ONLINE RETAIL STORE

Group 50

Samanyu Kamra 2021487

Shriya Verma 2021490

Embedded Queries:

1. **Changes prices of all products of the chosen category.**

```
def change_price():
    n = float(input("Enter the price multiplier: "))
    s = str(input("Enter Category Name: "))
    sql = "UPDATE product SET price = price * %s WHERE product.category_id =
(SELECT category.category_id FROM category WHERE
category.category_name = %s)"
    val = (n,s)

    mycursor.execute(sql,val)
    for i in mycursor:
        print(i)

    mydb.commit()
    print(mycursor.rowcount, "record(s) affected")

    print("Price Updated")
```

2. **Add A delivery Agent:**

```
def add_agent():
    n = int(input("Enter your Admin ID: "))
    m = int(input("Assign an ID to the new Delivery Agent: "))
    s = str(input("Enter delivery agent name: "))
    s1 = str(input("Create a Default Password: "))
    k = int(input("Enter Phone no: "))
    s2 = str(input("Enter E-mail: "))
```

```
    sql = "INSERT INTO Delivery(admin_id,agent_id ,full_name ,pass_word
,avg_rating  ,phone ,email) VALUES(%s,%s,%s,%s,%s,%s,%s)"
    val = (n,m,s,s1,2,k,s2)
    mycursor.execute(sql,val)
    for i in mycursor:
        print(i)

    mydb.commit()
    print(mycursor.rowcount, "record(s) affected")

    print("Delivery Agent added")
```

3. **Allows customer to view all their carts**

```
def my_cart():
    n = int(input("Enter the Customer ID: "))
    sql = "SELECT * FROM Customer WHERE customer.customer_id = %s
UNION SELECT * FROM cart WHERE cart.customer_id = %s"
    val = (n,n)
    mycursor.execute(sql,val)
    for i in mycursor:
        print(i)

    mydb.commit()
    print("Here are the details.")
    print(mycursor.rowcount, "record(s) affected")
```

OLAP Queries:

1. **What is the total revenue the online retail store generates in a given time period?**

```
SELECT SUM(amount) AS total_revenue
FROM Order
WHERE date_ BETWEEN [start_date] AND [end_date];
```

2. **What is the average price of products in each category?**

```
SELECT
   c.category_name AS Category,
   AVG(p.price) AS AvgPrice
FROM
   Product p
   JOIN Category c ON p.category_id = c.category_id
GROUP BY
   c.category_name

ORDER BY avg_price DESC;
```

3. **What are the Top 10 products?**

```
SELECT product.product_name , COUNT(*) AS FEEDBACK_COUNT
FROM product
JOIN product_feedback
ON product.product_id = product_feedback.product_id
GROUP BY product.product_id
ORDER BY FEEDBACK_COUNT DESC
LIMIT 10
```

4. **What are the top three delivery agents with the highest average rating?**

```
SELECT Delivery_Agent.first_name, Delivery_Agent.last_name,
Delivery_Agent.average_rating
FROM Delivery_Agent
ORDER BY Delivery_Agent.average_rating DESC
LIMIT 3;
```

Triggers:

1. **<u>If a customer is removed from the database, all feedback associated with that customer is also deleted :</u>**

```
DELIMITER $$
CREATE
TRIGGER remove_customer_corresponding_product_feedback BEFORE
DELETE
ON Customer FOR EACH ROW

BEGIN
DELETE FROM product_feedback WHERE
OLD.customer_id=customer_id;
END$$
DELIMITER ;
```

2. **<u>If a cart associated with an order does not exist, then this order can not be placed:</u>**

```
CREATE TRIGGER t1
BEFORE INSERT ON _order
FOR EACH ROW
BEGIN
DECLARE cart_count INT;
SELECT COUNT(*) INTO cart_count FROM Cart WHERE cart_id =
NEW.cart_id;
IF cart_count = 0 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot create
order without a corresponding cart record.';
 END IF;
END;
```

3. **<u>If a category is deleted, all products that belong to that category will also be deleted:</u>**

```
DELIMITER $$ CREATE
TRIGGER delete_product_from_category BEFORE DELETE ON category
FOR EACH ROW
```

```
BEGIN
DELETE from product
WHERE OLD.Category_ID=product.Category_ID; END$$
DELIMITER ;
```