**DBMS - MINI PROJECT**

**“Water Refill Station Management System”**

Submitted By:

Name: Adithya M

SRN: PES1UG20CS621

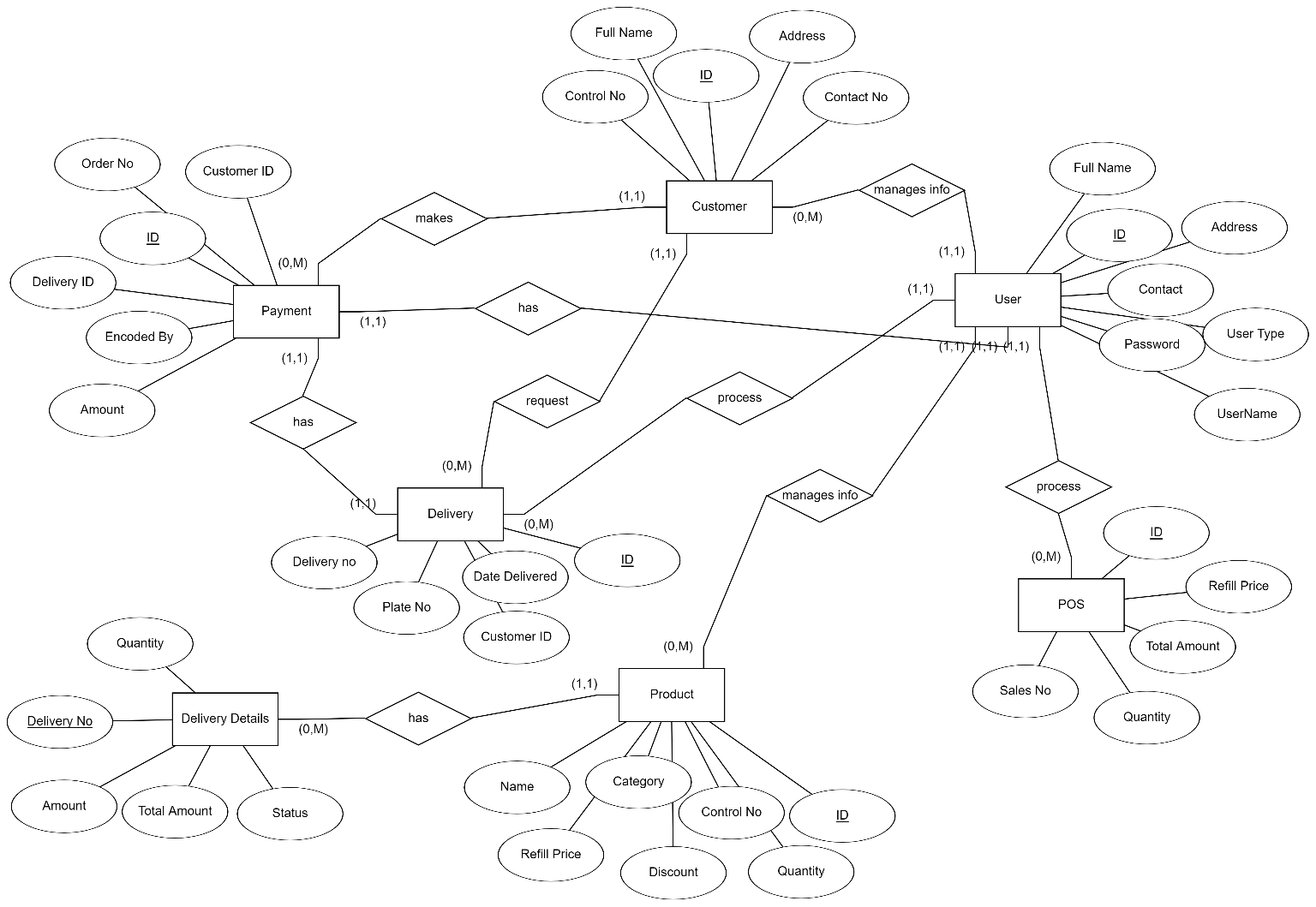
V Semester Section K

**ABSTRACT**

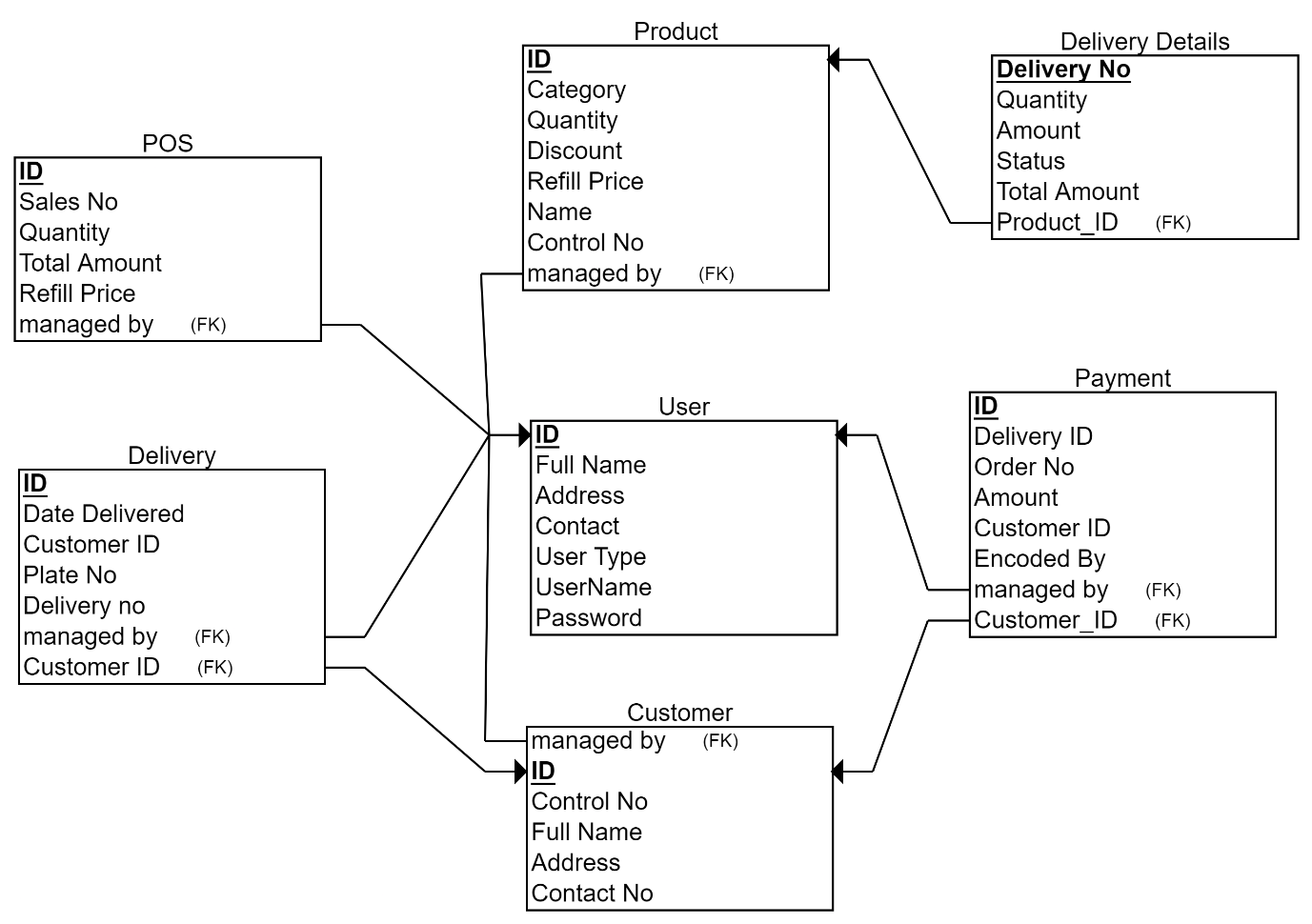
The purpose of this system is to help its clients for an easier and efficient management of stations without sacrificing costs or output.

This project enables the user to record things that are being purchased by the client. The water refilling station management is capable of viewing the item that is already delivered. Setting up a water refilling station whereby the station will cater to the reproduction of water through selling large and small amounts of volume.

**ER Diagram**

****

**Relational Schema**

****

**DDL statements - Building the database**

CREATE TABLE User(

user\_id int,

name varchar(20) not null,

address varchar(20),

contact int,

user\_type varchar(20) not null,

user\_name varchar(20) not null,

password varchar(20) not null,

primary key(user\_id)

);

CREATE TABLE Product(

product\_id int,

category varchar(20) not null,

quantity int not null,

discount float,

refill\_price float,

name varchar(20) not null,

managed\_by int not null,

primary key(product\_id),

foreign key(managed\_by) references User(user\_id)

);

CREATE TABLE Customer(

customer\_id int,

name varchar(20) not null,

address varchar(20),

contact\_no int,

managed\_by int not null,

primary key(customer\_id),

foreign key(managed\_by) references User(user\_id)

);

CREATE TABLE POS(

pos\_id int,

sales\_no int not null,

quantity int not null,

total\_price float not null,

refill\_price float not null,

managed\_by int not null,

primary key(pos\_id),

foreign key(managed\_by) references User(user\_id)

);

CREATE TABLE Delivery\_Details(

delivery\_no int,

quantity int not null,

total\_price float not null,

status varchar(20) not null,

product\_id int not null,

primary key(delivery\_no),

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

);

CREATE TABLE Payment(

payment\_id int,

delivery\_id int not null,

order\_no int not null,

total\_price float not null,

customer\_id int not null,

managed\_by int not null,

primary key(payment\_id),

foreign key(customer\_id) references Customer(customer\_id),

foreign key(managed\_by) references User(user\_id)

);

CREATE TABLE Delivery(

delivery\_id int,

delivery\_date date not null,

customer\_id int not null,

plate\_no varchar(20) not null,

delivery\_no int not null,

managed\_by int not null,

primary key(delivery\_id),

foreign key(managed\_by) references User(user\_id),

foreign key(customer\_id) references Customer(customer\_id)

);

**Populating the Database**

|  |  |
| --- | --- |
| INSERT into User  VALUES(  1234,  "Adi",  "F Block",  1234567890,  "admin",  "adi",  "pass"  );  INSERT into User  VALUES(  1357,  "Rahul",  "G Block",  123234240,  "moderator",  "rahul",  "pass2"  );  INSERT into User  VALUES(  4321,  "Suhas",  NULL,  924367840,  "intern",  "suhas",  "pass3"  );  INSERT INTO Product  VALUES(  4244,  "Can",  100,  5.0,  10.0,  "Bislerii",  1234  );  INSERT INTO Product  VALUES(  2342,  "Bottle",  75,  1.0,  2.0,  "Aqua",  1357  );  INSERT INTO Customer  VALUES(  53531,  "yehaw",  "Lmao Block",  4834393322,  1234  );  INSERT INTO Customer  VALUES(  53532,  "yehaw2",  "lol Block",  483334422,  1357  );  INSERT INTO Customer  VALUES(  53533,  "yes3",  NULL,  4353322,  4321  );  INSERT INTO POS  VALUES(  3242,  1,  2,  100.0,  10.0,  1234  );  INSERT INTO POS  VALUES(  3243,  2,  3,  200.0,  20.0,  1357  ); | INSERT INTO POS  VALUES(  3244,  3,  4,  300.0,  30.0,  4321  );  INSERT INTO Delivery\_Details  VALUES(  1,  2,  100.0,  "pending",  4244  );  INSERT INTO Delivery\_Details  VALUES(  2,  3,  200.0,  "on the way",  2342  );  INSERT INTO Delivery\_Details  VALUES(  3,  4,  300.0,  "delivered",  3422  );  INSERT INTO Payment  VALUES(  243242,  1,  1,  100.0,  53531,  1234  );  INSERT INTO Payment  VALUES(  243243,  2,  2,  200.0,  53532,  1357  );  INSERT INTO Payment  VALUES(  243244,  3,  3,  300.0,  53533,  4321  );  INSERT INTO Delivery  VALUES(  124213,  DATE("2022-11-20"),  53531,  "KA50HP1234",  4248234,  1234  );  INSERT INTO Delivery  VALUES(  124214,  DATE("2022-01-03"),  53532,  "KA50BC2434",  4248234,  1357  );  INSERT INTO Delivery  VALUES(  124215,  DATE("2022-04-11"),  53533,  "TS50AD2524",  4248241,  4321  ); |

**Tool Used**

* Streamlit
* MySQL
* Python

**Queries**

**Join queries (at least 6)**

**--find product names whose delivery status is pending with join**

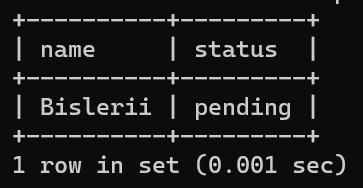
SELECT name,

status

FROM product

INNER JOIN delivery\_details ON product.product\_id = delivery\_details.product\_id

WHERE status = 'pending';



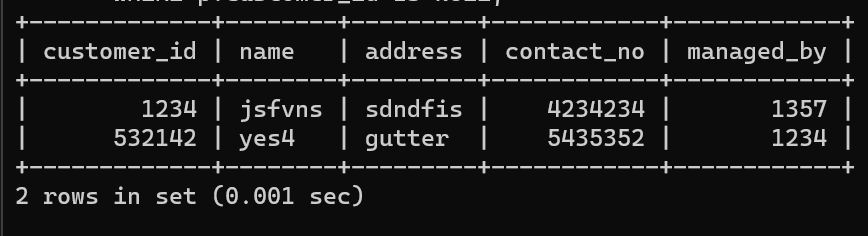
**-- using right join find payment details of customers who have not made any payment**

SELECT c.customer\_id, c.name, c.address, c.contact\_no, c.managed\_by

FROM customer as c

LEFT JOIN payment as p ON c.customer\_id = p.customer\_id

WHERE p.customer\_id IS NULL;



**-- using correlated subquery find the delivery details of the product with the highest refill\_price**

select \*

from delivery\_details

where product\_id = (

select product\_id

from product

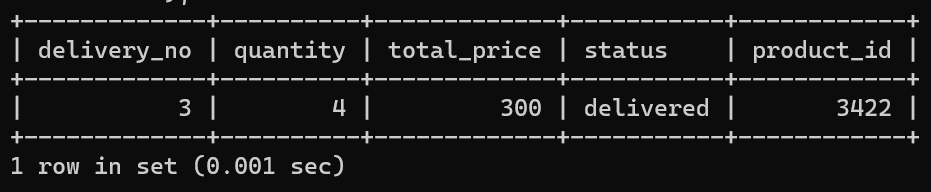
where refill\_price = (

select max(refill\_price)

from product

)

);



**-- using correlated subquery find the username and password of the user who has highest number of pos**

select user\_name,

password

from user

where user\_id = (

select user\_id

from pos

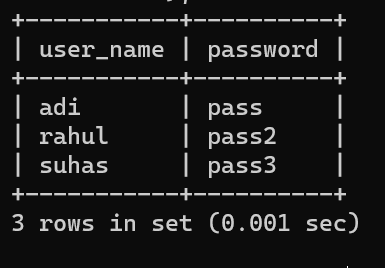
where pos\_id = (

select max(pos\_id)

from pos

)

);



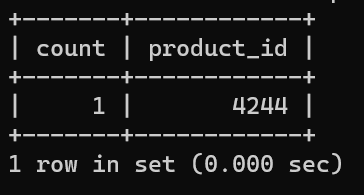
**Aggregate Functions (at least 2)**

**-- count the number of pending deliveries**

select count(delivery\_no) as count, product\_id

from delivery\_details

where status = 'pending';

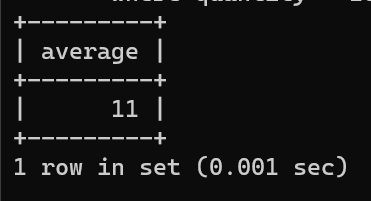


**--average refill price of products whose quantity is less than 10 and category is either bottle or tank**

select avg(refill\_price) as average

from product

where quantity < 1000 and category in ('bottle', 'tank');



**Set Operations (at least 2)**

**-- using union find the names of the products whose quantity is less than 10 or category is either bottle or tank**

select name

from product

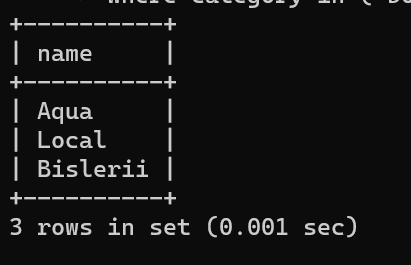
where quantity < 1000

union

select name

from product

where category in ('bottle', 'tank', 'dirty water');



**--using set difference find the payment details of customers who have not made any payment**

select c.customer\_id,

c.name,

c.address,

c.contact\_no

from customer as c

except

select c.customer\_id,

c.name,

c.address,

c.contact\_no

from customer as c

inner join payment as p on c.customer\_id = p.customer\_id;



**View (atleast 1)**

CREATE VIEW heavy\_ticket\_items AS

SELECT product\_id,

name,

refill\_price

FROM product

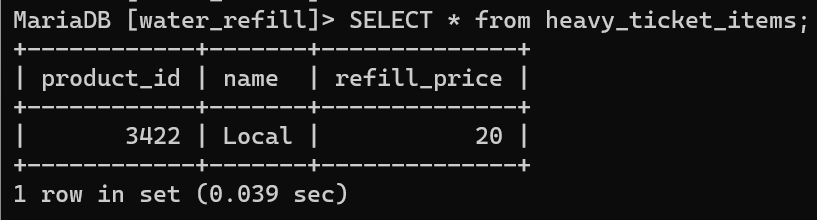
WHERE refill\_price > (

select avg(refill\_price)

from product

);

SELECT \* from heavy\_ticket\_items;

****

**Triggers (Functions or Procedures)**

**--decrement quantity after delivery status is changed to delivered**

DELIMITER $$

CREATE or replace procedure decrement\_quantity(IN p integer, IN q integer) BEGIN

UPDATE product

SET quantity = quantity - q

WHERE product\_id = p;

END;$$

DELIMITER ;

DROP TRIGGER update\_quantity;

DELIMITER $$

CREATE TRIGGER IF NOT EXISTS update\_quantity BEFORE

UPDATE ON delivery\_details

FOR EACH ROW BEGIN IF NEW.status = 'delivered'

THEN

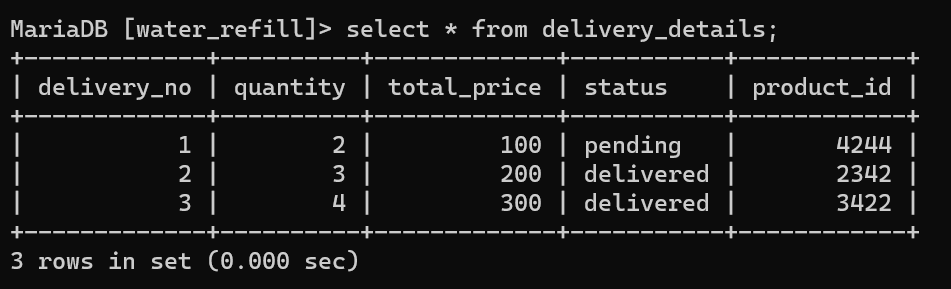
CALL decrement\_quantity(NEW.product\_id, NEW.quantity);

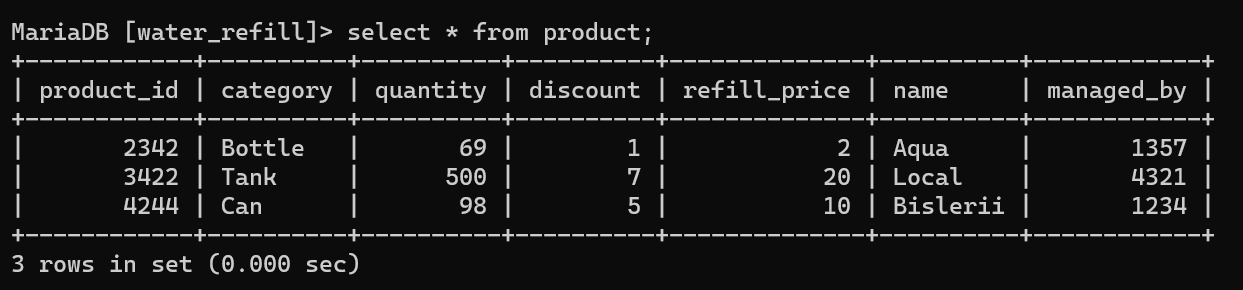
END IF;

END $$

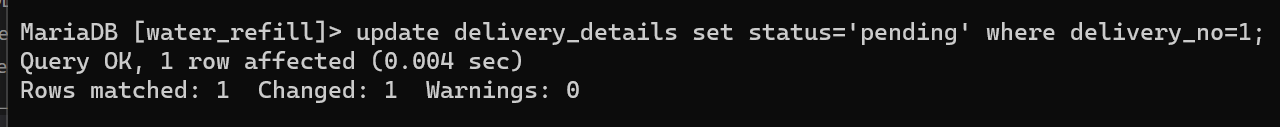
DELIMITER ;

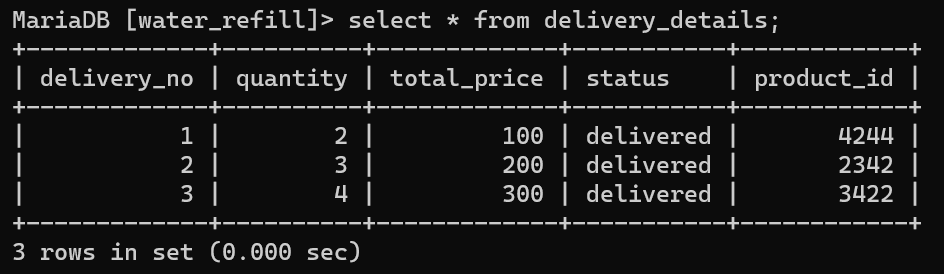
**Before update**

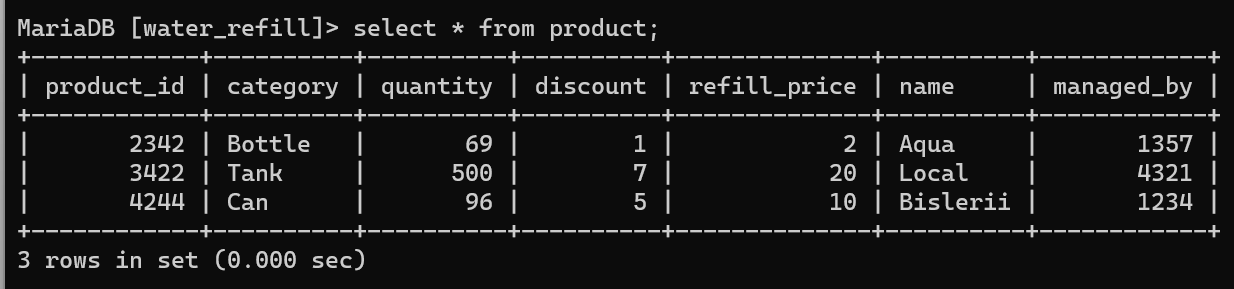
****

****

**After update**

****

****

****

**Developing a Frontend**

