DBMS Project Deadline5

REPORT

Embedded SQL, OLAP queries and Triggers

Siddharth Gupta 2021355

Tony Thomas 2021360

Embedded SQL:

//To make connection between python and SQL using mysql connector

```
import mysql.connector
mydb=mysql.connector.connect(
    host='127.0.0.1',
    database="onlineretailstore",
    user='root',
    password='tony022002@Kuku'
)
cursor=mydb.cursor()
```

//Command Line Interface starting

```
print("""Choose the query to be performed:""")
print("""1.login and view order history \n2.search product catalogue by key
word\n""")
query_num=int(input())
```

Embedded QUERY1

This query first makes user login by entering their valid credentials, then it shows the order ids of the previously placed orders. The user is made to choose from these to view the order history details showing all the order items of the chosen order along with the order_date and order status based on order_id. This will be useful while viewing the order history

```
#Embedded Query 1
login_id=int(input("Enter the login id :"))
password=input("Enter the password :").strip()
cmd="""SELECT * FROM authentication_credentials
    WHERE login_id={0} AND password='{1}';""".format(login_id,password)

cursor.execute(cmd);
result=cursor.fetchone()
customer_id=result[0]
print(customer_id)
cmd="""SELECT * FROM `order`
    WHERE customer_id='{0}'""".format(customer_id)
```

```
cursor.execute(cmd)
all_order_id=[]
for i in cursor:
    all_order_id.append(i[0])
print("""Which Order do you want to see the order history for?""")
num=1
for i in all_order_id:
    print(num," = ",i)
    num+=1
choice=int(input("Enter your choice : "))
order_id=all_order_id[choice-1]

cmd="""SELECT OI.*, O.order_date, O.order_status
FROM Order_item OI, `Order` O
WHERE OI.order_id = O.order_id AND O.order_id='{0}';
""".format(order_id)
cursor.execute(cmd)
for i in cursor:
    print(i)
```

Embedded QUERY2

Query acts as a make-do search bar, asks user to enter a word then searches and returns all products having that word anywhere in its name (checking for substring).

```
#Embedded Query 2
print("Search bar will display all the item names that has the following
search key")
search_key=input("Enter the item name/key word to be searched :")
cmd="""SELECT P.name, P.description, P.price
FROM Product P
WHERE P.name LIKE "%{0}%";""".format(search_key)
cursor.execute(cmd)
for i in cursor:
    print(i)
```

Command line interface has been designed for these queries.

OLAP Queries:

An OLAP query that gives the total number of items (products*quantity) in that category as well as the net monetary amount present in that category. It adds the stock quantities of all the products belonging to that category and also multiplies the price with the stock and adds that.

```
SELECT category_id, SUM(stock) AS 'Number of Quantities in this category',
SUM(price*stock) AS 'Total Cost of this Category'
FROM product
GROUP BY (category_id) WITH ROLLUP;
```

```
SELECT COALESCE(category_id, 'All categories') AS 'Category Id', SUM(stock) AS
'Number of Quantities in this category', SUM(price*stock) AS 'Total Cost of this Category'
FROM product
GROUP BY (category_id) WITH ROLLUP;
```

An OLAP query to return the total quantity of a product in a customer's cart that may be present as separate cart_items currently, it checks for multiple cart_items having same products, and present in the same cart then adds their quantity

```
SELECT cart_id, product_id, SUM(quantity)
FROM cart_item
GROUP BY cart_id, product_id WITH ROLLUP;
```

An OLAP query to return the total amount spent by customers till now on the shopping platform, it also gives the total amount spent on all orders till now

```
SELECT O.customer_id, SUM(O.net_amount) AS 'Total amount spent'
FROM `Order` O
GROUP BY O.customer_id WITH ROLLUP
```

```
SELECT COALESCE(O.customer_id, 'All customers') AS 'CustomerID',
SUM(O.net_amount) AS 'Total amount spent'
FROM `Order` O
GROUP BY O.customer_id WITH ROLLUP
```

An OLAP query to return the total number of products in a category and the total number of items present in that category.

```
SELECT COALESCE(category_id,'All Categories Combined: ') AS CategoryID,
COUNT(category_id) AS 'Total Number of products in that category',SUM(stock)  AS "Total
quantity of products in that category"
FROM product
GROUP BY category_id WITH ROLLUP;
```

Triggers:

//SHOW TRIGGERS used to view current active triggers

1. **Trigger to update the total amount of a cart when a new cart_item is added by the customer**

```
//actual trigger body
DELIMITER $$
CREATE TRIGGER cart_item_add_and_cart_total_amount_update
AFTER INSERT
ON cart_item
FOR EACH ROW
BEGIN
        UPDATE cart
   SET cart.cart_total_amount= cart.cart_total_amount + ((SELECT P.price FROM
product P WHERE P.product_id = NEW.product_id) * NEW.quantity)
   WHERE cart.cart_id= NEW.cart_id;
END $$
DELIMITER ;

//for dropping
DROP TRIGGER onlineretailstore.cart_item_add_and_cart_total_amount_update;

//for testing, should add Rs.10040 to cart447's total amount
INSERT INTO cart_item
(cart_id, product_id, quantity)
VALUES
('CART447', 'PROD268', 2);
```

2. **Checks stock value before adding a new cart_item, useful when a customer wishes to add a new item to their cart, there will be an automatic check imposed which would ensure that the item is added only if there is enough stock**

```
//actual trigger body
DELIMITER $$
CREATE TRIGGER cart_item_check_if_quantity_exists
BEFORE INSERT
ON cart_item
FOR EACH ROW
BEGIN
```

```
        IF (NEW.quantity >= (SELECT P.stock FROM product P WHERE
P.product_id = NEW.product_id))
   THEN
   SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Not enough stock';
        END IF;
END $$
DELIMITER ;

//for dropping
DROP TRIGGER onlineretailstore.cart_item_check_if_quantity_exists;

//for testing, should add Rs.10040 to cart447's total amount
INSERT INTO cart_item
(cart_id, product_id, quantity)
VALUES
('CART447', 'PROD268', 325);
```

Previous Queries:

1. **Query to authenticate the user and fetch the customer name for welcome message in case authenticated**
   78793 and 'PtSHC6' are the login_id and password entered by the user. If authenticated, it returns the first and last name of the customer for the welcome message. If not, it returns an empty row (this also shows one-one and total relation shared by these two entities)

```
SELECT C.first_name, C.last_name
FROM Customer C, Authentication_credentials A
WHERE A.login_id=78793 AND A.password='PtSHC6' AND
C.customer_id=A.customer_id;
```

2. **Query to search a product by name and return all products having that word anywhere in its name (checking for substring)**

```
SELECT P.name, P.description, P.price
FROM Product P
WHERE P.name LIKE "%Wine%";
```

3. **This query will return all the order items for a given order along with the order_date and order status based on order_id. This will be useful while viewing the order history**

```
SELECT OI.*, O.order_date, O.order_status
FROM Order_item OI, `Order` O
WHERE OI.order_id = O.order_id AND O.order_id='ORDER117';
```

4. **Query to update the password, here two update queries have been write so as to keep the data unaffected**

```
UPDATE Authentication_credentials A
SET A.password="newPwd"
WHERE A.login_id=78793 AND A.password='PtSHC6' AND
A.customer_id='CUST001';

UPDATE Authentication_credentials A
SET A.password="PtSHC6"
WHERE A.login_id=78793 AND A.password='newPwd' AND
A.customer_id='CUST001';
```

5. **Query to delete cart_item from a customer's cart, to simulate the delete button against a product one has added in one's cart**

```
DELETE FROM cart_item C
WHERE C.cart_id="CART447" AND C.product_id="PROD248";
```

6. **Add a new cart_item to a customer's cart, similar to adding to cart function for products on any online shopping site**

```
INSERT INTO cart_item
(cart_id, product_id, quantity)
VALUES
('CART447', 'PROD248', 35);
```

7. **Query to get all orders with coupon discounts greater than a number**

```
SELECT C.first_name, O.order_id, O.order_date, O.net_amount,
CO.percentage_discount
FROM Customer C, `Order` O, Coupon CO
WHERE C.customer_id = O.customer_id
AND O.coupon_id = CO.coupon_id
AND O.coupon_id
IN (SELECT COU.coupon_id
FROM Coupon COU
```

```
WHERE COU.percentage_discount >=20);
```

8. **Increase the quantity of a particular product by 1 in a customer's cart (to simulate the increment done when a customer clicks on the plus button against the cart item in their cart)**

```
UPDATE Cart_item CI INNER JOIN Customer C ON CI.cart_id = C.cart_id
SET CI.quantity = CI.quantity + 1
WHERE CI.product_id = 'PROD124'
AND C.customer_id ='CUST141';

UPDATE Cart_item CI INNER JOIN Customer C ON CI.cart_id = C.cart_id
SET CI.quantity = CI.quantity - 1
WHERE CI.product_id = 'PROD124'
AND C.customer_id ='CUST141';
```

9. **Query to get total amount spent by a customer on the store**

```
SELECT SUM(O.net_amount)
FROM `Order` O, Customer C
WHERE C.customer_id = O.customer_id
AND C.customer_id = 'CUST408';
```

10. **If the admin wants to know a list of all customers who have spent a total amount of money greater than Rs.7000 and sort this in descending order of amount spent.**
    The group by command in this query can be used in other ways as well, we could have grouped by pincode to analyse which area is pending the most (to improve delivery services) or least (to advertise or market)

```
SELECT C.customer_id, C.first_name, C.last_name, SUM(O.net_amount)
FROM `Order` O, Customer C
WHERE O.customer_id = C.customer_id
GROUP BY O.customer_id
HAVING SUM(O.net_amount) > 7000
ORDER BY SUM(O.net_amount) desc;
```

11. **Query to get average amount spent by customers per order and sort it in ascending order**

```
SELECT C.customer_id, C.first_name, C.last_name, AVG(O.net_amount)
```

```
FROM `Order` O, Customer C
WHERE O.customer_id = C.customer_id
GROUP BY O.customer_id
ORDER BY AVG(O.net_amount) asc;
```

12. **To get the order with maximum amount for the given customer. This customer had 2 orders of amounts 10013 and 8645, so 10013 was returned**

```
SELECT C.customer_id, C.first_name, C.last_name, MAX(O.net_amount)
FROM `Order` O, Customer C
WHERE O.customer_id = C.customer_id AND C.customer_id = 'CUST408';
```