Ramzy Izza Wardhana
21/472698/PA/20322
Database Lab CS B

Homework 9
Function and Trigger

1. Create a function to show the wholesale price of a product that based on the count. For example, a total discount of 5% if buyer buy a dozen and a total discount of 10% if the buyer buys minimum 100 items.

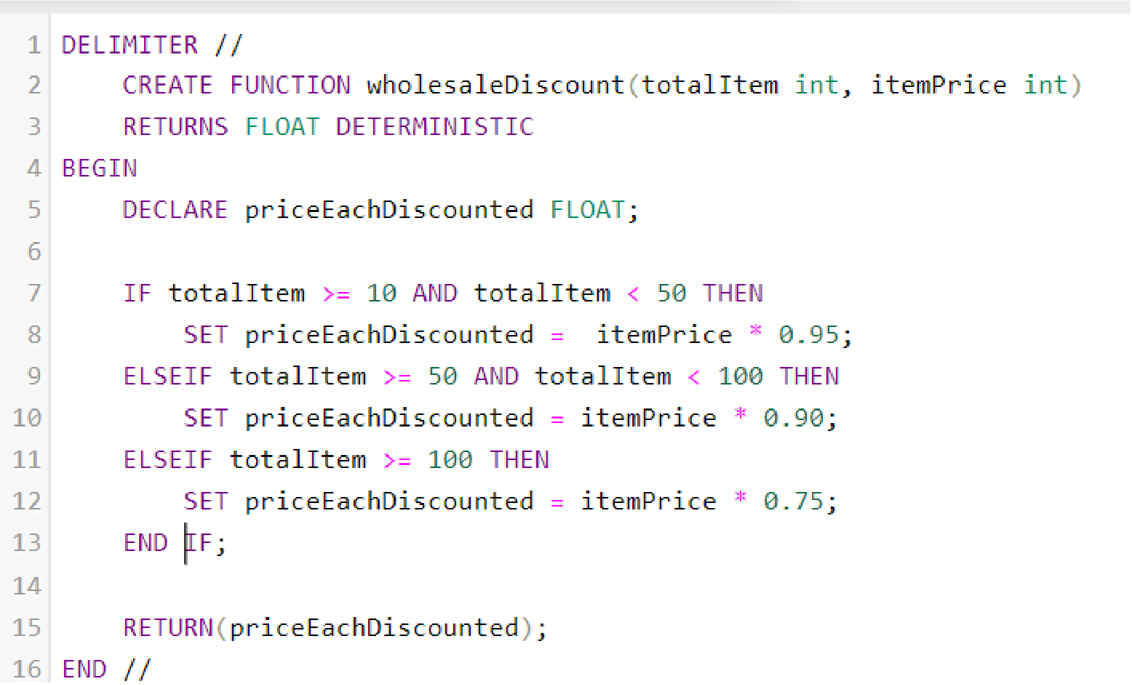
Answer:

Return: Wholesale **price** (per item) after discount based on the count

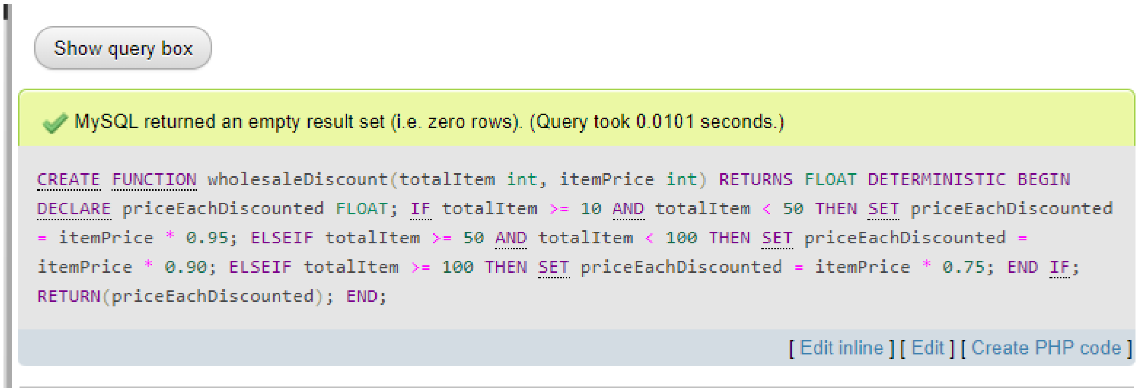
Constraint:

1. If buyAmount ≥ 10 && < 50, Discount = 5%, Wholesale Price per item = priceEach * 0.95
2. If buyAmount ≥ 50 && < 100, Discount = 10%, Wholesale Price per item = priceEach * 0.90
3. If buyAmount ≥ 100, Discount = 25%, Wholesale Price per item = priceEach * 0.75

Step 1: Create the Function



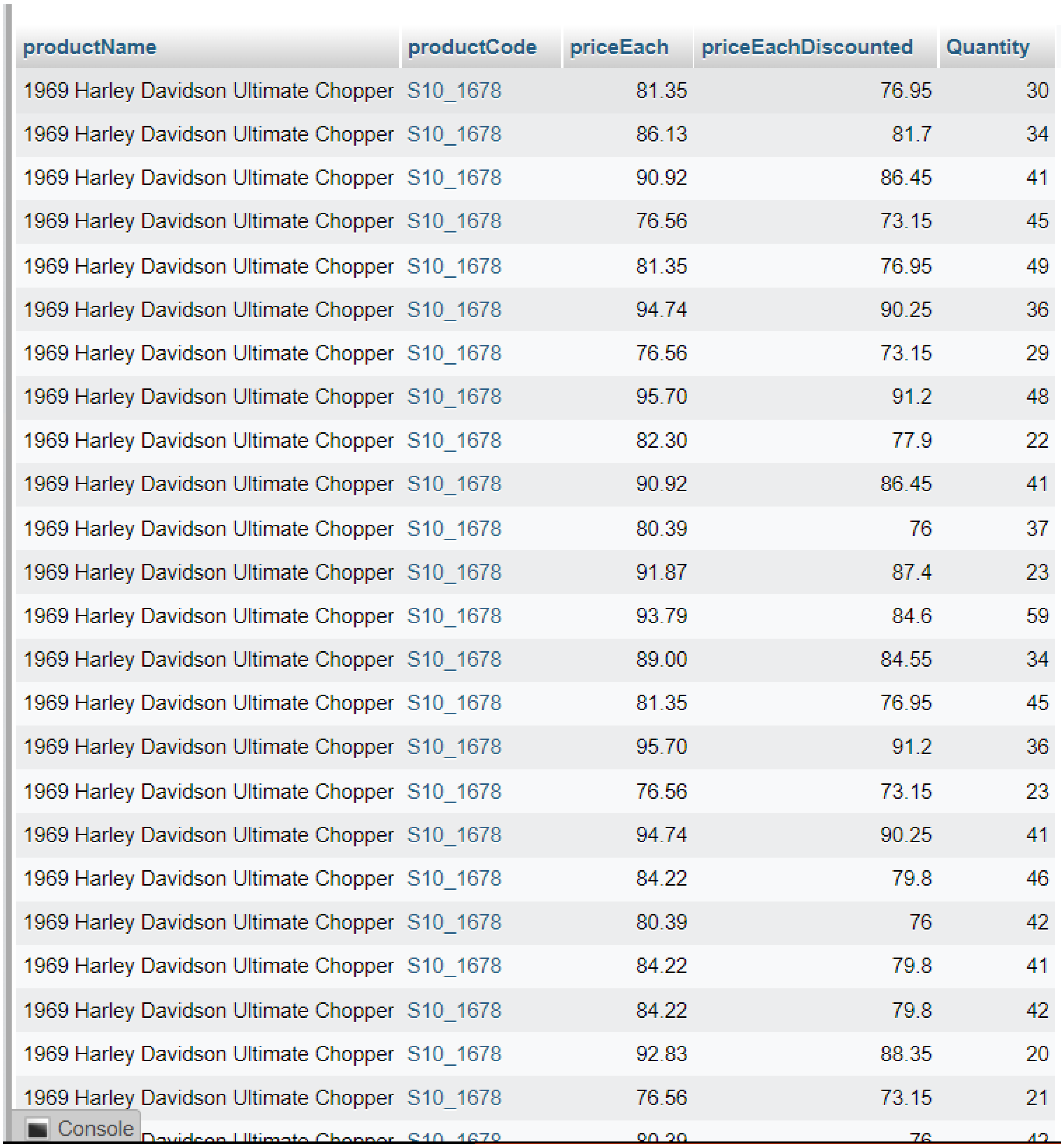
```
DELIMITER //
    CREATE FUNCTION wholesaleDiscount(totalItem int, itemPrice int)
    RETURNS FLOAT DETERMINISTIC
BEGIN
    DECLARE priceEachDiscounted FLOAT;

    IF totalItem >= 10 AND totalItem < 50 THEN
        SET priceEachDiscounted =  itemPrice * 0.95;
    ELSEIF totalItem >= 50 AND totalItem < 100 THEN
        SET priceEachDiscounted = itemPrice * 0.90;
    ELSEIF totalItem >= 100 THEN
        SET priceEachDiscounted = itemPrice * 0.75;
    END IF;

    RETURN(priceEachDiscounted);
END //
```



✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0101 seconds.)

```
CREATE FUNCTION wholesaleDiscount(totalItem int, itemPrice int) RETURNS FLOAT DETERMINISTIC BEGIN
DECLARE priceEachDiscounted FLOAT; IF totalItem >= 10 AND totalItem < 50 THEN SET priceEachDiscounted
= itemPrice * 0.95; ELSEIF totalItem >= 50 AND totalItem < 100 THEN SET priceEachDiscounted =
itemPrice * 0.90; ELSEIF totalItem >= 100 THEN SET priceEachDiscounted = itemPrice * 0.75; END IF;
RETURN(priceEachDiscounted); END;
```

Step 2: Running the function



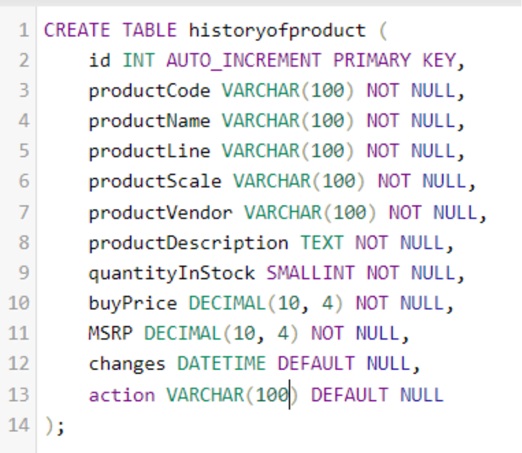
productName	productCode	priceEach	priceEachDiscounted	Quantity
1969 Harley Davidson Ultimate Chopper	S10_1678	81.35	76.95	30
1969 Harley Davidson Ultimate Chopper	S10_1678	86.13	81.7	34
1969 Harley Davidson Ultimate Chopper	S10_1678	90.92	86.45	41
1969 Harley Davidson Ultimate Chopper	S10_1678	76.56	73.15	45
1969 Harley Davidson Ultimate Chopper	S10_1678	81.35	76.95	49
1969 Harley Davidson Ultimate Chopper	S10_1678	94.74	90.25	36
1969 Harley Davidson Ultimate Chopper	S10_1678	76.56	73.15	29
1969 Harley Davidson Ultimate Chopper	S10_1678	95.70	91.2	48
1969 Harley Davidson Ultimate Chopper	S10_1678	82.30	77.9	22
1969 Harley Davidson Ultimate Chopper	S10_1678	90.92	86.45	41
1969 Harley Davidson Ultimate Chopper	S10_1678	80.39	76	37
1969 Harley Davidson Ultimate Chopper	S10_1678	91.87	87.4	23
1969 Harley Davidson Ultimate Chopper	S10_1678	93.79	84.6	59
1969 Harley Davidson Ultimate Chopper	S10_1678	89.00	84.55	34
1969 Harley Davidson Ultimate Chopper	S10_1678	81.35	76.95	45
1969 Harley Davidson Ultimate Chopper	S10_1678	95.70	91.2	36
1969 Harley Davidson Ultimate Chopper	S10_1678	76.56	73.15	23
1969 Harley Davidson Ultimate Chopper	S10_1678	94.74	90.25	41
1969 Harley Davidson Ultimate Chopper	S10_1678	84.22	79.8	46
1969 Harley Davidson Ultimate Chopper	S10_1678	80.39	76	42
1969 Harley Davidson Ultimate Chopper	S10_1678	84.22	79.8	41
1969 Harley Davidson Ultimate Chopper	S10_1678	84.22	79.8	42
1969 Harley Davidson Ultimate Chopper	S10_1678	92.83	88.35	20
1969 Harley Davidson Ultimate Chopper	S10_1678	76.56	73.15	21
Davidson Ultimate Chopper	S10_1678	80.39	76	42

Observe that the output given above is the product name, the quantity, the price each of quantity before discount, and price each quantity after Wholesale discount.

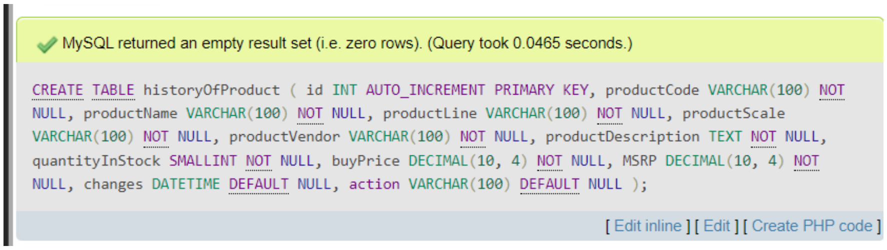
2. Create a trigger to record the change history of the product and create new table to contain those changes.

Answer:

Step 1: Create the table to provide the history list of changes



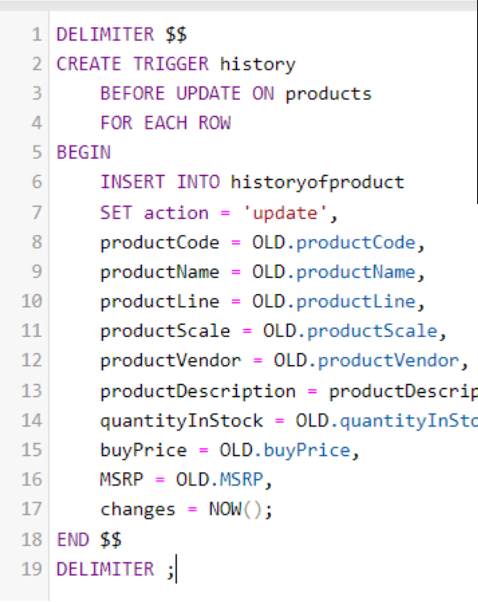
```
CREATE TABLE historyofproduct (
    id INT AUTO_INCREMENT PRIMARY KEY,
    productCode VARCHAR(100) NOT NULL,
    productName VARCHAR(100) NOT NULL,
    productLine VARCHAR(100) NOT NULL,
    productScale VARCHAR(100) NOT NULL,
    productVendor VARCHAR(100) NOT NULL,
    productDescription TEXT NOT NULL,
    quantityInStock SMALLINT NOT NULL,
    buyPrice DECIMAL(10, 4) NOT NULL,
    MSRP DECIMAL(10, 4) NOT NULL,
    changes DATETIME DEFAULT NULL,
    action VARCHAR(100) DEFAULT NULL
);
```



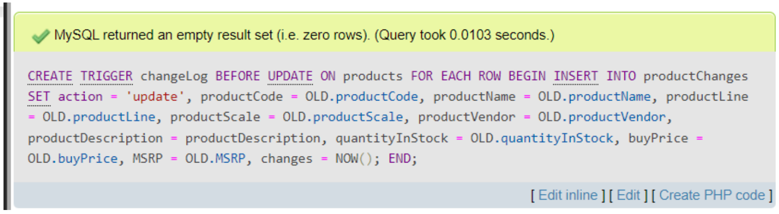
✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0465 seconds.)

```
CREATE TABLE historyOfProduct ( id INT AUTO_INCREMENT PRIMARY KEY, productCode VARCHAR(100) NOT
NULL, productName VARCHAR(100) NOT NULL, productLine VARCHAR(100) NOT NULL, productScale
VARCHAR(100) NOT NULL, productVendor VARCHAR(100) NOT NULL, productDescription TEXT NOT NULL,
quantityInStock SMALLINT NOT NULL, buyPrice DECIMAL(10, 4) NOT NULL, MSRP DECIMAL(10, 4) NOT
NULL, changes DATETIME DEFAULT NULL, action VARCHAR(100) DEFAULT NULL );
```

Step 2: Create the trigger to store the changed value into the table historyOfProduct



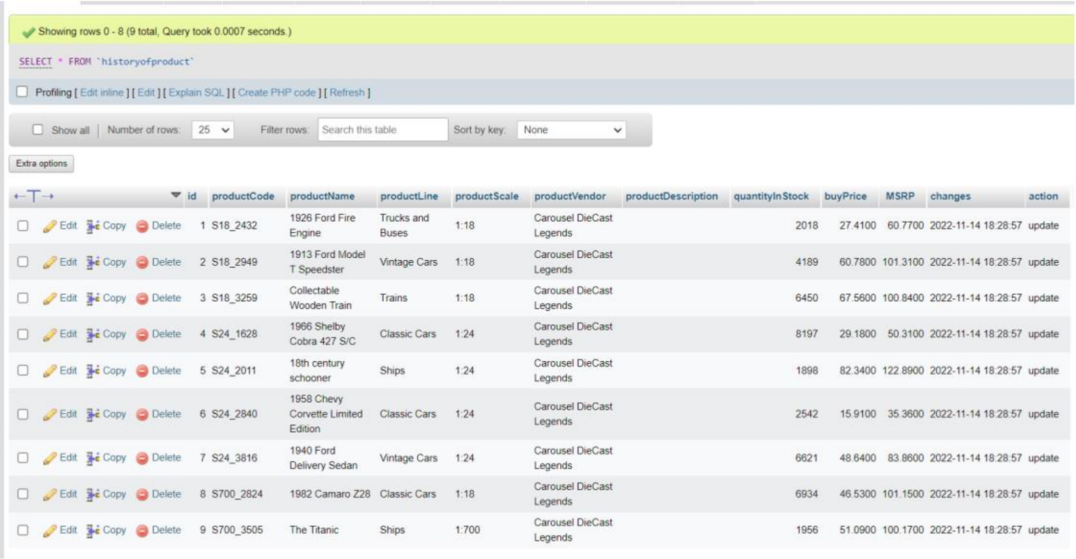
```
DELIMITER $$
CREATE TRIGGER history
    BEFORE UPDATE ON products
    FOR EACH ROW
BEGIN
    INSERT INTO historyofproduct
    SET action = 'update',
    productCode = OLD.productCode,
    productName = OLD.productName,
    productLine = OLD.productLine,
    productScale = OLD.productScale,
    productVendor = OLD.productVendor,
    productDescription = productDescription,
    quantityInStock = OLD.quantityInStock,
    buyPrice = OLD.buyPrice,
    MSRP = OLD.MSRP,
    changes = NOW();
END $$
DELIMITER ;
```



✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0103 seconds.)

```
CREATE TRIGGER changeLog BEFORE UPDATE ON products FOR EACH ROW BEGIN INSERT INTO productChanges
SET action = 'update', productCode = OLD.productCode, productName = OLD.productName, productLine
= OLD.productLine, productScale = OLD.productScale, productVendor = OLD.productVendor,
productDescription = productDescription, quantityInStock = OLD.quantityInStock, buyPrice =
OLD.buyPrice, MSRP = OLD.MSRP, changes = NOW(); END;
```

Step 3: Output the table of historyOfProduct



✓ Showing rows 0 - 8 (9 total, Query took 0.0007 seconds.)

```
SELECT * FROM `historyofproduct`
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

□ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

←T→	id	productCode	productName	productLine	productScale	productVendor	productDescription	quantityInStock	buyPrice	MSRP	changes	action
□ Edit Copy Delete	1	S18_2432	1926 Ford Fire Engine	Trucks and Buses	1:18	Carousel DieCast Legends		2018	27.4100	60.7700	2022-11-14 18:28:57	update
□ Edit Copy Delete	2	S18_2949	1913 Ford Model T Speedster	Vintage Cars	1:18	Carousel DieCast Legends		4189	60.7800	101.3100	2022-11-14 18:28:57	update
□ Edit Copy Delete	3	S18_3259	Collectable Wooden Train	Trains	1:18	Carousel DieCast Legends		6450	67.5600	100.8400	2022-11-14 18:28:57	update
□ Edit Copy Delete	4	S24_1628	1966 Shelby Cobra 427 S/C	Classic Cars	1:24	Carousel DieCast Legends		8197	29.1800	50.3100	2022-11-14 18:28:57	update
□ Edit Copy Delete	5	S24_2011	18th century schooner	Ships	1:24	Carousel DieCast Legends		1898	82.3400	122.8900	2022-11-14 18:28:57	update
□ Edit Copy Delete	6	S24_2840	1958 Chevy Corvette Limited Edition	Classic Cars	1:24	Carousel DieCast Legends		2542	15.9100	35.3600	2022-11-14 18:28:57	update
□ Edit Copy Delete	7	S24_3816	1940 Ford Delivery Sedan	Vintage Cars	1:24	Carousel DieCast Legends		6621	48.6400	83.8600	2022-11-14 18:28:57	update
□ Edit Copy Delete	8	S700_2824	1982 Camaro Z28	Classic Cars	1:18	Carousel DieCast Legends		6934	46.5300	101.1500	2022-11-14 18:28:57	update
□ Edit Copy Delete	9	S700_3505	The Titanic	Ships	1:700	Carousel DieCast Legends		1956	51.0900	100.1700	2022-11-14 18:28:57	update

Notice that all the productName that has has the vendor of Carousel DieCast Legends that had been updated previously will be stored in this new table which done by the trigger we have made before.