***AHMEDABAD UNIVERSITY***

**Program Name : B.Tech - ICT**

**Semester : 4th**

**Course Name : Database Management System Lab**

**Project Title : Online Shopping Management**

**System**

**Group Members : Dhara Vora - 1741046**

**Krushna Shah - 1741086**

**Description of Project :**

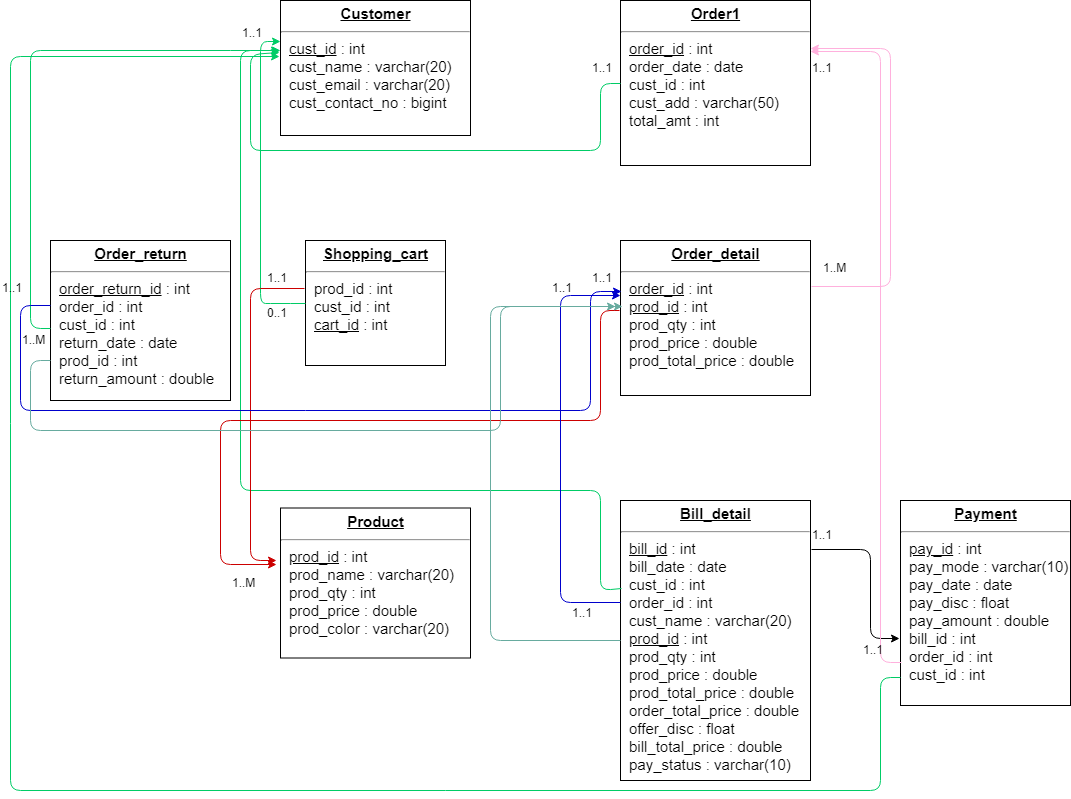
This is a small project for Online Shopping System. The basic idea is that the customer’s can buy products using online. And the administrator can enter the name and generate the receipt of the purchased product and the administrator can also view the yearly,monthly and daily reports of the products.

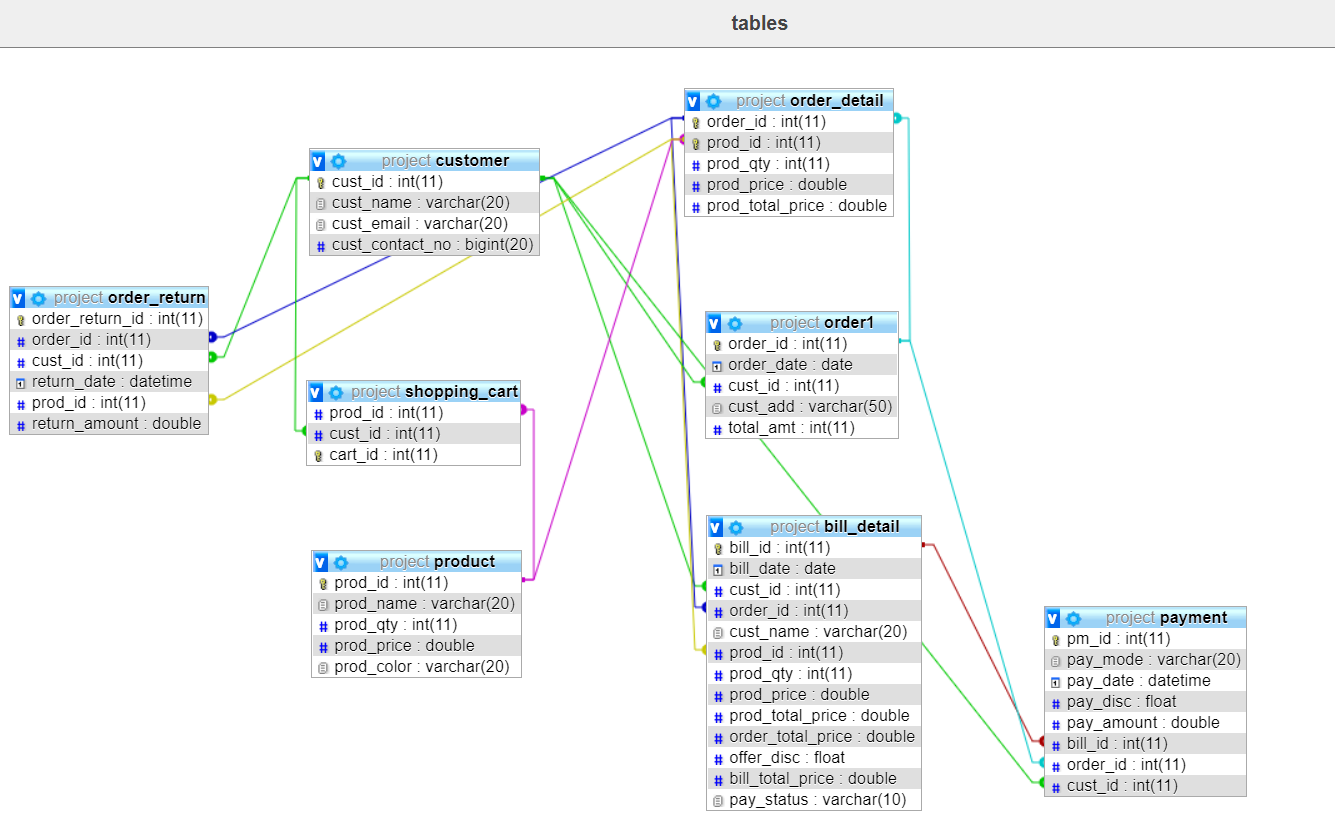
This project is an attempt to provide the advantages of online shopping to customers of a real shop. It helps buying the products in the shop anywhere through internet by using an android device. Thus the customer will get the service of online shopping and home delivery from his favorite shop. This system can be implemented to any shop in the locality or to multinational branded shops having retail outlet chains.

The central concept of the application is to allow the customer to shop virtually using the Internet and allow customers to buy the items and articles of their desire from the store. The information pertaining to the products are stores on the MySQL database. The Server process the customers and the items are shipped to the address submitted by them.

The system was designed into two modules, first is administrator who maintains and updates the information of the product. And the second is customer who wish to buy products. Order which are placed by the customer, will store into the database and according to the order detail, bill will be generated and the payment will be paid by the customer. According to our system, administrator can view the different records of the products, orders, bill details and the payments. Like, year wise order details , day wise placed orders , maximum pay mode used by customers , over all order details , etc.

**Relational Diagram :**

****

****

**Stored Procedures :**

**(1). This procedure will insert the product details into the**

**‘order\_detail’ table. Only those customer can add the data who**

**has placed the order**.

Delimiter $

drop procedure insert\_order\_detail$

create procedure insert\_order\_detail(in order\_id int,in pid int,in qty int)

begin

declare prod\_total\_price,prod\_final\_price,price, order\_total\_price double;

declare discount float;

declare id int;

declare b int;

declare cur1 cursor for select prod\_id, prod\_price from product where prod\_id = pid;

declare continue handler for not found set b = 1;

set order\_total\_price = 0;

open cur1;

set b = 0;

fetch cur1 into id, price;

while b = 0 do

set prod\_total\_price = price\*qty;

insert into order\_detail values(order\_id, pid, qty, price, prod\_total\_price);

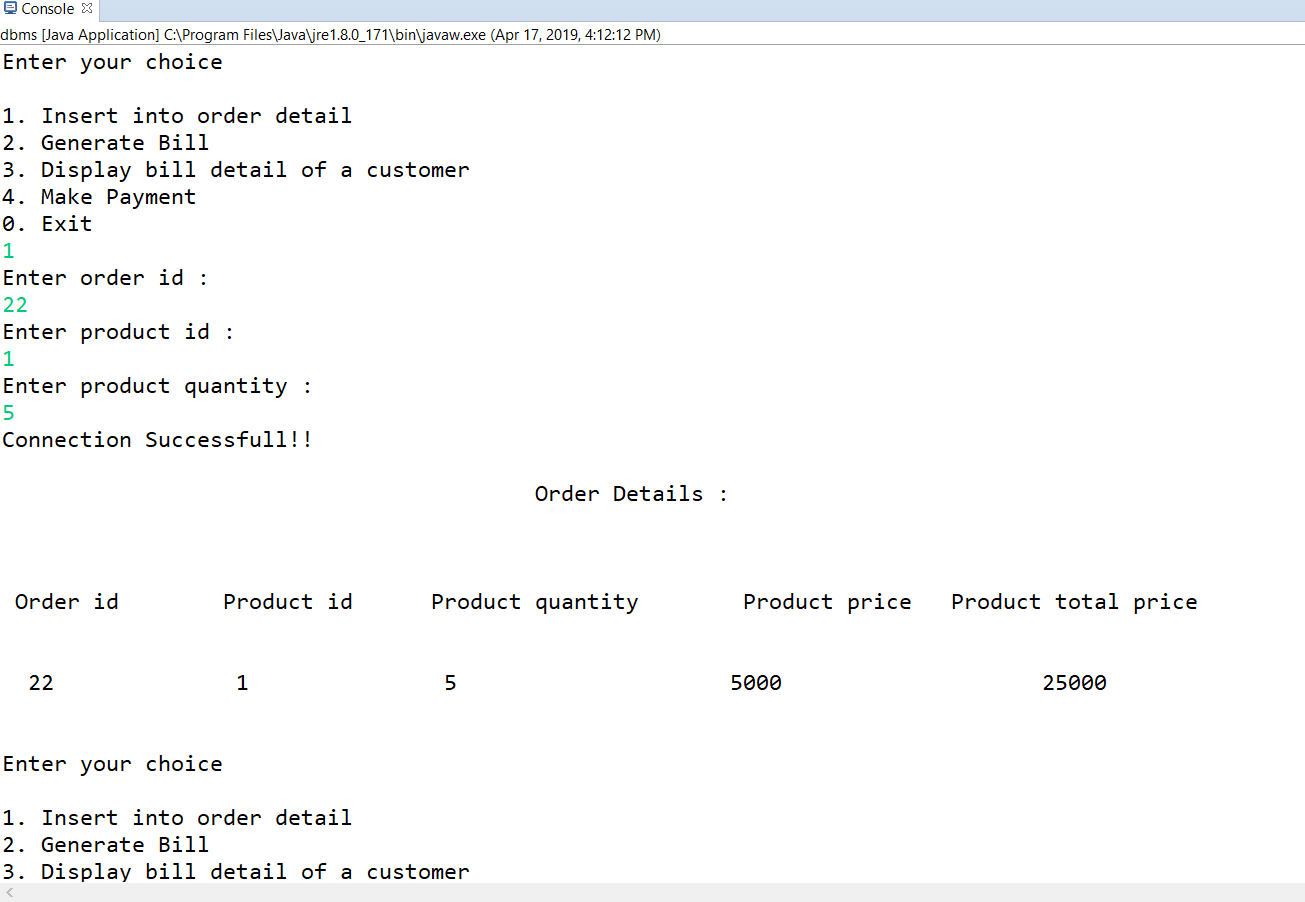
fetch cur1 into id,price;

end while;

close cur1;

end$

call insert\_order\_detail(19, 2, 2)$



**(2) . This procedure will generate the bill of a customer based on the products purchased.**

Delimiter $

drop procedure generate\_bill$

create procedure generate\_bill(in oid int, in i int)

begin

declare b,o\_id,b\_date,day,month,c\_id,pid,pqty , out\_value int;

declare disc float;

declare btp, pprice,ptotal\_price,t\_amt double;

declare c\_name varchar(20);

declare cur1 cursor for select prod\_id,prod\_qty,prod\_price,prod\_total\_price from order\_detail where oid = order\_id;

declare continue handler for not found set b = 1;

open cur1;

set b = 0;

set i = i + 1;

fetch cur1 into pid,pqty,pprice,ptotal\_price;

while b = 0 do

select cust\_id into c\_id from order1 where oid = order1.order\_id;

select cust\_name into c\_name from customer where c\_id = cust\_id;

select total\_amt into t\_amt from order1 where oid = order\_id;

insert into bill\_detail(bill\_id,cust\_id, order\_id, cust\_name, prod\_id, prod\_qty, prod\_price, prod\_total\_price,

order\_total\_price) values(i,c\_id, oid, c\_name, pid, pqty,pprice,ptotal\_price, t\_amt);

select order\_date into b\_date from order1 where oid = order1.order\_id;

update bill\_detail set bill\_date = b\_date where oid = order\_id;

set day = extract(day from(b\_date));

set month = extract(month from(b\_date));

if day = 15 and month = 8 then

set disc = 10.0;

elseif day > 24 and day < 28 and month = 10 then

set disc = 15.0;

elseif day = 14 and month = 1 then

set disc = 20.0;

else

set disc = 5.0;

end if;

update bill\_detail set offer\_disc = disc where oid = order\_id;

set btp = t\_amt - (((t\_amt)\*disc)/100);

update bill\_detail set bill\_total\_price = btp where oid = order\_id;

fetch cur1 into pid,pqty,pprice,ptotal\_price;

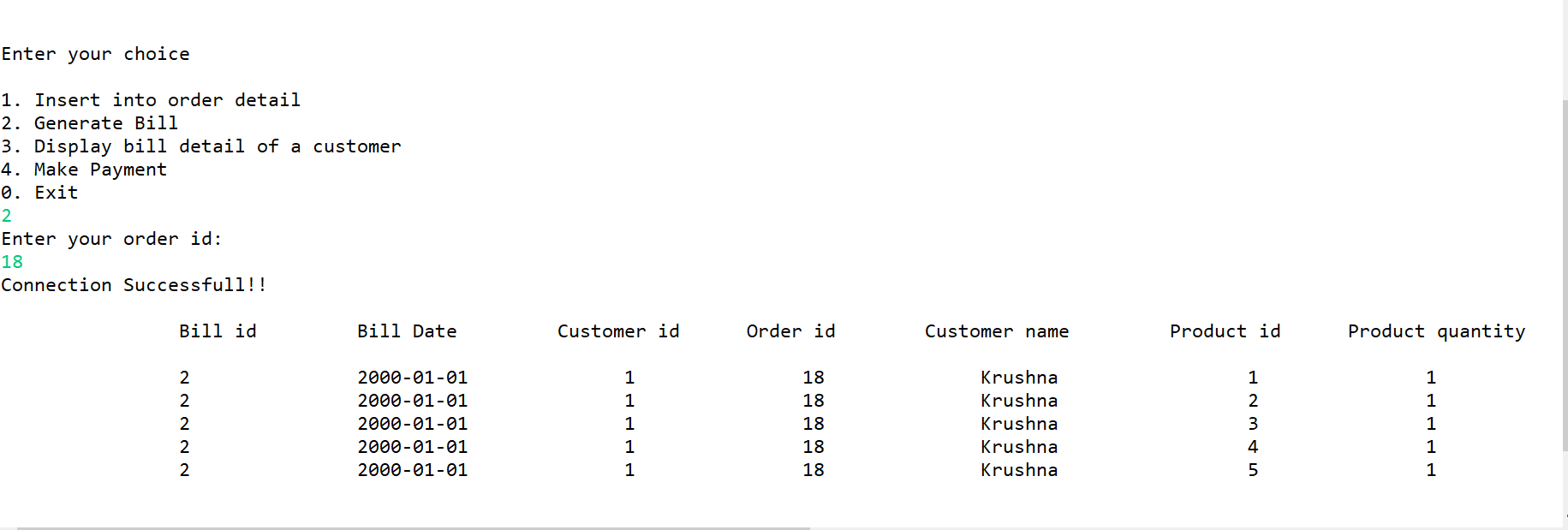
end while;

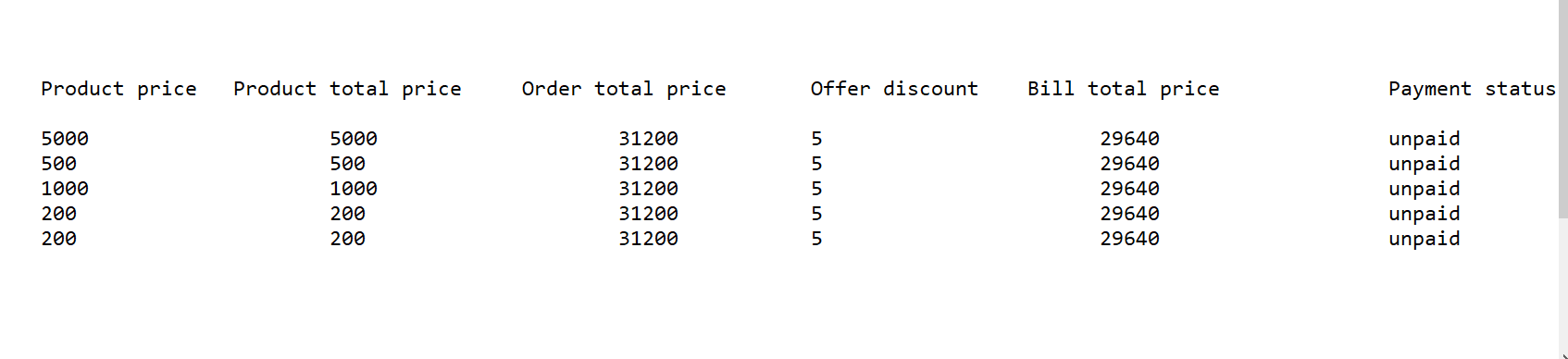
close cur1;

select \* from bill\_detail where oid = order\_id;

end$

call generate\_bill(20, 1)$





**(3) . This procedure will display the year wise order details to the**

**administrator.**

Delimiter $

drop procedure year\_wise\_order\_detail$

create procedure year\_wise\_order\_detail(year year)

begin

select year;

#select \* from order1 where year = extract(year from order1.order\_date);

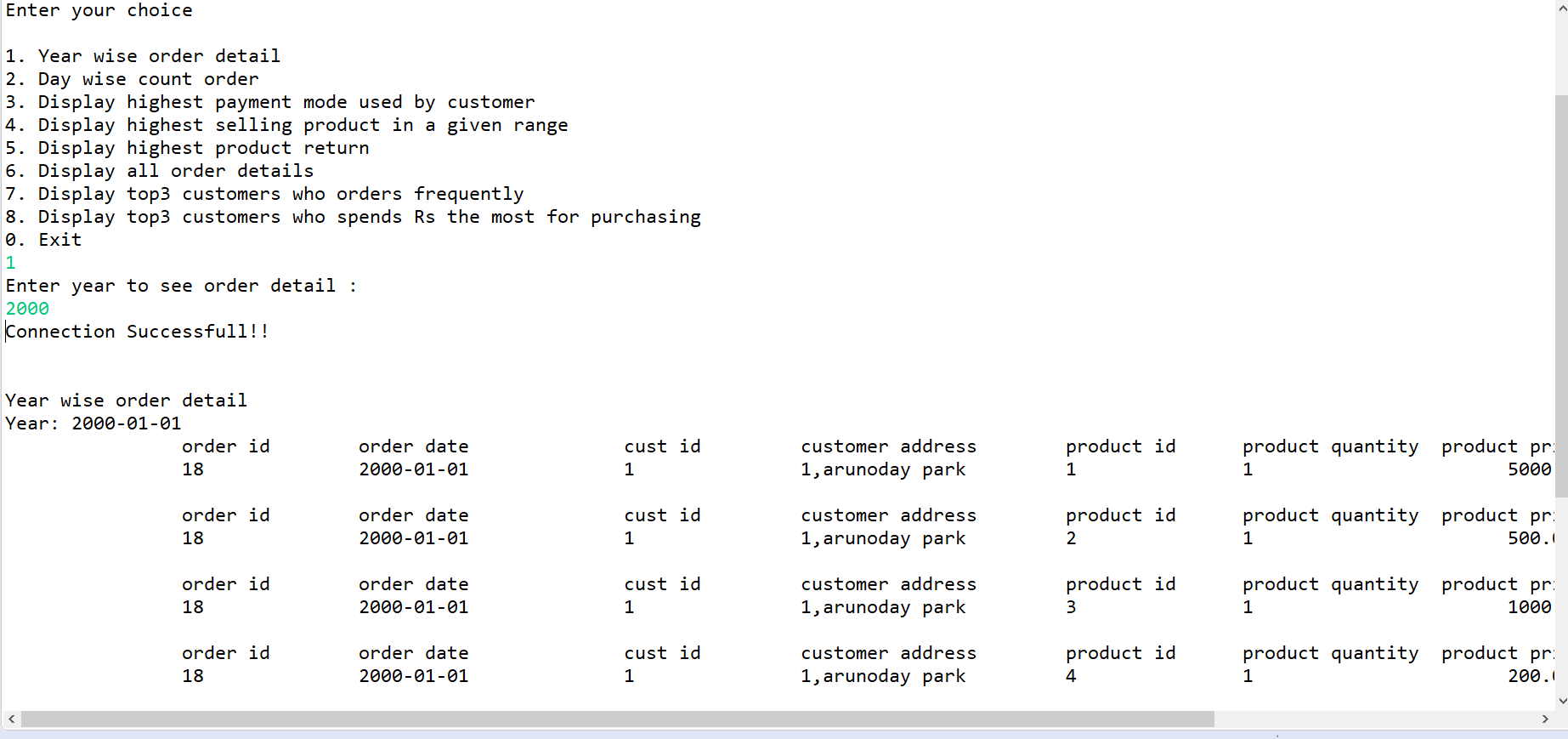
select o.order\_id, o.order\_date, o.cust\_id, o.cust\_add, od.prod\_id, od.prod\_qty, od.prod\_price,

od.prod\_total\_price, o.total\_amt from order1 o inner join order\_detail od

on o.order\_id = od.order\_id and year = extract(year from o.order\_date) ;

end$

call year\_wise\_order\_detail(2000)$



**(4). This procedure will display the customer id wise bill detail to the**

**customer.**

Delimiter $

drop procedure cust\_wise\_bill\_detail$

create procedure cust\_wise\_bill\_detail(in c\_id int)

begin

select distinct cust\_id,cust\_name,order\_id from bill\_detail where c\_id = cust\_id;

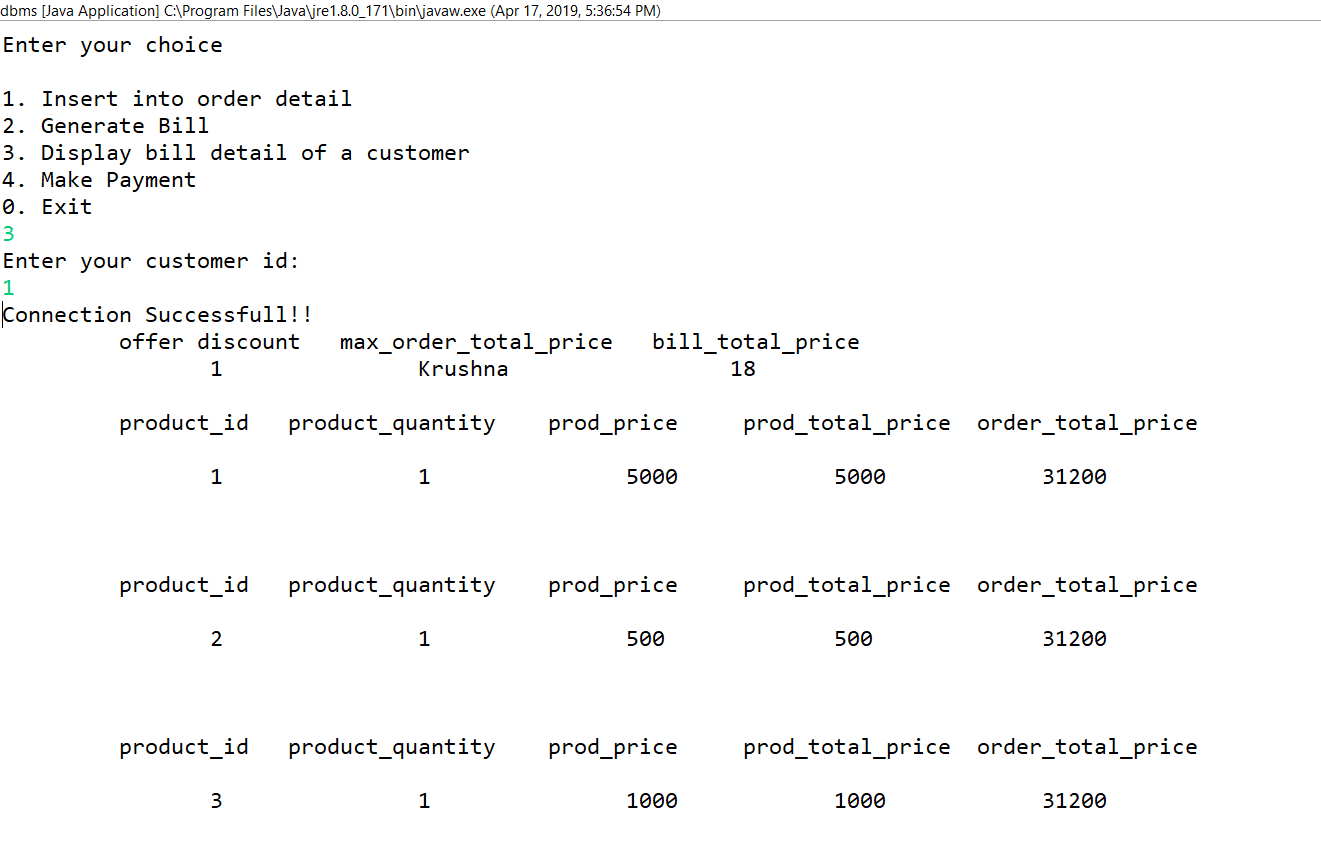
select prod\_id, prod\_qty, prod\_price, prod\_total\_price,

order\_total\_price from bill\_detail where c\_id = cust\_id;

select offer\_disc,max(order\_total\_price),bill\_total\_price from bill\_detail where c\_id = cust\_id;

end$

call cust\_wise\_bill\_detail(3)$



**(5). This procedure will take bill id and the payment mode input from**

**the customer and according to that insert the data into the**

**payment table and according to the bill total amount , offer**

**discount will be generated and display the final amount.**

Delimiter $

drop procedure payment$

create procedure payment(in b\_id int, in pay\_mode varchar(20))

begin

declare cid,oid int;

declare btp double;

declare disc float;

declare note varchar(40);

declare bdate date;

declare pm\_amount double;

select bill\_total\_price into btp from bill\_detail where b\_id = bill\_id;

select cust\_id into cid from bill\_detail where b\_id = bill\_id;

select order\_id into oid from bill\_detail where b\_id = bill\_id;

select bill\_date into bdate from bill\_detail where b\_id = bill\_id;

if btp >1000 then

if pay\_mode = 'cc' then

set disc = 10;

set note = 'Remain valid till 5 days!!';

select disc 'Discount' ;

select note 'Note';

elseif pay\_mode = 'cod' then

set disc = 5;

set note = 'Remain valid till 3 days!!';

select disc 'Discount' ;

select note 'Note';

else

set disc = 0;

set note = 'No Discount';

end if;

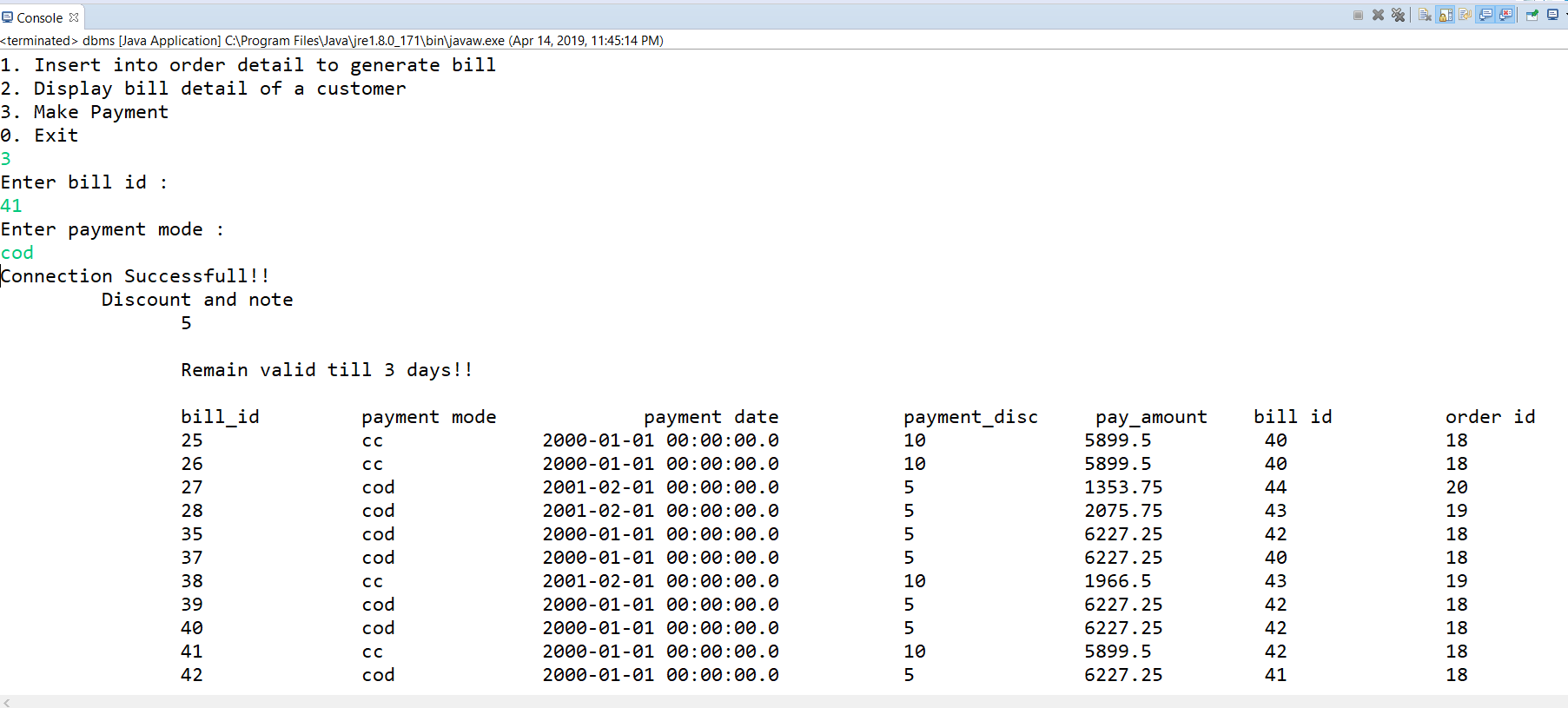
end if;

set pm\_amount = btp - (btp\*disc/100);

insert into payment(pay\_mode,pay\_date,pay\_disc,pay\_amount,bill\_id,order\_id,cust\_id) values(pay\_mode,bdate,disc,pm\_amount,b\_id,oid,cid);

end$

call payment(23, 'cc')$



**(6). This procedure will display the payment\_mode which was more**

**used by the customers.**

Delimiter $

drop procedure pay\_mode$

create procedure pay\_mode()

begin

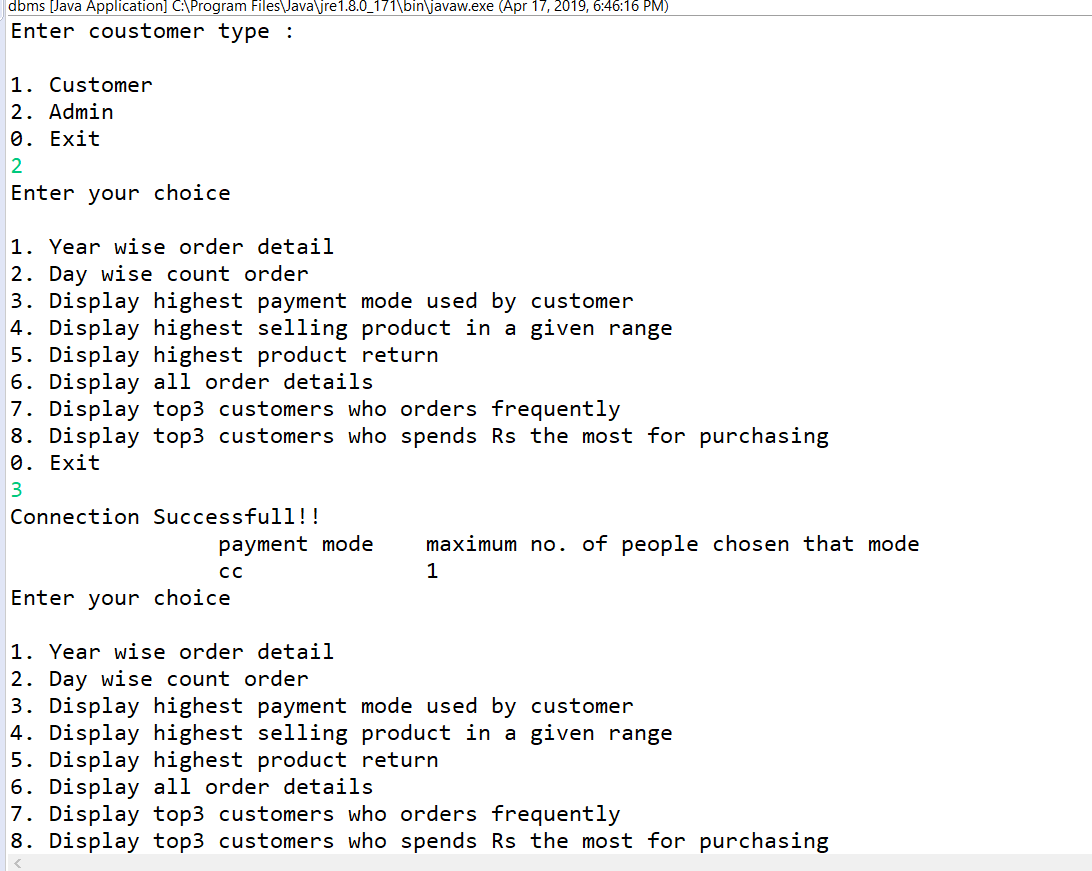
declare cnt int;

select count(pay\_mode) into cnt from payment group by pay\_mode order by count(pay\_mode) desc limit 1;

select pay\_mode, count(pay\_mode) from payment group by pay\_mode having count(pay\_mode) = cnt order by count(pay\_mode);

end$

call pay\_mode$



**(7) . This procedure will display the product which was highest sold.**

Delimiter $

drop procedure highest\_sell$

create procedure highest\_sell(in to\_date date,in from\_date date)

begin

declare o\_date date;

declare b,cnt int;

declare cur1 cursor for select order\_date from order1,order\_detail where

order1.order\_id = order\_detail.order\_id and order\_date >= to\_date and order\_date <= from\_date;

declare continue handler for not found set b = 1;

open cur1;

set b = 0;

fetch cur1 into o\_date;

select order\_id,prod\_id,prod\_qty from order\_detail;

select sum(order\_detail.prod\_qty) into cnt from order\_detail,product where order\_detail.prod\_id = product.prod\_id

group by order\_detail.prod\_id order by sum(order\_detail.prod\_qty) desc limit 1;

select order\_detail.prod\_id as 'Product ID', product.prod\_name as 'Product Name',

sum(order\_detail.prod\_qty) as 'Maximum quantity sold' from order\_detail,product

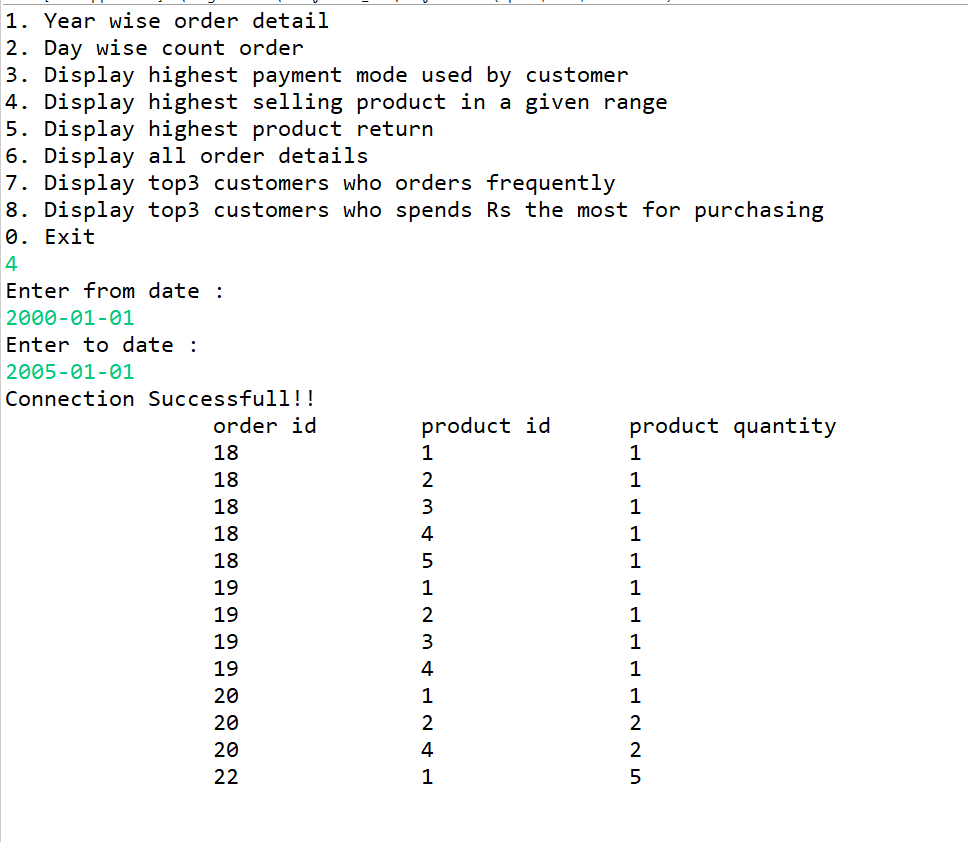
where order\_detail.prod\_id = product.prod\_id group by order\_detail.prod\_id

having sum(order\_detail.prod\_qty) = cnt order by sum(order\_detail.prod\_qty);

close cur1;

end$

call highest\_sell('2000-02-02','2002-02-02')$



**(8). This procedure will display the product which was highest return**

**by the different or same customers.**

Delimiter $

drop procedure highest\_prod\_return$

create procedure highest\_prod\_return()

begin

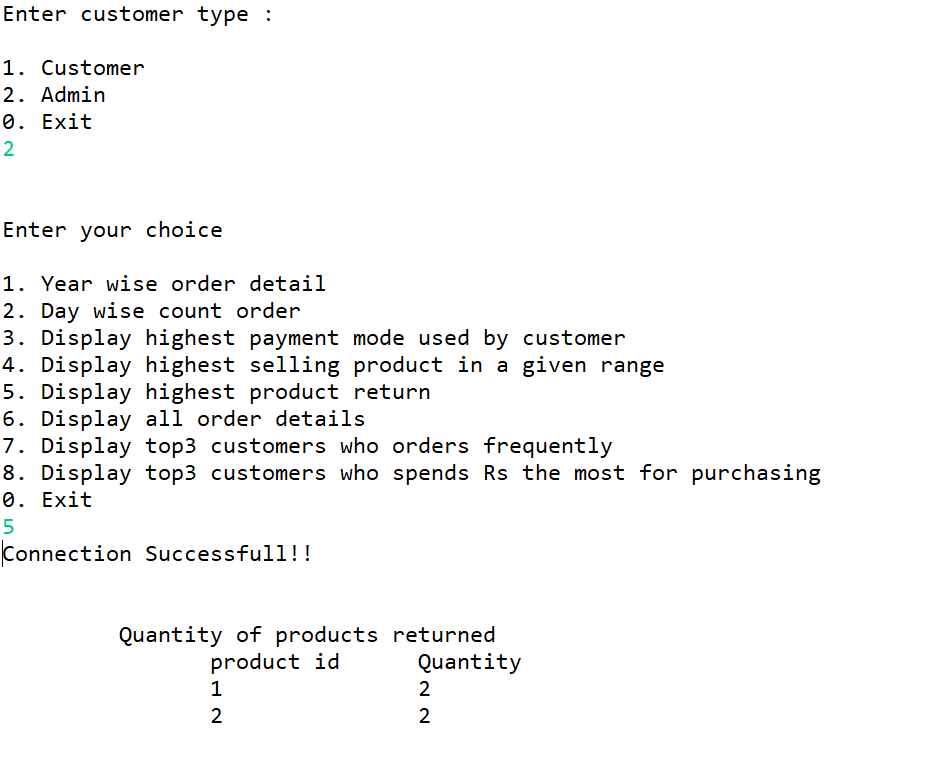
declare cnt int;

select count(prod\_id) into cnt from order\_return group by prod\_id order by count(prod\_id) desc limit 1;

select prod\_id, count(prod\_id) from order\_return group by prod\_id having count(prod\_id) = cnt order by count(prod\_id);

end$

call highest\_prod\_return$



**(9). This procedure will manage the products which are in the**

**shopping Cart. As soon as the shopping cart product will add into**

**the order detail, the product will be delete from the shopping cart.**

Delimiter $

drop procedure manage\_shopping\_cart$

create procedure manage\_shopping\_cart(in pid int, in cid int)

begin

delete from shopping\_cart where prod\_id = pid and cust\_id = cid;

end$

Call manage\_shopping\_cart$

**NOTE**: This procedure is being called from the trigger (i.e. after insert on order\_detail).

**(10). This procedure will display the overall order details based on year**

**and month.**

Delimiter $

drop procedure orderdetail$

create procedure orderdetail()

begin

declare b int;

declare odate date;

declare cur1 cursor for select order1.order\_date from order1 inner join order\_detail

on order1.order\_id= order\_detail.order\_id group by order\_detail.order\_id order by order1.order\_date;

declare continue handler for not found set b = 1;

open cur1;

set b = 0;

fetch cur1 into odate;

while b = 0 do

select extract(year from odate);

select extract(month from odate);

select \* from order1 inner join order\_detail on order1.order\_id = order\_detail.order\_id

where order1.order\_date = odate;

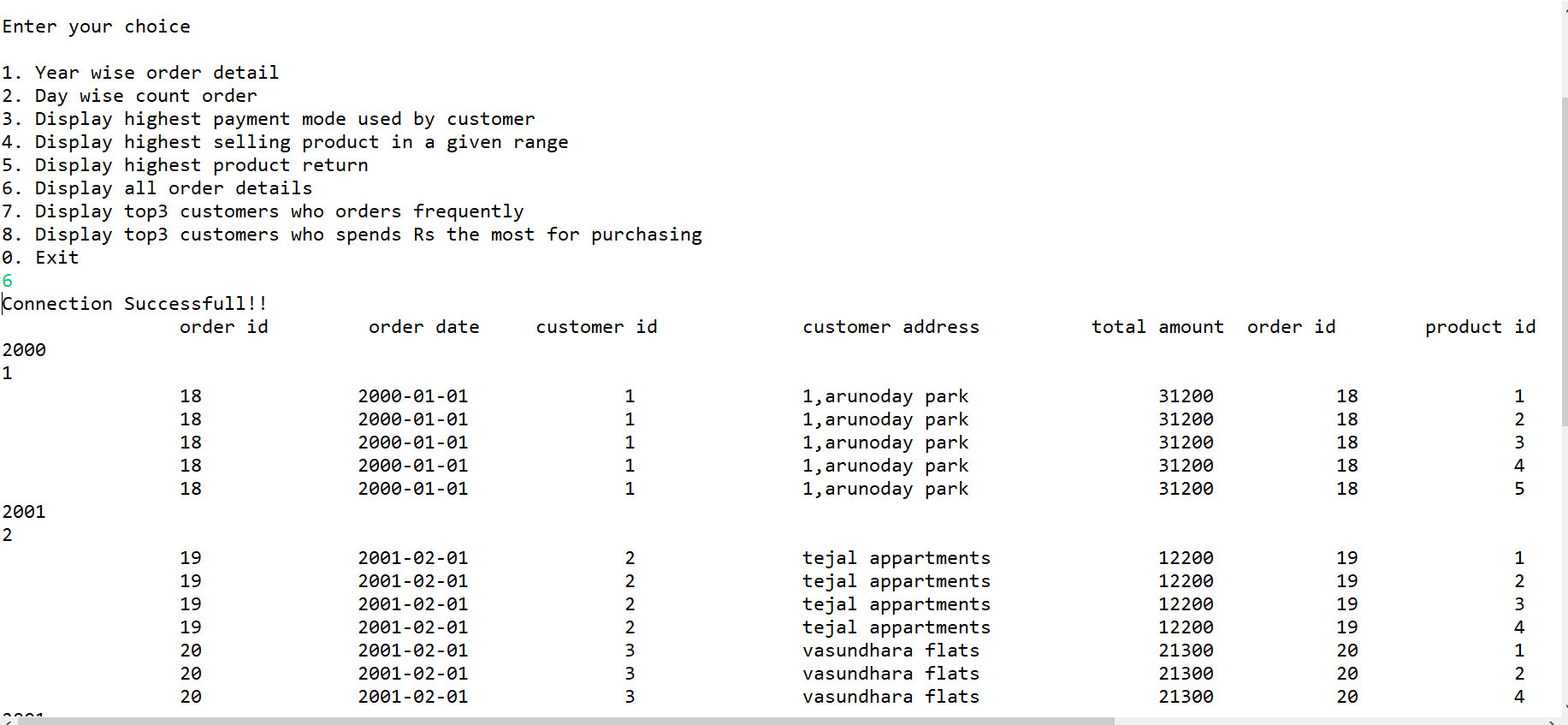
fetch cur1 into odate;

end while;

close cur1;

end$

call orderdetail$

****

**(11). This procedure will display the list of top 3 customers who orders more frequently.**

Delimiter $

drop procedure top3\_customers\_based\_on\_order$

create procedure top3\_customers\_based\_on\_order()

begin

declare count int;

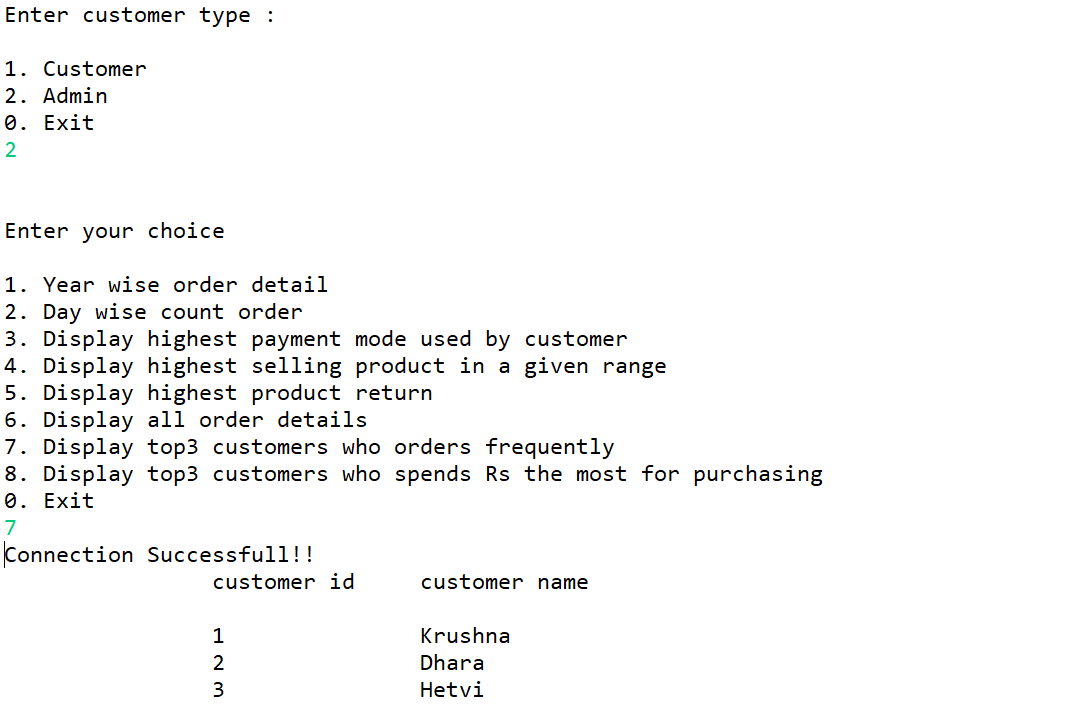
select distinct order1.cust\_id, customer.cust\_name, count(order\_detail.order\_id) as count from order1,customer,order\_detail

where order1.order\_id = order\_detail.order\_id and customer.cust\_id = order1.cust\_id

group by order1.cust\_id order by count(order\_detail.order\_id) desc limit 3;

end$

call top3\_customers\_based\_on\_order$

****

**(12). This procedure will display top 3 customers who spends more money for purchasing the products.**

Delimiter $

drop procedure top3\_customers\_based\_on\_amount$

create procedure top3\_customers\_based\_on\_amount()

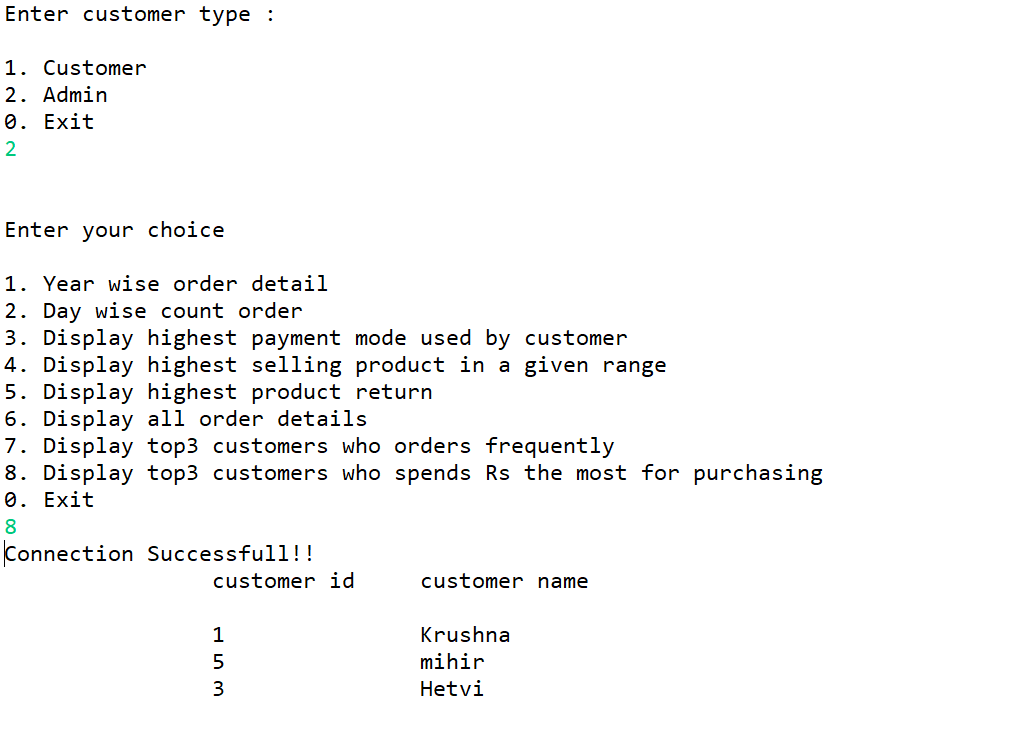
begin

select distinct order1.cust\_id, customer.cust\_name, total\_amt from order1, order\_detail, customer

where order1.order\_id = order\_detail.order\_id and customer.cust\_id = order1.cust\_id order by total\_amt desc limit 3;

end$

call top3\_customers\_based\_on\_amount$

****

**Stored Function :**

**(1). This function will return the number of orders placed in a given day.**

Delimiter $

drop function day\_wise\_cnt\_order$

create function day\_wise\_cnt\_order(date date) returns int

begin

declare order\_cnt int;

#select date;

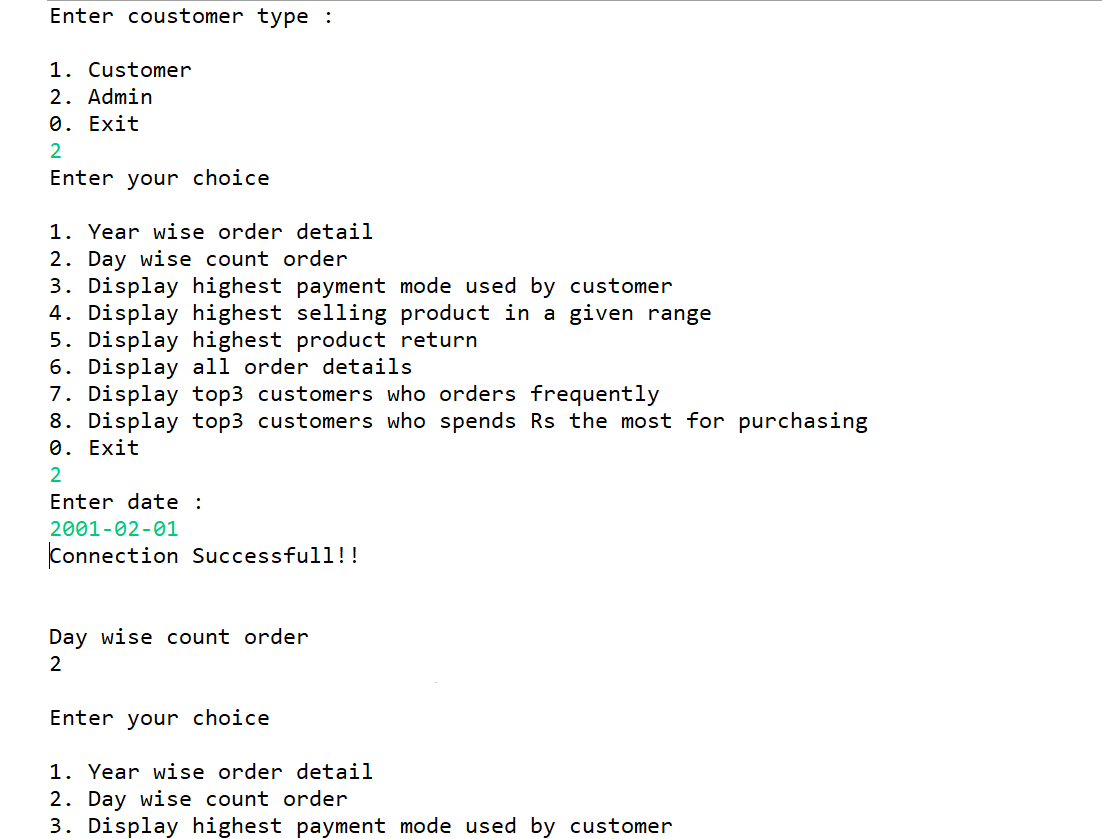
select count(distinct order1.order\_id) into order\_cnt from order1 inner join order\_detail

on order1.order\_id = order\_detail.order\_id and date = order1.order\_date group by order\_date;

return (order\_cnt);

end$

Select day\_wise\_cnt\_order(‘2001-02-02’)$



**Stored Triggers :**

**(1). This trigger will fired after insert on the order detail table. This**

**trigger Will update the total amount value in the order table,**

**update the , Product quantity in the product table and insert data**

**into the bill details.**

delimiter $

drop trigger err\_ins1$

create trigger err\_ins1 after insert on order\_detail

for each row

begin

declare c\_id int;

update order1 set total\_amt = total\_amt + new.prod\_total\_price where order\_id=new.order\_id;

update product set prod\_qty = prod\_qty - new.prod\_qty where prod\_id = new.prod\_id;

select cust\_id into c\_id from order1 where order1.order\_id = new.order\_id;

call manage\_shopping\_cart(new.prod\_id, c\_id);

end$

call insert\_order\_detail(20,1,1)$

**(2). This trigger will fired if the customer’s desired quantity is greater**

**than the total available quantity.**

delimiter $

drop trigger err\_ins2$

create trigger err\_ins2 before insert on order\_detail

for each row

begin

declare msg varchar(128);

declare p\_qty int;

select distinctrow product.prod\_qty into p\_qty from product inner join order\_detail on new.prod\_id = product.prod\_id;

#set pid = select prod\_id from product where prod\_id = new.prod\_id;

if p\_qty < new.prod\_qty then

set msg = 'Not enough quantity.....';

elseif new.prod\_qty < 0 then

set msg = 'Quantity can not be negative.....';

end if;

signal sqlstate '45001' set message\_text = msg;

end$

**(3). This trigger will update the payment status after the payment**

**make by the customer.**

Delimiter $

drop trigger pay\_status$

create trigger pay\_status after insert on payment

for each row

begin

update bill\_detail set pay\_status = 'Paid' where bill\_id = new.bill\_id;

end$

**(4). This trigger will update the product quantity in the product table**

**after the order return.**

delimiter $

drop trigger qty\_return$

create trigger qty\_return after insert on order\_return

for each row

begin

declare p\_qty int;

select prod\_qty into p\_qty from order\_detail where order\_detail.order\_id = new.order\_id and order\_detail.prod\_id = new.prod\_id;

update product set prod\_qty = prod\_qty + p\_qty where prod\_id = new.prod\_id;

end$