U19CS076 DBMS ASSIGNMENT 9

KRITHIKHA BALAMURUGAN

CURSORS

Q1. Create a cursor to fetch the count of customers and sellers.

```
DECLARE
 CURSOR count_c IS SELECT CUSTOMER_ID FROM CUSTOMER;
 CURSOR count_s IS SELECT SELLER_ID FROM SELLER;
 t1 CUSTOMER.CUSTOMER_ID%type;
 t2 SELLER.SELLER_ID%type;
 ccount number;
 scount number;
BEGIN
 ccount := 0;
 scount := 0;
 OPEN count_c;
 LOOP
 FETCH count_c INTO t1;
 EXIT WHEN count_c%NOTFOUND;
 ccount := ccount + 1;
 END LOOP;
 dbms_output.put_line('Number of customers: ' || ccount);
 CLOSE count_c;
 OPEN count_s;
 LOOP
 FETCH count_s INTO t2;
 EXIT WHEN count_s%NOTFOUND;
 scount := scount + 1;
 END LOOP;
 dbms_output.put_line('Number of sellers: ' || scount);
 CLOSE count_s;
END;
```

Results Explain Describe

```
Number of customers: 10
Number of sellers: 6

Statement processed.
```

0.01 seconds

krithikhabala@gmail.com r

Q2. Create a cursor to display all the product details with rating more than 3.5.

```
DECLARE
 CURSOR rating_filter IS SELECT * FROM PRODUCT WHERE Rating>3.5;
BEGIN
 FOR X IN rating_filter
 loop
 DBMS_OUTPUT.PUT_LINE(X.Product_id||' | '||X.Product||' | '||X.Amount|
|' | '||X.Quantity_Rem||' | '||X.Category_id||' | '||X.Seller_id||' |
'||X.Rating);
 end loop;
END;
```

Results | Explain | Describe | Saved SQL | History

```
1P | The Programming language of ORACLE | 350 | 4 | 1C | 1S | 4.5
3P | White Lamp | 800 | 3 | 3C | 5S | 4
8P | Portico King size bedsheet | 1999 | 1 | 3C | 1S | 5

Statement processed.


0.01 seconds
```

Q3. Create a cursor to display all the products category wise.

```
DECLARE
 CURSOR disp_products IS SELECT * FROM PRODUCT ORDER BY Category_id AS
C;
BEGIN
 FOR X IN disp_products
 loop
 DBMS_OUTPUT.PUT_LINE(X.Product_id||' | '||X.Product||' | '||X.Amount|
|' | '||X.Quantity_Rem||' | '||X.Category_id||' | '||X.Seller_id||' |
'||X.Rating);
 end loop;
END;
```

Results | Explain | Describe | Saved SQL | History

```
10P | Artificial Intelligence 3rd Edition | 570 | 9 | 1C | 2S |
1P | The Programming language of ORACLE | 350 | 4 | 1C | 1S | 4.5
7P | Introduction to Java | 650 | 8 | 1C | 5S | 3
11P | Introduction to python | 630 | 10 | 1C | 5S | 1.5
6P | Catwalk leather flats | 1599 | 3 | 2C | 4S | 1
2P | Nike White shoes | 7000 | 2 | 2C | 3S |
9P | Book rack | 999 | 7 | 3C | 4S | 2.5
8P | Portico King size bedsheet | 1999 | 1 | 3C | 1S | 5
3P | White Lamp | 800 | 3 | 3C | 5S | 4
5P | Antique Silver Bracelet | 700 | 5 | 4C | 6S |
4P | Antique Silver Earrings | 400 | 7 | 4C | 2S | 3

Statement processed.
```

0.01 seconds

TRIGGERS:

1. Create a trigger to update the remaining quantity of product in the product table, when a new entry in order_products table is inserted

```
CREATE OR REPLACE TRIGGER update_quant
AFTER INSERT ON ORDER_PRODUCT
FOR EACH ROW
BEGIN
 dbms_output.put_line('trigger 1 - triggered');
 UPDATE PRODUCT SET QUANTITY_REM = QUANTITY_REM -
 :new.QUANTITY WHERE QUANTITY_REM > 0 AND PRODUCT_ID = :new.PRODUCT_ID
;
 IF SQL%ROWCOUNT=0 THEN
 dbms_output.put_line('No row affected');
 END IF;
END update_quant;
```

SQL commands for execution:

```
BEGIN
INSERT INTO order_product(ORDER_ID, PRODUCT_ID, QUANTITY, SELLER_ID, O
RIGINAL_AMT, DISCOUNT, PROD_RATING) values('11O' , '9P' , 1 , '4S' , 9
99 , 0 , 4);
END
```

Results Explain Descri

```
trigger 1 - triggered
Query Success

Statement processed.
```

0.04 seconds

krithikhabala@gmail.com

Q2. Create a trigger to update product rating and seller rating when a new entry in the order_products table is inserted.

```
CREATE OR REPLACE TRIGGER update_rating
AFTER INSERT ON ORDER_PRODUCT
BEGIN
 dbms_output.put_line('trigger 2 ->triggered');
 UPDATE product p SET p.rating = (SELECT AVG(prod_rating) FROM order_p
roduct GROUP BY product_id HAVING product_id = p.product_id);
 UPDATE seller s SET s.rating = (SELECT AVG(prod_rating) FROM order_pr
oduct GROUP BY seller_id HAVING seller_id = s.seller_id);
 IF SQL%ROWCOUNT=0 THEN
 dbms_output.put_line('No row affected');
 END IF;
END update_rating;
```

SQL commands for execution:

```
BEGIN
INSERT INTO order_product(ORDER_ID, PRODUCT_ID, QUANTITY, SELLER_ID, O
RIGINAL_AMT, DISCOUNT, PROD_RATING) values('12O' , '6P' , 1 , '4S' , 1
599 , 0 , 5);
END
```

Results Explain Desc

```
trigger 1 - triggered
trigger 2 ->triggered

Statement processed.
```

0.03 seconds

krithikhabala@gmail.com

3. Create a trigger to check when a new entry is to be inserted in the order_products table the quantity column satisfies the remaining quantity column from the product table.

```
CREATE OR REPLACE TRIGGER check_quantity
BEFORE INSERT
ON order_product
FOR EACH ROW ENABLE
DECLARE
q PRODUCT.quantity_rem%type;
BEGIN
 DBMS_OUTPUT.PUT_LINE('Q3 Trigger');
 SELECT quantity_rem INTO q FROM PRODUCT WHERE product_id=:new.product
_id;
 IF (:new.QUANTITY < q ) THEN UPDATE product SET Quantity_Rem = Quanti
ty_Rem-:NEW.QUANTITY where product_id=:NEW.product_id;
 dbms_output.put_line('Remaining quantity satisfied');
 ELSE
 dbms_output.put_line('Remaining quantity not satisfied');
 END IF;
 IF SQL%ROWCOUNT=0 THEN
 dbms_output.put_line('No row affected');
 END IF;
END check_quantity;
```

SQL commands for execution:

```
BEGIN
INSERT INTO order_product(ORDER_ID, PRODUCT_ID, QUANTITY, SELLER_ID, O
RIGINAL_AMT, DISCOUNT, PROD_RATING) values('14O' , '6P' , 10 , '4S' ,
1599 , 0 , 5);
END
```