Sardar Vallabhbhai National Institute of Technology, Surat Subject: DATABASE MANAGEMENT SYSYTEM

- **DBMS Assignment-11**
- **Name: Nehal Jhajharia**
- **Roll No.: B093**
- **Admission No.: U20CS093**

Cursor:

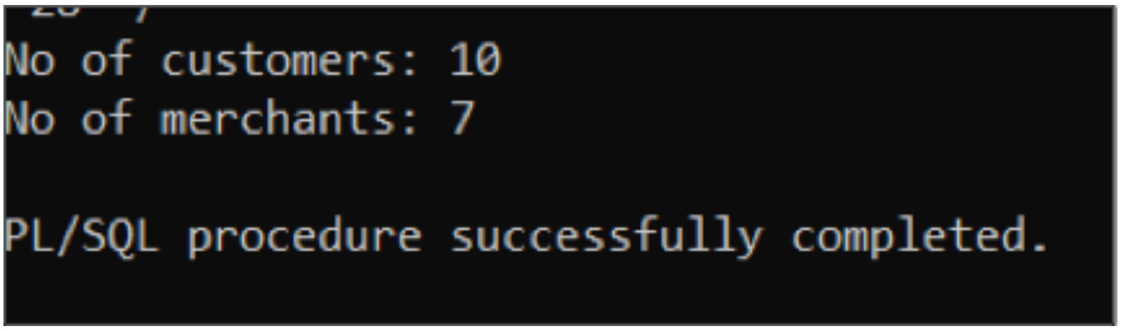
Q1. Create a cursor to fetch the count of customers and sellers.

```
Declare
cus customer.customer_id%type;
mer merchant.m_id%type;
count1 number;
count2 number;
Cursor count_customer is select customer_id from customer; Cursor
count_merchant is select m_id from merchant;
begin
count1:=0;
count2:=0;
open count_customer;
loop
fetch count_customer into cus;
exit when count_customer%NOTFOUND;
count1:=count1+1;
end loop;
close count_customer;
dbms_output.put_line('No of customers: '|| count1);
open count_merchant;
loop
fetch count_merchant into mer;
```

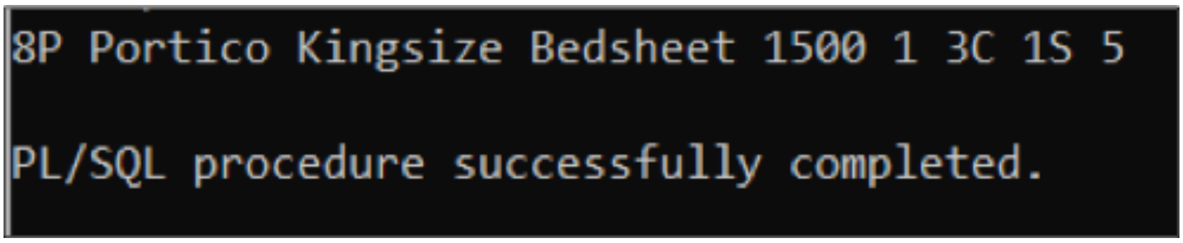
```
        exit when count_merchant%NOTFOUND;
        count2:=count2+1;

end loop;
close count_merchant;
dbms_output.put_line('No of merchants: '|| count2);

end;
/
```

Q2. Create a cursor to display all the product details with rating more than 4.5.

```
Declare
cursor prod_details is select product_id, product, amount,
quantity_remaining, category_id, m_id,
rating from product where rating > 4.5;
p_id product.product_id%type;
p_name product.product%type;
p_amount product.amount%type;
p_qrem product.quantity_remaining%type;
p_catid product.category_id%type;
p_mid product.m_id%type;
p_rating product.rating%type;
begin
open prod_details;
loop
fetch prod_details into
p_id,p_name,p_amount,p_qrem,p_catid,p_mid,p_rating;
exit when prod_details%NOTFOUND;
dbms_output.put_line(p_id ||' '||p_name||' '||p_amount||' '||p_qrem||' '||p_catid||'
'||p_mid||' '||p_rating); end loop;
close prod_details;
end;
/
```



```
No of customers: 10
No of merchants: 7

PL/SQL procedure successfully completed.
```



```
8P Portico Kingsize Bedsheet 1500 1 3C 1S 5

PL/SQL procedure successfully completed.
```

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

Declare

cursor prod_category is select product_id, product, amount, product.quantity_remaining,

product. category_id, m_id, rating ,category from product, category where

product.category_id=category.category_id order by category_id;

p_id product.product_id%type;

p_name product.product%type;

p_amount product.amount%type;

p_qrem product.quantity_remaining%type;

p_catid product.category_id%type;

p_mid product.m_id%type;

p_rating product.rating%type;

p_catname category.category%type;

begin

open prod_category;

loop

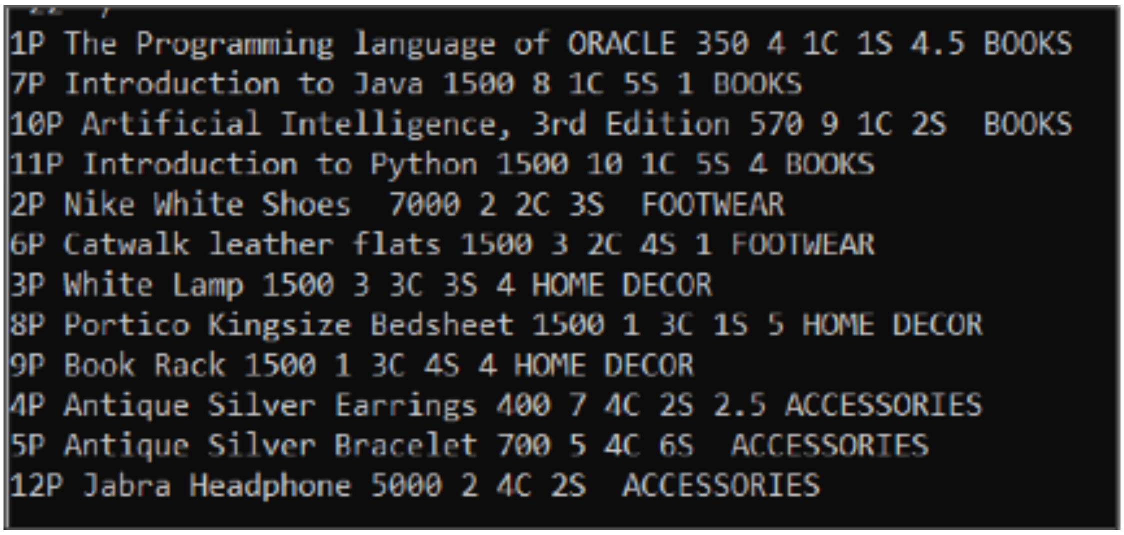
fetch prod_category into p_id,p_name,p_amount,p_qrem,p_catid,p_mid,p_rating,p_catname;

exit when prod_category%NOTFOUND;

dbms_output.put_line(p_id ||' '||p_name||' '||p_amount||' '||p_qrem||' '||p_catid||' '||p_mid||' '||p_rating || ' '||p_catname);

end loop;
close prod_category;

end;
/

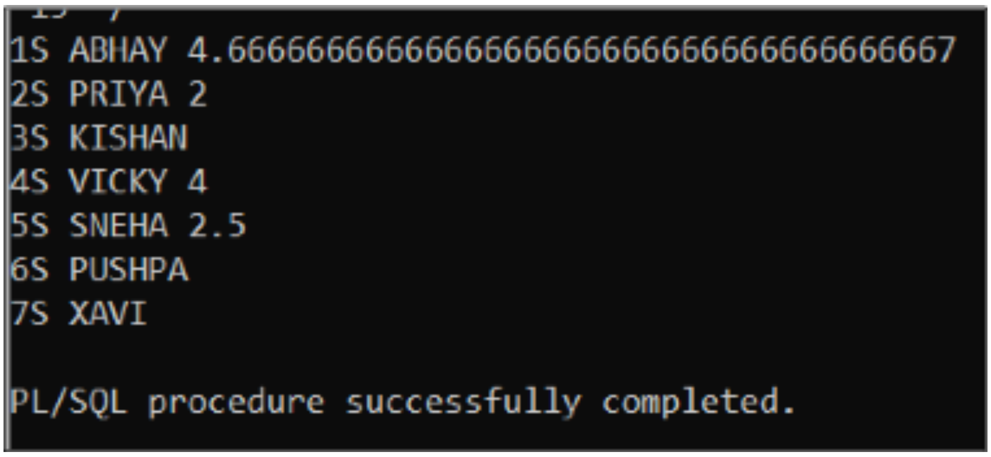


```
1P The Programming language of ORACLE 350 4 1C 1S 4.5 BOOKS
7P Introduction to Java 1500 8 1C 5S 1 BOOKS
10P Artificial Intelligence, 3rd Edition 570 9 1C 2S  BOOKS
11P Introduction to Python 1500 10 1C 5S 4 BOOKS
2P Nike White Shoes  7000 2 2C 3S  FOOTWEAR
6P Catwalk leather flats 1500 3 2C 4S 1 FOOTWEAR
3P White Lamp 1500 3 3C 3S 4 HOME DECOR
8P Portico Kingsize Bedsheet 1500 1 3C 1S 5 HOME DECOR
9P Book Rack 1500 1 3C 4S 4 HOME DECOR
4P Antique Silver Earrings 400 7 4C 2S 2.5 ACCESSORIES
5P Antique Silver Bracelet 700 5 4C 6S  ACCESSORIES
12P Jabra Headphone 5000 2 4C 2S  ACCESSORIES
```

Q4. Display Seller ID, Seller name and Rating of all employees using cursors.

Declare
Cursor merchant_details is select m_id, m_name, rating from merchant;
mer_id merchant.m_id%type;
mer_name merchant.m_name%type;
mer_rating merchant.rating%type;
begin
open merchant_details;
loop
fetch merchant_details into mer_id, mer_name, mer_rating;
exit when merchant_details%NOTFOUND; dbms_output.put_line(mer_id||' '||mer_name||' '||mer_rating);

end loop;
close merchant_details;
end;
/



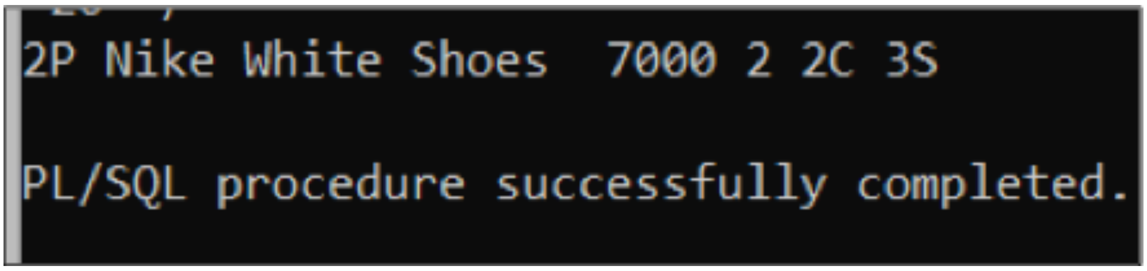
```
1S ABHAY 4.66666666666666666666666666666666667
2S PRIYA 2
3S KISHAN
4S VICKY 4
5S SNEHA 2.5
6S PUSHPA
7S XAVI

PL/SQL procedure successfully completed.
```

Q5. Display Product details having highest Amount using cursor.

Declare
cursor prod_amount is select product_id, product, amount,
quantity_remaining, category_id, m_id,
rating from product where amount=(select max(amount) from product);
p_id product.product_id%type;
p_name product.product%type;
p_amount product.amount%type;
p_qrem product.quantity_remaining%type;
p_catid product.category_id%type;
p_mid product.m_id%type;
p_rating product.rating%type;
begin
open prod_amount;
loop
fetch prod_amount into
p_id,p_name,p_amount,p_qrem,p_catid,p_mid,p_rating;
exit when prod_amount%NOTFOUND;
dbms_output.put_line(p_id ||' '||p_name||' '||p_amount||' '||p_qrem||' '||p_catid||' '||p_mid||' '||p_rating); end loop;
close prod_amount;
end;
/

Q6. Display Rating of all Sellers in descending order using cursor.

```
Declare
Cursor merchant_rating is select rating from merchant order by rating desc;
mer_rating merchant.rating%type;
begin
open merchant_rating;
```



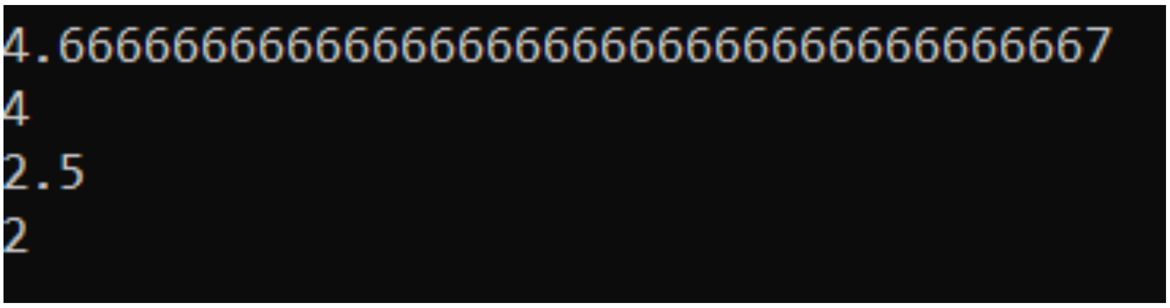
```
2P Nike White Shoes  7000 2 2C 3S

PL/SQL procedure successfully completed.
```

```
loop
fetch merchant_rating into mer_rating; exit when
merchant_rating%NOTFOUND; dbms_output.put_line(mer_rating);
end loop;
close merchant_rating;
end;
/
```

Trigger

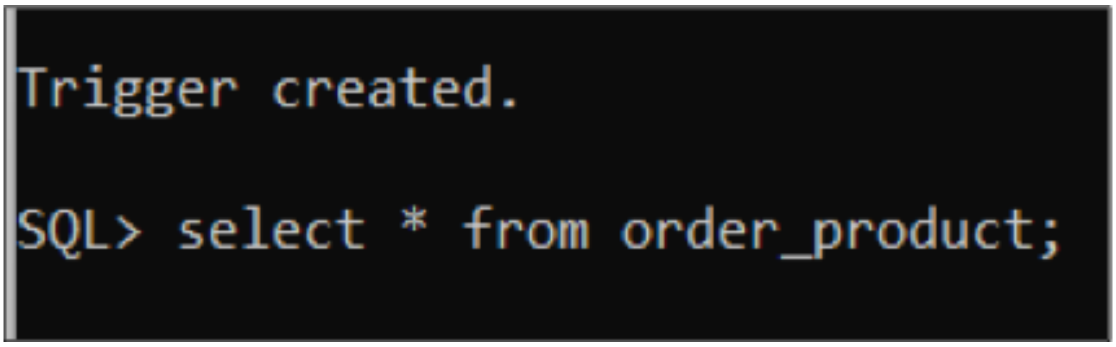
Q1. Create a trigger to update the remaining quantity of product in the product table, when a new entry in order_products table is inserted.

```
Create or replace trigger update_quantity
After insert on order_product
for each row
begin
dbms_output.put_line('Quantity triggered fired. ');
update product set quantity_remaining=quantity_remaining – :new.quantity
where quantity_remaining > 0 AND product_id = :new.product_id;
if sql%rowcount=0 then
dbms_output.put_line('No rows affected. ');
end if;
end update_quantity;
```



```
4.6666666666666666666666666666666666667
4
2.5
2
```

____



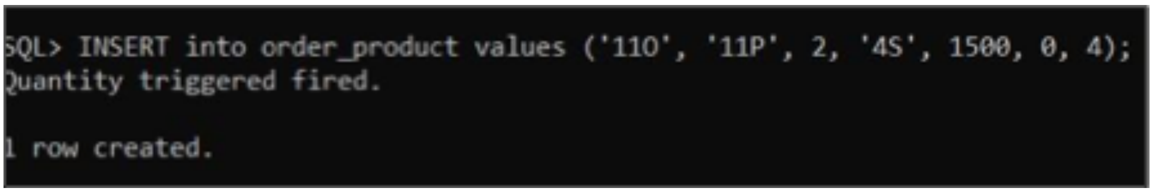
```
Trigger created.

SQL> select * from order_product;
```

INSERT into orders values('11O','5CU',1500,'12-JAN-22'); // inserting in parent table; INSERT into order_product values ('11O', '11P', 2, '4S', 1500, 0, 4);

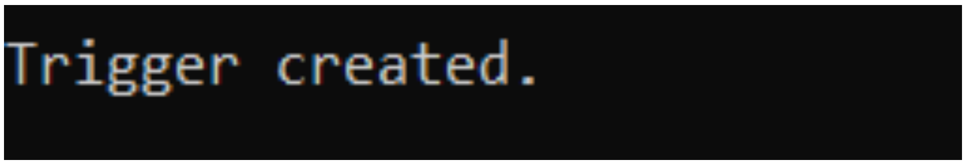
Q2. Create a trigger to update product rating and seller rating when a new entry in the order_products table is inserted.

```
Create or replace trigger rating_update
After insert on order_product

Begin
dbms_output.put_line('Update rating triggered fired. ');
update product p set p.rating = (select avg(rating) from order_product group by product_id having product_id=p.product_id);
update merchant m set m.rating=(select avg(rating) from order_product group by m_id having m_id=m.m_id);
if sql%rowcount=0 then
dbms_output.put_line('No rows affected. ');
end if;
end rating_update;
/
```

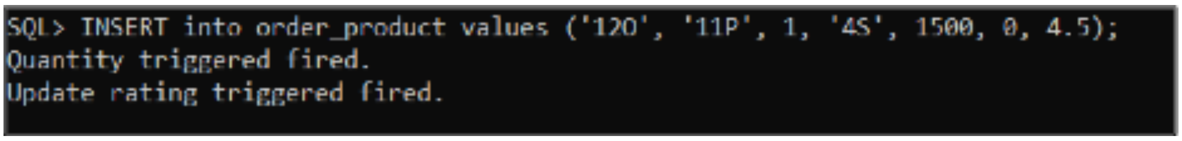
INSERT into orders values('12O','6CU',1500,'18-JAN-22'); // inserting in parent table; INSERT into order_product values ('12O', '11P', 1, '4S', 1500, 0, 4.5);



```
SQL> INSERT into order_product values ('11O', '11P', 2, '4S', 1500, 0, 4);
Quantity triggered fired.

1 row created.
```



```
Trigger created.
```



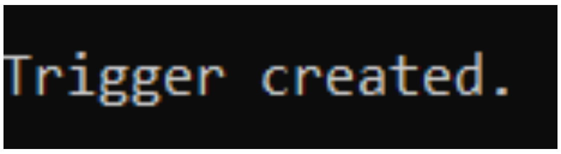
```
SQL> INSERT into order_product values ('12O', '11P', 1, '4S', 1500, 0, 4.5);
Quantity triggered fired.
Update rating triggered fired.
```

Q3. 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.

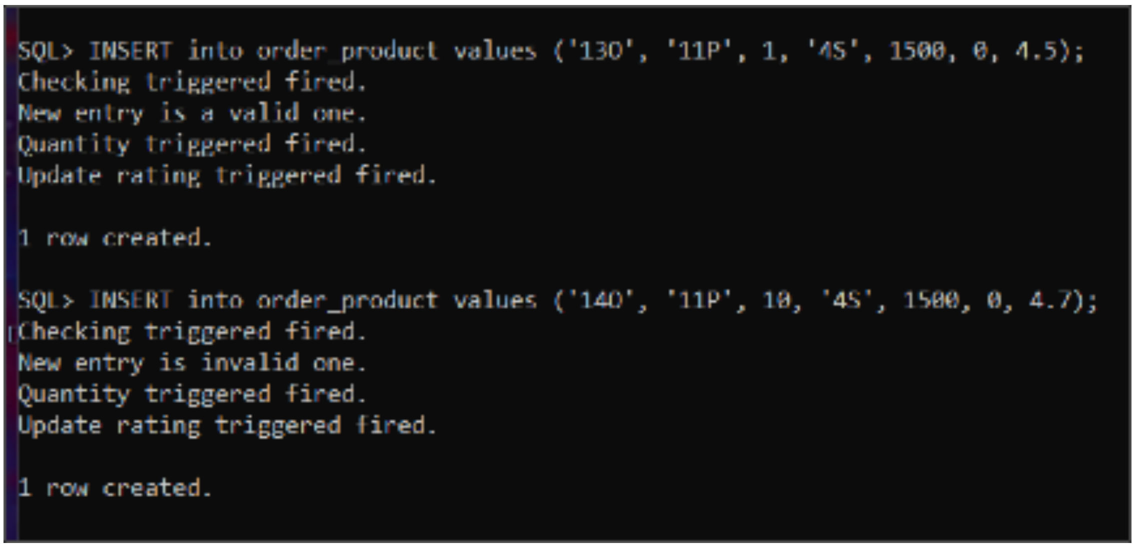
Create or replace trigger check_quantity
After insert on order_product

for each row
declare
quan product.quantity_remaining%type;
begin
dbms_output.put_line('Checking triggered fired. ');
select quantity_remaining into quan from product where product_id=:new.product_id; if(:new.quantity < quan) then
update product set quantity_remaining=quantity_remaining – :new.quantity;
dbms_output.put_line('New entry is a valid one. ');
else
dbms_output.put_line('New entry is invalid one. ');
end if;
if sql%rowcount=0 then
dbms_output.put_line('No rows affected. ');
end if;
end check_quantity;
/

INSERT into orders values('13O','6CU',1500,'18-JAN-22'); // inserting in parent table; INSERT into orders values('14O','7CU',1500,'24-MAR-22'); // inserting in parent table; INSERT into order_product values ('13O', '11P', 1, '4S', 1500, 0, 4.5);
INSERT into order_product values ('14O', '11P', 10, '4S', 1500, 0, 4.7);



```
Trigger created.
```



```
SQL> INSERT into order_product values ('13O', '11P', 1, '4S', 1500, 0, 4.5);
Checking triggered fired.
New entry is a valid one.
Quantity triggered fired.
Update rating triggered fired.

1 row created.

SQL> INSERT into order_product values ('14O', '11P', 10, '4S', 1500, 0, 4.7);
Checking triggered fired.
New entry is invalid one.
Quantity triggered fired.
Update rating triggered fired.

1 row created.
```