text input field containing 'data:image/jpeg;base64,/9j/4AAQSkZJRgAI'.
- Name:** A text input field containing 'Laptop ABC'.
- Price:** A text input field containing '25000'.
- Manufacture Date:** A date picker showing '2024-12-11' with a 'Today' button and a calendar icon. A note below says 'Note: You are 7 hours ahead of server time.'
- Expiry Date:** A date picker showing '2024-12-31' with a 'Today' button and a calendar icon. A note below says 'Note: You are 7 hours ahead of server time.'
- Status:** A text input field containing 'New'.

At the bottom of the form, there are three buttons: 'SAVE', 'Save and add another', and 'Save and continue editing'.

Figure 1: Add a new laptop into product list

- Filling it's detail



The screenshot shows a web application interface for adding a new laptop. The breadcrumb trail is 'Add laptop'. The form contains the following fields:

- Id:** A dropdown menu with a value of '123456'.
- Ram:** A text input field containing '16'.
- Cpu:** A text input field containing 'Core i9'.
- Graphic card:** A text input field containing 'Nvidia 2050'.
- Purpose:** A text input field containing 'Gaming'.

At the bottom of the form, there are three buttons: 'SAVE', 'Save and add another', and 'Save and continue editing'.

Figure 2: Add the laptop's specs

- After that, the laptop will appear on the interface and in the database and ready to be bought by customer.

Select laptop to change

Action: 0 of 91 selected

<input type="checkbox"/>	LAPTOP
<input type="checkbox"/>	123456 - Laptop ABC
<input type="checkbox"/>	1150 - Laptop gaming ASUS TUF Gaming FA401WV RG062WS
<input type="checkbox"/>	1149 - Laptop gaming Gigabyte AORUS 5 SE4 73VN313SH
<input type="checkbox"/>	1148 - Laptop Dell Inspiron T7430 N7430I58W1 Silver
<input type="checkbox"/>	1147 - Laptop gaming Gigabyte G6 KF H3VN853SH
<input type="checkbox"/>	1146 - Laptop gaming Acer Predator Helios 300 PH315 55 751D
<input type="checkbox"/>	1145 - Laptop gaming Acer Nitro V ANV15 41 R7AP

Figure 3: New laptop appear on interface

Update the laptop's information:

- Change the information to anything you want and click submit to save the changes

Change laptop

1147 - Laptop gaming Gigabyte G6 KF H3VN853SH

Id:

Ram:

Cpu:

Graphic card:

Purpose:

Image:

Name:

Price:

Manufacture Date:
Note: You are 7 hours ahead of server time.

Expiry Date:
Note: You are 7 hours ahead of server time.

Status:

B name:

W addr:

Figure 4: Laptop information update

Delete a product:

- On the manage page, We can delete any product we want just by click the delete icon, a pop up will appear to ask for confirmation.

Are you sure?

Are you sure you want to delete the selected products? All of the following objects and their related items will be deleted:

Summary

- Products: 2

Objects

- Product: 123456
- Product: 1150

Figure 5: Product delete confirm?

- After the product have been deleted.

Select laptop to change

Action: 0 of 89 selected

<input type="checkbox"/>	LAPTOP
<input type="checkbox"/>	1149 - Laptop gaming Gigabyte AORUS 5 SE4 73VN313SH
<input type="checkbox"/>	1148 - Laptop Dell Inspiron T7430 N7430I58W1 Silver
<input type="checkbox"/>	1147 - Laptop gaming Gigabyte G6 KF H3VN853SH
<input type="checkbox"/>	1146 - Laptop gaming Acer Predator Helios 300 PH315 55 751D
<input type="checkbox"/>	1145 - Laptop gaming Acer Nitro V ANV15 41 R7AP
<input type="checkbox"/>	1144 - Laptop gaming Acer Nitro V ANV16 41 R6NA
<input type="checkbox"/>	1143 - Laptop gaming Acer Predator Helios Neo 14 PHN14 51 96HG
<input type="checkbox"/>	1142 - Laptop Dell Inspiron 15 3530 71011775


Figure 6: Product list after delete

1.2.5.b Interface to display the output of section 1.2.3


In requirement 1.2.3 there are procedure to search for laptop base on their information, here is the interface on the app to perform that. Based on the customer needs and preference, they can choose the brand they like, the laptop's specs needed for their work, customer can also choose to buy new or pick a second handed one for the discount and finally choose to sort buy price (ascending or descending). Note that the same can be do for other product type (keyboard, mouse, headphone).

Brand: Gigabyte
CPU: Core i5
RAM: 8
Graphic Card:
Purpose: Gaming
Price Range: -
Product Status: New
Sort Price by order: Ascending


Laptop



Laptop gaming Gigabyte G5 MF F2VN3335H
Gigabyte
 9995000đ
RAM: 8 GB
CPU: Core i5
Graphic Card: NVIDIA 4050
Purpose: Gaming



Laptop gaming Gigabyte G5 MF F2VN3335H
Gigabyte
 9995000đ
RAM: 8 GB
CPU: Core i5
Graphic Card: NVIDIA 4050
Purpose: Gaming



Laptop gaming Gigabyte G5 KF E3PH3335H
Gigabyte
 11495000đ
RAM: 8 GB
CPU: Core i5
Graphic Card: NVIDIA 4060
Purpose: Gaming

Figure 7: Query for laptops

1.2.5.c Interface to demonstrate procedure in requirement 1.2.3/1.2.4 (pick one)

In requirement 1.2.3 there are procedure to update certain tables when a customer add a product into their card, here is the interface of that. On the page the customer can see all the product they have put to card together with their status (New_Ordering for product that is new and currently in card, 2nd_Ordering for product that is second handed and currently in card), the quantity and the final price of that product, on the top there are total amount and price and the checkout button for customer to confirm the buying.

Items: 5 Total: 24127875đ






Item	Price	Status	Remove Item	Total
 Laptop gaming Lenovo Legion 5 16IRX9 83DG004XVN 9260125đ	9260125đ	2nd_Ordering	1 ▼	9260125đ
 Laptop Dell Inspiron 3530 N3530I716W1 Silver 11395000đ	11395000đ	New_Ordering	1 ▼	11395000đ
 Chuột Rapoo MT760 Mini Không Dây Đen 395000đ	395000đ	New_Ordering	1 ▼	395000đ
 Tai nghe Corsair HS80 RGB Wireless 1895000đ	1895000đ	New_Ordering	1 ▼	1895000đ
 Bàn phím không dây Logitech MX Keys Mini for Mac - Pale Gray 1182750đ	1182750đ	2nd_Ordering	1 ▼	1182750đ

Figure 8: Customer's card