**DBMS - MINI PROJECT**

**INVENTORY MANAGEMENT SYSTEM**

Submitted By:

Name: Jayasimha S

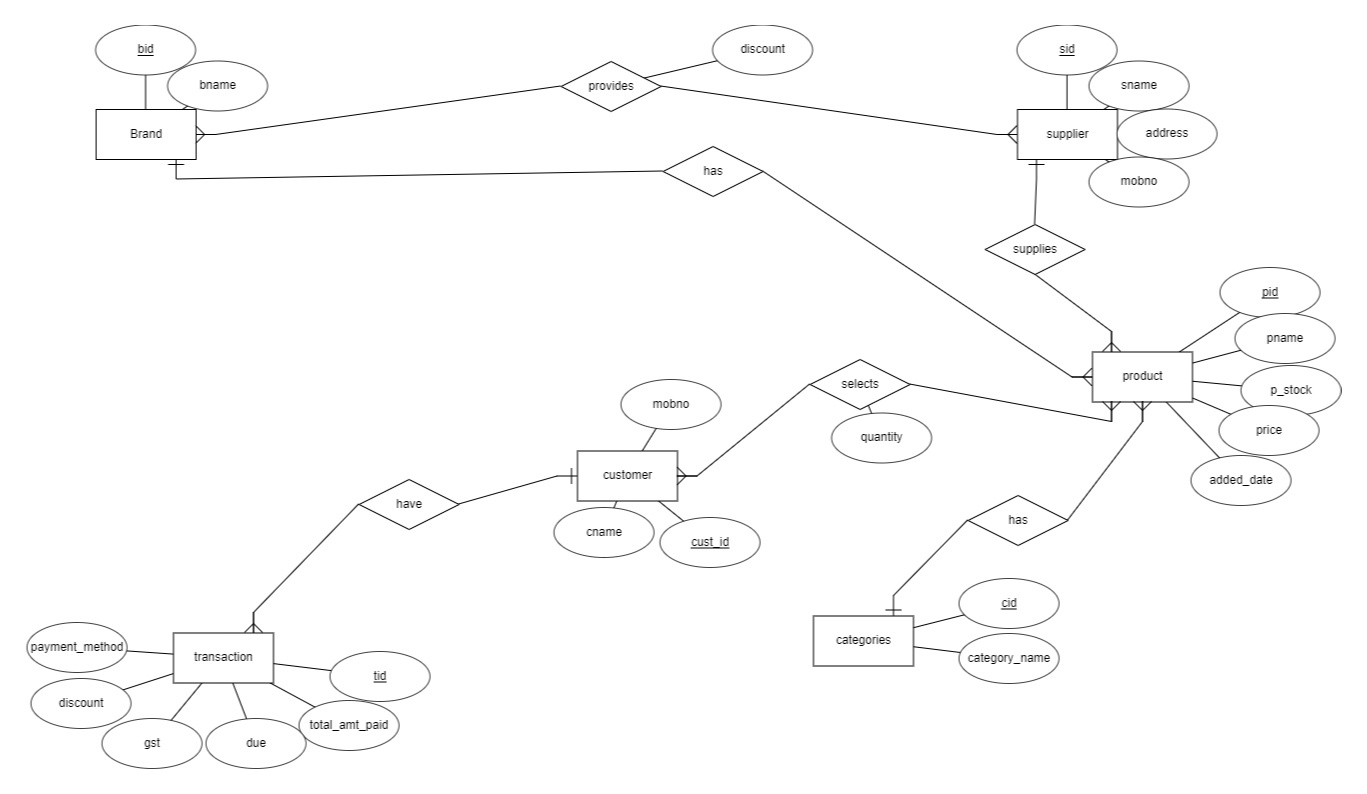
SRN: PES1UG20CS177

V Semester Section = C Section

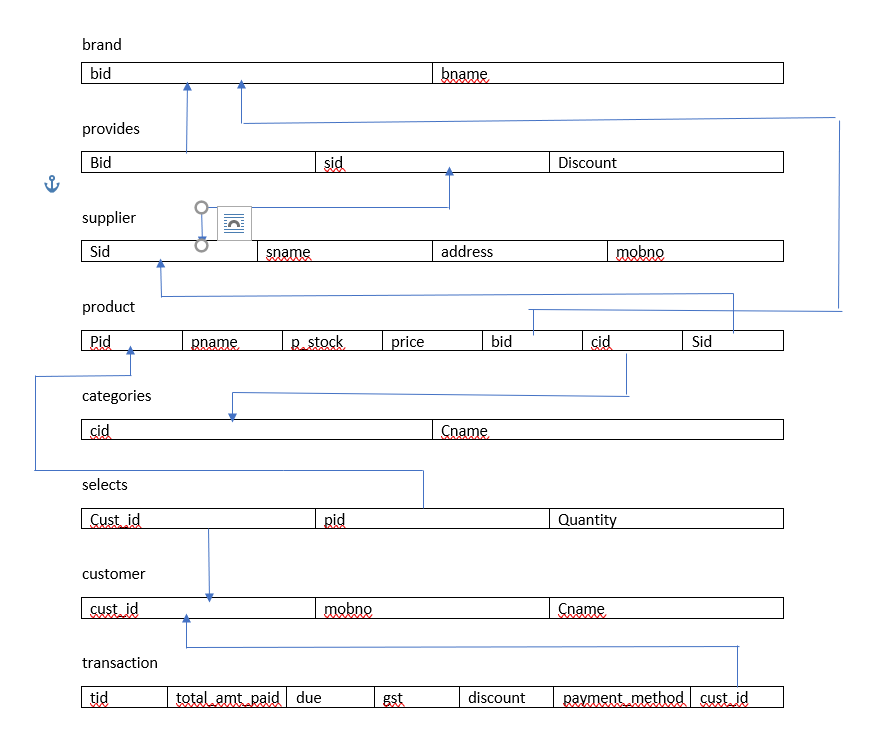
**ABSTRACT**

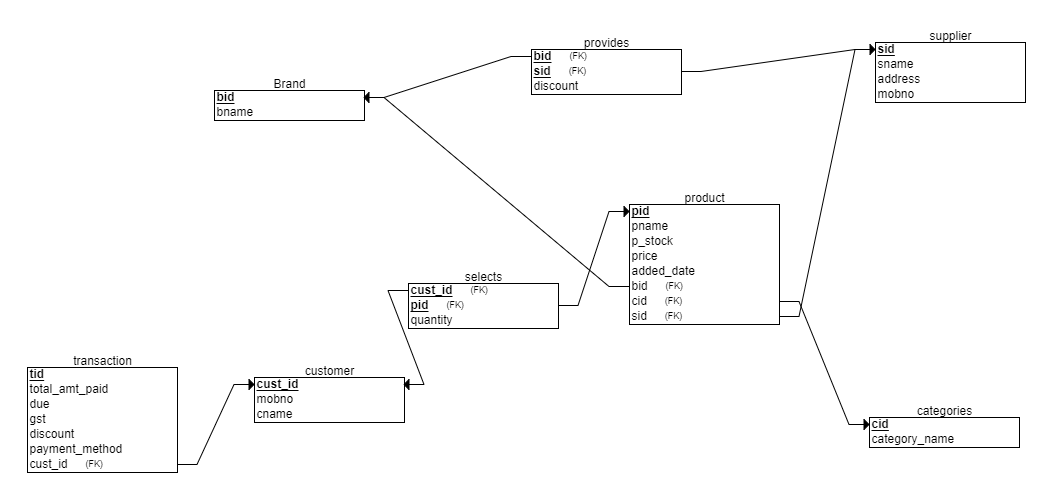
The inventory management system consists of 5 basic entities namely brand, categories, product, supplier, customer, transaction. The product table consists of the product details of all the products of different brands, different categories and from different suppliers whose details are present in the brand, categories and supplier table respectively. The customer table consists the details of all the customers and when the customer selects a product, his selection is recorded in the selects table and the transaction details of his selected is item are recorded by the inventory manager in the transaction table. The front end for this system is developed using python and backend using mysql database.

**ER Diagram**



**Relational Schema**





**DDL statements - Building the database**

CREATE TABLE brand

(

bid INT NOT NULL,

bname VARCHAR(20) NOT NULL,

PRIMARY KEY (bid)

);

CREATE TABLE categories

(

cid INT NOT NULL,

category\_name VARCHAR(20) NOT NULL,

PRIMARY KEY (cid)

);

CREATE TABLE supplier

(

sid INT NOT NULL,

sname VARCHAR(20) NOT NULL,

address VARCHAR(20) NOT NULL,

mobno INT NOT NULL,

PRIMARY KEY (sid)

);

CREATE TABLE customer

(

mobno INT NOT NULL,

cust\_id INT NOT NULL,

cname VARCHAR(20) NOT NULL,

PRIMARY KEY (cust\_id)

);

CREATE TABLE transaction

(

tid INT NOT NULL,

total\_amt\_paid INT NOT NULL,

due INT NOT NULL,

gst INT NOT NULL,

discount INT NOT NULL,

payment\_method VARCHAR(20) NOT NULL,

cust\_id INT NOT NULL,

PRIMARY KEY (tid),

FOREIGN KEY (cust\_id) REFERENCES customer(cust\_id)

);

CREATE TABLE provides

(

discount INT NOT NULL,

bid INT NOT NULL,

sid INT NOT NULL,

PRIMARY KEY (bid, sid),

FOREIGN KEY (bid) REFERENCES Brand(bid),

FOREIGN KEY (sid) REFERENCES supplier(sid)

);

CREATE TABLE product

(

pid INT NOT NULL,

pname VARCHAR(20) NOT NULL,

p\_stock INT NOT NULL,

price INT NOT NULL,

bid INT NOT NULL,

cid INT NOT NULL,

sid INT NOT NULL,

PRIMARY KEY (pid),

FOREIGN KEY (bid) REFERENCES brand(bid),

FOREIGN KEY (cid) REFERENCES categories(cid),

FOREIGN KEY (sid) REFERENCES supplier(sid)

);

CREATE TABLE selects

(

quantity INT NOT NULL,

cust\_id INT NOT NULL,

pid INT NOT NULL,

PRIMARY KEY (cust\_id, pid),

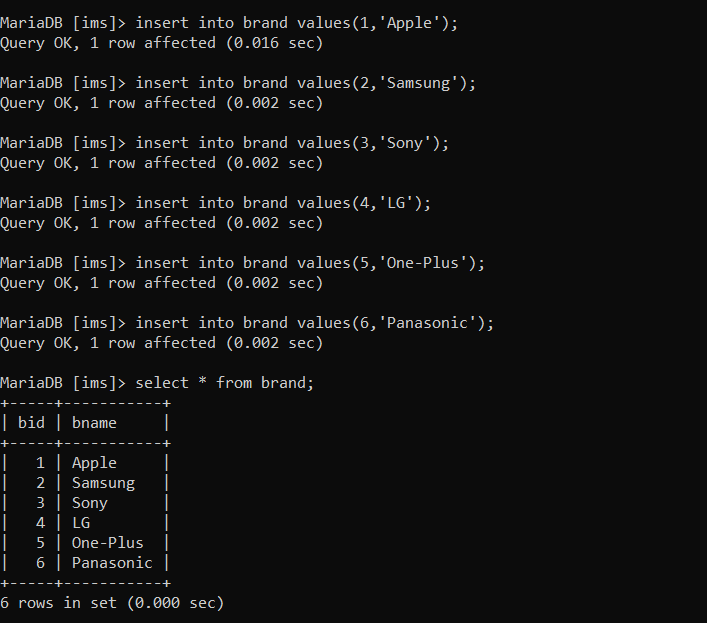
FOREIGN KEY (cust\_id) REFERENCES customer(cust\_id),

FOREIGN KEY (pid) REFERENCES product(pid)

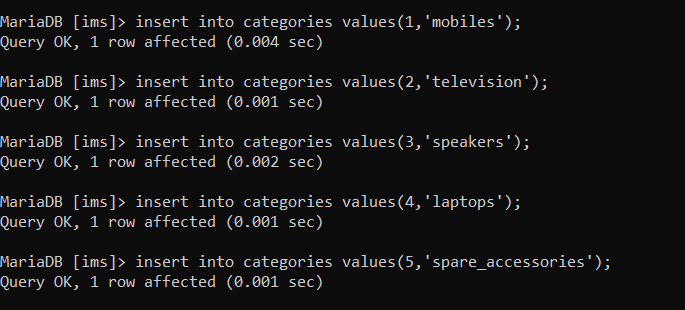
);

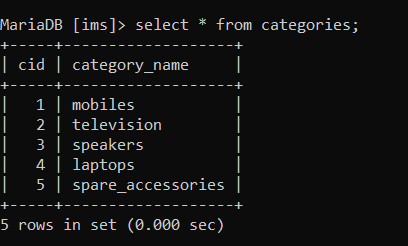
**Populating the Database**

**Brand table:**

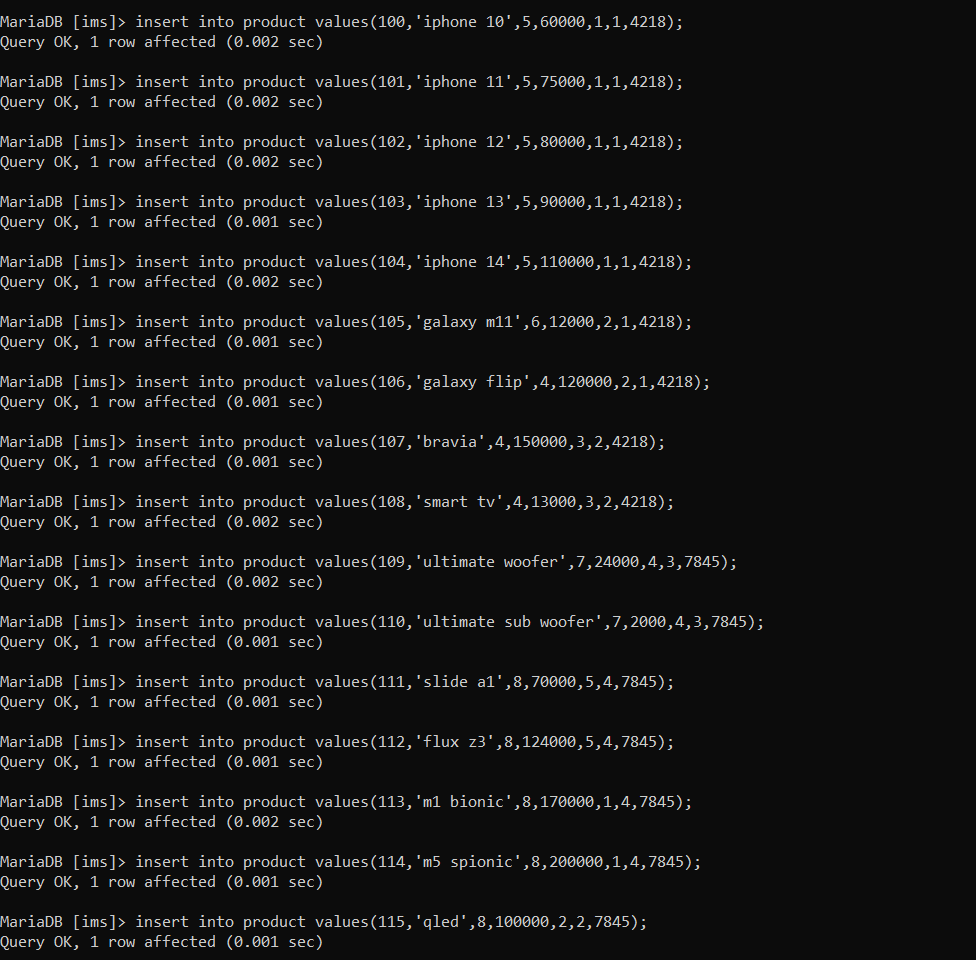


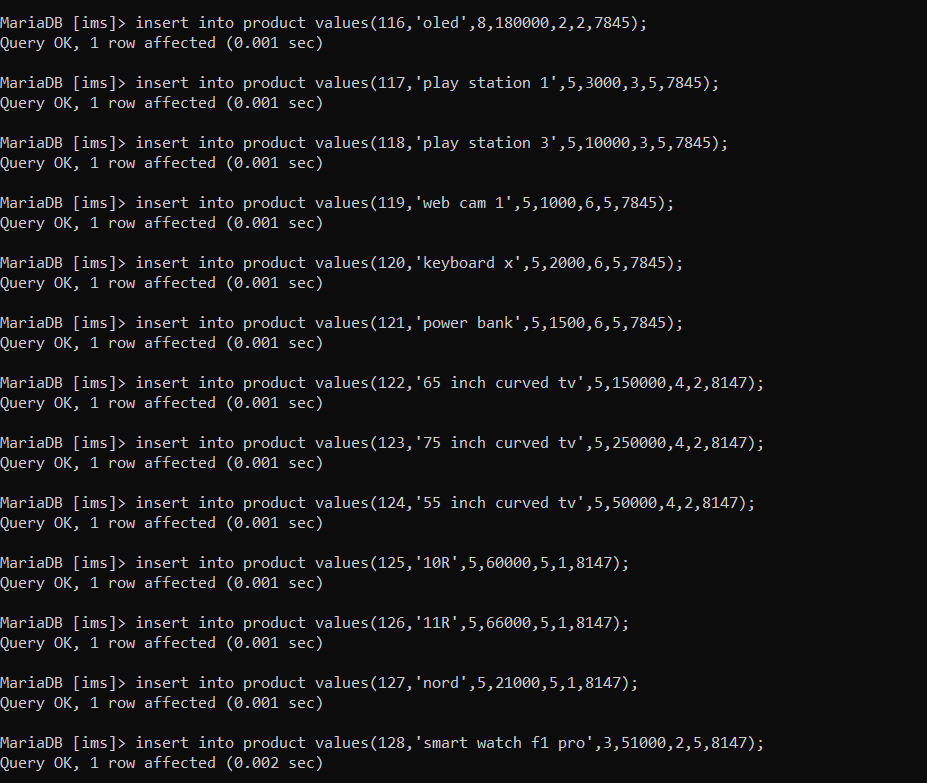
**Categories table:**

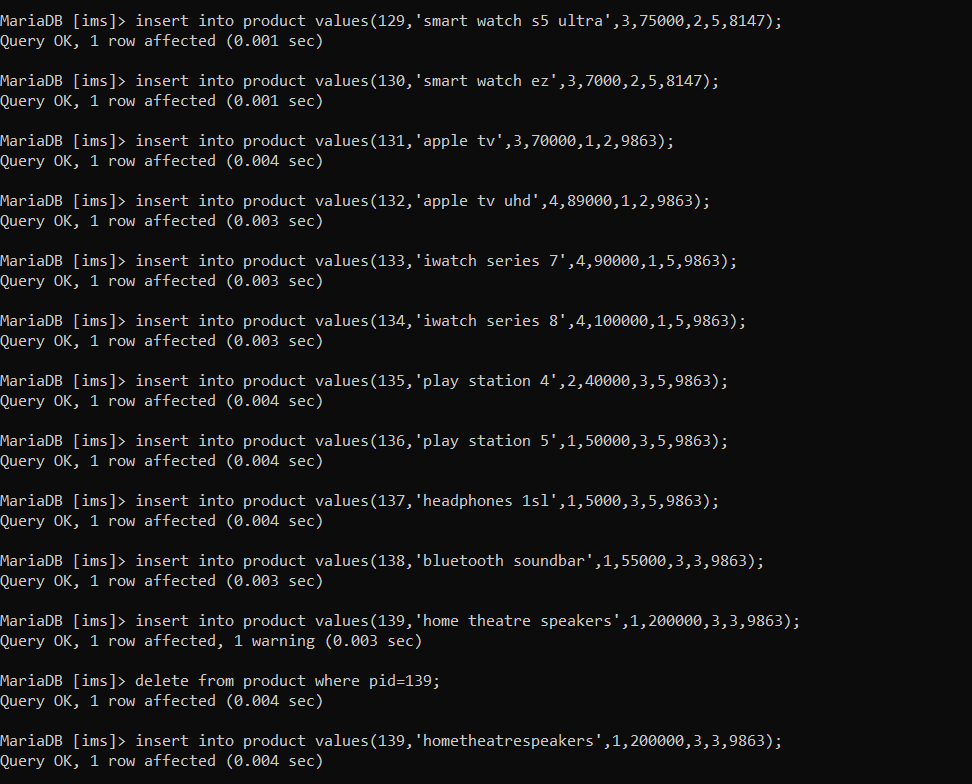


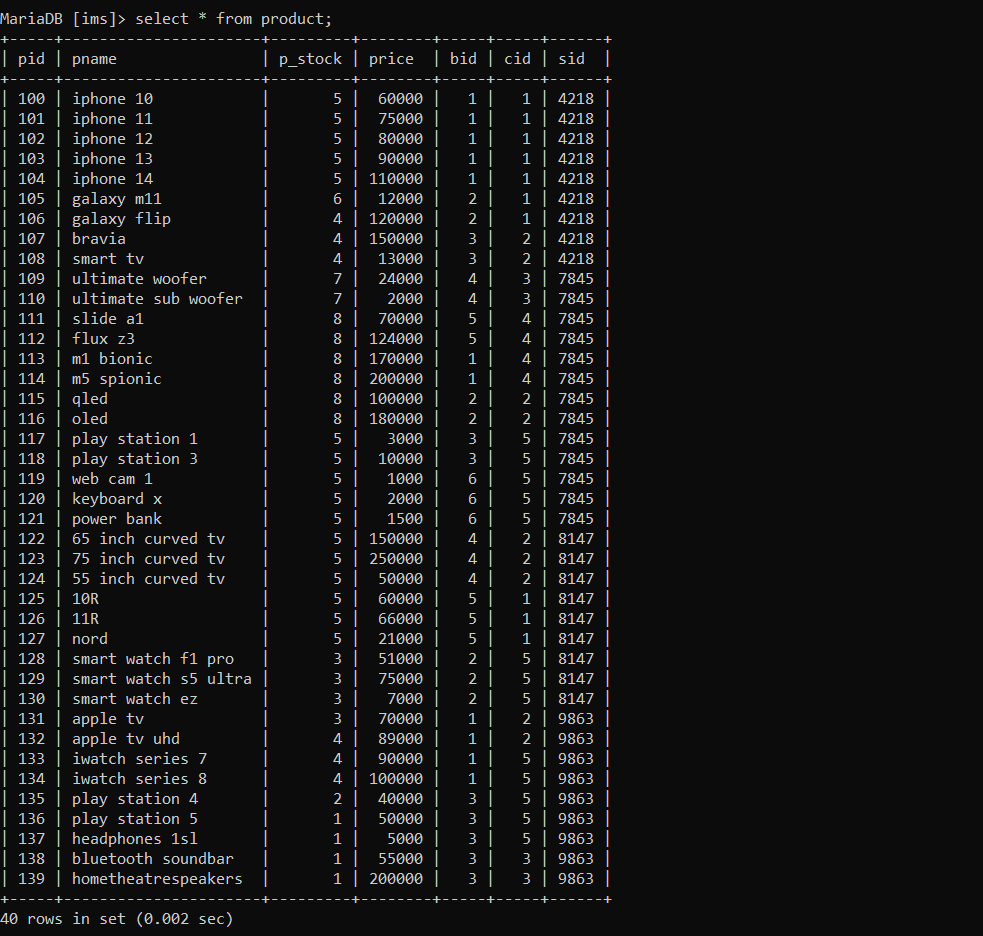


**Product table:**

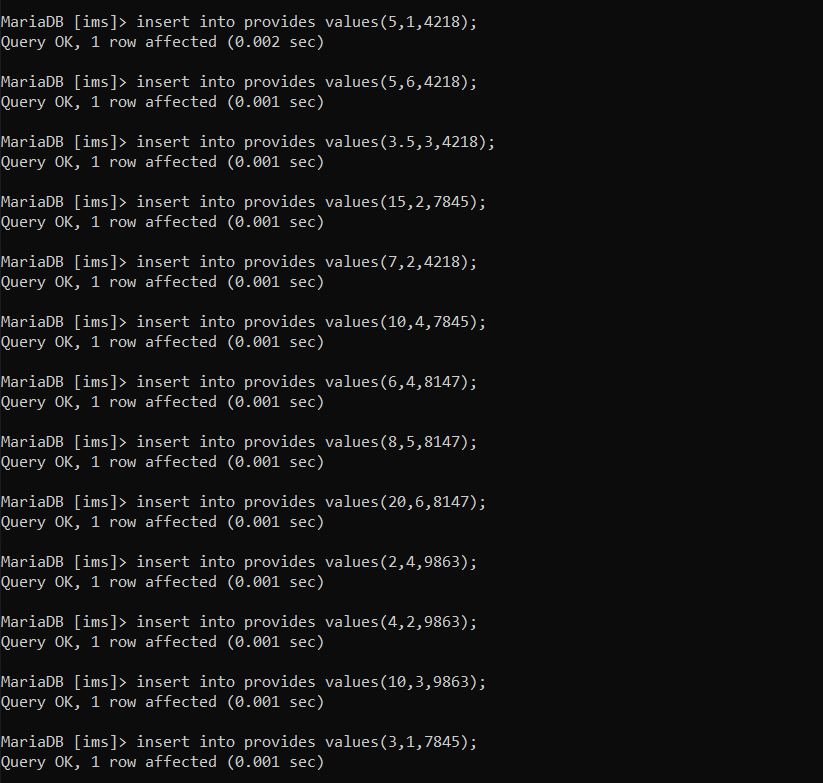


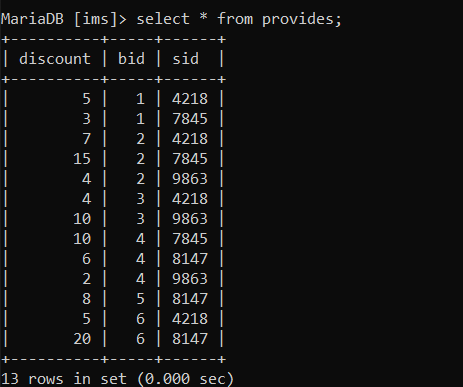




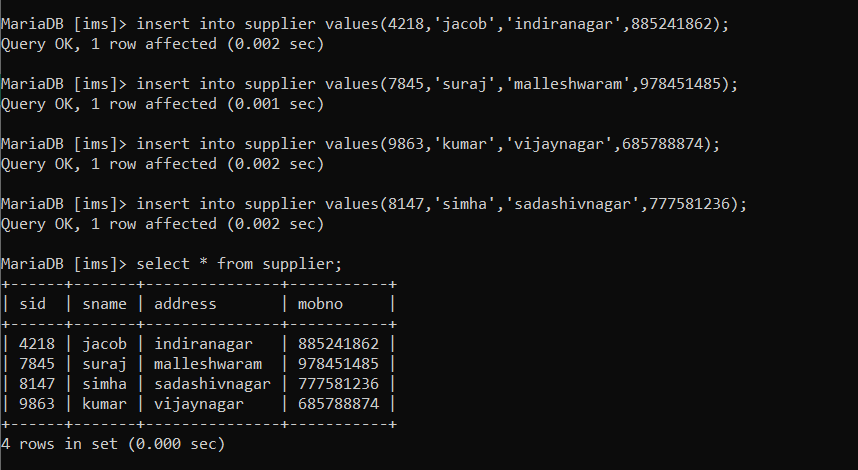


**Provides table:**

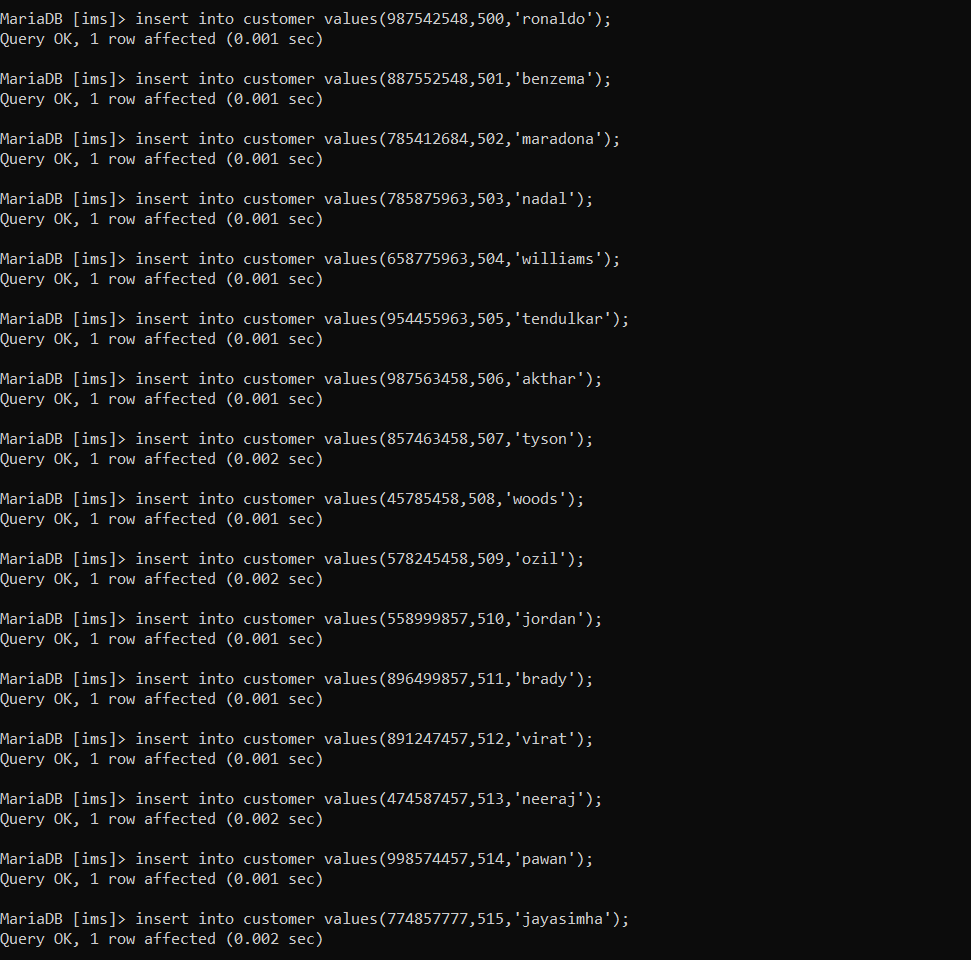


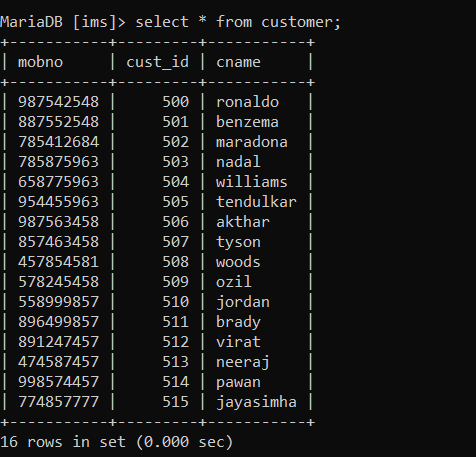


**Supplier table:**

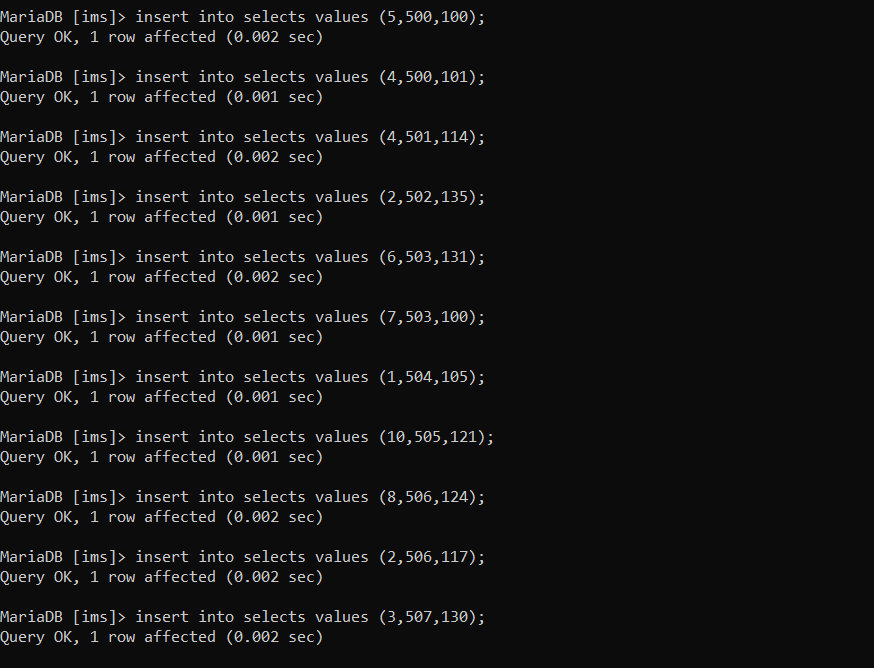


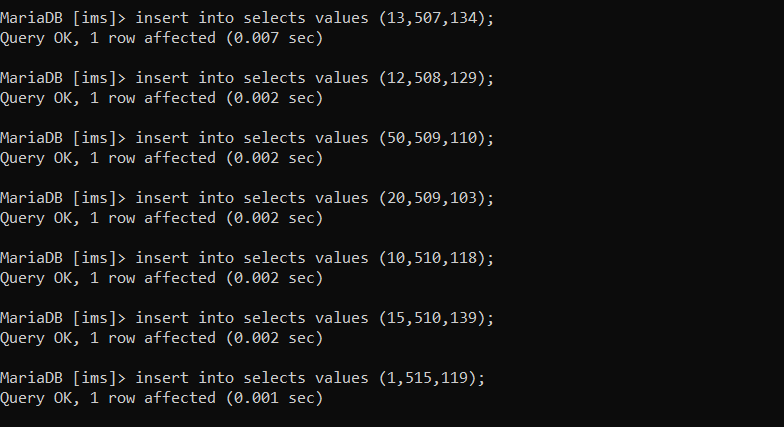
**Customer table:**

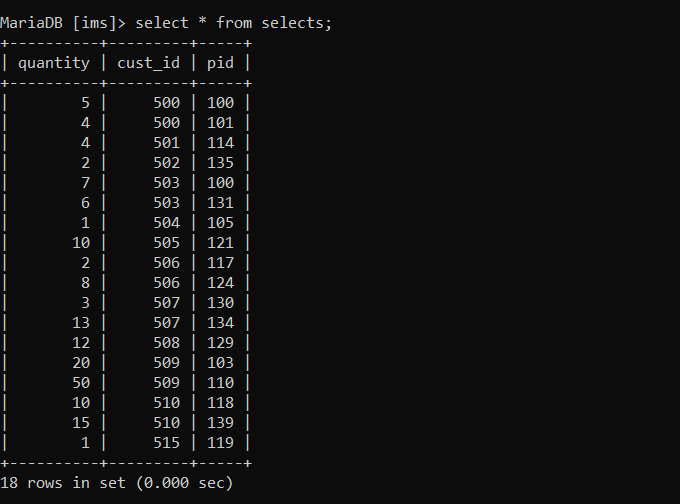




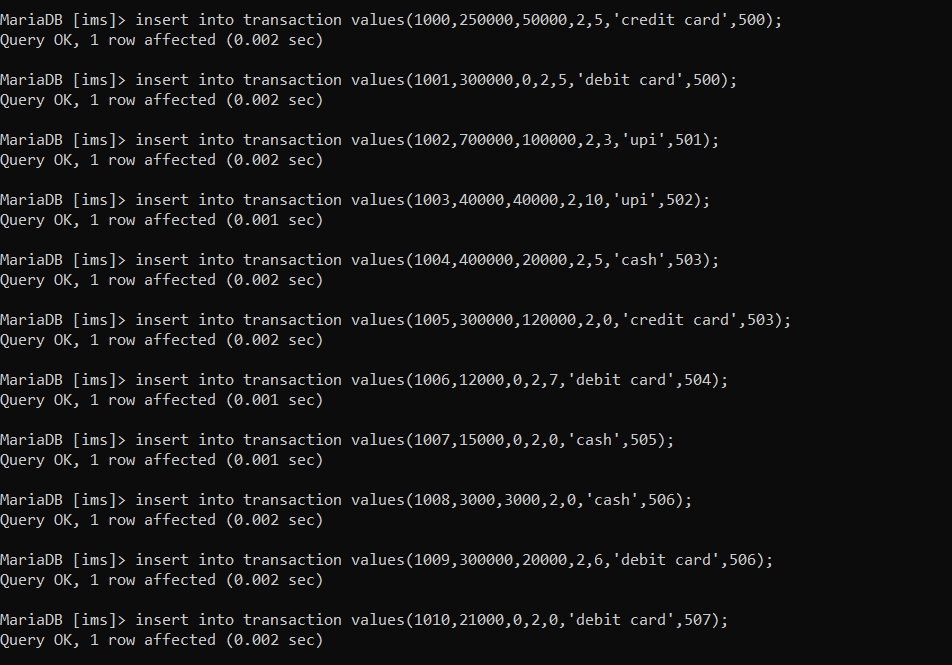
**Selects table:**

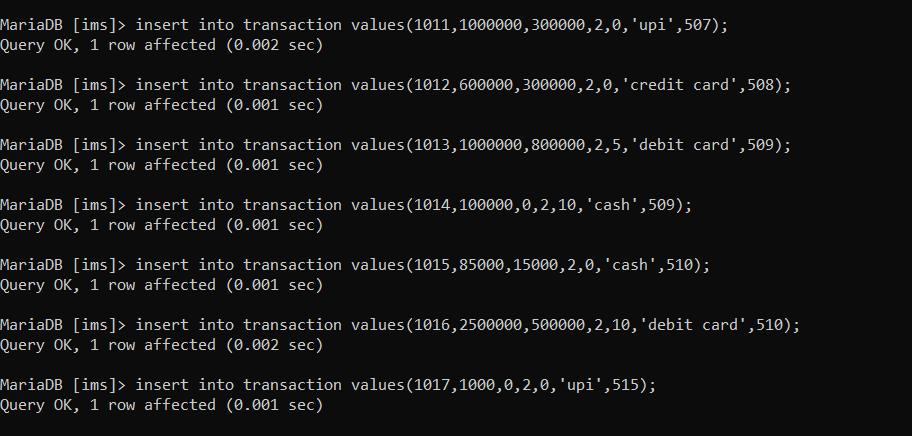


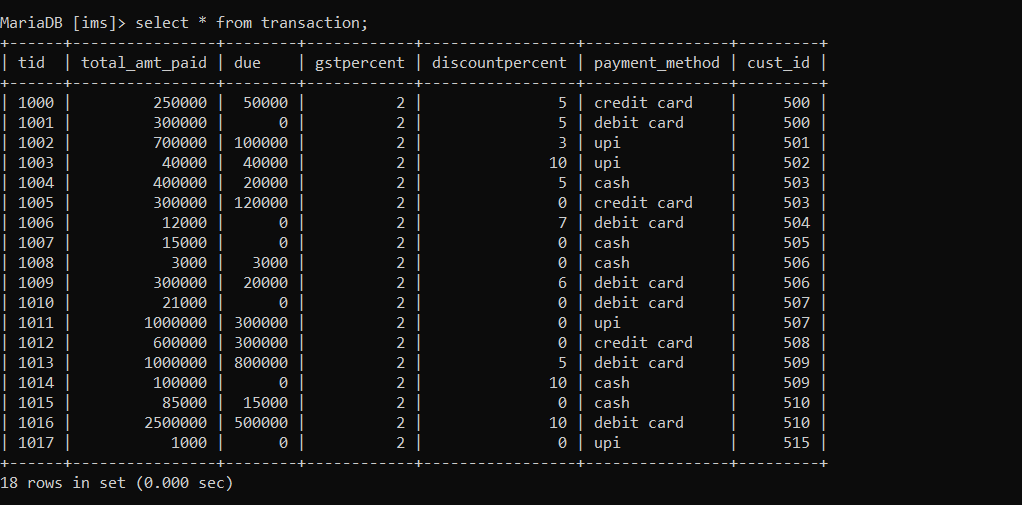




**Transaction table:**







**Tools Used**

Tool used for front end – Python (Tkinter)

Tool used for back end – mysql database

**Queries**

**Join queries (at least 6)**

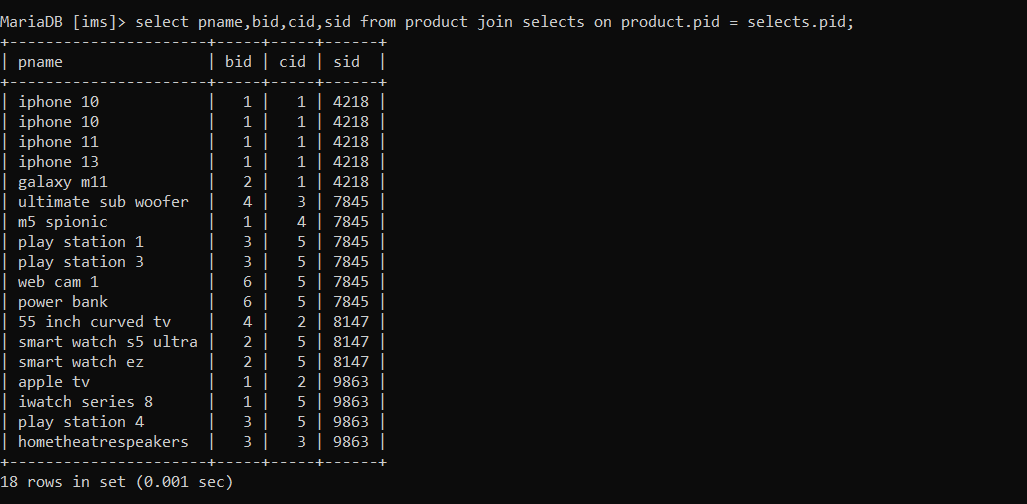
Write the query in English Language, Show the equivalent SQL statement and also screenshot of the query and the results.

Include 2 regular join, 2 co-related and 2 nested queries

**Regular joins:**

1) Display the product name, brand id, category id and supplier id of all the products that have been bought by the customers.

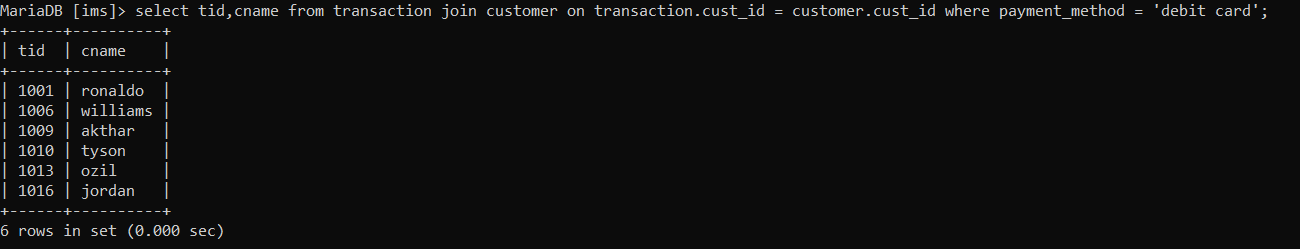
Query: select pname,bid,cid,sid from product join selects on product.pid = selects.pid;



2) select the transaction id and customer name of all the customers who have completed their payment using ‘debit card’

Query: select tid,cname from transaction join customer on transaction.cust\_id = customer\_cust\_id where

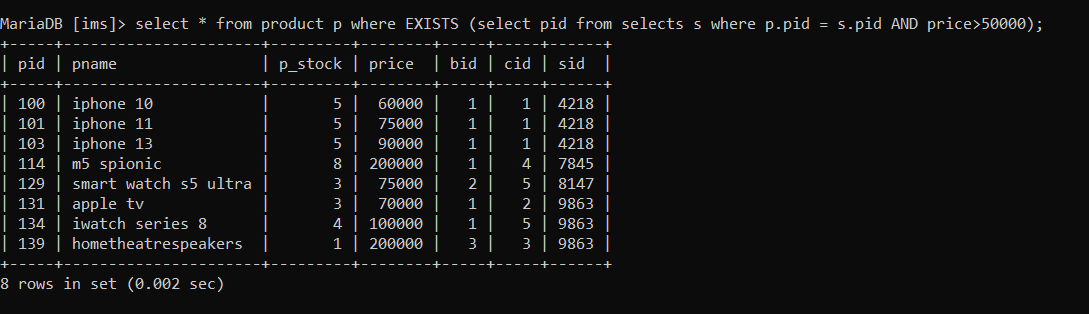
Payment\_method = ‘debit card’;



**Co - related Queries:**

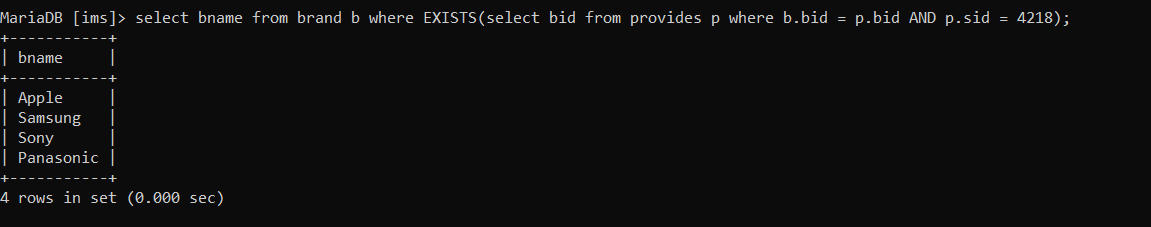
1) Display the product details of the all the products bought by the customers and whose price is greater than 50000.

Query: select \* from product p where EXISTS (select pid from selects s where p.pid = s.pid AND price > 50000);



2) select the names of all the brands which provide discount to the products sold by the supplier with supplier id 4218.

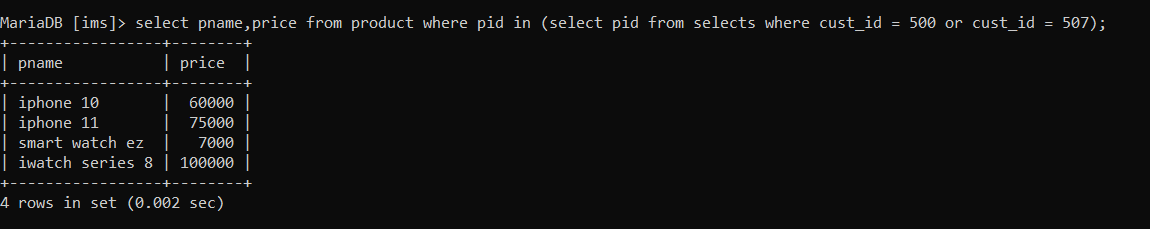
Query: select bname from brand b where EXISTS (select bid from provides p where b.bid = p.bid AND p.sid = 4218);



**Nested Queries:**

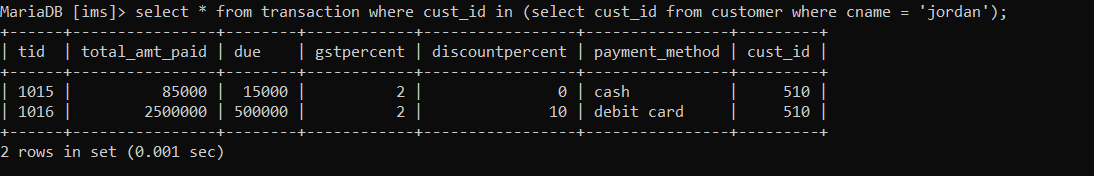
1) select the name and price of all the products which have been bought by the customers with customer id 500 or 507.

Query: select pname.price from products where pid in (select pid from selects where cust\_id = 500 or cust\_id = 507);



2) Display all the transaction details of the customer by name ‘jordan’.

Query: select \* from transaction where cust\_id in (select cust\_id from customer where cname = ‘jordan’).

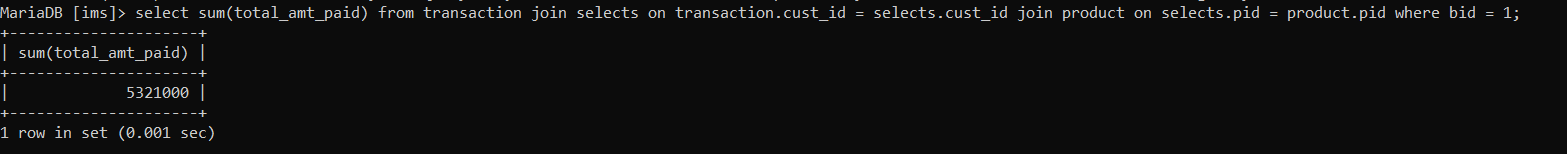


**Aggregate Functions (at least 2)**

Showcase at least 2 Aggregate function queries. Write the query in English Language, Show the equivalent SQL statement and also screenshot of the query and the results

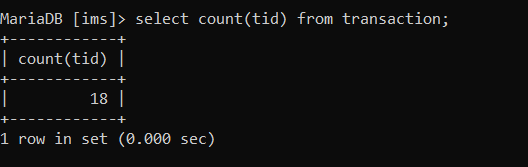
1) Total revenue from the sale of the brand with brand id = 1 i.e apple products.

Query: select sum(total\_amt\_paid) from transaction join selects on transaction.cust\_id = selects.cust\_id join product on selects.pid = product.pid where bid = 1;



2) Display the count of the total number of transactions the inventory management system has recorded.

Query: select count(tid) from transaction;

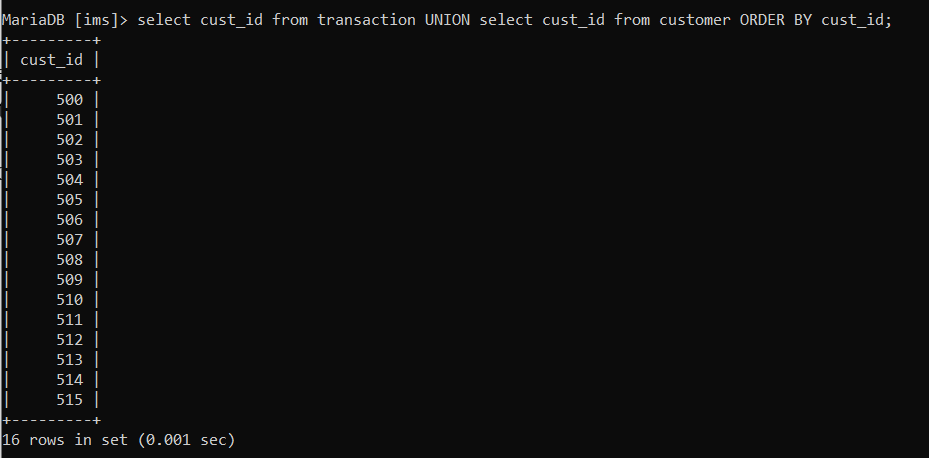


**Set Operations (at least 2)**

Showcase at least 2 Set Operations queries. Write the query in English Language, Show the equivalent SQL statement and also screenshot of the query and the results.

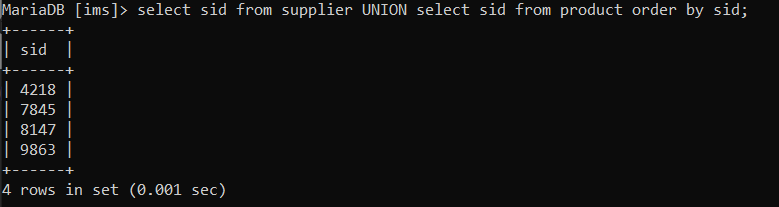
1) Display the customer id’s of all the customers who have made transactions and order them by their respective customer id’s

Query: select cust\_id from transaction UNION select cust\_id from customer ORDER BY cust\_id;



2) Display the supplier id of all the suppliers who supply the products and order them by their sid.

Query: select sid from supplier UNION select sid from product order by sid;



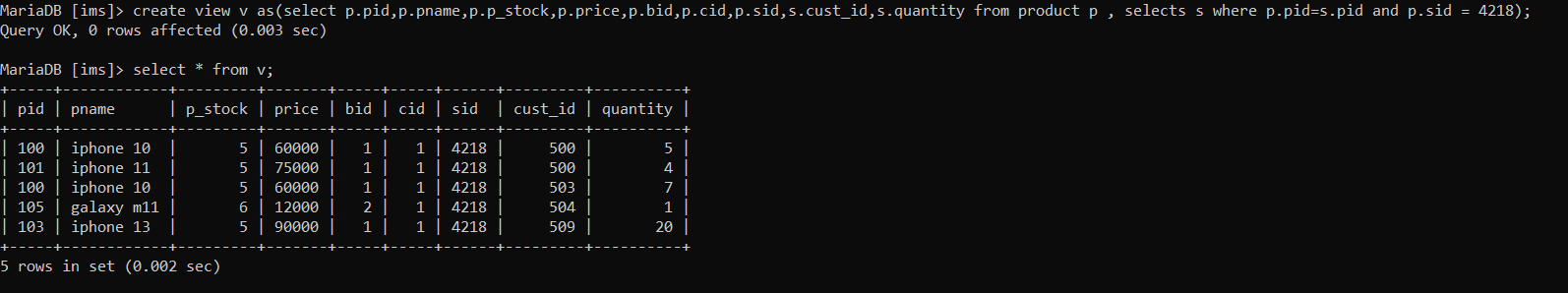
**View (atleast 1)**

Demonstrate creation and querying one view

View to display the product details, customer id, quantity of all products sold which have been supplied by the supplier with supplier id = 4218.

Creation: create view v as(select p.pid,p.pname,p.pstock,p.price,p.bid,p.cid,p.sid,cust\_id,quantity from product p , selects s where p.pid = s.pid and p.sid = 4218);

Querying: select \* from v;



**Triggers (Functions or Procedures)**

**PROCEDURE:**

Create a Function or a Procedure. State the objective of the function / Procedure. Run and

display the results.

Procedure: This procedure takes one input parameter named var where the user provides the id of the respective brand (bid), and the procedure displays all the products and their details of that particular brand and also displays the products that have incurred sales in that particular brand.

Creation:

DELIMITER &&

CREATE PROCEDURE Sales\_Details(IN var INT)

BEGIN

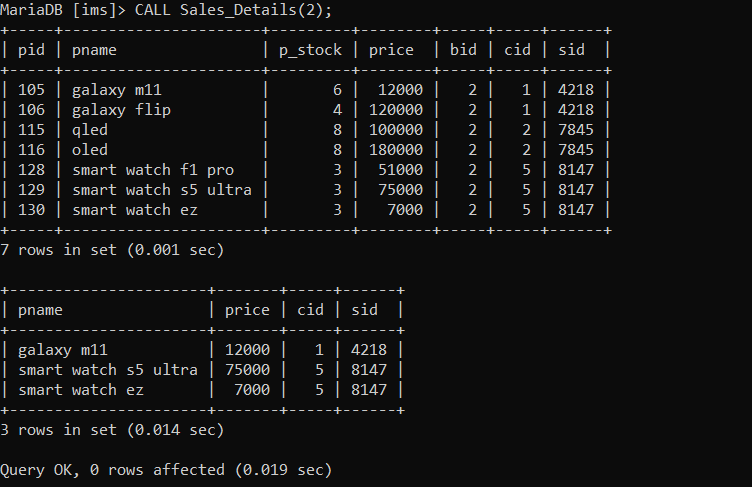
select \* from product where bid = var;

select pname,price,cid,sid from product p join selects s on p.pid = s.pid where bid = var;

END &&

DELIMITER ;

Calling the stored Procedure: CALL Sales\_Details(2);



**TRIGGER:**

Here a trigger has been created to check weather the user has entered a valid number for the product stock.

Creation:

DELIMITER $$

CREATE TRIGGER stock\_check

BEFORE INSERT ON product FOR EACH ROW

BEGIN

DECLARE error\_msg VARCHAR(255);

SET error\_msg = ('enter valid product stock');

IF NEW.p\_stock < 0

THEN

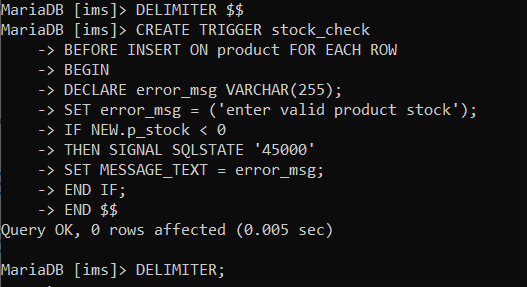
SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = error\_msg;

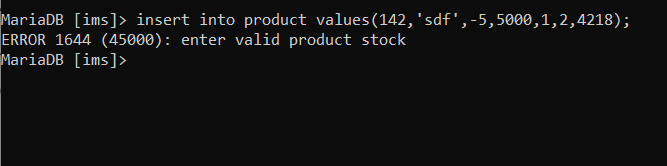
END IF;

END $$

DELIMITER;



This trigger raises an error when invalid stock is entered i.e less than zero.



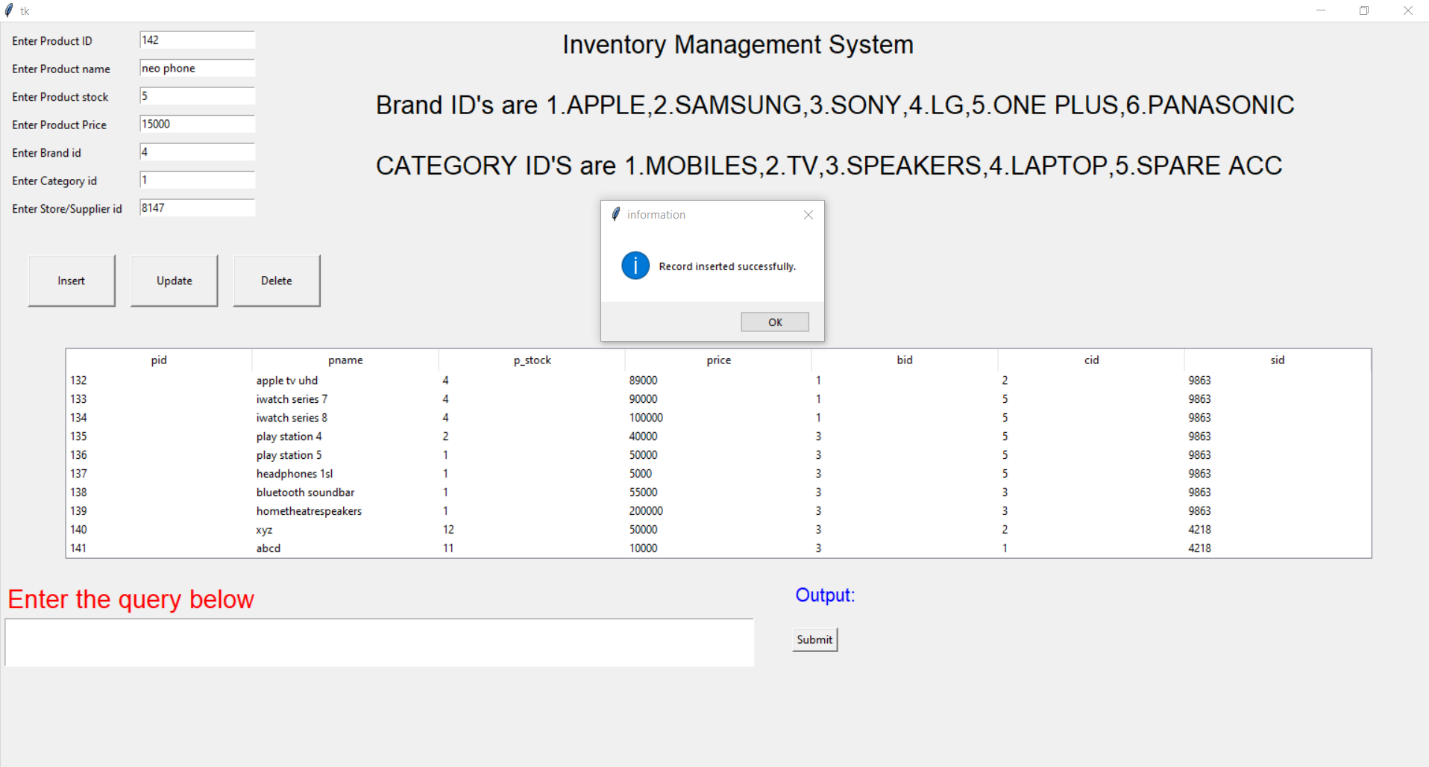
**Developing a Frontend**

The frontend should support

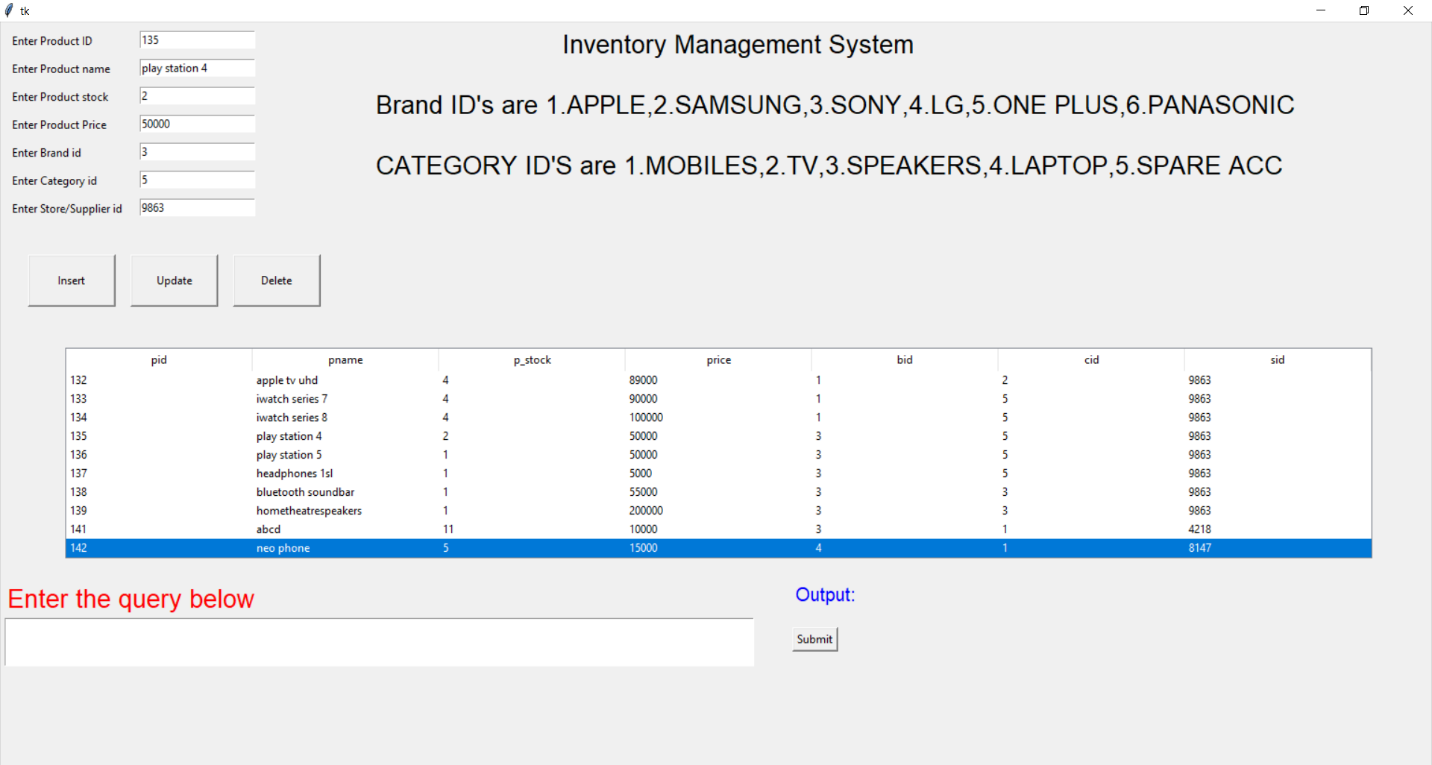
1. Addition, Modification and Deletion of records from any chosen table

2. There should be a window to accept and run any SQL statement and display the result

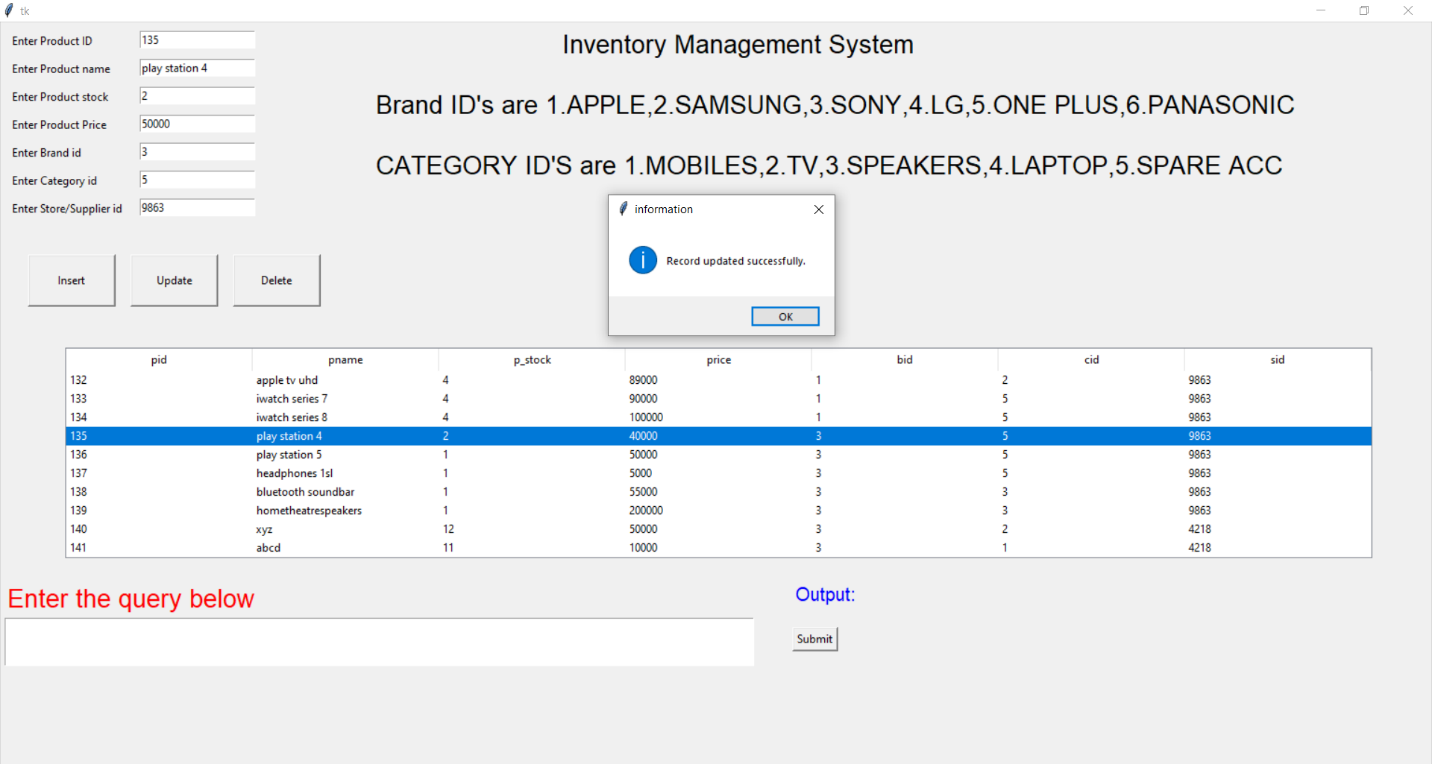
Addition:



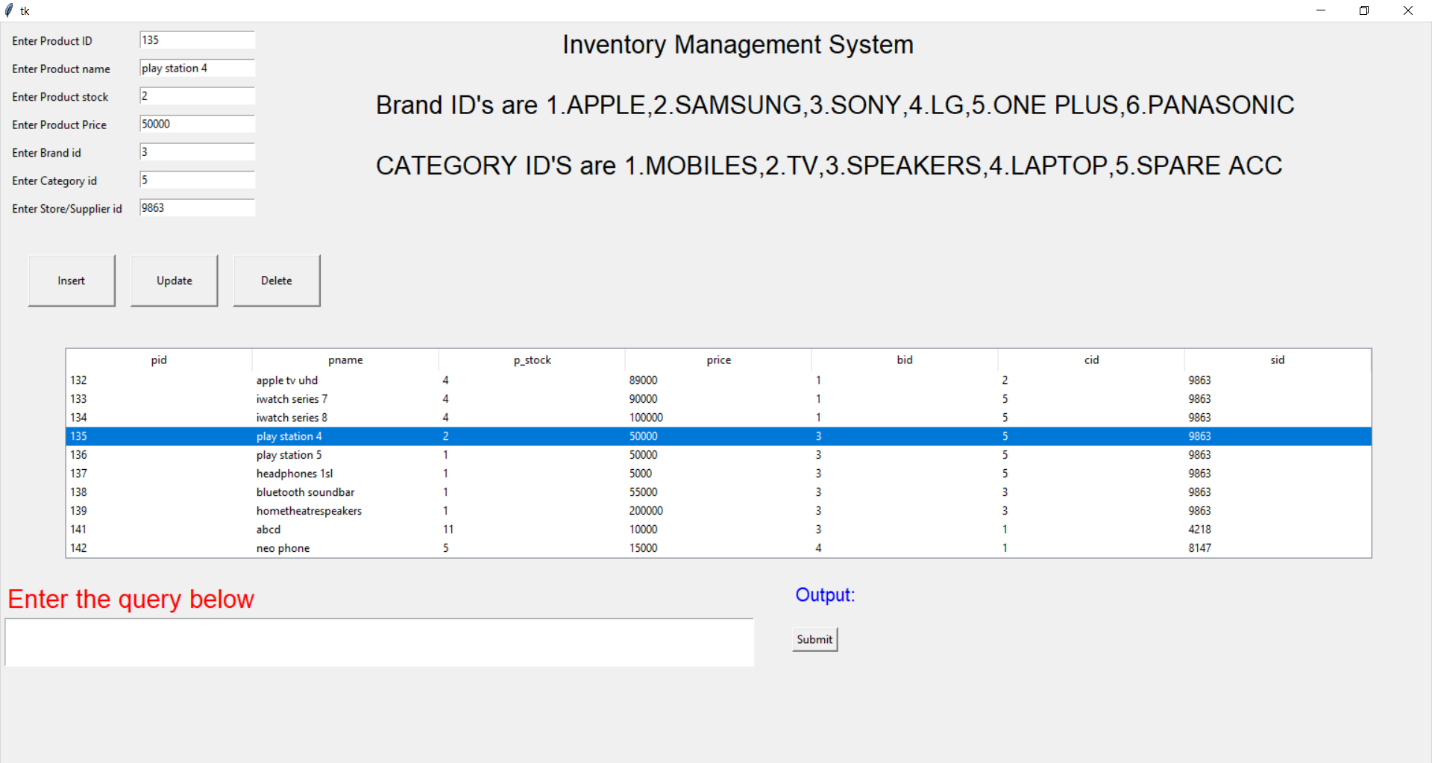
Record inserted successfully:



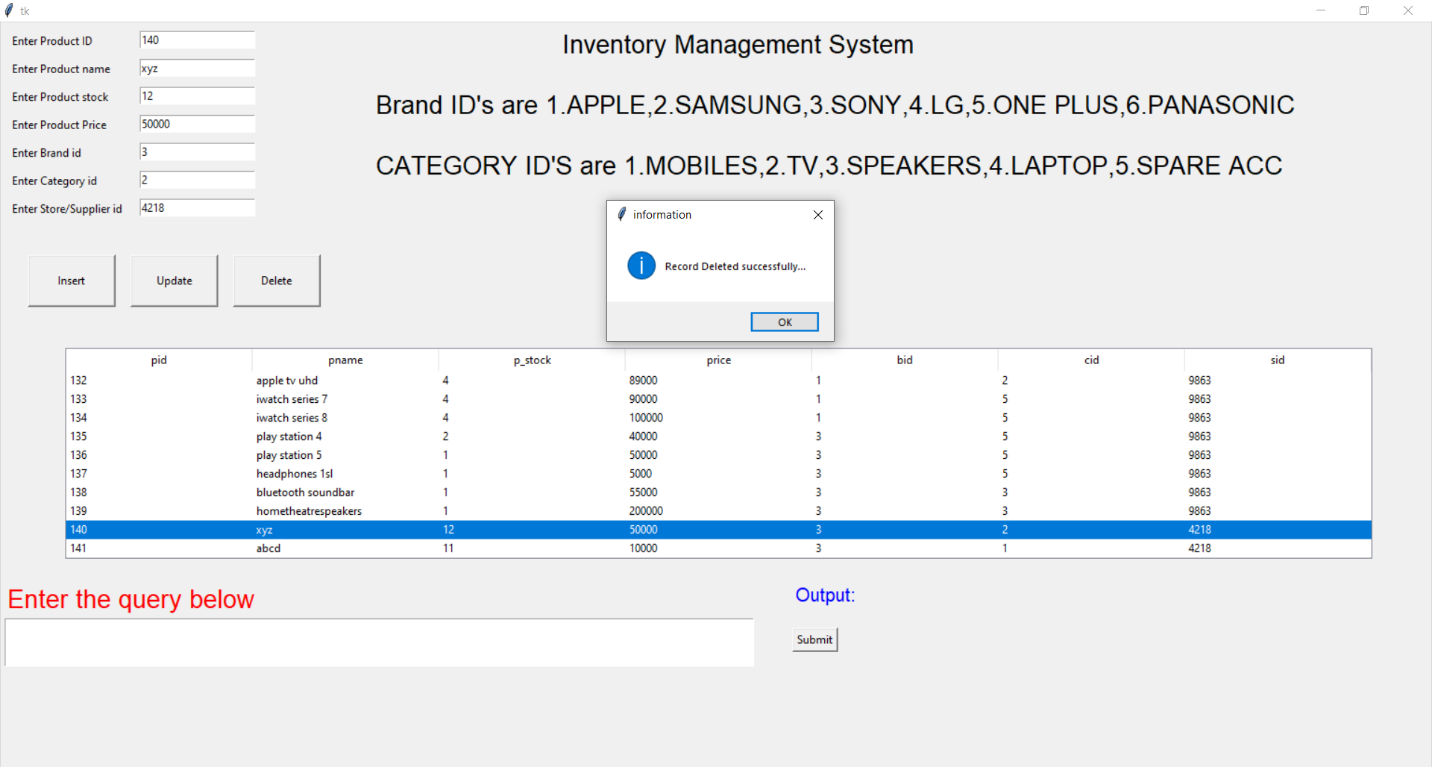
Modification:



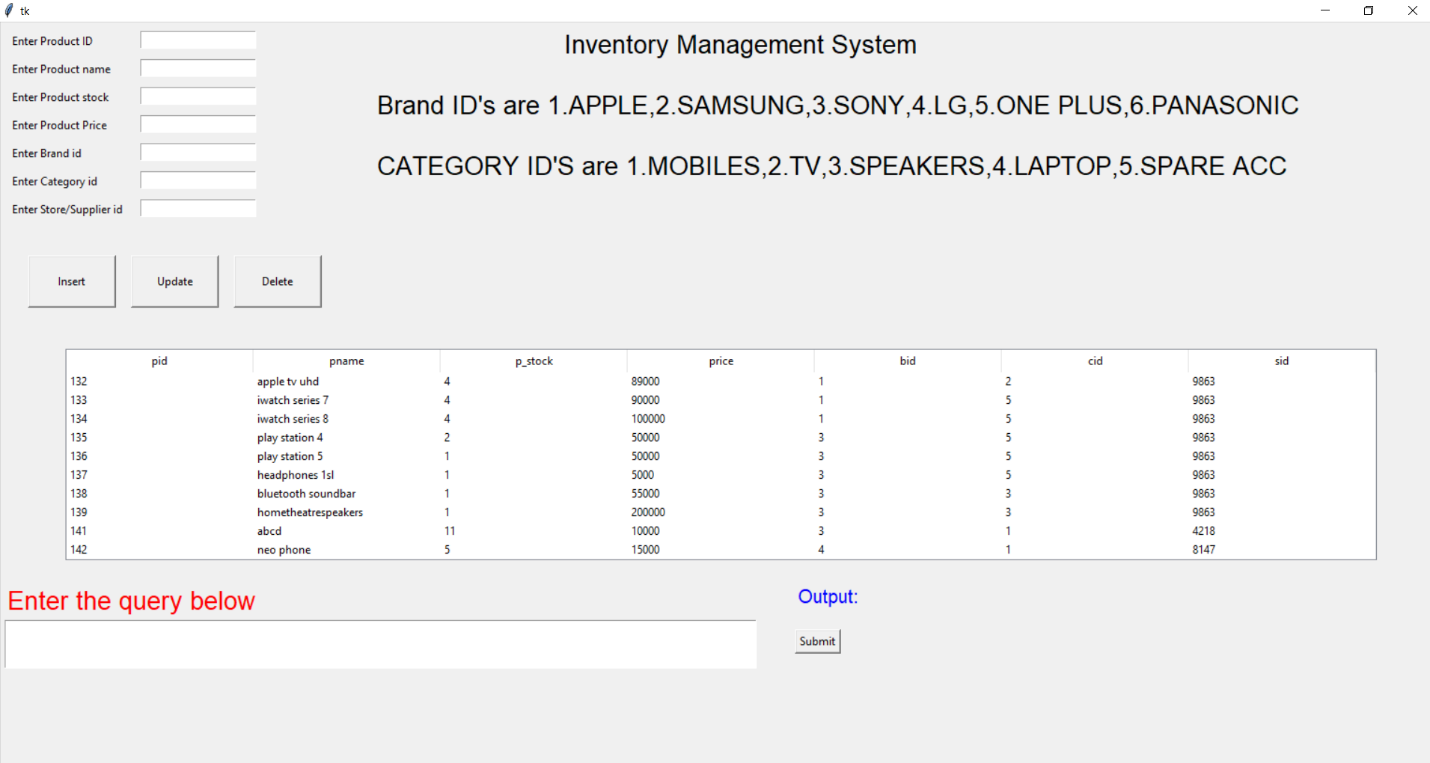
Record Updated successfully:



Deletion:



Record deleted successfully:



2) Window to accept and run any SQL statement and display the result.

