TRIGGER

<u>TRIGGER 1</u>

→Updates the cost column of the table when the productID and productCost is Inserted

```
DELIMITER $$
CREATE TRIGGER update_cart_item_price
BEFORE INSERT ON cart_item
FOR EACH ROW
BEGIN
   IF NEW.quantity < 0 THEN
      SET NEW.quantity = 0;
   END IF;
   SET NEW.cost = NEW.quantity * (SELECT productPrice FROM Product WHERE productID =
NEW.productID);
END$$
DELIMITER ;
```

<u>TRIGGER 2</u>

→Updates the Total Cost column of the table when the corresponding cart_item is inserted

```
DELIMITER $$
CREATE TRIGGER update_cart
AFTER INSERT ON cart_item
FOR EACH ROW
BEGIN
  IF NOT EXISTS (SELECT * FROM cart WHERE customerID = NEW.customerID) THEN
   INSERT INTO cart (customerID, total_Cost, discount) VALUES (NEW.customerID, NEW.cost,
0);
  ELSE
   -- If a cart already exists, update the totalCost
   UPDATE cart
   SET total_Cost = (SELECT SUM(cost) FROM cart_item WHERE cart_item.customerID =
NEW.customerID)*(1-(discount/100))
   WHERE customerID=NEW.customerID;
   END IF;
END$$
DELIMITER ;
```

TRIGGER 3

```
-----------------------------------------------------------
→ Extra 10% discount if the cart price is above 10k
DELIMITER $$
CREATE TRIGGER increase_discount
BEFORE UPDATE ON cart
FOR EACH ROW
BEGIN
   IF NEW.total_Cost +(NEW.total_Cost*(OLD.discount/100)) > 10000 THEN
      SET NEW.discount = OLD.discount+10;
      SET NEW.total_Cost=
(NEW.total_Cost+(NEW.total_Cost*(OLD.discount/100)))*(1-(NEW.discount/100));
   END IF;
END$$
DELIMITER ;
```

OLAP:

1)Query to get the total number of orders and revenue generated by each delivery agent over the years(Slicing OLAP Query)

```
SELECT deliveryagent.deliveryID,deliveryagent.first_name, COUNT(DISTINCT
order_detail.orderID) as total_orders, SUM(order_detail.totalCost) as total_revenue
FROM deliveryagent
JOIN order_detail ON deliveryagent.deliveryID = order_detail.deliveryID
JOIN payment ON order_detail.orderID = payment.orderID
WHERE YEAR(payment.dateTransaction) = YEAR(payment.dateTransaction)
GROUP BY deliveryagent.deliveryID
ORDER BY total_revenue DESC
```

//---
2)Query to get the details of all the customers in ranking of orders placed(Customer corresponding to highest orders placed comes first)
(Drill down OLAP query, as it drills down on the customer dimension by ranking the customers based on the number of orders they have place)(identify which customers are the most valuable or active in terms of placing orders.)

```
SELECT customer.customerID,customer.first_name, COUNT(order_detail.orderID) AS
total_orders
```

```
FROM customer
LEFT JOIN order_detail ON customer.customerID = order_detail.customerID
GROUP BY customer.customerID
ORDER BY total_orders DESC
//--------------------------------------------------------------------------------
```
3)Query to get how many products are in each product category (Slicing)

```
SELECT pc.categoryName, COUNT(belongs.productID) AS total_products
FROM product_category pc
LEFT JOIN belongs  ON pc.categoryID = belongs.categoryID
GROUP BY pc.categoryName;
//--------------------------------------------------------------------------------
```
4)Query to get the total order value,number of instances of usage, and average order value for each payment method.

```
SELECT modePayment as Mode_Of_Payment,SUM(order_detail.totalCost) as
Total_Order_Value,COUNT(order_detail.orderID) as NumberOf_Times_Used,
AVG(order_detail.totalCost) AS Average_Order_Value
FROM payment
JOIN order_detail ON order_detail.orderID=payment.orderID
GROUP BY modePayment WITH ROLLUP;
//--------------------------------------------------------------------------------
```
5) Query to get the revenue Generated by Year and Month

```
SELECT YEAR(payment.dateTransaction) as Year, MONTH(payment.dateTransaction)
as Month, SUM(order_detail.totalCost) as Revenue_Generated
FROM payment
INNER JOIN order_detail ON payment.orderID = order_detail.orderID
GROUP BY Year,Month WITH ROLLUP
```

//NULL NULL represents the sum of revenue generated over the 2 years while 2022 NULL means the revenue generated in 2022

<u>EMBEDDED SQL QUERIES:</u>

1) Query to get the specific details of the orders that were placed within a specific date range (Between '2022-01-01' AND '2022-03-01)

```
  mycursor = db.cursor()
   mycursor.execute('''SELECT order_detail.orderID, payment.DateTransaction,
order_detail.totalCost,
customer.first_name
FROM order_detail
JOIN payment ON order_detail.orderID = payment.orderID
JOIN customer ON order_detail.customerID = customer.customerID
WHERE payment.DateTransaction BETWEEN '2022-01-01' AND '2022-03-01';''')

    results = mycursor.fetchall()
    for x in results:
        x = str(x)
        x = x.replace("datetime.date", "")
        x = x.replace("'", "")
        print(x[1:len(x) - 1])
```

2) Query to get the details of all the customers who have spent more than a specific amount of money.(For giving extra discount to the customer)

```
 mycursor = db.cursor()
    mycursor.execute('''SELECT customer.customerID, customer.first_name,
SUM(order_detail.totalCost) as
Total_Spent
FROM customer
JOIN order_detail ON customer.customerID = order_detail.customerID
GROUP BY customer.customerID
HAVING SUM(order_detail.totalCost) > 10000;''')

    results = mycursor.fetchall()
    for x in results:
        x = str(x)
        x = x.replace("Decimal", "")
        x = x.replace("(", "")
        x = x.replace(")", "")
        x = x.replace("'", "")
        print(x)
```

3) Query to get to know the what products are in Cart(Summary).
   For customerID=10

```
 mycursor = db.cursor()
   mycursor.execute('''SELECT
cart_item.productID,
(SELECT productName FROM product WHERE productID = cart_item.productID) AS
productName, cart_item.quantity,
(SELECT productPrice FROM product WHERE productID = cart_item.productID) AS
productPrice,cart_item.cost
FROM cart, cart_item
WHERE
cart.customerID = cart_item.customerID AND cart.customerID = 10;''')

   results = mycursor.fetchall()
   for x in results:
      x = str(x)
      x = x.replace("Decimal", "")
      x = x.replace("(", "")
      x = x.replace(")", "")
      x = x.replace("'", "")
      print(x)
```

4)Display the products available:

```
   cursor = db.cursor()
   cursor.execute('SELECT * FROM product')
   results = cursor.fetchall()
```