**Project Name:**

Online Supershop Management System

**Description:**

A supershop management basically provides opportuinities to buy product online. It maintains a list of products that it offers. When a user places an order, the shop keeps records of the user like user’s name, address, contact information etc. and payment information. As the name indicates, a supershop management also provides online order facility. Information of online orders are saved to the system database.

**Schema Diagram:**

|  |
| --- |
| User\_info  user\_id(pk)  user\_name  email  password  phone\_no address  occupation |

|  |
| --- |
| Products\_info  product\_id(pk)  product\_name  price  stock  details  product\_status |

|  |
| --- |
| Order\_request |

order\_id(pk)

user\_id(fk)

product\_id(fk)

order\_date

quantity

req\_product\_name

req\_amount

pk=primary key; fk = foreign key

Figure 1: Schema diagram for Online Shopping Database

**Implementation:**

3 tables.

* user\_info
* product\_info
* order\_request\_list

Create table,trigger and insert data on table

drop table order\_request\_list;

drop table user\_info;

drop table product\_info;

create table user\_info(

user\_id number,

user\_namevarchar(20),

emailvarchar(20) not null,

passwordvarchar(15) not null,

phone\_no number(15) not null,

addressvarchar(30) not null,

primary key(user\_id)

);

create table product\_info(

product\_id number,

product\_namevarchar(20) not null,

price number not null,

stock number,

details varchar(100),

primary key(product\_id)

);

create table order\_request\_list(

order\_id number,

user\_id number,

product\_id number,

order\_date date,

quantity number not null,

req\_productvarchar(20),

req\_amount number check(req\_amount>0),

primary key(order\_id),

foreign key(user\_id) references user\_info(user\_id),

foreign key(product\_id) references product\_info(product\_id)

);

--lab2 alter command

alter table user\_info add occupation VARCHAR(10);

alter table product\_info add product\_status varchar(20) default null;

alter table order\_request\_list

modify order\_date date default sysdate;

alter table order\_request\_list rename column req\_product to req\_product\_name;

describe user\_info;

describe product\_info;

describe order\_request\_list;

--lab9 trigger

--check the product to buy is available in stock

create or replace trigger product\_update before insert or update on order\_request\_list

for each row

declare

product\_id number;

P1 number :=0;

begin

select stock into P1 from product\_info where product\_id = :new.product\_id;

P1:=P1-:new.quantity;

updateproduct\_info set stock = P1 where product\_id =:new.product\_id;

if(P1<0) then

raise\_application\_error(-20000,'Product are not available');

end if;

end;

/

create or replace trigger check\_price before insert or update on product\_info

for each row

declare

c\_min constant number(8,2):= 20000.0;

c\_max constant number(8,2):= 200000.0;

begin

if:new.price>c\_max or :new.price<c\_min then

raise\_application\_error(-20000,'New product is too expensive or cheap');

end if;

end;

/

CREATE TRIGGER TR\_PRODUCT\_STATUS

BEFORE UPDATE OR INSERT ON product\_info

FOR EACH ROW

BEGIN

IF :new.stock=0 THEN

:NEW.product\_status:='Unavailable';

ELSIF :new.stock>0 THEN

:NEW.product\_status:='Avaiable';

END IF;

END TR\_PRODUCT\_STATUS;

/

insert into user\_info (user\_id,user\_name,email,password,phone\_no,address,occupation) values(1207006,'karima','Kareema@yahoo.com','12345',0183958844,'khulna','student');

insert into user\_info (user\_id,user\_name,email,password,phone\_no,address,occupation)

values(1207007,'sharmi','sharmiabc@yahoo.com','23456',01688358851,'khulna','student');

insert into user\_info

(user\_id,user\_name,email,password,phone\_no,address,occupation)

values(1207008,'muna','tanjim@yahoo.com','34567',01670313775,'kuet','student');

insert into user\_info

(user\_id,user\_name,email,password,phone\_no,address,occupation)

values(1207009,'soma','Shahnaz@yahoo.com','45678',01689060063,'kuet','student');

insert into user\_info

(user\_id,user\_name,email,password,phone\_no,address,occupation) values(1207010,'mousumi','mousumi@yahoo.com','56789',01521456320,'kuet','student');

insert into product\_info (product\_id,product\_name,price,stock,details,product\_status) values(001,'samsung tab',32000,10,'it has fm radio and video player','null');

insert into product\_info (product\_id,product\_name,price,stock,details,product\_status) values(002,'sony ericson',30000,20,'it has a voice recorder','null');

insert into product\_info (product\_id,product\_name,price,stock,details,product\_status) values(003,'nokia lumia',21000,25,'it has internet facilities','null');

insert into product\_info (product\_id,product\_name,price,stock,details,product\_status) values(004,'hp laptop',60000,15,'it has a 5 gb ram and core i5','null');

insert into product\_info (product\_id,product\_name,price,stock,details,product\_status) values(005,'dslr camera',60000,0,'it has high resolution camera','null');

insert into order\_request\_list (order\_id,user\_id,product\_id,quantity,req\_product\_name, req\_amount) values (11,1207006,004,5,'abc',1);

insert into order\_request\_list (order\_id,user\_id,product\_id,quantity,req\_product\_name, req\_amount) values(12,1207006,002,2,'def',2);

insert into order\_request\_list (order\_id,user\_id,product\_id,quantity,req\_product\_name, req\_amount) values(13,1207008,003,3,'efghjds',3);

insert into order\_request\_list (order\_id,user\_id,product\_id,quantity,req\_product\_name, req\_amount) values(14,1207008,002,3,'dfvsn',4);

insert into order\_request\_list (order\_id,user\_id,product\_id,quantity,req\_product\_name, req\_amount) values(15,1207009,001,3,'hij',5);

select \* from user\_info;

select \* from product\_info;

select \* from order\_request\_list;

commit;

Use queries,pl/sql,procedure,function on table

--lab3

--distinct data find

select distinct(product\_id) from order\_request\_list;

--using order by

selectproduct\_name,price from product\_info order by price;

--calculating field

select (price-price\*.05) as price\_with\_discount from product\_info;

--applying condition

selectproduct\_name from product\_info where price<50000;

--find product using string or pattern matching

selectproduct\_name,price from product\_info where product\_name like '%samsung tab';

--and or etccomparision

selectproduct\_name from product\_info where price>20000 AND price<40000;

--between

selectproduct\_name from product\_info where price BETWEEN 20000 AND 40000;

selectproduct\_name,product\_id from product\_info where price NOT BETWEEN 20000 AND 40000;

--in operation or set membership

selectproduct\_name,price from product\_info where product\_name in ('samsung tab','sonyericson');

--lab4 aggregate function

--max

select price from product\_info;

select max(price) from product\_info;

--count(\*),sum,avg

select count(\*),sum(price),avg(price) from product\_info;

--group by

selectprice,count(price) from product\_info group by price;

--having

selectprice,count(price) from product\_info group by price having price>30000;

--lab-5 subquery and set operations starts

selectuser\_id,user\_name from user\_info

whereuser\_name IN (select user\_name from user\_info where address='khulna');

selects.user\_name,s.email,s.address from user\_info s where s.user\_id in(select o.user\_id from order\_request\_list o where o.product\_id=002);

selects.user\_name from user\_info s where address='kuet'

union

selects.user\_name from user\_info s where s.user\_id in(select o.user\_id from order\_request\_list o where o.product\_id=004);

selects.user\_name from user\_info s where address='kuet'

intersect

selects.user\_name from user\_info s where s.user\_id in(select o.user\_id from order\_request\_list o where o.product\_id=001);

selects.user\_name from user\_info s where s.user\_id in(select o.user\_id from order\_request\_list o where o.product\_id=004)

minus

selects.user\_name from user\_info s where address='kuet';

--lab6 join operation

-- using clause

selectp.product\_name,p.price,o.order\_id from product\_info p join order\_request\_list o using (product\_id);

--normal join operation

selectc.user\_name , d.product\_id from order\_request\_list d join user\_info c on d.user\_id=c.user\_id;

--natural join

selectc.user\_name,c.email,d.product\_id,d.order\_date from order\_request\_list d natural join user\_info c;

--outer joins

selectc.user\_name , d.product\_id from order\_request\_list d left outer join user\_info c on d.user\_id=c.user\_id;

selectc.user\_name , d.product\_id from order\_request\_list d right outer join user\_info c on d.user\_id=c.user\_id;

selectc.user\_name , d.product\_id from order\_request\_list d full outer join user\_info c on d.user\_id=c.user\_id;

--lab7 pl/sql

--no of user

SET SERVEROUTPUT ON

DECLARE

count\_user\_iduser\_info.user\_id%type;

BEGIN

SELECT count(user\_id) INTO count\_user\_id

FROM user\_info;

DBMS\_OUTPUT.PUT\_LINE('The number of user id is : ' || count\_user\_id);

END;

/

--no of product in stock

SET SERVEROUTPUT ON

DECLARE

count\_productproduct\_info.stock%type;

BEGIN

SELECT sum(stock) INTO count\_product

FROM product\_info;

DBMS\_OUTPUT.PUT\_LINE('The number of product in stock is : ' || count\_product);

END;

/

--maximum quantity sold at a time

SET SERVEROUTPUT ON

DECLARE

max\_quantityorder\_request\_list.quantity%type;

BEGIN

SELECT MAX(quantity) INTO max\_quantity

FROM order\_request\_list;

DBMS\_OUTPUT.PUT\_LINE('The maximum number of products sold at a time is : ' || max\_quantity);

END;

/

--maximum quantity is requested not available in stock

SET SERVEROUTPUT ON

DECLARE

max\_quantityorder\_request\_list.req\_amount%type;

BEGIN

SELECT MAX(req\_amount) INTO max\_quantity

FROM order\_request\_list;

DBMS\_OUTPUT.PUT\_LINE('The maximum number of products requested is : ' || max\_quantity);

END;

/

--product order date

SET SERVEROUTPUT ON

DECLARE

Date\_orderedorder\_request\_list.order\_date%type;

ProductNameproduct\_info.product\_name%type := 'samsung tab';

BEGIN

SELECT order\_date INTO Date\_ordered

FROM order\_request\_list, product\_info

WHERE product\_info.product\_id = order\_request\_list.product\_id AND

product\_name = ProductName;

DBMS\_OUTPUT.PUT\_LINE('Product Name : ' || ProductName || ' ordered on '|| Date\_ordered);

END;

/

--conditional discount

SET SERVEROUTPUT ON

DECLARE

full\_priceproduct\_info.price%type;

ProductName VARCHAR2(100);

discount\_priceproduct\_info.price%type;

BEGIN

ProductName := 'hp laptop';

SELECT price INTOfull\_price

FROM product\_info

WHERE product\_name like ProductName ;

IF full\_price<25000 THEN

discount\_price := full\_price;

ELSIF full\_price>= 250000 and full\_price<40000 THEN

discount\_price := full\_price - (full\_price\*0.25);

ELSIF full\_price>= 40000 and full\_price<=50000 THEN

discount\_price := full\_price - (full\_price\*0.4);

ELSE

discount\_price := full\_price - (full\_price\*0.5);

END IF;

DBMS\_OUTPUT.PUT\_LINE (ProductName || 'Full Price: '||full\_price||' DisountedPice: '|| ROUND(discount\_price,2));

EXCEPTION

WHEN others THEN

DBMS\_OUTPUT.PUT\_LINE (SQLERRM);

END;

/

--lab8

--while loop

setserveroutput on

declare

counter number(7):=1207006;

Name user\_info.user\_name%type;

Email user\_info.email%type;

begin

while counter<=1207010

loop

selectuser\_name , email into Name, Email from user\_info where user\_id=counter;

dbms\_output.put\_line('Record '||counter);

dbms\_output.put\_line(Name ||' ' ||Email);

dbms\_output.put\_line('-----------------------');

counter :=counter+1;

end loop;

exception

when others then dbms\_output.put\_line(sqlerrm);

end;

/

--for

setserveroutput on

declare

counter number(3);

ProductNameproduct\_info.product\_name%type;

Price product\_info.price%type;

begin

for counter in 001..005

loop

selectproduct\_name,price into ProductName,Price from product\_info where product\_id in (select product\_id from product\_info where product\_id=counter);

dbms\_output.put\_line('Record : '||counter);

dbms\_output.put\_line(ProductName||' '||Price);

dbms\_output.put\_line('----------------------');

end loop;

exception

when others then

dbms\_output.put\_line(sqlerrm);

end;

/

--loop

setserveroutput on

declare

counter number(7):=1207005;

Name user\_info.user\_name%type;

Address user\_info.address%type;

begin

loop

counter:=counter+1;

selectuser\_name,address into Name,Address from user\_info where user\_id=counter;

dbms\_output.put\_line(Name||' '||Address);

exit when counter>1207010;

end loop;

exception

when others then dbms\_output.put\_line(sqlerrm);

end;

/

--cursor

SET SERVEROUTPUT ON

DECLARE

CURSOR product\_cur IS

SELECT product\_name,price FROM product\_info;

product\_recordproduct\_cur%ROWTYPE;

BEGIN

OPEN product\_cur;

LOOP

FETCH product\_cur INTO product\_record;

EXIT WHEN product\_cur%ROWCOUNT> 4;

DBMS\_OUTPUT.PUT\_LINE ('Product Name : ' || product\_record.product\_name || ' Price : ' || product\_record.price);

END LOOP;

CLOSE product\_cur;

END;

/

--create procedure to update particular id

CREATE OR REPLACE PROCEDURE UPD\_PRODUCTS (

productidproduct\_info.product\_id%TYPE,

product\_priceproduct\_info.price%TYPE) IS

BEGIN

UPDATE product\_info SET price = product\_price WHERE product\_id = productid;

IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR(-20202, 'No product updated.');

END IF;

COMMIT;

END UPD\_PRODUCTS;

/

SHOW ERRORS

SET SERVEROUTPUT ON

BEGIN

UPD\_PRODUCTS (004,50000);

END;

/

select price from product\_info where product\_id=004;

--create function and find amount user order to buy

CREATE OR REPLACE FUNCTION GET\_ID (useridorder\_request\_list.user\_id%TYPE) RETURN number IS

amountorder\_request\_list.quantity%TYPE;

BEGIN

SELECT quantity INTO amount FROM order\_request\_list WHERE user\_id = userid;

RETURN amount;

END;

/

SET SERVEROUTPUT ON

DECLARE

product\_quantity number;

BEGIN

product\_quantity := GET\_ID (1207009);

dbms\_output.put\_line('Users ordered quantity : ' || product\_quantity);

END;

/

create or replace function avg\_price return number is

avg\_priceproduct\_info.price%type;

begin

selectavg(price) into avg\_price from product\_info;

returnavg\_price;

end;

/

setserveroutput on

begin

dbms\_output.put\_line('average price: '||avg\_price);

end;

/

--lab9

--date

selectsysdate from dual;

selectcurrent\_date from dual;

selectsystimestamp from dual;

SELECT ADD\_MONTHS (order\_date,12) AS Warrenty

FROM order\_request\_list;

commit;

**Discussion:**

In the above project, I implemented almost all the basic features of Oracle database. I applied DDL (Database Definition Language) to create table, alter column name or column type. I also used DML (Database Modification Language) to insert, update and delete rows of the tables. For calculation purpose, I used PL/SQL, function and procedure. Trigger was used to validate data before inserting into or updating rows of a table. This mini project (Supershop Management System) helped me learn about the basics of database.