Database Management Systems

UCS310

Shopping Cart

Project Report

Submitted by:

Tanish Modi	102203956
Shivanshu Garg	102253004
Navoneel Nath	102383075
Shresth Raj	102383077

Submitted to:

Ms. Damini Arora

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

DEPARTMENT OF ENGINEERING
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
PATIALA

Jan-May 2024

INDEX

Problem Statement

Every e-commerce platform employs a shopping cart feature to facilitate customers in assembling their desired items and completing their purchases collectively. We have endeavored to grasp this system's mechanics and recreate it utilizing our expertise in normalization and PL/SQL.

Description

The data schema consists of 6 tables: Customer, Product, Cart, Orders, order_items, payments.

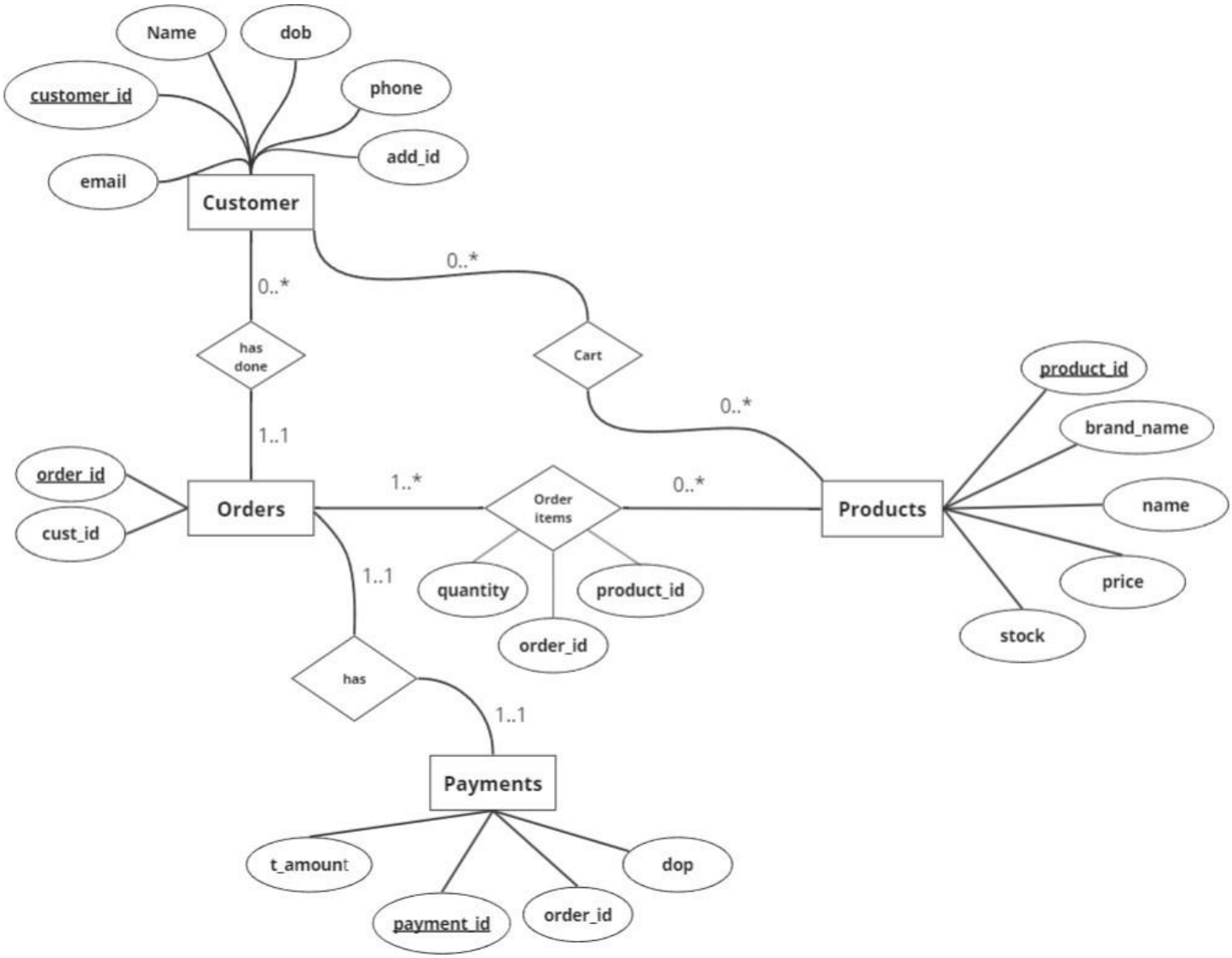
We have also included 2 triggers, 3 procedures, and 2 functions.

The triggers are set on table cart before insert/update and after insertion on payments table.

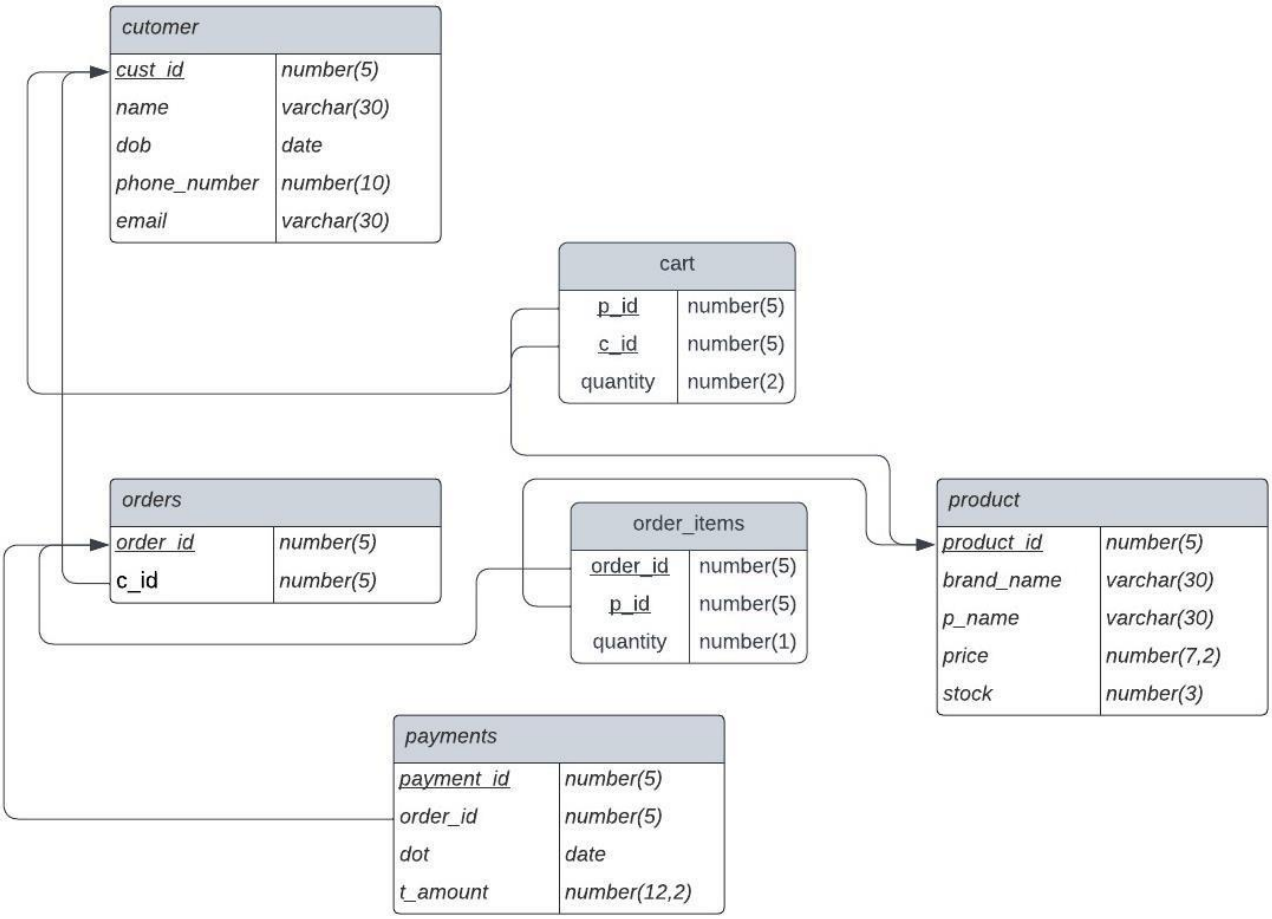
Procedures are used to update cart, remove items from cart and make payments.

Two functions are used to calculate total cart and other to calculate total sales.

ER Diagram


Name
dob
customer_id
phone
add_id
email
Customer
0..*
0..*
has done
Cart
1..1
0..*
product_id
brand_name
order_id
cust_id
Orders
1..*
Order items
0..*
Products
name
price
stock
quantity
product_id
order_id
1..1
has
1..1
Payments
t_amount
dop
payment_id
order_id

ER to Table


cutomer
cust_id number(5)
name varchar(30)
dob date
phone_number number(10)
email varchar(30)
cart
p_id number(5)
c_id number(5)
quantity number(2)
orders
order_id number(5)
c_id number(5)
order_items
order_id number(5)
p_id number(5)
quantity number(1)
product
product_id number(5)
brand_name varchar(30)
p_name varchar(30)
price number(7,2)
stock number(3)
payments
payment_id number(5)
order_id number(5)
dot date
t_amount number(12,2)

Normalization

Normalisation is the process to eliminate data redundancy and enhance data integrity in the table. Normalisation also helps to organize the data in the database. It is a multi-step process that sets the data into tabular form and removes the duplicated data from the relational tables.

Normalisation organizes the columns and tables of a database to ensure that database integrity constraints properly execute their dependencies. It is a systematic technique of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update, and Deletion anomalies.

1NF: None of the tables created have any multi-valued attributes and hence, the tables are in First Normal Form.

2NF: All the partial dependencies have been resolved and any partial dependency does not exist. The tables are now in Second Normal Form.

3NF: All the attributes having transitive dependencies have been shifted to different tables and now no transitive dependencies exist. The tables are now in Third Normal Form.

SQL and PL/SQL CODES

- ## Creating Tables

--customer table

```
create table customer(
    cust_id number(5) primary key,
    name varchar(30) not null,
    dob date not null,
    phone_number number(10) not null,
    email varchar(30) not null
);
```

--products table

```
create table product(
    product_id number(5) primary key,
    brand_name varchar(30) not null,
    p_name varchar(30) not null,
    price number(7,2) not null,
    stock number(3) not null
);
```

--orders table

```
create table orders(
    order_id number(5) primary key,
    c_id number(5) not null constraint cust_check4 references customer(cust_id)
);
```

--payments table

```
create table payments(
    payment_id number(5) primary key,
    order_id number(5) not null constraint cust_check REFERENCES orders(order_id),
    dot date not null,
    t_amount number(12,2) not null
);
```

--cart table

```
create table cart(
    p_id number(5) not null constraint product_check REFERENCES product(product_id),
    c_id number(5) not null constraint cust_check3 REFERENCES customer(cust_id),
    quantity number(2) not null
);
```

--order items table

```
create table order_items(
    order_id number(5) not null constraint order_check references orders(order_id),
     p_id number(5) not null constraint product_check2 REFERENCES product(product_id),
    quantity number(1) not null
);
```

- ## Inserting Values

--product table

```
insert into product values(1, 'iPhone 13', 'Apple', 799.00, 60);
insert into product values(2, 'Galaxy S21', 'Samsung', 699.99, 50);
insert into product  values(3, 'Pixel 6', 'Google', 599.00, 80);
insert into product values(4, 'OnePlus 9 Pro', 'OnePlus', 969.00, 70);
insert into product values(5, 'Xperia 1 III', 'Sony', 1299.99, 20);
insert into product  values(6, 'Mi Mix Fold', 'Xiaomi', 1499.00, 50);
insert into product values(7, 'ROG Phone 5s Pro', 'Asus', 1199.99, 40);
insert into product values(8, 'Zenfone 8 Flip', 'Asus', 999.99, 30);
insert into product  values(9, 'Find X3 Pro', 'Oppo', 1199.00, 20);
insert into product values(10,'Mate X2 Pro', 'Huawei', 2599.00,10);
insert into product  values(11,'T-Shirt','Nike',20.00,200);
insert into product  values(12,'Jeans','Levis',50.00,150);
insert into product  values(13,'Sneakers','Adidas',100.00,100);
insert into product values(14,'Boots','Timberland',150.00,75);
insert into product values(15,'Dress Shoes','Cole Haan',200.00,50);
```

```
insert into product values(16,'Socks','Hanes',10.00,50);

insert into product values(17,'Underwear','Calvin Klein',25.00,30);

insert into product values(18,'Sweatshirt','Champion',40.00,250);

insert into product values(19,'Jacket','The North Face',150.00,200);

insert into product  values(20,'Backpack','JanSport',50.00,150);
```

--customer table

```
insert into customer values(1,'Anni',to_date('1990-07-07', 'yyyy-mm-dd'),9753124680,'anni@gmail.com');

insert into customer values(2,'Derek',to_date('1991-05-12', 'yyyy-mm-dd'),9864213570,'derek@gmail.com');

insert into customer values(3,'Acid', to_date('1990-02-18', 'yyyy-mm-dd'),9765432180,'acid@gmail.com');

insert into customer values(4,'Raggie',to_date('1991-05-24', 'yyyy-mm-dd'),9966785430,'raggie@gmail.com');
```

Triggers, Procedures and Functions

- **Triggers**

--**quantity check**

```
create or replace trigger quantity_check
before
insert or update
on cart
for each row
declare
stocks number;
begin
   select stock into stocks from product where product_id=:new.p_id;
if stocks<:new.quantity-:old.quantity or :new.quantity>9 or :new.quantity<0 then
   raise_application_error(-20005,'Wrong Input');
end if;
end;
```

--**clear cart**

```
create or replace trigger clear_cart
after
insert
on payments
for each row
declare
cid orders.c_id%type;
begin
   select c_id into cid from orders where order_id=:new.order_id;
   delete from cart where c_id= cid;
if sql%found then
   null;
else
   raise_application_error(-20003,'No Item found in the cart');
end if;
end;
```

- **Procedures**

--**add product to cart**

```
create or replace procedure add_to_cart(customer_id in number,prod_id in number,quant in
number) as
prd_id cart.p_id%type;
begin
   select p_id into prd_id from cart where p_id=prod_id and c_id=customer_id;
   if sql%found then
      --before updation the quantity_check trigger will be triggered to check that the quantity
      --is correct or not.
      update cart set quantity=quantity+quant where p_id=prod_id and c_id=customer_id;
      update product set stock=stock-quant where product_id=prod_id;
      dbms_output.put_line('Product Added Successfully in Cart');
   end if;
exception
   when no_data_found then
   insert into cart values(prod_id,customer_id,quant);
   update product set stock=stock-quant where product_id=prod_id;
   dbms_output.put_line('Product Added Successfully in Cart');
end;
```

--**remove from cart**

```
create or replace procedure remove_from_cart(customer_id in number,prod_id in number,quant in
number) as
prd_id cart.p_id%type;
temp number;
begin
   select p_id into prd_id from cart where p_id=prod_id and c_id=customer_id;
   if sql%found then
      update cart set quantity=quantity-quant where p_id=prod_id and c_id=customer_id;
      update product set stock=stock+quant where product_id=prod_id;
   end if;
   select quantity into temp from cart where p_id=prod_id and c_id=customer_id;
   if sql%found and temp=0 then
      delete from cart where p_id=prod_id and c_id=customer_id;
   end if;
   dbms_output.put_line('Product Removed Successfully from Cart');
exception
   when no_data_found then
   raise_application_error(-20007,'No Item Found with the current product id');

end;
```

--do payments

```
create or replace procedure dopayments(customer_id in number) as
CURSOR items(cust_id number) IS SELECT * FROM cart WHERE cart.c_id=cust_id;
oid number;
temp number;
total number;
begin
total:=0;
for dataa in items(customer_id) loop
select price*dataa.quantity into temp from product where product.product_id=dataa.p_id;
if sql%found then
   total:=total+temp;
end if;
end loop;
select count(order_id) into oid from orders;
select count(payment_id) into temp from payments;
insert into orders values(oid+1,customer_id);
for dataa in items(customer_id) loop
   insert into order_items values(oid+1,dataa.p_id,dataa.quantity);
end loop;
insert into payments values(temp+1,oid+1,to_date (to_char (sysdate, 'dd/mon/yyyy'),
'dd/mon/yyyy'),total);
dbms_output.put_line('Payment Done Successfully with order id '|| oid+1);
end;
```

- # Functions

--function to print total cart amount of the customer

```
create or replace function total_cart_amount(total_amount out number,cust_id in number) return
number as
    begin
    select sum((select price from product where product.product_id= cart.p_id)* cart.quantity) into
    total_amount from cart where c_id=cust_id;
    if total_amount is null then
        raise_application_error(-20034,'No Product Found in the Cart');
   end if;
    return (total_amount);

end;
```

--function to print total sales done till date

```
create or replace function total_sales(total_amount out number) return number as
    begin
    select sum(t_amount) into total_amount from payments;
     if total_amount is null then
        raise_application_error(-20034,'No Product Found in the Cart');
    end if;
    return (total_amount);
end;
```

Queries

```
--show all products
  select *  from product;

--show all product added in the cart of the customer
  select * from cart where c_id=1;

--find total amount of products in the cart
declare
   amount number;
begin
   amount:=total_cart_amount(amount,2);
   dbms_output.put_line('Total Cart Amount is '||amount);
end;

--add product to cart
begin
    --add_to_cart(customer_id,product_id,quantity)
      add_to_cart(1,2,5);
end;

--remove product from cart
begin
    --remove_from_cart(customer_id,product_id,quantity)
      Remove_from_cart(1,2,5);
end;

--do payments

begin
    --dopayments (customer_id)
      dopayments (1);
   --after payment clear cart trigger will be triggered and the products will be removed from
   --the cart and an order_id and payment_id will be generated.
   --All the products whose amount is payment will be added to order_items as a record with
   --order_id linked.
end;
--find total amount of products in the cart
declare
   amount number;
begin
   amount:=total_sales(amount);
   dbms_output.put_line('Total Sales is '||amount);
end;
```

OUTPUT SCREENSHOTS

- **Show all Tables Data**

--query

```
select * from customer;
select * from product;
select * from cart;
select * from orders;
select * from order_items;
select * from payments;
```

--output

--product table

PRODUCT_ID	BRAND_NAME	P_NAME	PRICE	STOCK
6	Mi Mix Fold	Xiaomi	1499	41
1	iPhone 13	Apple	799	60
4	OnePlus 9 Pro	OnePlus	969	65
3	Pixel 6	Google	599	80
5	Xperia 1 III	Sony	1299.99	20
7	ROG Phone 5s Pro	Asus	1199.99	40
8	Zenfone 8 Flip	Asus	999.99	30
9	Find X3 Pro	Oppo	1199	20
10	Mate X2 Pro	Huawei	2599	5
11	T-Shirt	Nike	20	200
12	Jeans	Levis	50	150
13	Sneakers	Adidas	100	100
14	Boots	Timberland	150	75
15	Dress Shoes	Cole Haan	200	50
16	Socks	Hanes	10	50
17	Underwear	Calvin Klein	25	30
18	Sweatshirt	Champion	40	250
19	Jacket	The North Face	150	200
20	Backpack	JanSport	50	150
2	Galaxy S21	Samsung	699.99	45

Download CSV

--customer table

CUST_ID	NAME	DOB	PHONE_NUMBER	EMAIL
1	Anni	07-JUL-90	9753124680	anni@gmail.com
2	Derek	12-MAY-91	9864213570	derek@gmail.com
3	Acid	18-FEB-90	9765432180	acid@gmail.com
4	Raggie	24-MAY-91	9966785430	raggie@gmail.com

--cart table

P_ID	C_ID	QUANTITY
6	2	9
10	2	5
4	1	5
2	1	5

--payments table

PAYMENT_ID	ORDER_ID	DOT	T_AMOUNT
1	1	03-MAY-23	8344.95

--orders table

ORDER_ID	C_ID
1	1

--order item tables

ORDER_ID	P_ID	QUANTITY
1	4	5
1	2	5

Download CSV

2 rows selected.

Queries Output

- --show all products

```
167     select * from product;
```

PRODUCT_ID	BRAND_NAME	P_NAME	PRICE	STOCK
6	Mi Mix Fold	Xiaomi	1499	41
1	iPhone 13	Apple	799	60
4	OnePlus 9 Pro	OnePlus	969	65
3	Pixel 6	Google	599	80
5	Xperia 1 III	Sony	1299.99	20
7	ROG Phone 5s Pro	Asus	1199.99	40
8	Zenfone 8 Flip	Asus	999.99	30
9	Find X3 Pro	Oppo	1199	20
10	Mate X2 Pro	Huawei	2599	5
11	T-Shirt	Nike	20	200
12	Jeans	Levis	50	150
13	Sneakers	Adidas	100	100
14	Boots	Timberland	150	75
15	Dress Shoes	Cole Haan	200	50
16	Socks	Hanes	10	50
17	Underwear	Calvin Klein	25	30
18	Sweatshirt	Champion	40	250
19	Jacket	The North Face	150	200
20	Backpack	JanSport	50	150
2	Galaxy S21	Samsung	699.99	45

- --show customer products present in the cart

```
218   select * from cart where c_id=1;
```

P_ID	C_ID	QUANTITY
4	1	5
2	1	5

Download CSV

2 rows selected.

- --find total amount of products in the cart

```
declare
    amount number;
begin
    amount:=total_cart_amount(amount,2);
    dbms_output.put_line('Total Cart Amount is '||amount);
end;
```

```
Statement processed.
Total Cart Amount is 9721
```

--products present in the cart

P_ID	C_ID	QUANTITY
4	2	9
15	2	5
15	1	5

Download CSV

3 rows selected.

- --add product to cart

```
begin
    add_to_cart(1,12,5);
end;
```

```
Statement processed.
Product Added Successfully in Cart
```

--product added to the cart

P_ID	C_ID	QUANTITY
4	2	9
15	2	5
12	1	5
15	1	5

Download CSV

4 rows selected.

- --remove product from cart

```
186  begin
187      remove_from_cart(1,15,3);
188  end;
```

```
Statement processed.
Product Removed Successfully from Cart
```

--item removed with certain quantity from the cart

P_ID	C_ID	QUANTITY
4	2	9
15	2	5
12	1	5
15	1	2

Download CSV

4 rows selected.

- --do payment

```
188  begin
189      dopayments(1);
190  end;
```

```
Statement processed.
Payment Done Successfully with order id 1and payment id 1
```

--products removed from cart

P_ID	C_ID	QUANTITY
4	2	9
15	2	5

Download CSV

2 rows selected.

--order items added in order items table

ORDER_ID	P_ID	QUANTITY
1	12	5
1	15	2

Download CSV

--orders added in order table

ORDER_ID	C_ID
1	1

Download CSV

--payment details added in payment table

PAYMENT_ID	ORDER_ID	DOT	T_AMOUNT
1	1	03-MAY-23	650

Download CSV

- --total sales till date

```
declare
    amount number;
begin
    amount:=total_sales(amount);
    dbms_output.put_line('Total Sales is '||amount);
end;
```

```
Statement processed.
Total Sales is 6196
```

--payments present in paymnets table

PAYMENT_ID	ORDER_ID	DOT	T_AMOUNT
1	1	03-MAY-23	200
2	2	03-MAY-23	2998
3	3	03-MAY-23	2998

Download CSV

3 rows selected.

Conclusion

Our project journey has been immensely enriching, particularly in understanding the intricacies of shopping cart schema, harnessing PL/SQL functionalities, and delving into the realm of comprehensive database design.

Throughout this endeavor, we've come to appreciate the depth of possibilities beyond the project's initial scope. The opportunity to explore these functionalities has been invaluable, fostering a deeper understanding of database systems and their potential applications.

We're grateful for the chance to engage with this project, as it has not only honed our technical skills but also broadened our perspective on the possibilities within the realm of database management.