DBMS - MINI PROJECT

“Water Refill Station Management System”

Submitted By:

Name: Adithya M

SRN: PES1UG20CS621

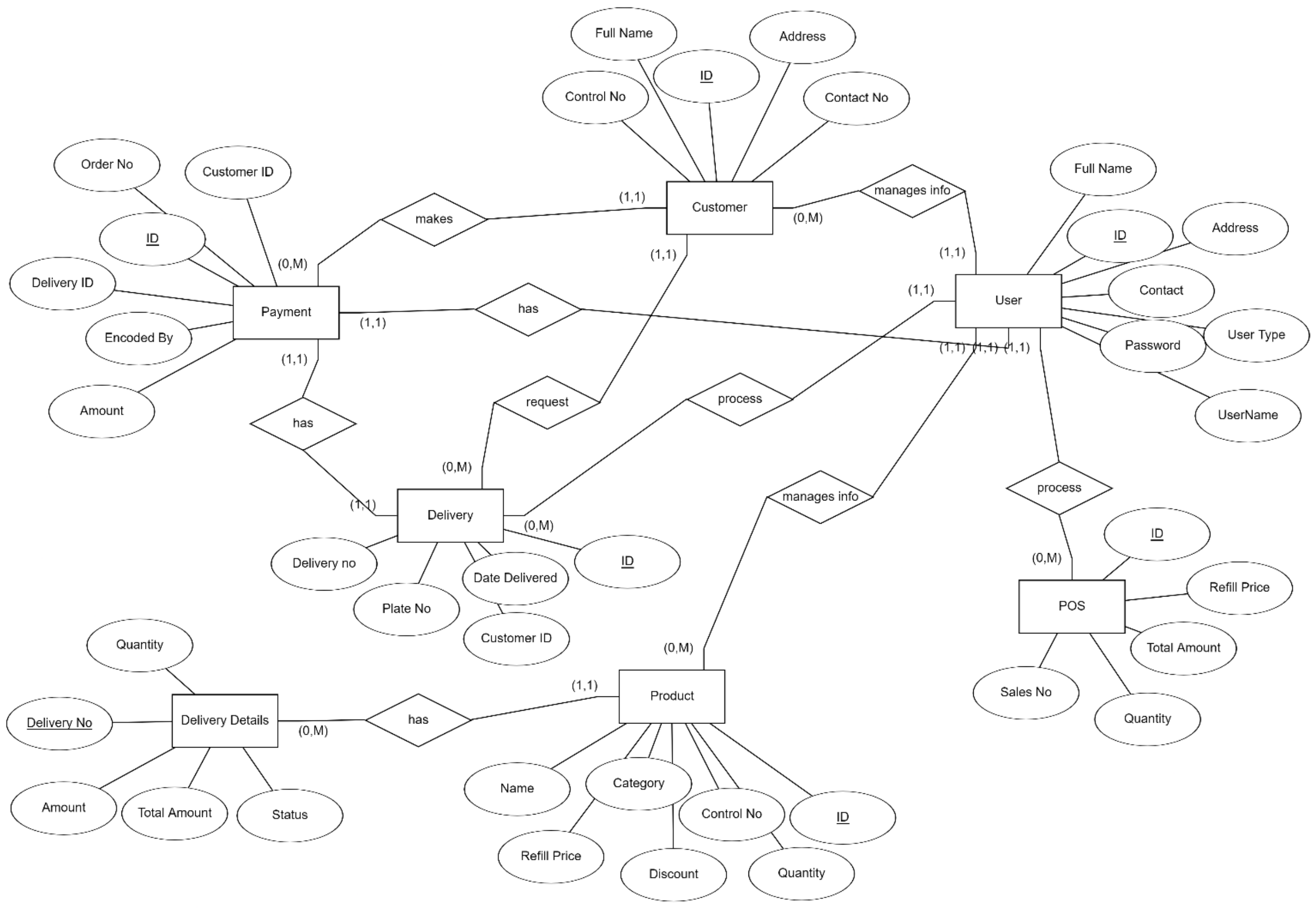
V Semester Section K

ABSTRACT

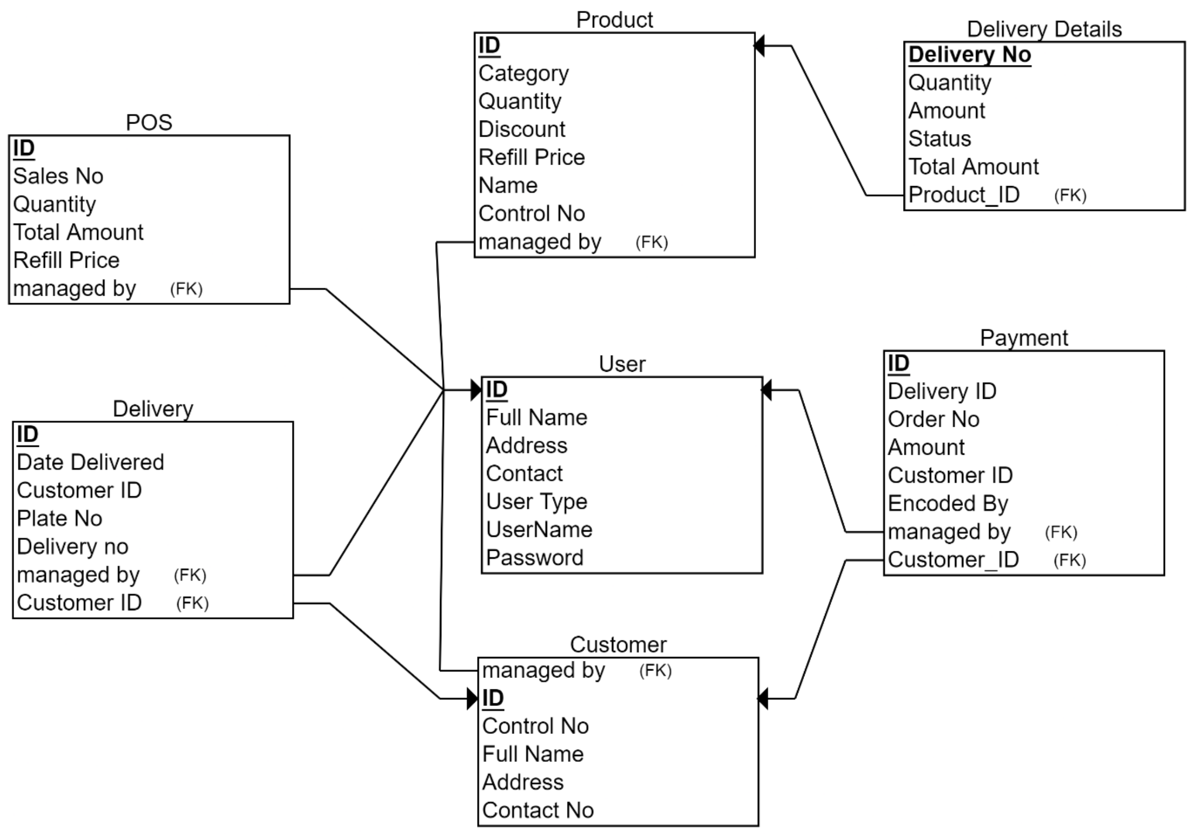
The purpose of this system is to help its clients for an easier and efficient management of stations without sacrificing costs or output.

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 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,
    );
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
    );
```

```
        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)

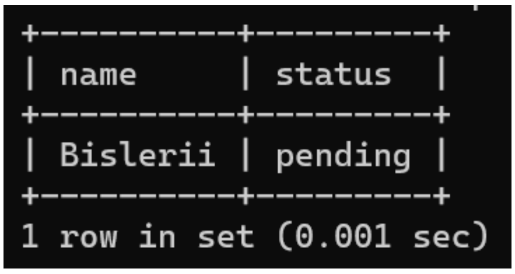
--find product names whose delivery status is pending with join

```
SELECT name,

    status

FROM product
```

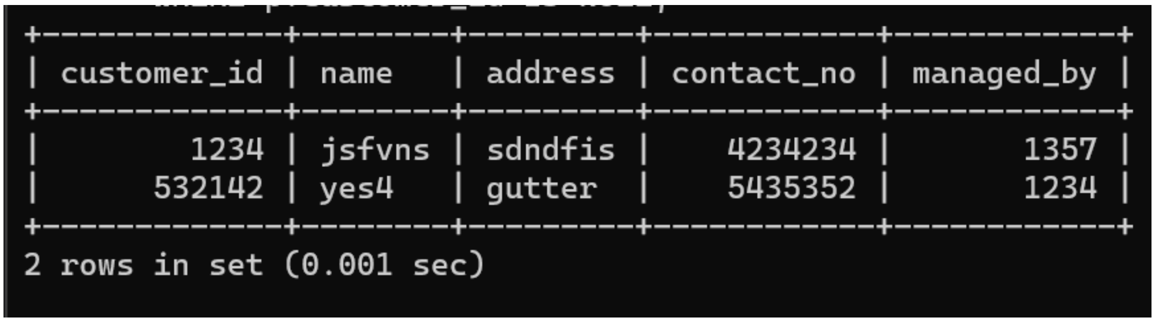
```
    INNER JOIN delivery_details ON product.product_id = delivery_details.product_id
WHERE status = 'pending';
```



```
+----------+---------+
| name     | status  |
+----------+---------+
| Bislerii | pending |
+----------+---------+
1 row in set (0.001 sec)
```

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

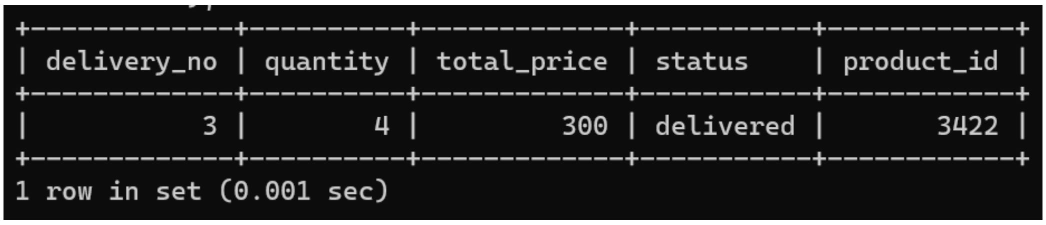
```
SELECT c.customer_id, c.name, c.address, c.contact_no, c.managed_by
FROM customer as c
    LEFT JOIN payment as p ON c.customer_id = p.customer_id
WHERE p.customer_id IS NULL;
```



```
+-------------+--------+---------+------------+------------+
| customer_id | name   | address | contact_no | managed_by |
+-------------+--------+---------+------------+------------+
|        1234 | jsfvns | sdndfis |    4234234 |       1357 |
|      532142 | yes4   | gutter  |    5435352 |       1234 |
+-------------+--------+---------+------------+------------+
2 rows in set (0.001 sec)
```

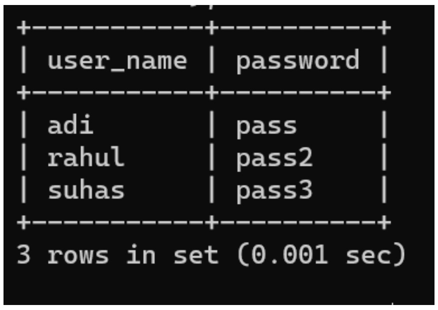
-- using correlated subquery find the delivery details of the product with the highest refill_price

```
select *
from delivery_details
where product_id = (
        select product_id
        from product
        where refill_price = (
                select max(refill_price)
                from product
            )
    );
```


```
+-------------+----------+-------------+-----------+------------+
| delivery_no | quantity | total_price | status    | product_id |
+-------------+----------+-------------+-----------+------------+
|           3 |        4 |         300 | delivered |       3422 |
+-------------+----------+-------------+-----------+------------+
1 row in set (0.001 sec)
```

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

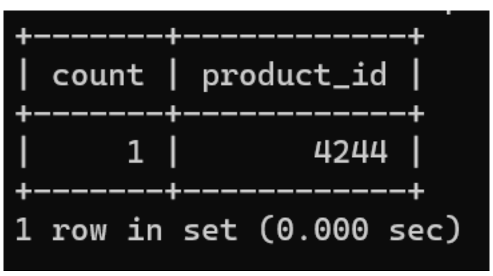
```
select user_name,
    password
from user
where user_id = (
        select user_id
        from pos
        where pos_id = (
                select max(pos_id)
                from pos
            )
    );
```


```
+-----------+----------+
| user_name | password |
+-----------+----------+
| adi       | pass     |
| rahul     | pass2    |
| suhas     | pass3    |
+-----------+----------+
3 rows in set (0.001 sec)
```

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';
```



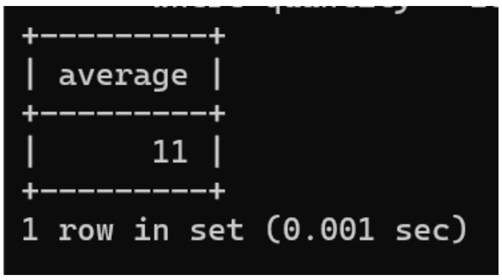
```
+-------+------------+
| count | product_id |
+-------+------------+
|     1 |       4244 |
+-------+------------+
1 row in set (0.000 sec)
```

--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');



```
+---------+
| average |
+---------+
|      11 |
+---------+
1 row in set (0.001 sec)
```

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

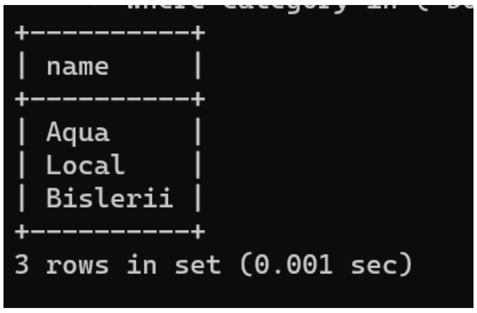
where quantity < 1000

union

select name

from product

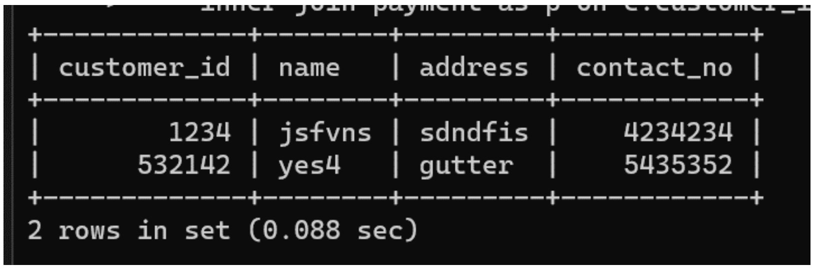
where category in ('bottle', 'tank', 'dirty water');



```
+----------+
| name     |
+----------+
| Aqua     |
| Local    |
| Bislerii |
+----------+
3 rows in set (0.001 sec)
```

--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;
```

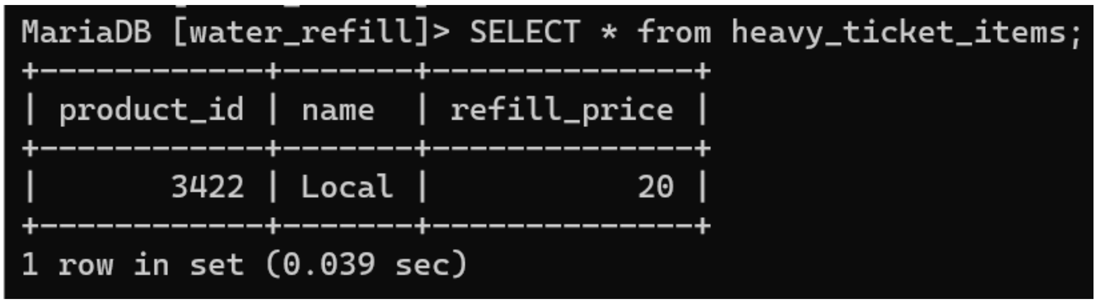


```
+-------------+--------+---------+------------+
| customer_id | name   | address | contact_no |
+-------------+--------+---------+------------+
|        1234 | jsfvns | sdndfis |    4234234 |
|      532142 | yes4   | gutter  |    5435352 |
+-------------+--------+---------+------------+
2 rows in set (0.088 sec)
```

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;



```
MariaDB [water_refill]> SELECT * from heavy_ticket_items;
+------------+-------+--------------+
| product_id | name  | refill_price |
+------------+-------+--------------+
|       3422 | Local |           20 |
+------------+-------+--------------+
1 row in set (0.039 sec)
```

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

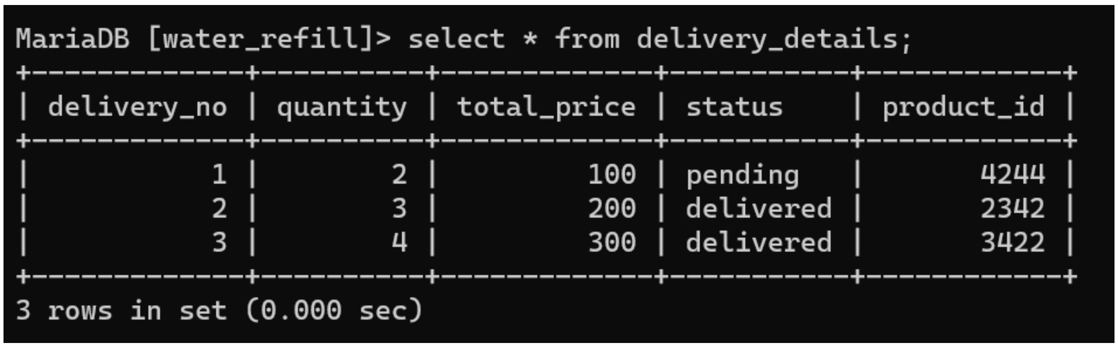
CALL decrement_quantity(NEW.product_id, NEW.quantity);

END IF;

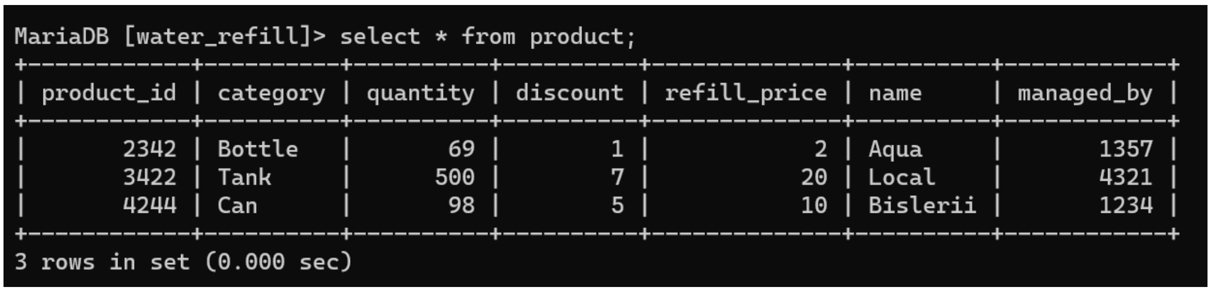
END $$

DELIMITER ;

Before update

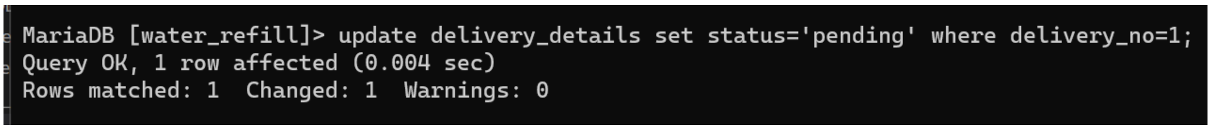


```
MariaDB [water_refill]> select * from delivery_details;
+-------------+----------+-------------+-----------+------------+
| delivery_no | quantity | total_price | status    | product_id |
+-------------+----------+-------------+-----------+------------+
|           1 |        2 |         100 | pending   |       4244 |
|           2 |        3 |         200 | delivered |       2342 |
|           3 |        4 |         300 | delivered |       3422 |
+-------------+----------+-------------+-----------+------------+
3 rows in set (0.000 sec)
```

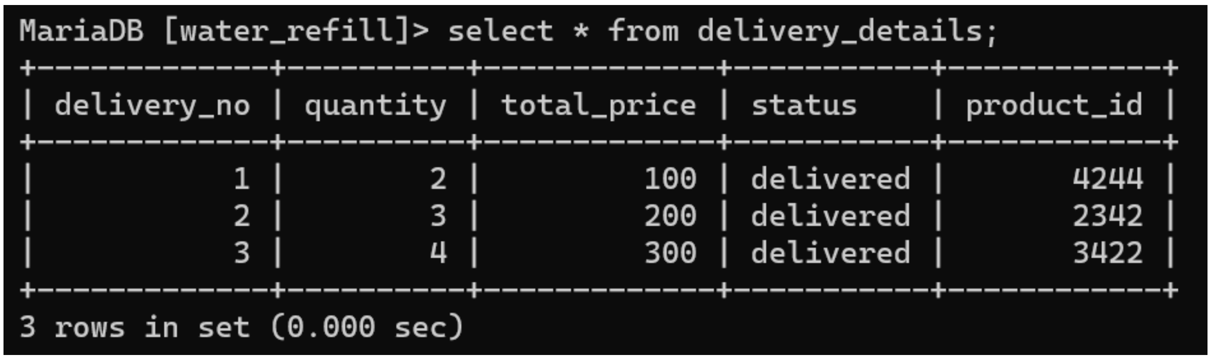


```
MariaDB [water_refill]> select * from product;
+------------+----------+----------+----------+--------------+----------+------------+
| product_id | category | quantity | discount | refill_price | name     | managed_by |
+------------+----------+----------+----------+--------------+----------+------------+
|       2342 | Bottle   |       69 |        1 |            2 | Aqua     |       1357 |
|       3422 | Tank     |      500 |        7 |           20 | Local    |       4321 |
|       4244 | Can      |       98 |        5 |           10 | Bislerii |       1234 |
+------------+----------+----------+----------+--------------+----------+------------+
3 rows in set (0.000 sec)
```

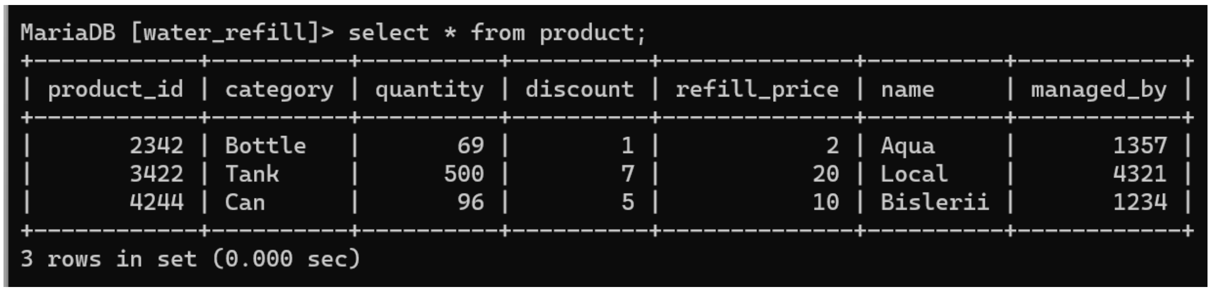
After update



```
MariaDB [water_refill]> update delivery_details set status='pending' where delivery_no=1;
Query OK, 1 row affected (0.004 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

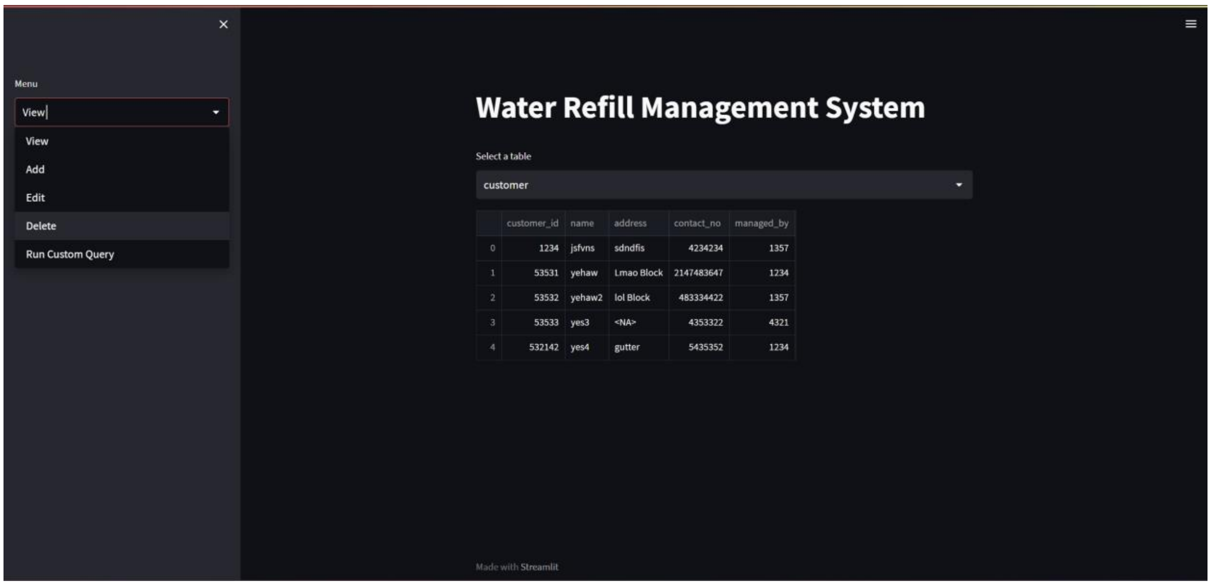


```
MariaDB [water_refill]> select * from delivery_details;
+-------------+----------+-------------+-----------+------------+
| delivery_no | quantity | total_price | status    | product_id |
+-------------+----------+-------------+-----------+------------+
|           1 |        2 |         100 | delivered |       4244 |
|           2 |        3 |         200 | delivered |       2342 |
|           3 |        4 |         300 | delivered |       3422 |
+-------------+----------+-------------+-----------+------------+
3 rows in set (0.000 sec)
```



```
MariaDB [water_refill]> select * from product;
+------------+----------+----------+----------+--------------+----------+------------+
| product_id | category | quantity | discount | refill_price | name     | managed_by |
+------------+----------+----------+----------+--------------+----------+------------+
|       2342 | Bottle   |       69 |        1 |            2 | Aqua     |       1357 |
|       3422 | Tank     |      500 |        7 |           20 | Local    |       4321 |
|       4244 | Can      |       96 |        5 |           10 | Bislerii |       1234 |
+------------+----------+----------+----------+--------------+----------+------------+
3 rows in set (0.000 sec)
```

Developing a Frontend

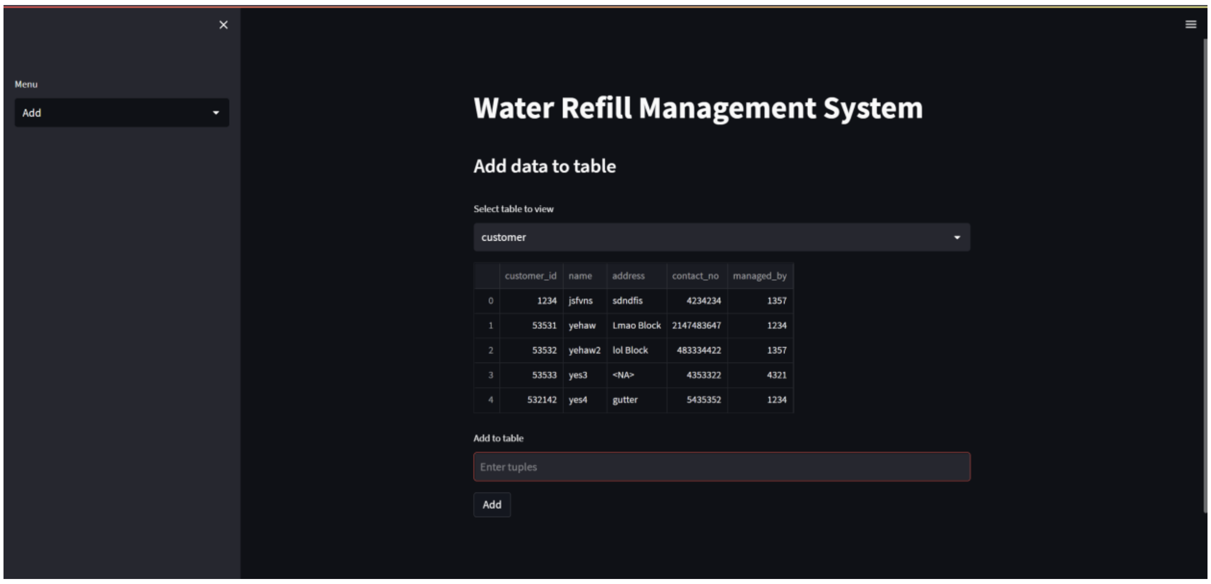


Water Refill Management System

Select a table

customer

	customer_id	name	address	contact_no	managed_by
0	1234	jsfvns	sdndfis	4234234	1357
1	53531	yehaw	Lmao Block	2147483647	1234
2	53532	yehaw2	lol Block	483334422	1357
3	53533	yes3	<NA>	4353322	4321
4	532142	yes4	gutter	5435352	1234



Water Refill Management System

Add data to table

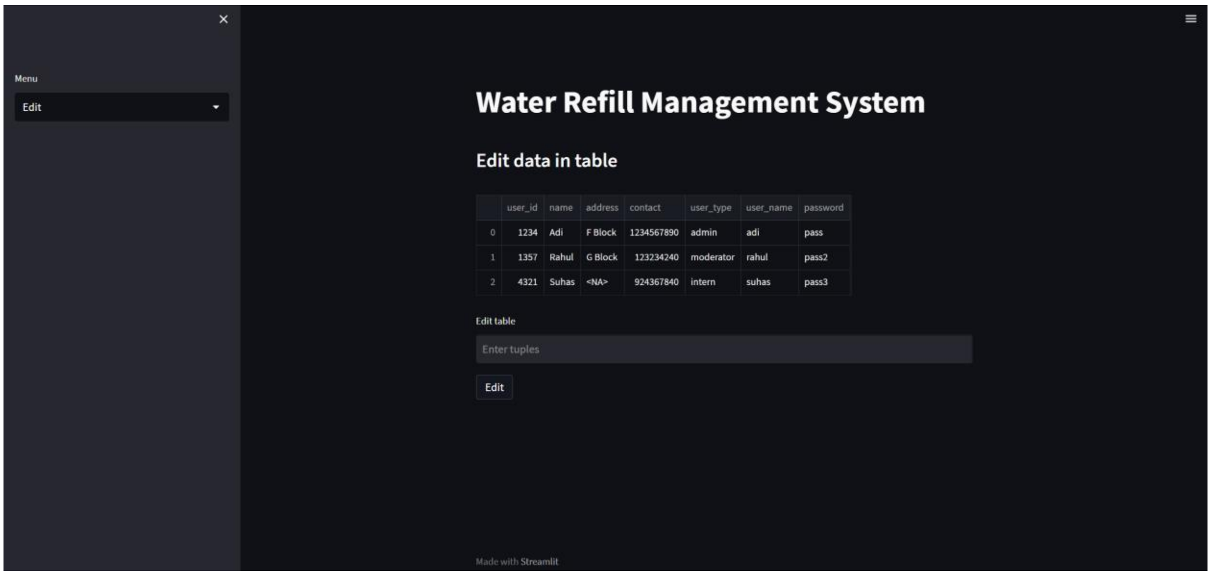
Select table to view

customer

	customer_id	name	address	contact_no	managed_by
0	1234	jsfvns	sdndfis	4234234	1357
1	53531	yehaw	Lmao Block	2147483647	1234
2	53532	yehaw2	lol Block	483334422	1357
3	53533	yes3	<NA>	4353322	4321
4	532142	yes4	gutter	5435352	1234

Add to table

Enter tuples



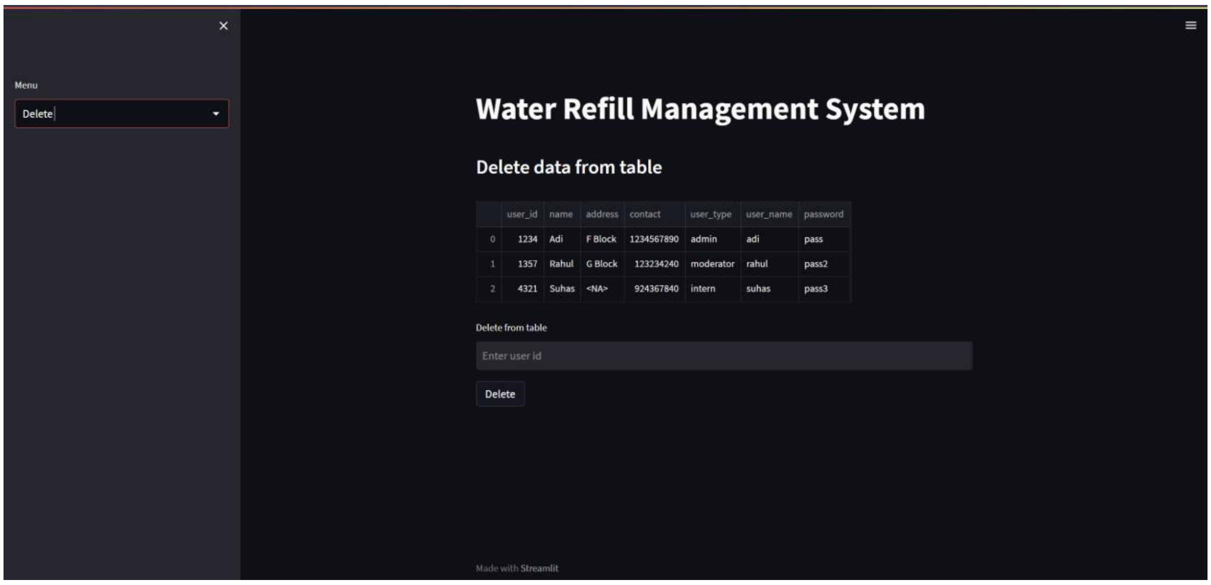
Water Refill Management System

Edit data in table

	user_id	name	address	contact	user_type	user_name	password
0	1234	Adi	F Block	1234567890	admin	adi	pass
1	1357	Rahul	G Block	123234240	moderator	rahul	pass2
2	4321	Suhas	<NA>	924367840	intern	suhas	pass3

Edit table

Enter tuples



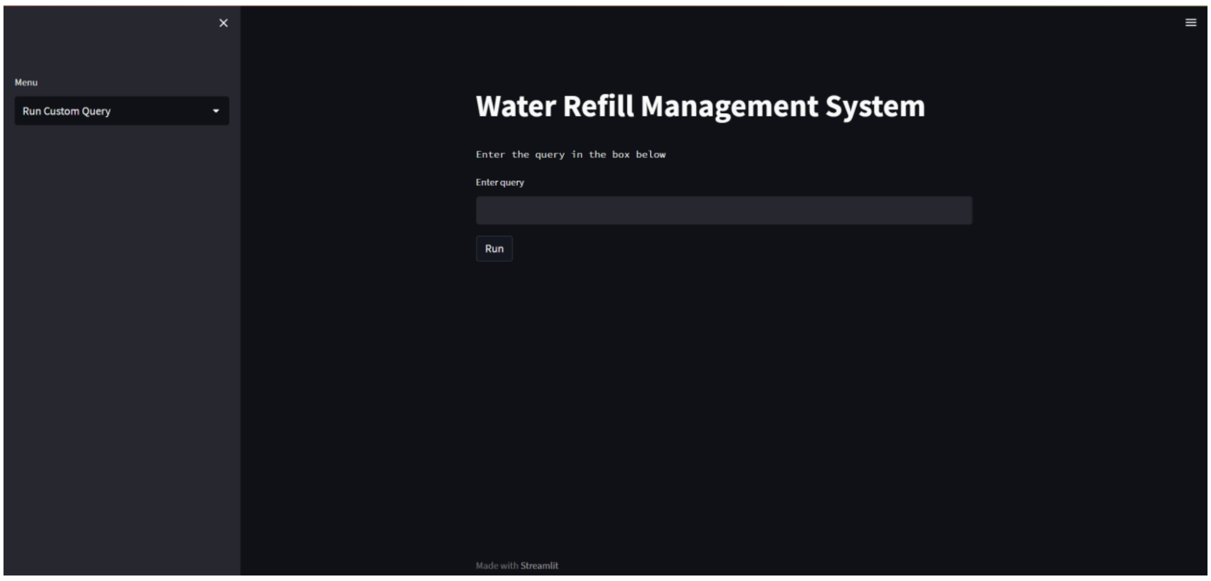
Water Refill Management System

Delete data from table

	user_id	name	address	contact	user_type	user_name	password
0	1234	Adi	F Block	1234567890	admin	adi	pass
1	1357	Rahul	G Block	123234240	moderator	rahul	pass2
2	4321	Suhas	<NA>	924367840	intern	suhas	pass3

Delete from table

Enter user id



Water Refill Management System

```
Enter the query in the box below
```

Enter query