Thapar Institute of Engineering and Technology

Electrical and computer Engineering Department

Database Management System - Project file

(UEC617)

Submitted to – Mr. Punit Kumar

Team members:

Chirag Arora (102065031)

Garvit Mittal(102195004)

INDEX

Ecommerce Management System

Problem Statement

In this advanced time of internet shopping no seller wants to be left behind, moreover due to its simplicity the shift from offline selling model to an online selling model is witnessing a rampant growth. Along these lines, as an engineer our responsibility is to facilitate the way of this change for the dealer or seller. Among numerous things that an internet based webpage requires the most significant is a data set framework. Henceforth in this venture we are wanting to plan a database where little attire merchants can sell their item on the web.

Structure of the Model

- A new user can register on the website.
- A customer can see details of the product present in the cart
- A customer can view his order history.
- Admin can start a sale with certain discount on every product.
- Customer can filter the product based on the product details.
- A customer can add or delete a product from the cart.
- A seller can unregister/ stop selling his product.
- A seller/ customer can update his details.
- Admin can view the products purchased on particular date.
- Admin can view number of products sold on a particular date.
- A customer can view the total price of product present in the cart unpurchased.
- Admin can view details of customer who have not purchased anything.
- Admin can view total profit earned from the website.

Entity Relationship Description

Entity Relation Diagram includes entity sets, attributes and relationships between the entity sets. An Entity could be described as an object of some kind and collection of such objects is called an entity set. Attributes are the properties of each entity set and relationships are the connections among two or more entity sets.

Our model consists of 6 entities namely Customer, Payment, Cart, Seller, Product and Cart_item.

1. Customer consists of following attributes: -
    - Customer_id (acts as a primary key)
    - C_pass
    - Name
    - Address
    - Pincode
    - Phone_no

2. Payment consists of following attributes: -
    - payment_id (acts as a primary key)
    - Total_Amount
    - Payment_date

3. Cart entity consists of cart_id acting as primary key.

4. Cart_item(Weak entity set) consists of following attributes: -
    - Purchased
    - Quantity_wished
    - Date_Added

5. Product consists of following attributes: -
    - Product_id (Primary Key)
    - size
    - Age_group
    - Type
    - Color
    - Gender
    - Commission
    - Cost

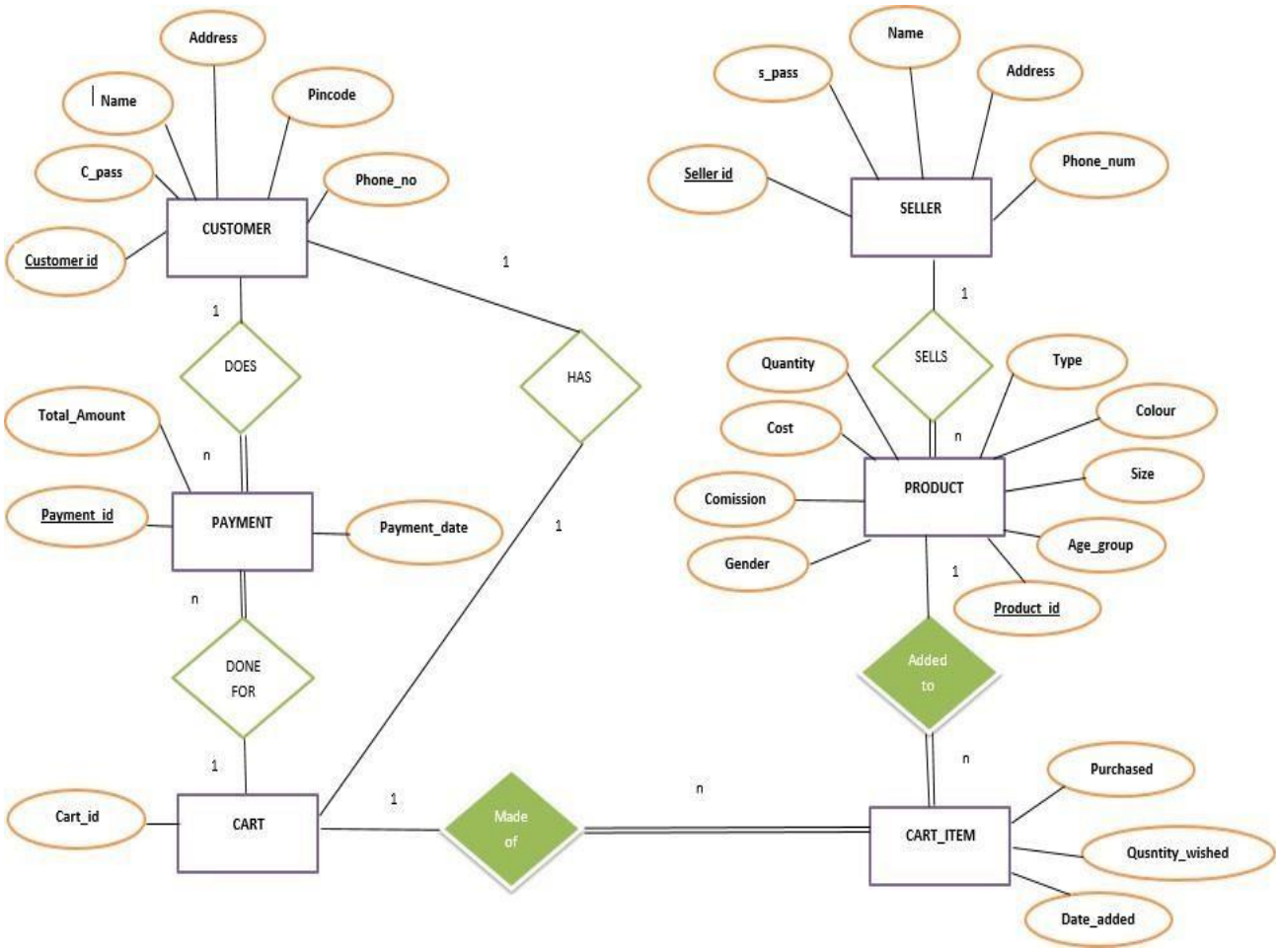
- Quantity

6. Seller consists of following attributes: -
    - Seller_id (Primary key)
    - s_pass
    - Name
    - Address
    - Phone_number(multivalued attribute)

The model consists of 6 relationships namely Does, Has, Done for, made of, Added to and Sets: -

- Customer and Payment have 1 to many “Does” relationship.
- Customer and Cart have 1 to 1 “Has” relationship.
- Payment and Cart have many to one “Done” for relationship.
- Cart and Cart_item have 1 to many “made of” relationship.
- Cart_item and Product have many to one “Added to” relationship.
- Product and Seller have many to one “Sets” relationship.

Entity Relation Diagram



ER Diagram to Tables

Customer_id	C_pass	Name	Address	Pincode	Phone_no.

Table schema: **CUSTOMER** (Customer_id, C_pass, Name, Address, Pincode, Phone_no.)

Payment_id	Total_Amount	Payment_date

Table schema: **PAYMENT** (Payment_id, Total_Amount, Payment_date)

Cart_id

Table schema: **CART** (Cart_id)

Purchased	Quantity_wished	Date_added

Table schema: **CART_ITEM** (Purchased, Quantity_wished, Date_added)

Product_id	Size	Age_group	Type	Colour	Gender	Commission	Cost	Quantity

Table schema: **PRODUCT** (Product_id, Size, Age_group , Type, Colour, Gender, Commission, Cost, Quantity)

Seller_id	s_pass	Name	Address	Phone_num

Table schema: **SELLER** (Seller_id, s_pass, Name, Address, Phone_num)

Converting Relationship Set to Table

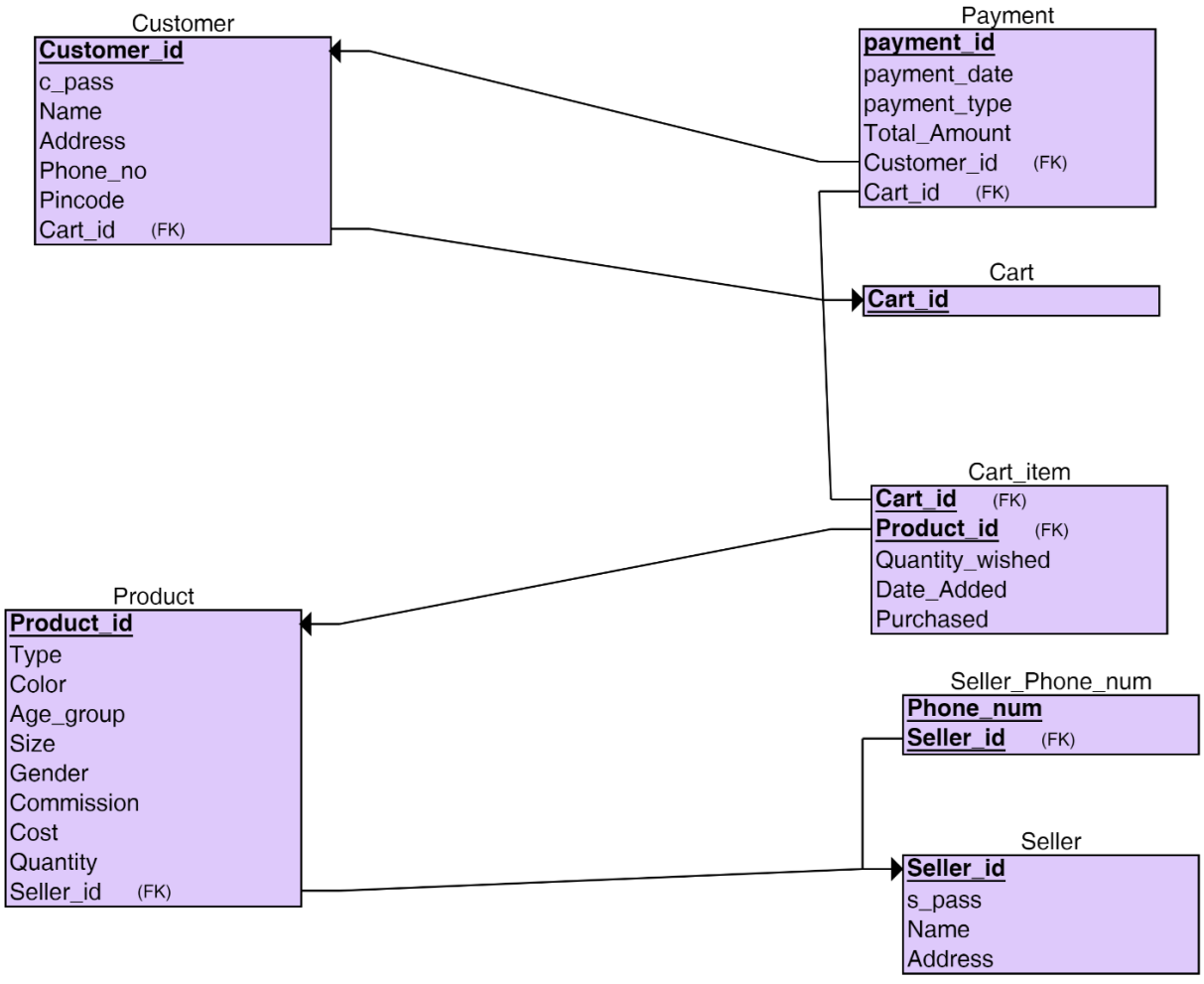
Customer_id	Payment_id

Payment_id	Cart_id

Cart_id	Product_id	Purchased	Quantity_wished	Date_added

Product_id	Seller_id

Relational Database Schema



SQL/PLSQL code

Creating Tables

```
CREATE TABLE Cart
   (
      Cart_id VARCHAR(7) NOT NULL,
      PRIMARY KEY(Cart_id)
   );

   CREATE TABLE Customer
   (
      Customer_id VARCHAR(6) NOT NULL,
      c_pass VARCHAR(10) NOT NULL,
      Name VARCHAR(20) NOT NULL,
      Address VARCHAR(20) NOT NULL,
      Pincode NUMBER(6) NOT NULL,
      Phone_number_s number(10) NOT NULL,
      PRIMARY KEY (Customer_id),
      Cart_id VARCHAR(7) NOT NULL,
      FOREIGN KEY(Cart_id) REFERENCES cart(Cart_id)
   );

   CREATE TABLE Seller
   (
      Seller_id VARCHAR(6) NOT NULL,
      s_pass VARCHAR(10) NOT NULL,
      Name VARCHAR(20) NOT NULL,
      Address VARCHAR(10) NOT NULL,
      PRIMARY KEY (Seller_id)
   );

   CREATE TABLE Seller_Phone_num
   (
      Phone_num NUMBER(10) NOT NULL,
      Seller_id VARCHAR(6) NOT NULL,
      PRIMARY KEY (Phone_num, Seller_id),
      FOREIGN KEY (Seller_id) REFERENCES Seller(Seller_id)
      ON DELETE CASCADE
   );

   CREATE TABLE Payment
```

```
(
   payment_id VARCHAR(7) NOT NULL,
   payment_date DATE NOT NULL,
   Payment_type VARCHAR(10) NOT NULL,
   Customer_id VARCHAR(6) NOT NULL,
   Cart_id VARCHAR(7) NOT NULL,
   PRIMARY KEY (payment_id),
   FOREIGN KEY (Customer_id) REFERENCES Customer(Customer_id),
   FOREIGN KEY (Cart_id) REFERENCES Cart(Cart_id),
   total_amount numeric(6)
);

CREATE TABLE Product
(
   Product_id VARCHAR(7) NOT NULL,
   Type VARCHAR(7) NOT NULL,
   Color VARCHAR(15) NOT NULL,
   P_Size VARCHAR(2) NOT NULL,
   Gender CHAR(1) NOT NULL,
   Commission NUMBER(2) NOT NULL,
   Cost NUMBER(5) NOT NULL,
   Quantity NUMBER(2) NOT NULL,
   Seller_id VARCHAR(6),
   PRIMARY KEY (Product_id),
   FOREIGN KEY (Seller_id) REFERENCES Seller(Seller_id)
   ON DELETE SET NULL
);

CREATE TABLE Cart_item
(
   Quantity_wished NUMBER(1) NOT NULL,
   Date_Added DATE NOT NULL,
   Cart_id VARCHAR(7) NOT NULL,
   Product_id VARCHAR(7) NOT NULL,
   FOREIGN KEY (Cart_id) REFERENCES Cart(Cart_id),
   FOREIGN KEY (Product_id) REFERENCES Product(Product_id),
   Primary key(Cart_id,Product_id)
);

alter table Cart_item add purchased varchar(3) default 'NO';
```

Inserting Values

These are some demo values. Full data will be updated in future commits

```
insert into Cart values('crt1011');

insert into Customer values('cid100','ABCM1235','rajat','G-453','632014',9893135876,
'crt1011');

insert into Seller values('sid100','12345','aman','delhi cmc');

insert into Product values('pid1001','jeans','red','32','M',10,10005,20,'sid100');

insert into Seller_Phone_num values('9943336206','sid100');

insert into Cart_item values(3,to_date('10-OCT-1999','dd-mon-
yyyy'),'crt1011','pid1001','Y');

insert into Payment values('pmt1001',to_date('10-OCT-1999','dd-mon-
yyyy'),'online','cid100','crt1011',NULL)
```

Basic Queries

1 If the customer wants to see details of product present in the cart

```
select * from product where product_id in(
  select product_id from Cart_item where (Cart_id in (
     select Cart_id from Customer where Customer_id='cid100'
  ))
and purchased='NO');
```

2. If a customer wants to see order history

```
select product_id,Quantity_wished from Cart_item where (purchased='Y' and Cart_id
in (select Cart_id from customer where Customer_id='cid101'));
```

3. **Customer wants to see filtered products on the basis of size,gender,type**

```
select product_id, color, cost, seller_id from product where (type='jeans' and
```

p_size='32' and gender='F' and quantity>0)

4. If customer wants to modify the cart

delete from cart_item where (product_id='pid1001' and Cart_id in (select cart_id from Customer where Customer_id='cid100'));

5. If a seller stops selling his product

delete from seller where seller_id = 'sid100';
update product set quantity = 00 where seller_id is NULL;

6. If admin want to see what are the product purchased on the particular date

select product_id from cart_item where (purchased='Y' and date_added='12-dec-2018')

PL/SQL function

Procedure which returns the type of product with the cost less than the given cost

```
create or replace procedure cost_filter(c in number,t in varchar)
is
cs product.cost%type;
ty product.type%type;
id product.product_id%type;
cursor cf is
select product_id,cost,type from product where cost<c and type=t;
begin
open cf;
loop
fetch cf into id,cs,ty;
exit when cf%notfound;
dbms_output.put_line('Product' || id || 'has cost ' || cs || ' and the type is' || ty);
end loop;
close cf;
exception
when no_data_found then
dbms_output.put_line('Sorry no such products exist');
```

Function which returns total number of products which a particular seller sells

```
create or replace function totalProducts(sId in varchar)
return number
is
total number(2):=0;
begin
select count(*) into total
from product
where seller_id=sId;
return total;
end;
```

Function execution:

```
declare
c number(2);
begin
c:=totalProducts('sid102');
dbms_output.put_line('Total products is : '|| c);
end;
```

Procedure which returns the total quantity of product with the given ID

Procedure with exception handling

```
create or replace procedure prod_details(p_id in varchar)
is
quan number(2);
begin
select quantity into quan from product where product_id=p_id;
exception
when no_data_found then
dbms_output.put_line('Sorry no such product exist !!');
end;
```

Triggers

Trigger that will execute before inserting new customer to database and inserting a new cartId to the cart_items table

Function to count number of cart items

```
create or replace function numCartId(cd in varchar)
return number
is
total number(2):=0;
begin
select count(*) into total
from cart_item
```

```
where cart_id=cd;
return total;
end;
Trigger
Create or replace trigger before_customer
before insert
on
customer
for each row
declare
c varchar(10);
n number(2);
begin
c:=  :new.cart_id;
n:=numCartId(c);
if n>0 then
dbms_output.put_line('Sorry');
end if;
insert into cart values(c);
end;
end;
```

Trigger to update the total amount of user everytime he adds something to payment table

```
create or replace function total_cost(cId in varchar)
return number
is
total number(2) :=0;
begin
select sum(cost) into total from product,cart_item where
product.product_id=cart_item.product_id and cart_id=cId;
return total;
end;

create or replace trigger before_pay_up
before insert
on
payment
for each row
declare
total number(3);
```

```
    begin
    total :=total_cost(:new.cart_id);
    insert into payment
values(:new.payment_id,:new.payment_date,:new.payment_type,:new.customer_id,:new
.cart_id,total);
    end;
```

Output screenshots

1. Creating tables

```
CREATE TABLE Cart
   (
      Cart_id VARCHAR(7) NOT NULL,
      PRIMARY KEY(Cart_id)
   );

   CREATE TABLE Customer
   (
      Customer_id VARCHAR(6) NOT NULL,
      c_pass VARCHAR(10) NOT NULL,
      Name VARCHAR(20) NOT NULL,
      Address VARCHAR(20) NOT NULL,
      Pincode NUMBER(6) NOT NULL,
      Phone_number_s number(10) NOT NULL,
      PRIMARY KEY (Customer_id),
      Cart_id VARCHAR(7) NOT NULL,
      FOREIGN KEY(Cart_id) REFERENCES cart(Cart_id)
   );

   CREATE TABLE Seller
   (
      Seller_id VARCHAR(6) NOT NULL,
      s_pass VARCHAR(10) NOT NULL,
      Name VARCHAR(20) NOT NULL,
      Address VARCHAR(10) NOT NULL,
      PRIMARY KEY (Seller_id)
   );

   CREATE TABLE Seller_Phone_num
   (
      Phone_num NUMBER(10) NOT NULL,
      Seller_id VARCHAR(6) NOT NULL,
      PRIMARY KEY (Phone_num, Seller_id),
      FOREIGN KEY (Seller_id) REFERENCES Seller(Seller_id)
      ON DELETE CASCADE
   );

   CREATE TABLE Payment
   (
      payment_id VARCHAR(7) NOT NULL,
      payment_date DATE NOT NULL,
      Payment_type VARCHAR(10) NOT NULL,
      Customer_id VARCHAR(6) NOT NULL,
      Cart_id VARCHAR(7) NOT NULL,
      PRIMARY KEY (payment_id),
      FOREIGN KEY (Customer_id) REFERENCES Customer(Customer_id),
      FOREIGN KEY (Cart_id) REFERENCES Cart(Cart_id),
      total_amount numeric(6)
   );

   CREATE TABLE Product
   (
      Product_id VARCHAR(7) NOT NULL,
      Type VARCHAR(7) NOT NULL,
      Color VARCHAR(15) NOT NULL,
      P_Size VARCHAR(2) NOT NULL,
      Gender CHAR(1) NOT NULL,
      Commission NUMBER(2) NOT NULL,
      Cost NUMBER(5) NOT NULL,
      Quantity NUMBER(2) NOT NULL,
      Seller_id VARCHAR(6),
      PRIMARY KEY (Product_id),
      FOREIGN KEY (Seller_id) REFERENCES Seller(Seller_id)
      ON DELETE SET NULL
   );

   CREATE TABLE Cart_item
   (
      Quantity_wished NUMBER(1) NOT NULL,
      Date_Added DATE NOT NULL,
      Cart_id VARCHAR(7) NOT NULL,
      Product_id VARCHAR(7) NOT NULL,
      FOREIGN KEY (Cart_id) REFERENCES Cart(Cart_id),
      FOREIGN KEY (Product_id) REFERENCES Product(Product_id),
      Primary key(Cart_id,Product_id)
   );

   alter table Cart_item add purchased varchar(3) default 'NO';
```

Output

```
Table created.

Table created.

Table created.

Table created.

Table created.

Table created.

Table created.

Table altered.
```

2. Inserting demo values

```
alter table Cart_item add purchased varchar(3) default 'NO';
insert into Cart values('crt1011');

insert into Customer values('cid100','ABCM1235','rajat','G-453','632014',9893135876, 'crt1011');

insert into Seller values('sid100','12345','aman','delhi cmc');

insert into Product values('pid1001','jeans','red','32','M',10,10005,20,'sid100');

insert into Seller_Phone_num values('9943336206','sid100');

insert into Cart_item values(3,to_date('10-OCT-1999','dd-mon-yyyy'),'crt1011','pid1001','Y');

insert into Payment values('pmt1001',to_date('10-OCT-1999','dd-mon-yyyy'),'online','cid100','crt1011',NULL);
```

Output

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

3. Basic Queries

```
133
134       select * from product where product_id in(
135         select product_id from Cart_item where (Cart_id in (
136             select Cart_id from Customer where Customer_id='cid100'
137         ))
138     and purchased='NO');
139
140 select product_id,Quantity_wished from Cart_item where (purchased='Y' and Cart_id in (select Cart_id from customer where Customer_id='cid101'));
141
142
143 select product_id, color, cost, seller_id from product where (type='jeans' and p_size='32' and gender='F' and quantity>0)
144
145  delete from cart_item where (product_id='pid1001' and Cart_id in (select cart_id from Customer where Customer_id='cid100'));
146
147  delete  from seller where seller_id = 'sid100';
148 update product set quantity = 00 where seller_id is NULL;
149
150 select product_id from cart_item where (purchased='Y' and date_added='12-dec-2018');
151 select count(product_id) count_pid,date_added from Cart_item where purchased='Y'  group by(date_added);
152
```

Output

PRODUCT_ID	TYPE	COLOR	P_SIZE	GENDER	COMMISSION	COST	QUANTITY	SELLER_ID
pid1001	jeans	red	32	M	10	10005	20	sid100

Download CSV

PRODUCT_ID	QUANTITY_WISHED
pid1001	3

Download CSV

PRODUCT_ID	COLOR	COST	SELLER_ID
pid1001	red	10005	sid100

Download CSV

```
1 row(s) deleted.

1 row(s) deleted.

1 row(s) updated.

no data found
```

4. PL/SQL function

```
create or replace procedure cost_filter(c in number,t in varchar)
is
cs product.cost%type;
ty product.type%type;
id product.product_id%type;
cursor cf is
select product_id,cost,type from product where cost<c and type=t;
begin
open cf;
loop
fetch cf into id,cs,ty;
exit when cf%notfound;
dbms_output.put_line('Product' || id || 'has cost ' || cs || ' and the type is' || ty);
end loop;
close cf;
exception
when no_data_found then
dbms_output.put_line('Sorry no such products exist');

create or replace function totalProducts(sId in varchar)
return number
is
total number(2):=0;
begin
select count(*) into total
from product
where seller_id=sId;
return total;
end;
```

Output

```
Function created.
```

Function execution

```
declare
    c number(2);
    begin
    c:=totalProducts('sid102');
    dbms_output.put_line('Total products is : '|| c);
    end;
```

Output

```
Statement processed.
Total products is : 0
```

Procedure which returns the total quantity of product with the given ID

```
    Procedure with exception handling
     create or replace procedure prod_details(p_id in varchar)
    is
    quan number(2);
    begin
    select quantity into quan from product where product_id=p_id;
    exception
    when no_data_found then
    dbms_output.put_line('Sorry no such product exist !!');
    end;
```

Output

```
Statement processed.
Total products is : 0
```

5. Triggers

```
226
227     create or replace function numCartId(cd in varchar)
228     return number
229     is
230     total number(2):=0;
231     begin
232     select count(*) into total
233     from cart_item
234     where cart_id=cd;
235     return total;
236     end;
237     Trigger
238     Create or replace trigger before_customer
239     before insert
240     on
241     customer
242     for each row
243     declare
244     c varchar(10);
245     n number(2);
246     begin
247     c:= :new.cart_id;
248     n:=numCartId(c);
249     if n>0 then
250     dbms_output.put_line('Sorry');
251     end if;
252     insert into cart values(c);
253     end;
254     end;
255  create or replace function total_cost(cId in varchar)
256     return number
```

```
255  create or replace function total_cost(cId in varchar)
256     return number
257     is
258     total number(2) :=0;
259     begin
260     select sum(cost) into total from product,cart_item where product.product_id=cart_item.product_id and cart_id=cId;
261     return total;
262     end;
263
264     create or replace trigger before_pay_up
265     before insert
266     on
267     payment
268     for each row
269     declare
270     total number(3);
271     begin
272     total :=total_cost(:new.cart_id);
273     insert into payment values(:new.payment_id,:new.payment_date,:new.payment_type,:new.customer_id,:new.cart_id,total);
274     end;
275
276
277
```

```
    create or replace function total_cost(cId in varchar)
return number
is
total number(2) :=0;
begin
select sum(cost) into total from product,cart_item where product.product_id=cart_item.product_id and cart_id=cId;
return total;
end;

create or replace trigger before_pay_up
before insert
on
payment
for each row
declare
total number(3);
begin
total :=total_cost(:new.cart_id);
insert into payment values(:new.payment_id,:new.payment_date,:new.payment_type,:new.customer_id,:new.cart_id,total);
end;
```

Conclusion : In this project, we gained knowledge with respect to database management and how the different features of SQl/PLSQL can be implemented in real life.