DBMS

MINI PROJECT

**RATION CARD**

**MANAGEMENT SYSTEM**

**NAME :** SANMAT SANJAYAKUMAR PAYAGOUDAR

**SRN :** PES1UG20CS385

**SECTION :** G

**Description and Scope**

Today, if a person goes to ration shop for buying anything he has to wait in queue till his chance comes, and ration shop person starts investigating about his details manually, which is very time consuming.

In present system there are many draw backs. For example

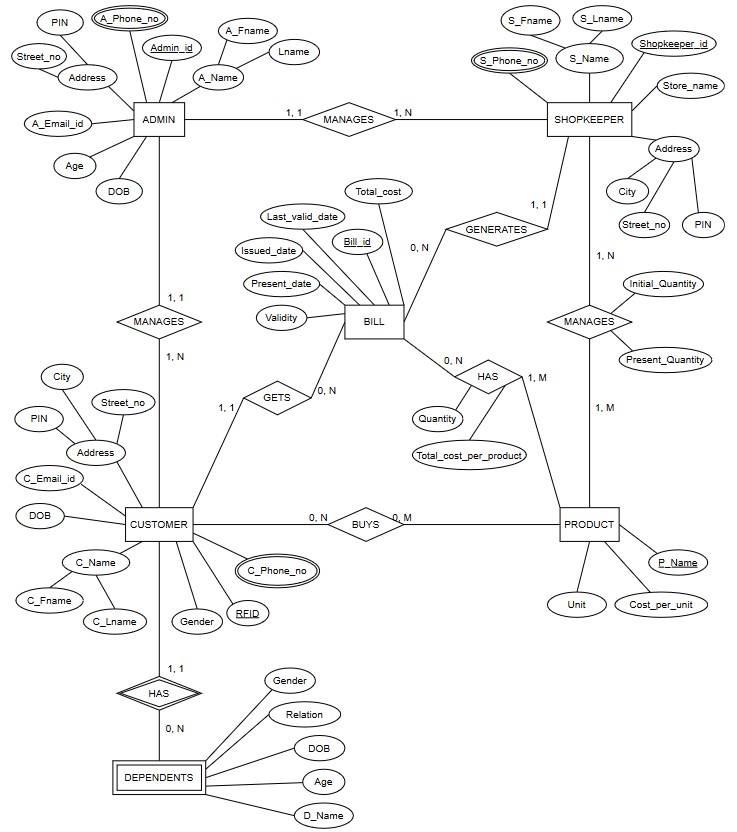
* Distributor undergoes illegal activities and fake transactions because of this reason end users are under loss and won’t get their proper sock
* Government doesn’t know how many users are there in particular city or district.
* All process done by manually using paper and this data can be lost in some accident like fire etc.
* Fast report generation is not possible.

**ABOUT THIS PROJECT :**

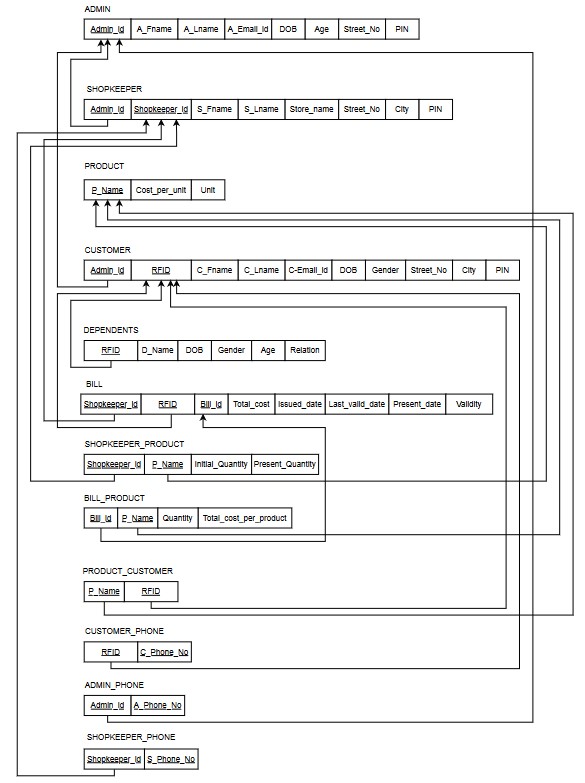
This “RATION CARS MANAGEMENT SYSTEM” is an efficient system to tackle the above problems. It has an interface for maintaining the database online. It provides services like rates of materials which is fixed by the state government and also provides the details of shopkeeper, customer and customer’s past history about buying products and also the dependents of the customer.

It maintains all the details of distributors and his transaction, which will prevent the distributor from doing illegal activities. And government can easily access the whole data about customer and shopkeeper and any other data. Risk of losing data is less because of online data maintenance. And generation of report regarding any data is easy here.

**ER – DIAGRAM**



**RELATIONAL SCHEMA**



**DDL Statements – Building the database**

CREATE TABLE ADMIN(

    Admin\_id VARCHAR(10),

    A\_Fname VARCHAR(20),

    A\_Lname VARCHAR(20),

    A\_Email\_id VARCHAR(50),

    DOB DATE,

    Age INT(3),

    Street\_no VARCHAR(40),

    PIN INT(6),

    PRIMARY KEY (Admin\_id)

);

CREATE TABLE SHOPKEEPER(

    Shopkeeper\_id VARCHAR(10),

    S\_Fname VARCHAR(20),

    S\_Lname VARCHAR(20),

    Store\_name VARCHAR(20),

    Street\_no VARCHAR(40),

    City VARCHAR(20),

    PIN INT(6),

    Admin\_id VARCHAR(10),

    PRIMARY KEY (Shopkeeper\_id),

    FOREIGN KEY (Admin\_id) REFERENCES ADMIN (Admin\_id)

);

CREATE TABLE PRODUCT(

    P\_Name VARCHAR(20),

    Cost\_per\_unit float(5, 2),

    Unit VARCHAR(20),

    PRIMARY KEY (P\_Name)

);

CREATE TABLE CUSTOMER(

    RFID BIGINT(20),

    C\_Fname VARCHAR(20),

    C\_Lname VARCHAR(20),

    C\_Email\_id VARCHAR(50),

    DOB DATE,

    Gender VARCHAR(10),

    Street\_no VARCHAR(40),

    City VARCHAR(20),

    PIN INT(6),

    Admin\_id VARCHAR(10),

    PRIMARY KEY (RFID),

    FOREIGN KEY (Admin\_id) REFERENCES ADMIN (Admin\_id)

);

CREATE TABLE DEPENDENT(

    RFID BIGINT(10),

    D\_Name VARCHAR(20),

    DOB DATE,

    Gender VARCHAR(10),

    Age INT(3),

    Relation VARCHAR(20),

    FOREIGN KEY (RFID) REFERENCES CUSTOMER (RFID)

    ON DELETE CASCADE

);

CREATE TABLE BILL(

    Bill\_id VARCHAR(10),

    Total\_cost float(5, 2),

    Issued\_date DATE,

    Last\_valid\_date DATE,

    Present\_date DATE,

    Validity VARCHAR(20),

    Shopkeeper\_id VARCHAR(10),

    RFID BIGINT(10),

    PRIMARY KEY (Bill\_id),

    FOREIGN KEY (Shopkeeper\_id) REFERENCES SHOPKEEPER (Shopkeeper\_id),

    FOREIGN KEY (RFID) REFERENCES CUSTOMER (RFID)

    ON DELETE CASCADE

);

CREATE TABLE SHOPKEEPER\_PRODUCT(

    Shopkeeper\_id VARCHAR(10),

    P\_Name VARCHAR(20),

    Initial\_Quantity float(5, 2),

    Present\_Quantity float(5, 2),

    PRIMARY KEY (Shopkeeper\_id, P\_Name),

    FOREIGN KEY (Shopkeeper\_id) REFERENCES SHOPKEEPER (Shopkeeper\_id),

    FOREIGN KEY (P\_Name) REFERENCES PRODUCT (P\_Name)

    ON DELETE CASCADE

);

CREATE TABLE BILL\_PRODUCT(

    Bill\_id VARCHAR(10),

    P\_Name VARCHAR(20),

    Quantity float(5, 2),

    Total\_cost\_per\_product float(5, 2),

    PRIMARY KEY (Bill\_id, P\_Name),

    FOREIGN KEY (Bill\_id) REFERENCES BILL (Bill\_id),

    FOREIGN KEY (P\_Name) REFERENCES PRODUCT (P\_Name)

    ON DELETE CASCADE);

CREATE TABLE PRODUCT\_CUSTOMER(

    P\_Name VARCHAR(10),

    RFID BIGINT(20),

    PRIMARY KEY (P\_Name, RFID),

    FOREIGN KEY (P\_Name) REFERENCES PRODUCT (P\_Name),

    FOREIGN KEY (RFID) REFERENCES CUSTOMER (RFID)

    ON DELETE CASCADE

);

CREATE TABLE CUSTOMER\_PHONE(

    RFID BIGINT(20),

    Phone\_no BIGINT(10),

    PRIMARY KEY (RFID, Phone\_no),

    FOREIGN KEY (RFID) REFERENCES CUSTOMER (RFID)

    ON DELETE CASCADE

);

CREATE TABLE ADMIN\_PHONE(

    Admin\_id VARCHAR(10),

    Phone\_no BIGINT(10),

    PRIMARY KEY (Admin\_id, Phone\_no),

    FOREIGN KEY (Admin\_id) REFERENCES ADMIN (Admin\_id)

    ON DELETE CASCADE

);

CREATE TABLE SHOPKEEPER\_PHONE(

    Shopkeeper\_id VARCHAR(10),

    Phone\_no BIGINT(10),

    PRIMARY KEY (Shopkeeper\_id, Phone\_no),

    FOREIGN KEY (Shopkeeper\_id) REFERENCES SHOPKEEPER (Shopkeeper\_id)

    ON DELETE CASCADE

);



**Populating the Database**

**Using INSERT command :**

**Query :**

INSERT INTO SHOPKEEPER(Shopkeeper\_id, S\_Fname, S\_Lname, Store\_name, Street\_No, City, PIN, Admin\_id) VALUES

('SHK\_001', 'Manisha',  'Solanki',  'Lakshmi FPS',  'M G Road', 'Bengaluru',    575030, 'ADM\_001'),

('SHK\_002', 'Bharti',   'Devgan',   'Mahaveer FPS', 'NAL Wind Tunnel Road', 'Mangaluru',    560059, 'ADM\_003'),

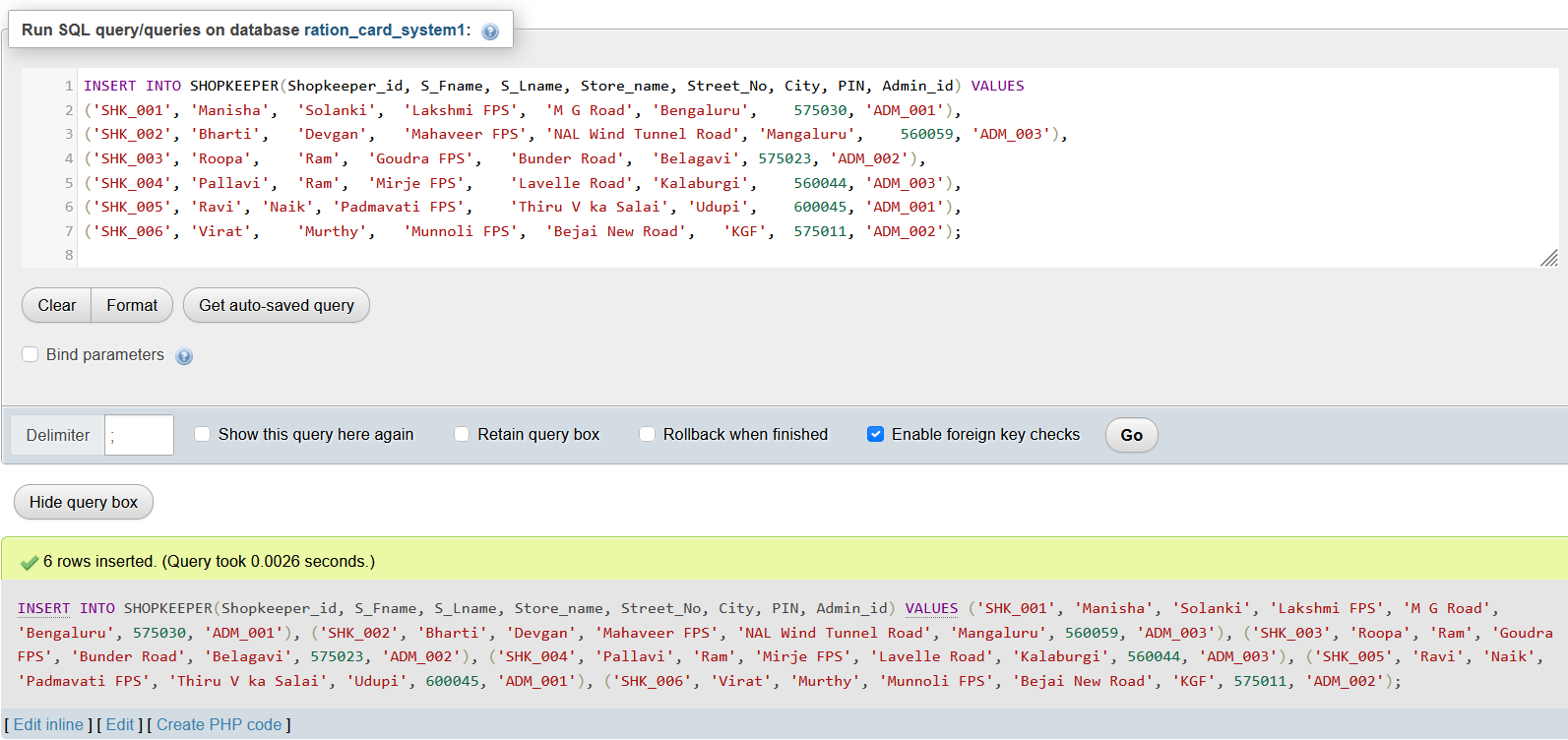
('SHK\_003', 'Roopa',    'Ram',  'Goudra FPS',   'Bunder Road',  'Belagavi', 575023, 'ADM\_002'),

('SHK\_004', 'Pallavi',  'Ram',  'Mirje FPS',    'Lavelle Road', 'Kalaburgi',    560044, 'ADM\_003'),

('SHK\_005', 'Ravi', 'Naik', 'Padmavati FPS',    'Thiru V ka Salai', 'Udupi',    600045, 'ADM\_001'),

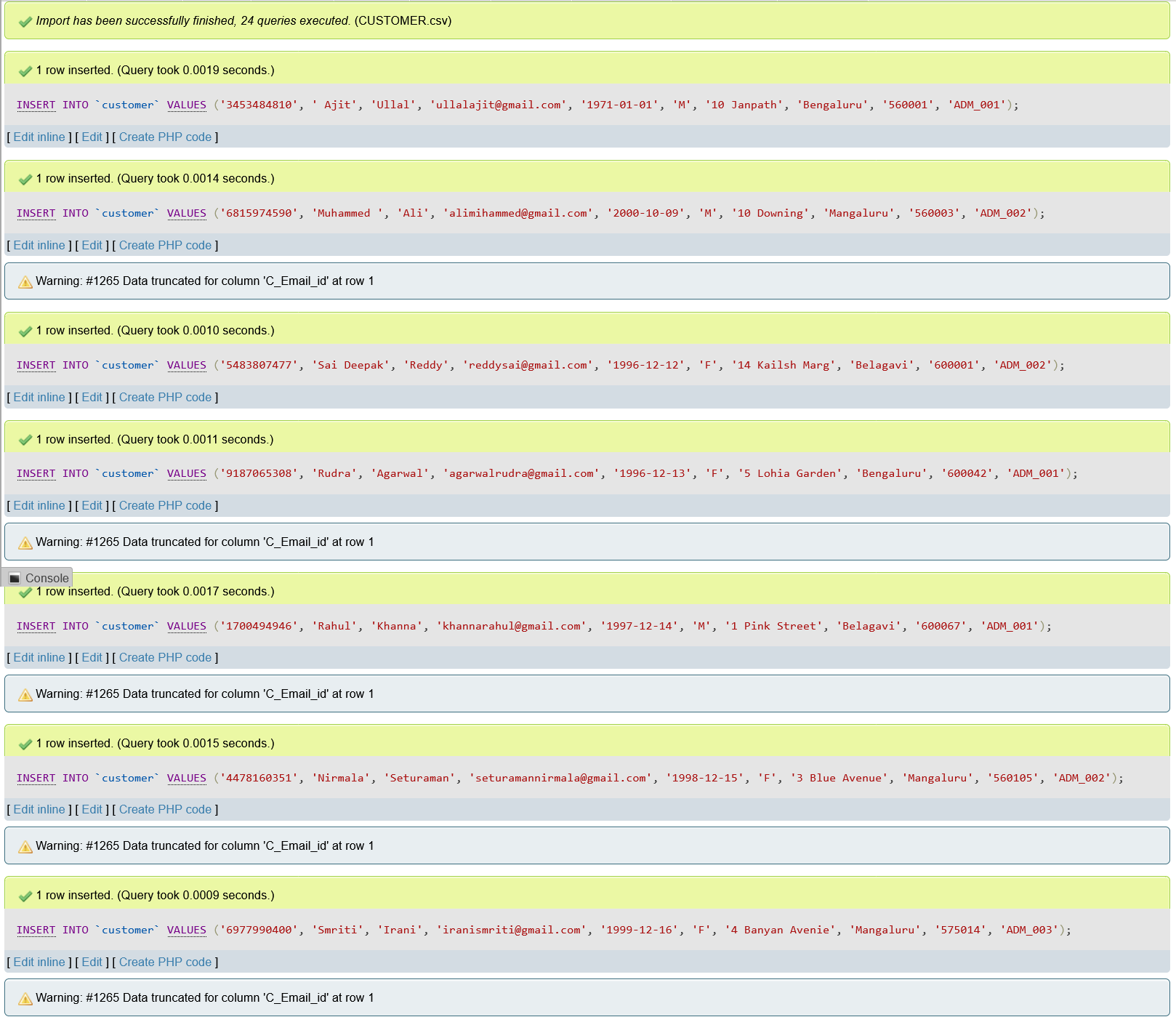
('SHK\_006', 'Virat',    'Murthy',   'Munnoli FPS',  'Bejai New Road',   'KGF',  575011, 'ADM\_002');

**OUTPUT :**



**Using IMPORT :**

**OUTPUT :**



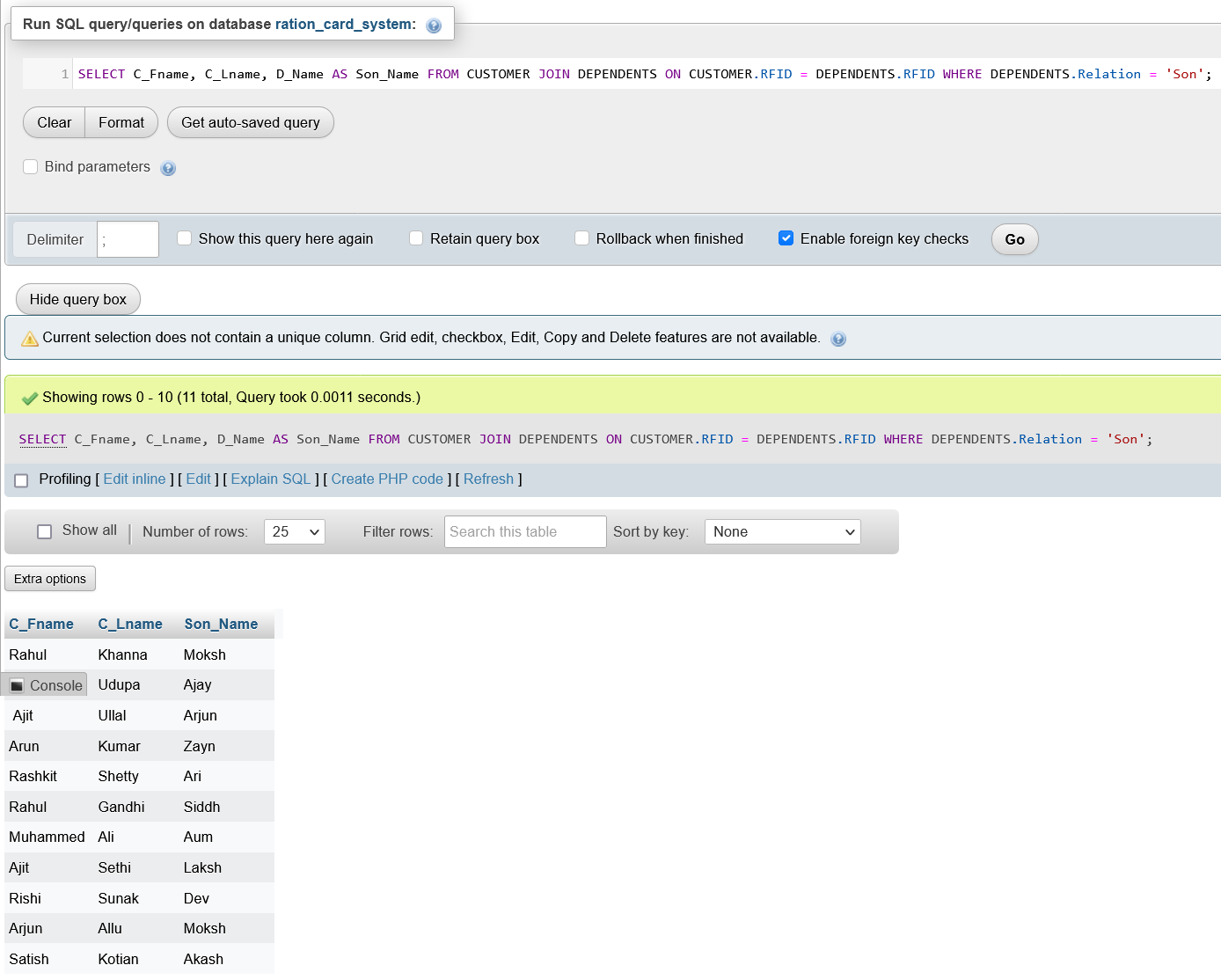
**JOIN Queries**

1. Retrieve Customer name and son name iff customer has a son.

Query :

SELECT C\_Fname, C\_Lname, D\_Name AS Son\_Name FROM CUSTOMER JOIN DEPENDENTS ON CUSTOMER.RFID = DEPENDENTS.RFID WHERE DEPENDENTS.Relation = 'Son';

Output :

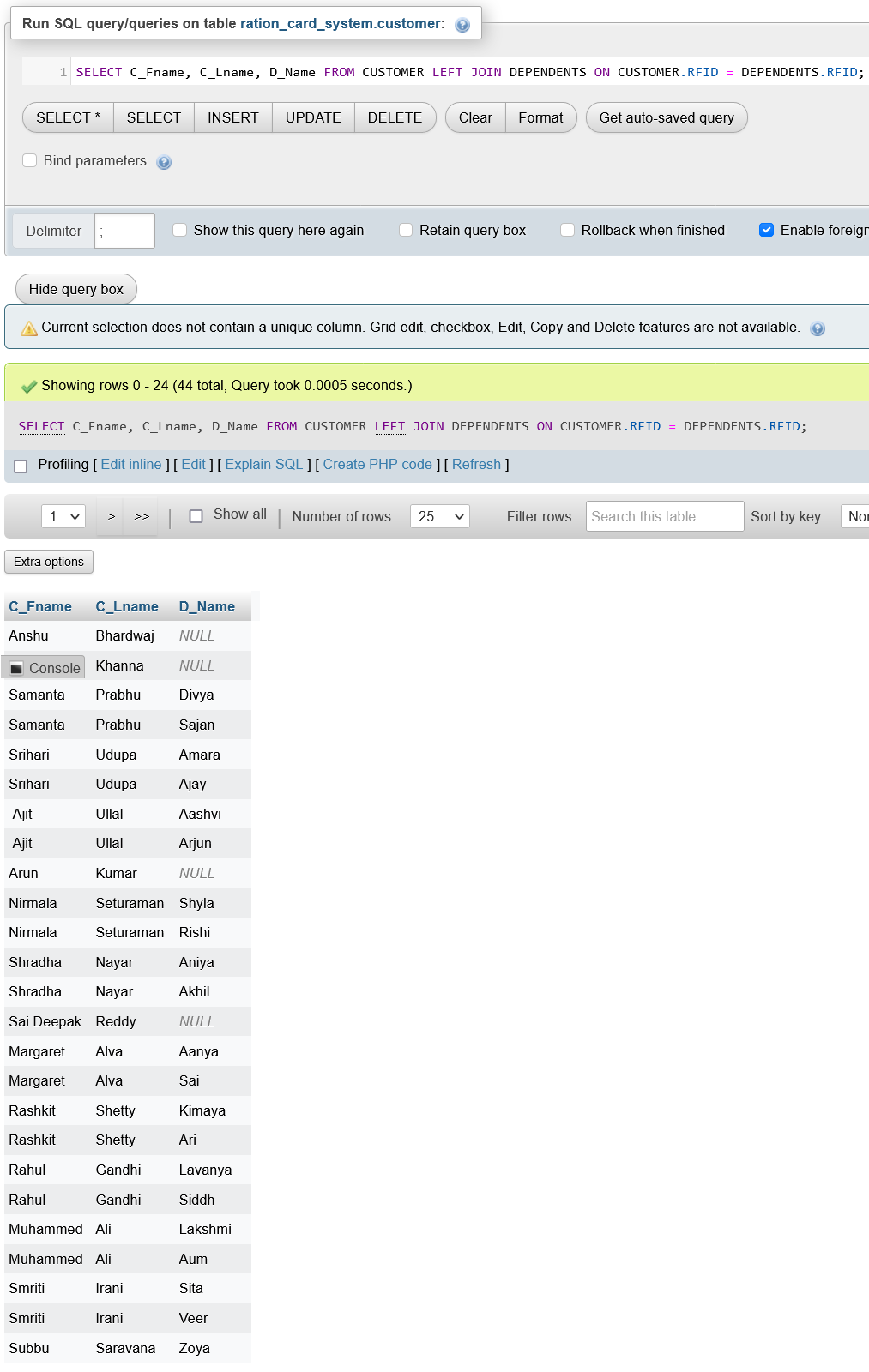


1. Retrieve Customer name and dependent name if customer has a dependent.

Query :

SELECT C\_Fname, C\_Lname, D\_Name FROM CUSTOMER LEFT JOIN DEPENDENTS ON CUSTOMER.RFID = DEPENDENTS.RFID;

Output :

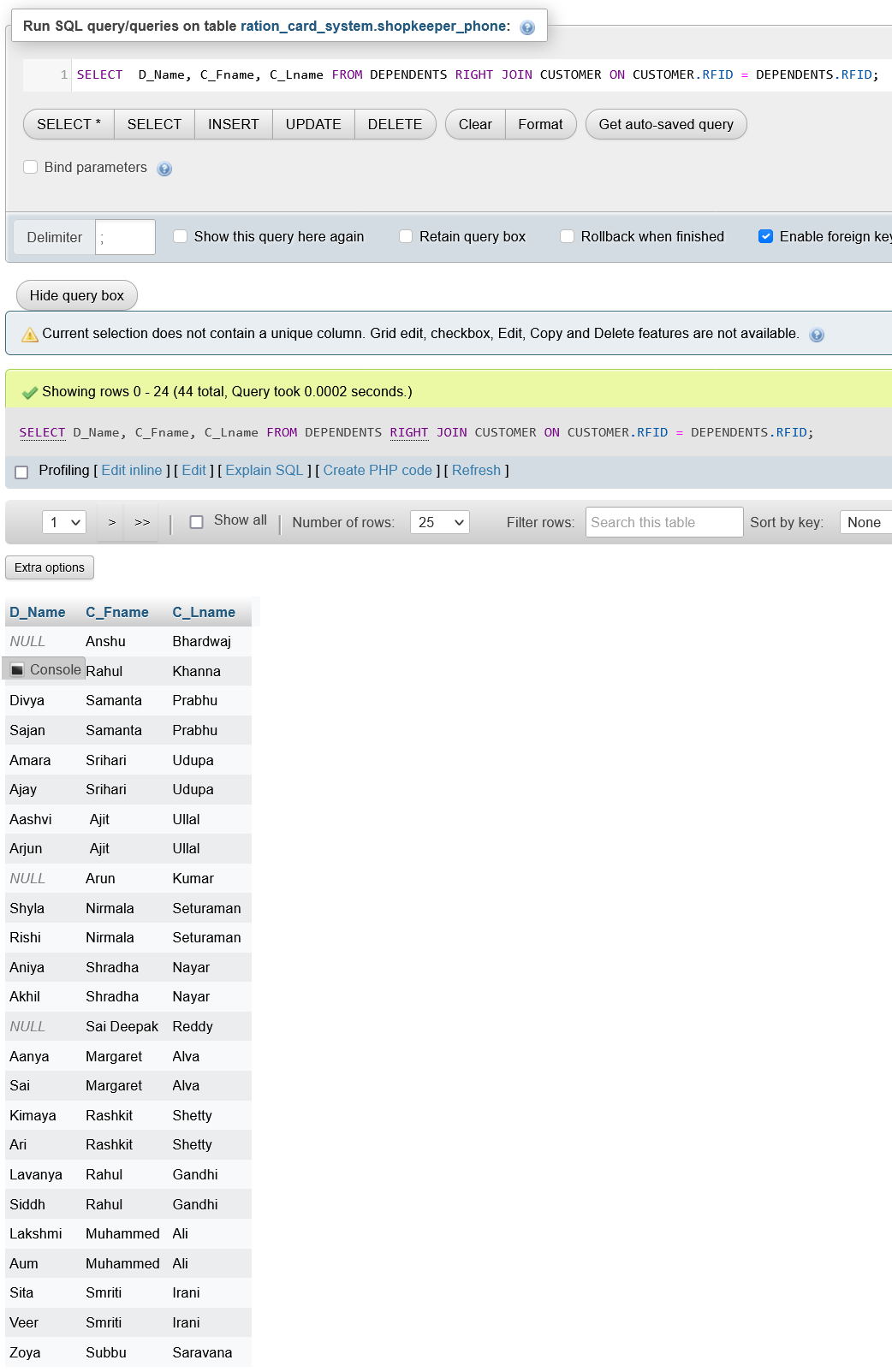


1. Retrieve Dependent and Customer name if customer has a dependent.

Query :

SELECT  D\_Name, C\_Fname, C\_Lname FROM DEPENDENTS RIGHT JOIN CUSTOMER ON CUSTOMER.RFID = DEPENDENTS.RFID;

Output :



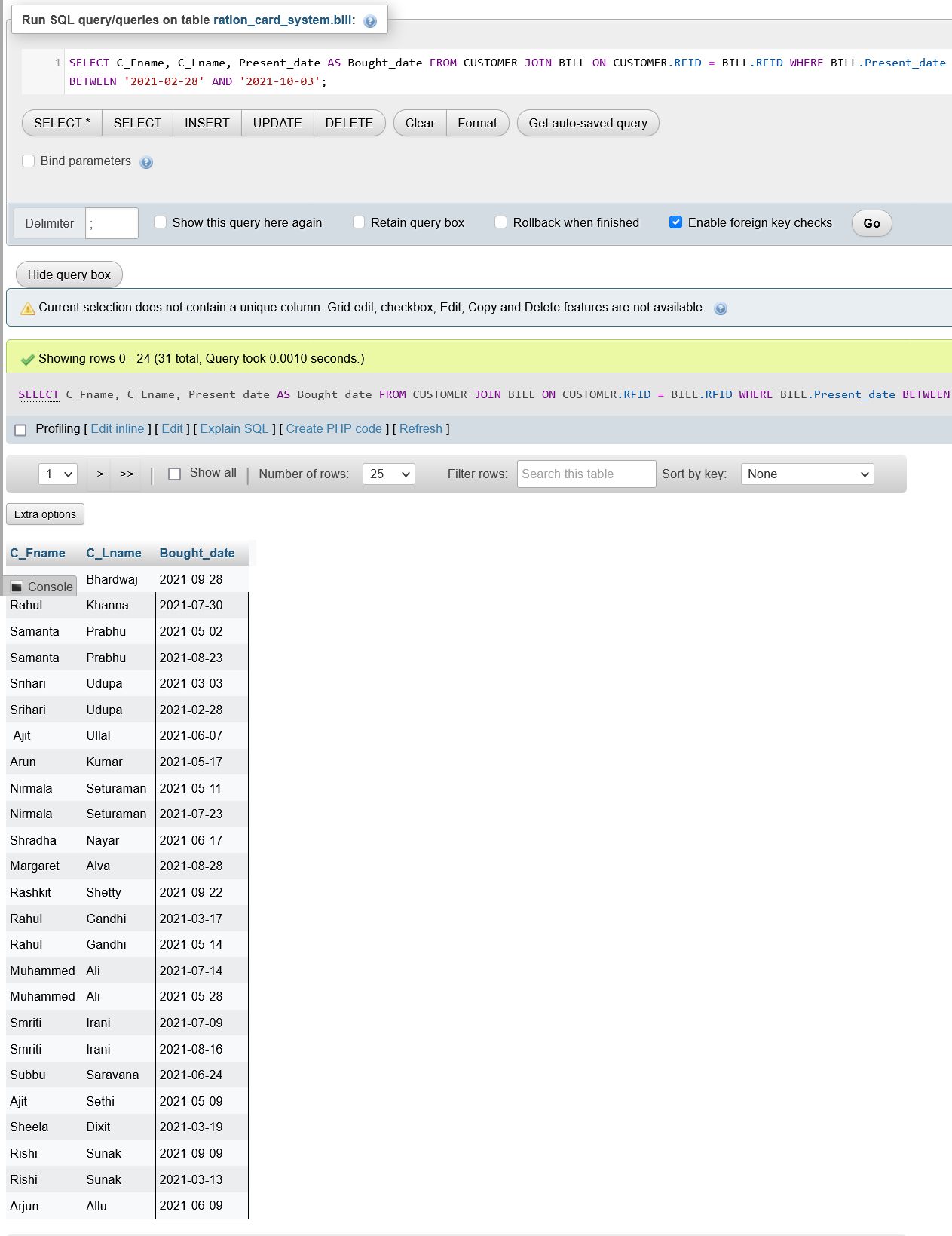
1. Retrieve customer name and product bought date for those customers who bought the products between

28th February 2021 and 3rd October 2021.

Query :

SELECT C\_Fname, C\_Lname, Present\_date AS Bought\_date FROM CUSTOMER JOIN BILL ON CUSTOMER.RFID = BILL.RFID WHERE BILL.Present\_date BETWEEN '2021-02-28' AND '2021-10-03';

Output :



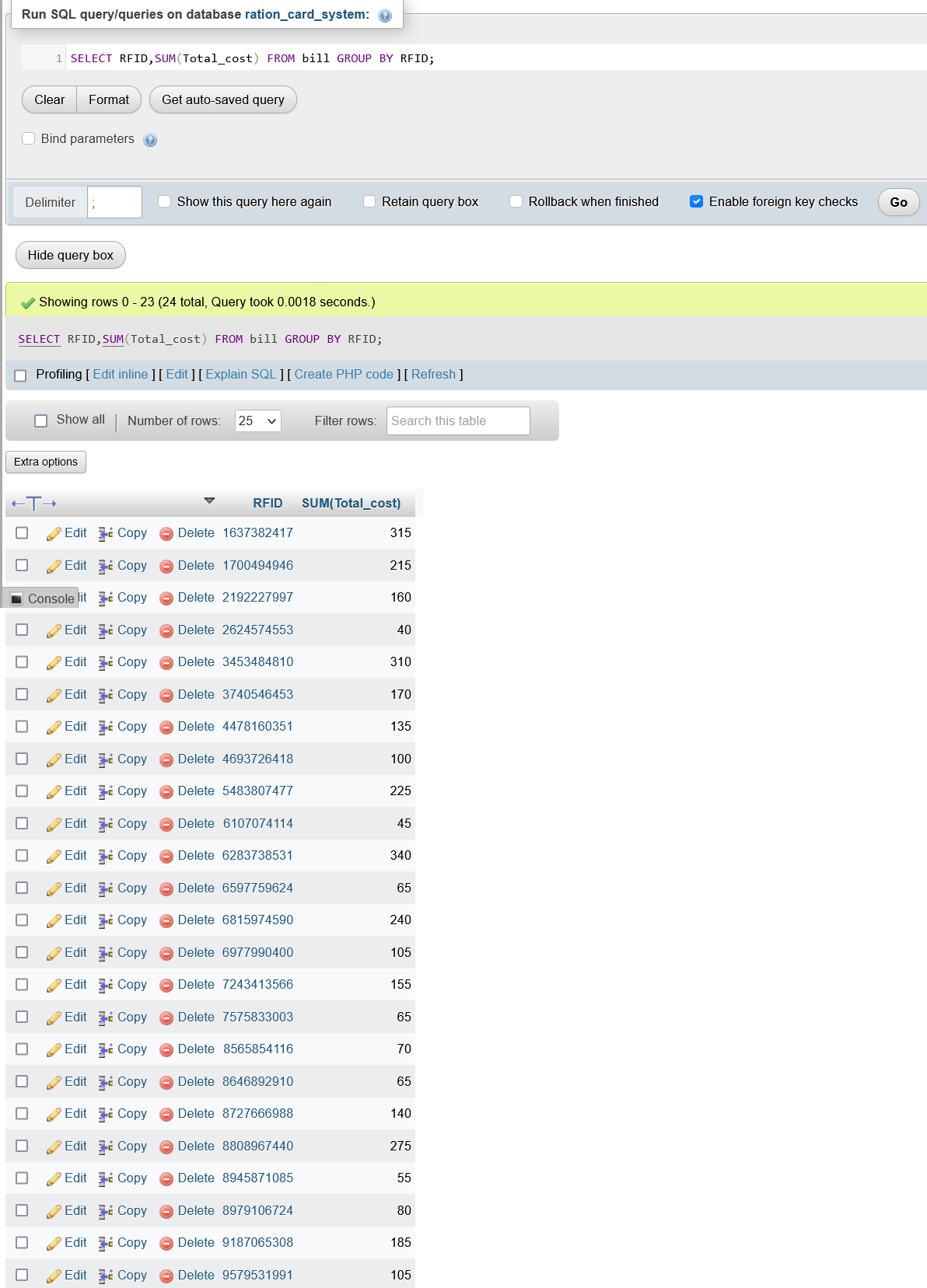
**Aggregate Functions**

1. Retrieve the total amount spent by each customer.

Query:

SELECT SUM(Total\_cost) FROM bill GROUP BY RFID;

Output:

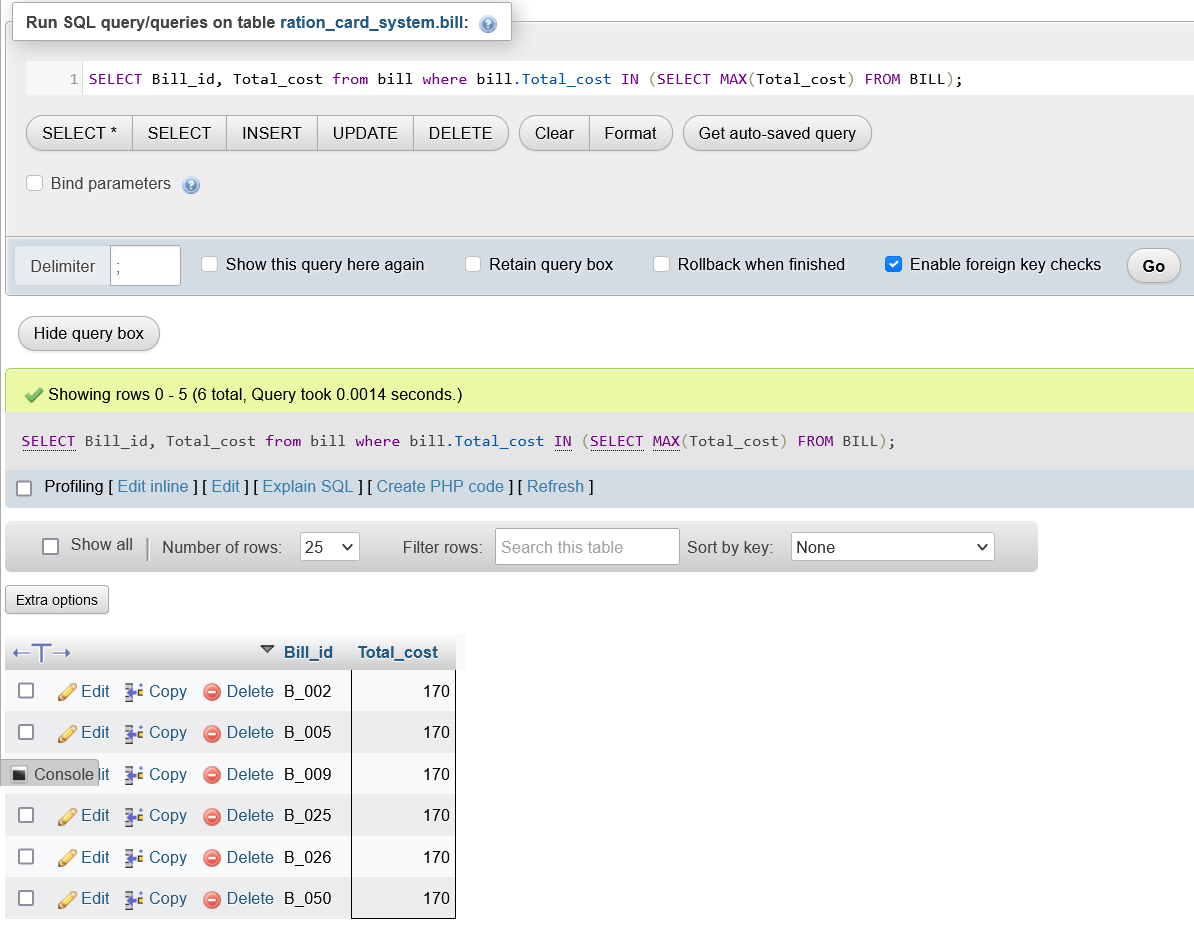


1. Retrieve the bill id and total cost for bill which have the maximum value for total cost.

Query :

SELECT Bill\_id, Total\_cost from bill where bill.Total\_cost IN (SELECT MAX(Total\_cost) FROM BILL);

Output:

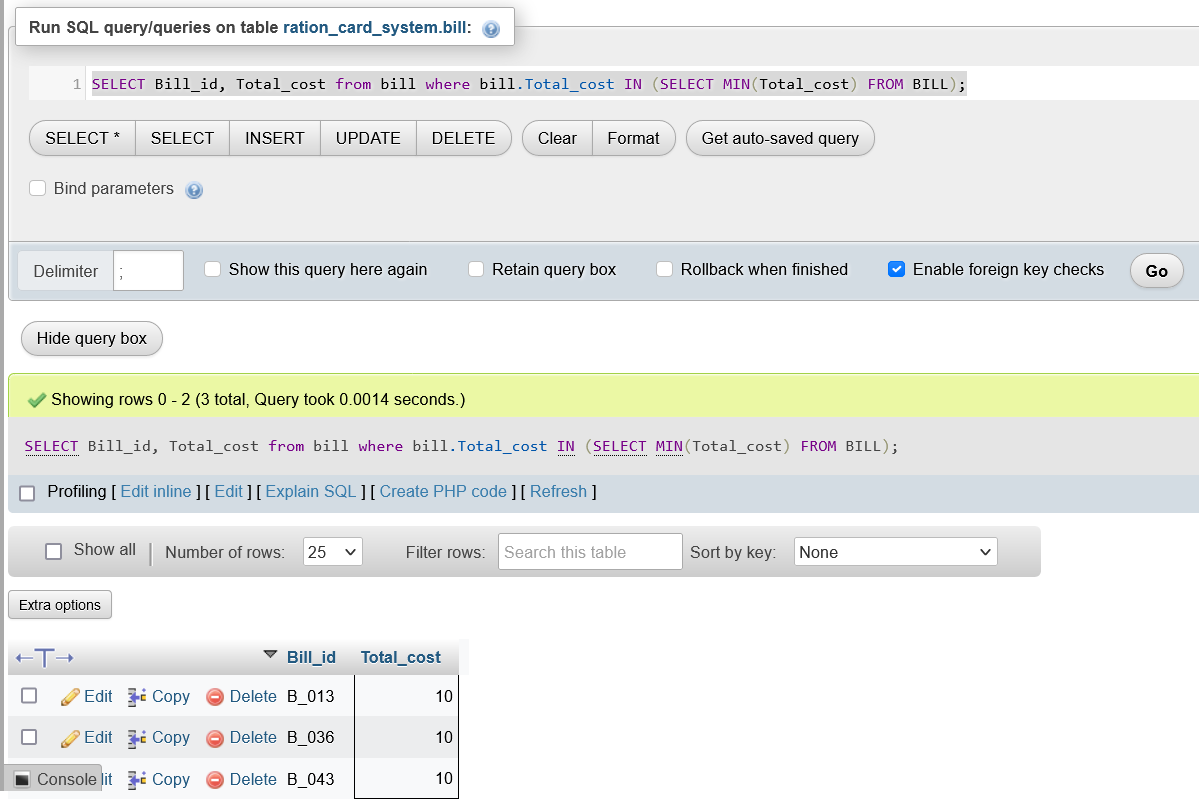


1. Retrieve the bill id and total cost for bill which have the minimum value for total cost.

Query :

SELECT Bill\_id, Total\_cost from bill where bill.Total\_cost IN (SELECT MIN(Total\_cost) FROM BILL);

Output:

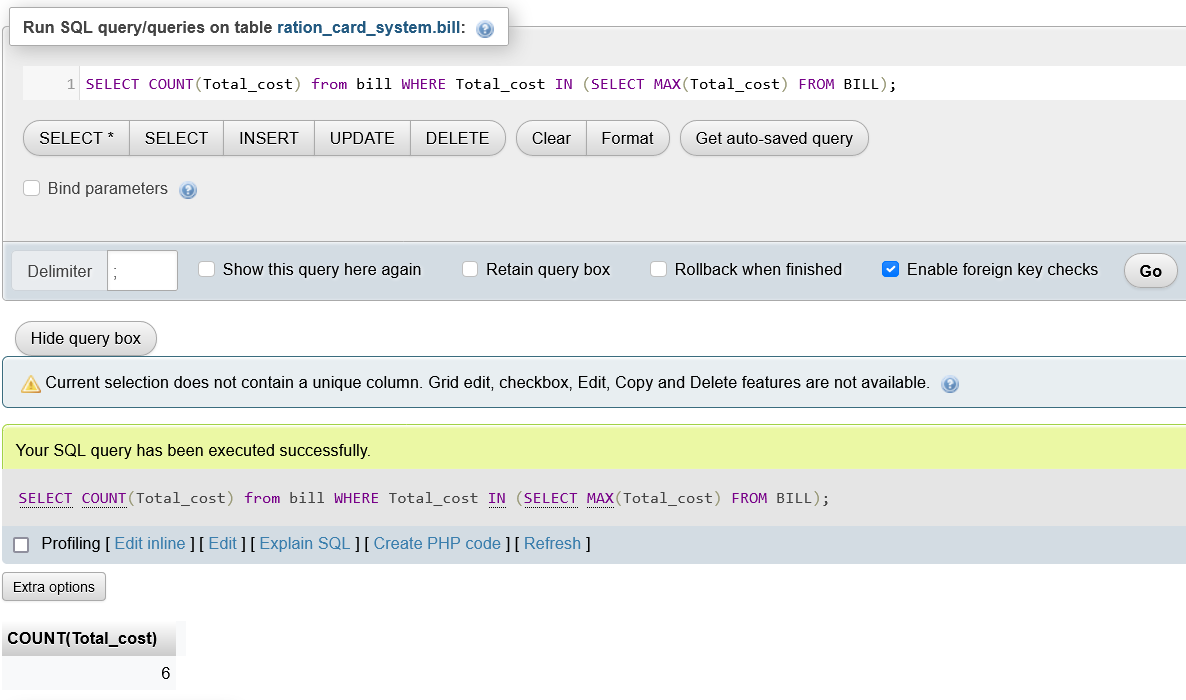


1. Retrieve the number of bills which have the total cost equal to maximum value for total cost.

Query :

SELECT COUNT(Total\_cost) from bill WHERE Total\_cost IN (SELECT MAX(Total\_cost) FROM BILL);

Output :

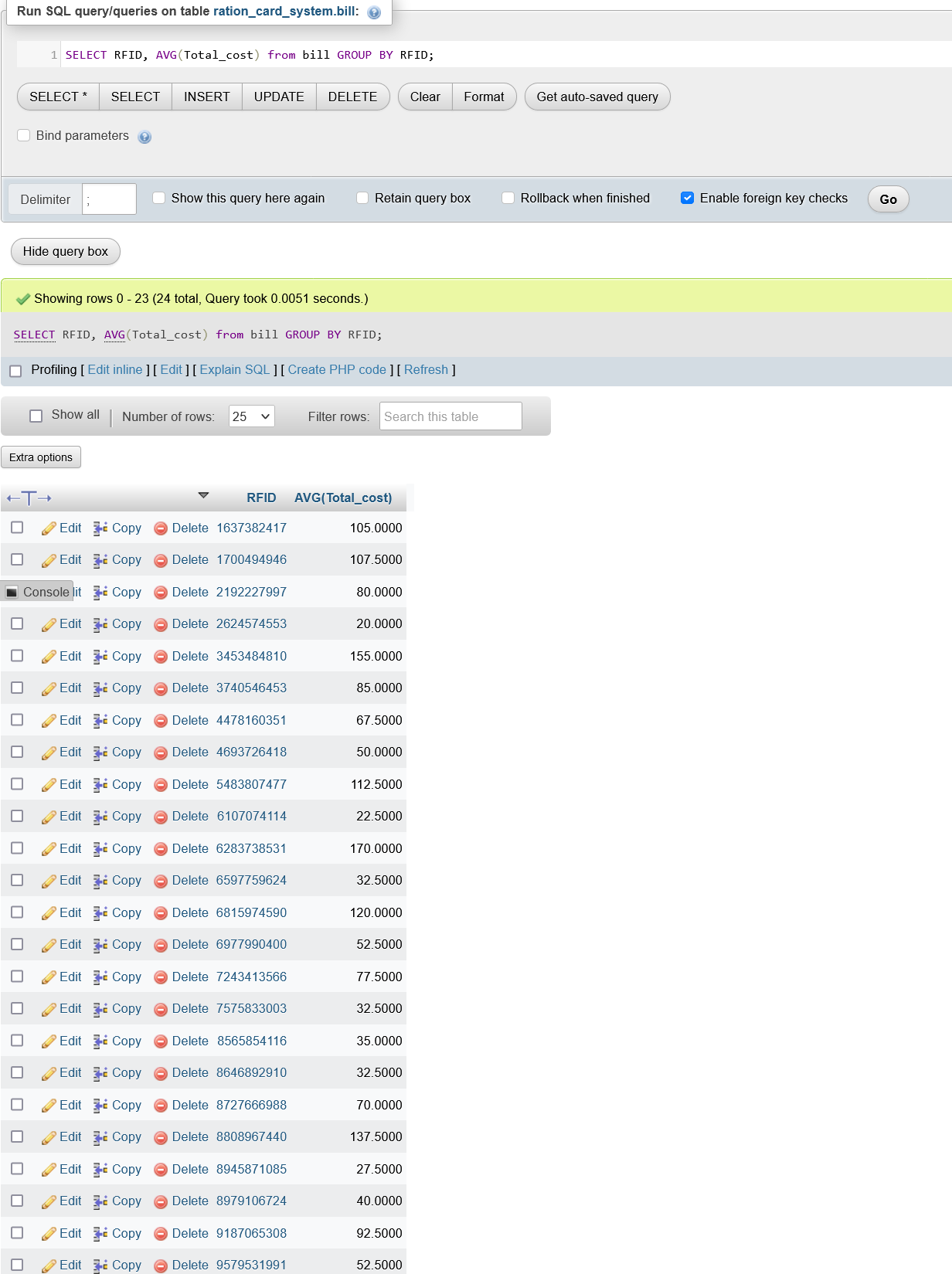


1. Retrieve the RFID of the customer and average cost spent by them.

Query :

SELECT RFID, AVG(Total\_cost) from bill GROUP BY RFID;

Output:



**SET Operations**

1. Retrieve the phone numbers that are either of admin or shopkeeper table.

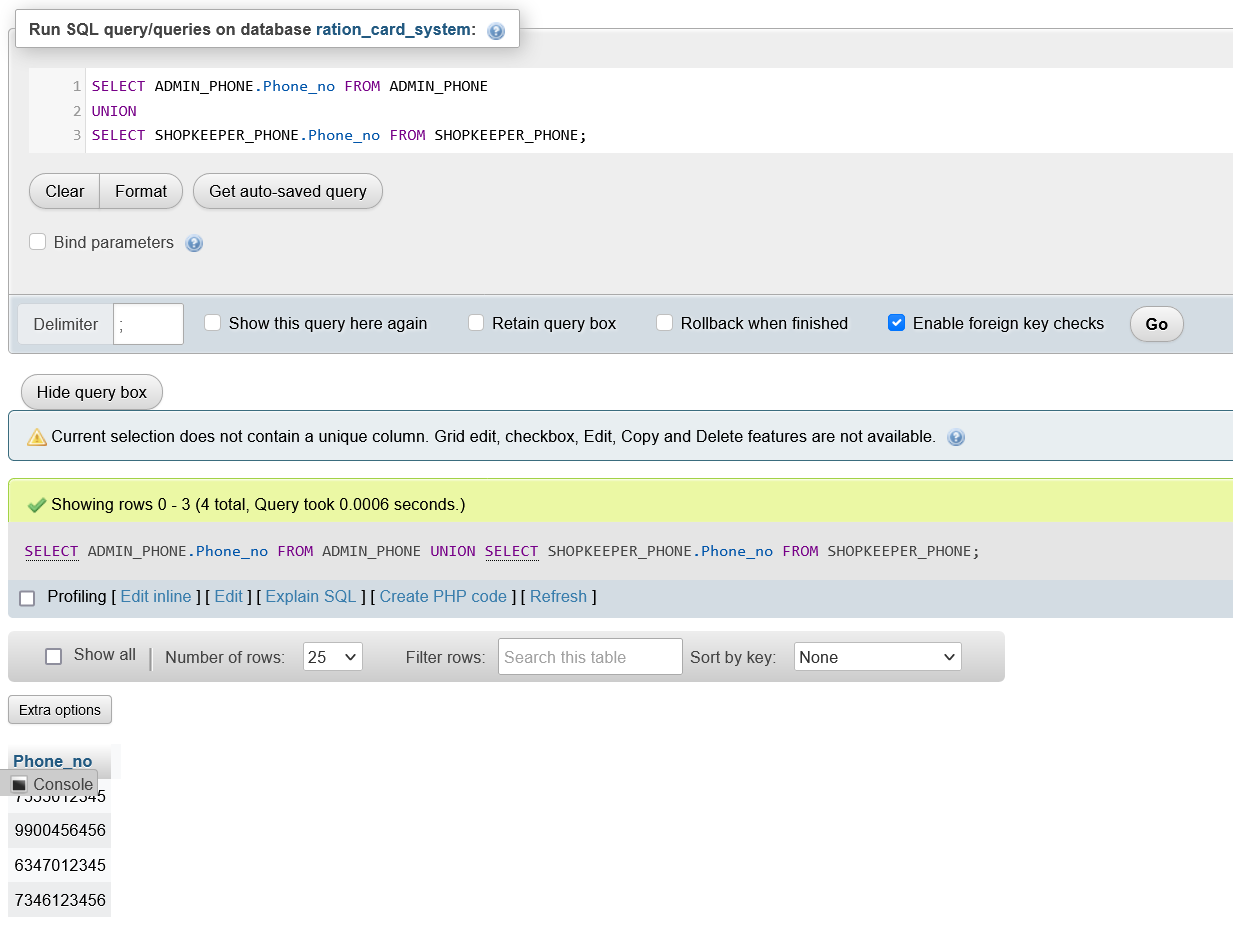
Query :

SELECT ADMIN\_PHONE.Phone\_no FROM ADMIN\_PHONE

UNION

SELECT SHOPKEEPER\_PHONE.Phone\_no FROM SHOPKEEPER\_PHONE;

Output :



1. Retrieve all the phone numbers that are in admin or shopkeeper table.

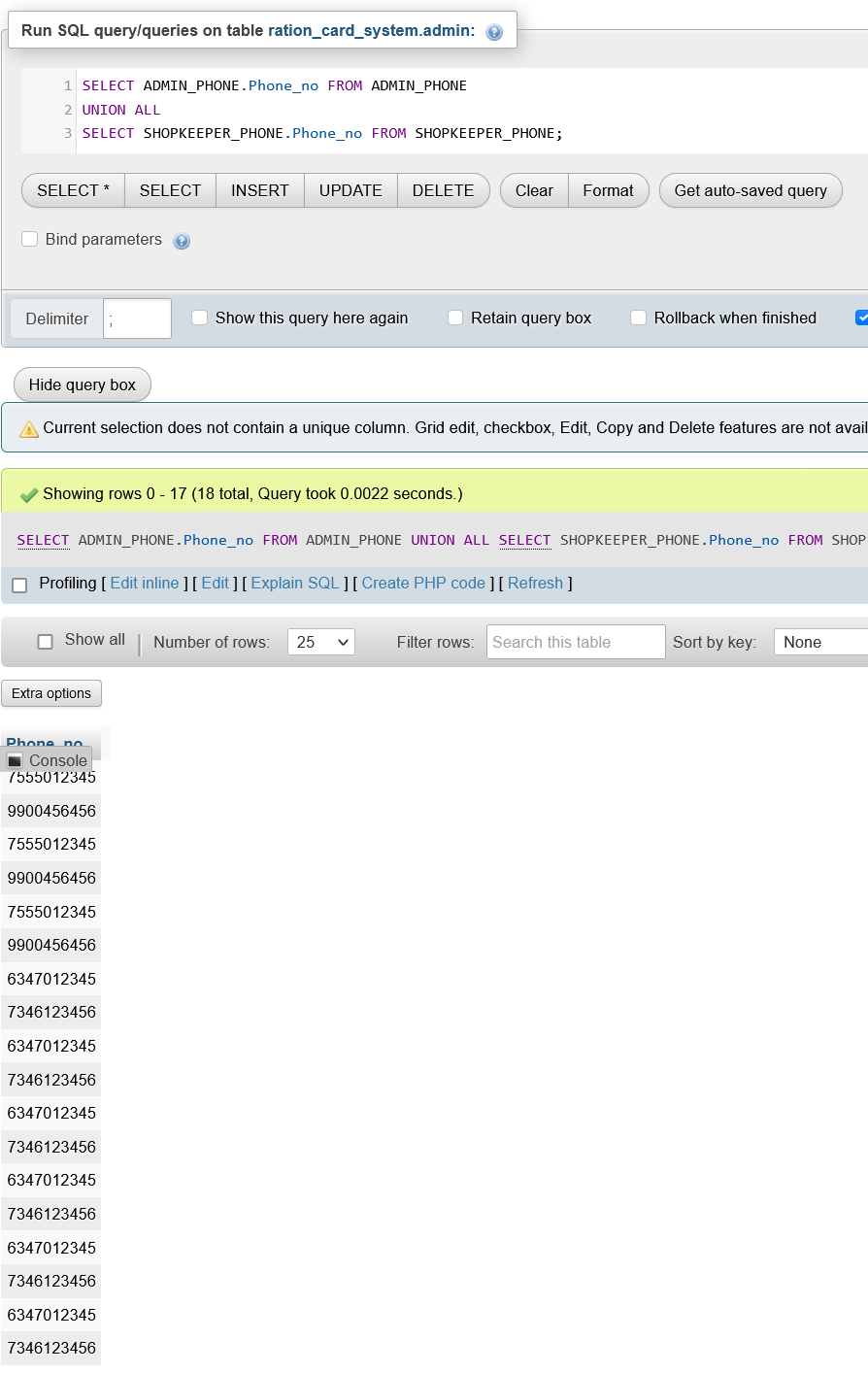
Query :

SELECT ADMIN\_PHONE.Phone\_no FROM ADMIN\_PHONE

UNION ALL

SELECT SHOPKEEPER\_PHONE.Phone\_no FROM SHOPKEEPER\_PHONE;

Output :



1. Retrieve the first name which are present in both customer and shopkeeper table.

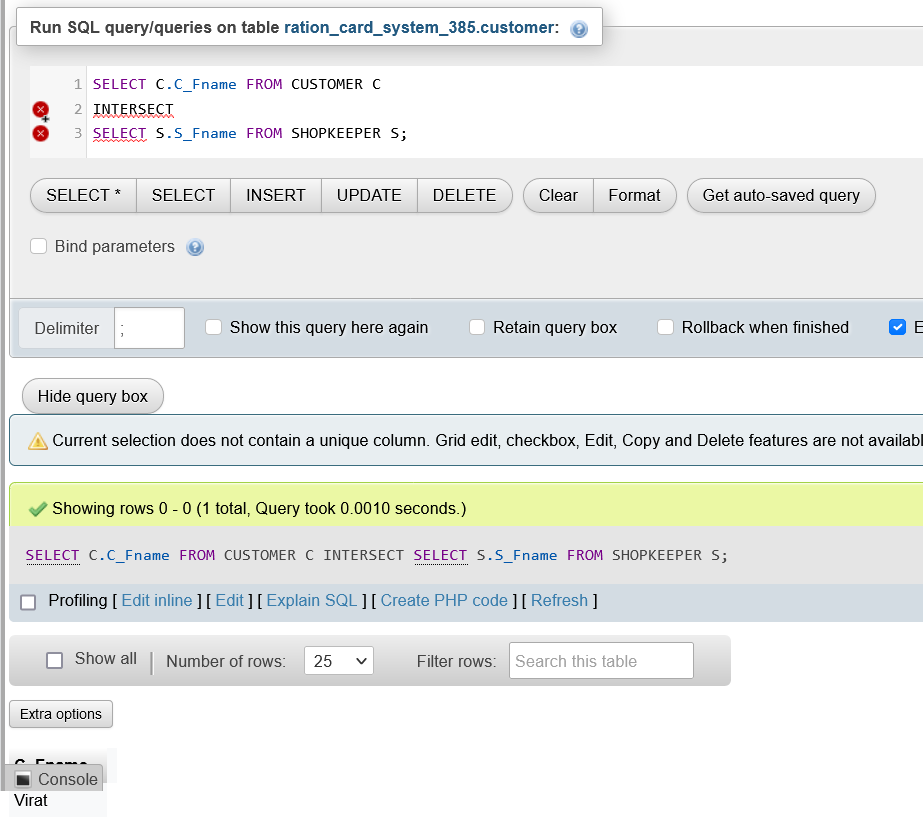
Query :

SELECT C.C\_Fname FROM CUSTOMER C

INTERSECT

SELECT S.S\_Fname FROM SHOPKEEPER S;

Output :



1. Retrieve the city names which are present in both customer and shopkeeper table.

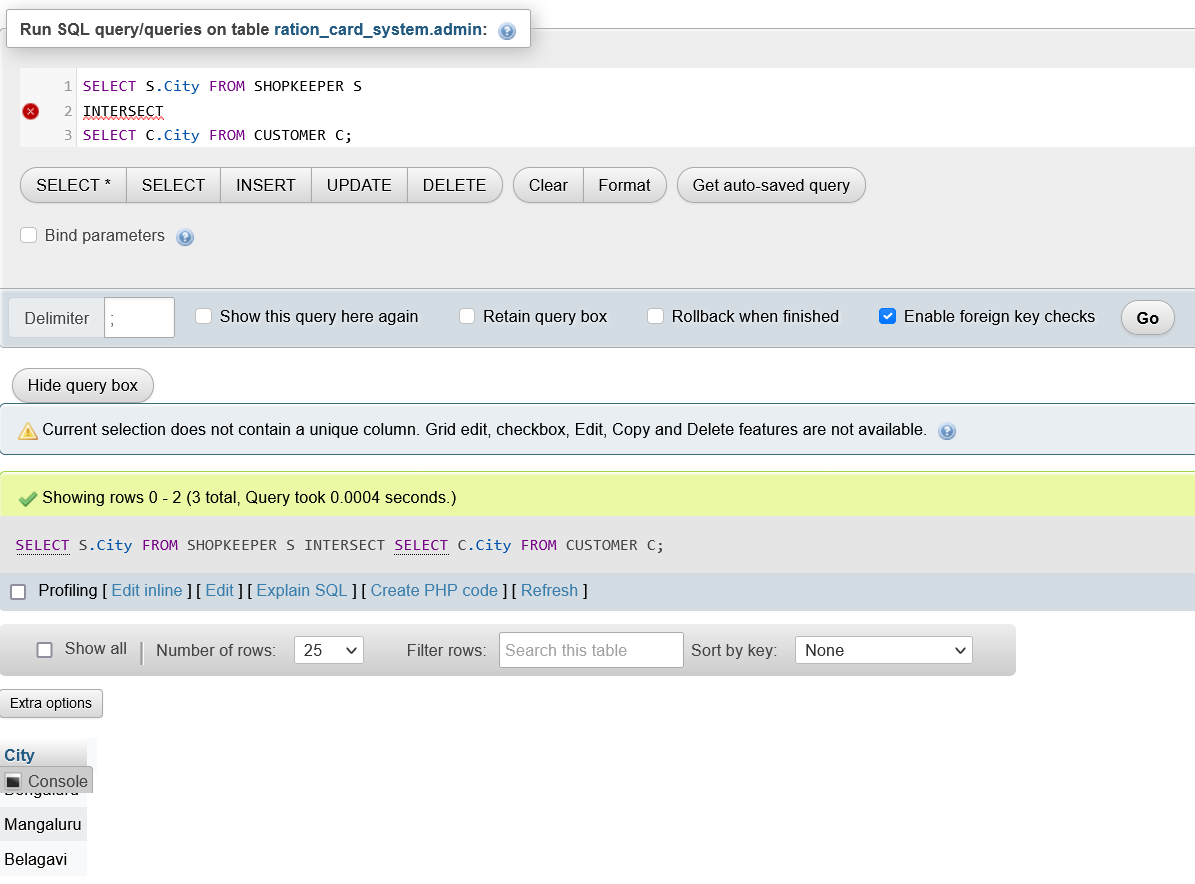
Query :

SELECT S.City FROM SHOPKEEPER S

INTERSECT

SELECT C.City FROM CUSTOMER C;

Output :



**Functions and Procedures**

1. Create a function to check the eligibility of a particular are for ration card.

Query :

DELIMITER $$

create function eligible(Age int)

    RETURNS varchar(20)

    DETERMINISTIC

    BEGIN

    IF Age > 10 THEN

    RETURN ("yes");

    ELSE

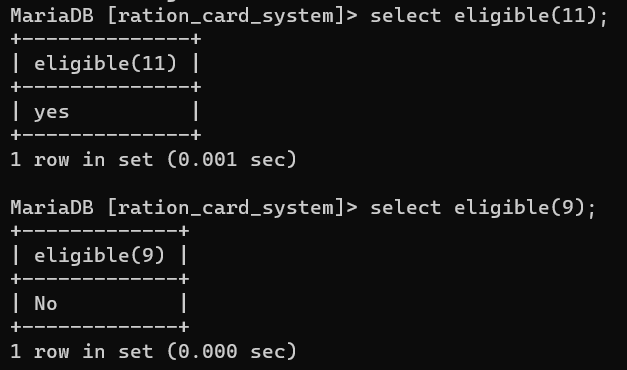
    RETURN ("No");

    END IF;

    end$$

DELIMITER ;

Output :



1. Create a procedure to count valid bills.

Query :

DELIMITER $$

CREATE PROCEDURE count\_valid\_bills(OUT p1 INT)

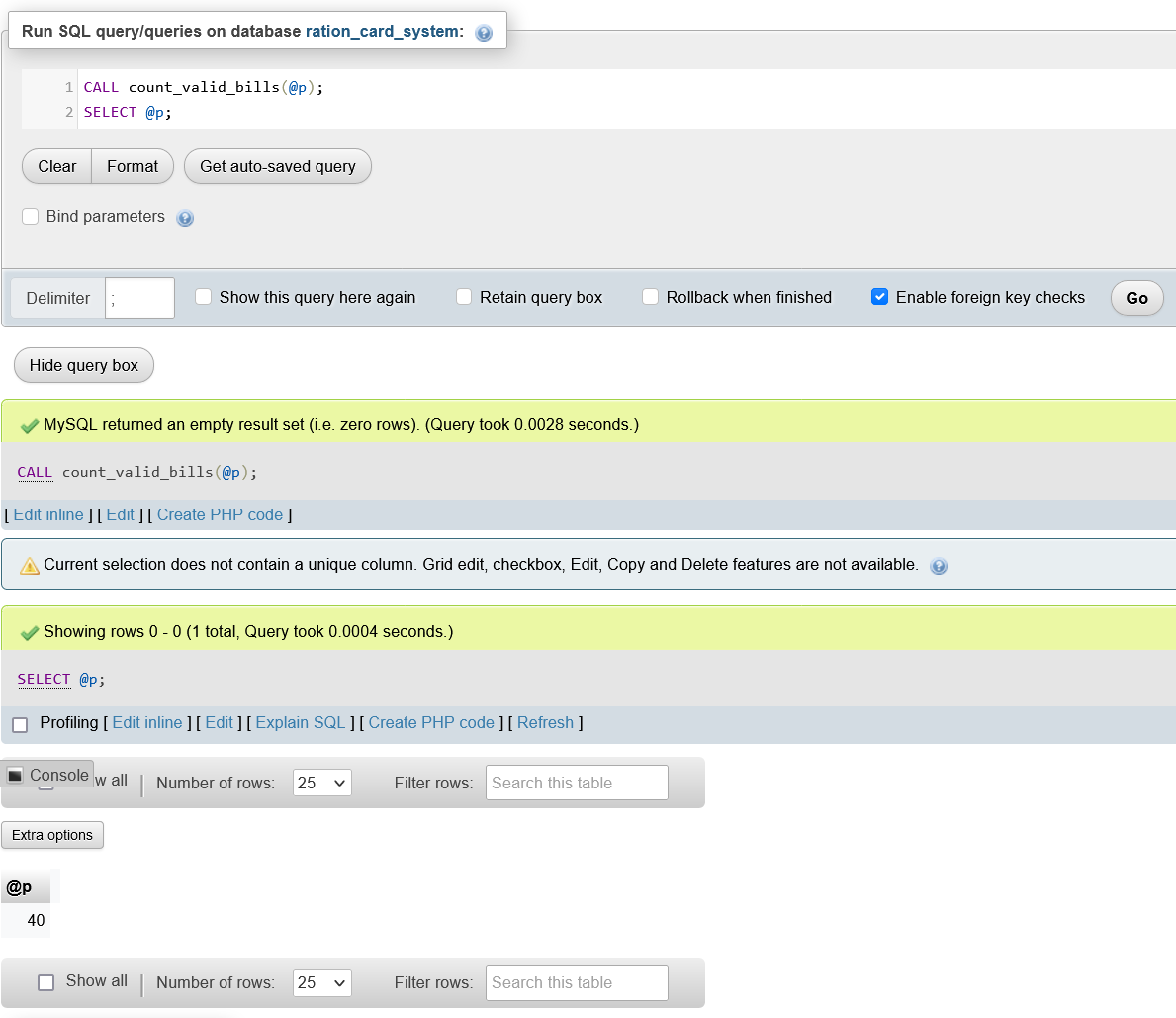
    BEGIN

    select COUNT(Bill\_id)INTO p1 from BILL WHERE Validity = 'Valid';

    END $$

DELIMITER ;

Output :



1. Create a procedure to retrieve given number of customers (Number of customer to be retrieved should be given as input).

Query :

DELIMITER $$

CREATE PROCEDURE get\_customer\_limit(IN var1 INT)

    BEGIN

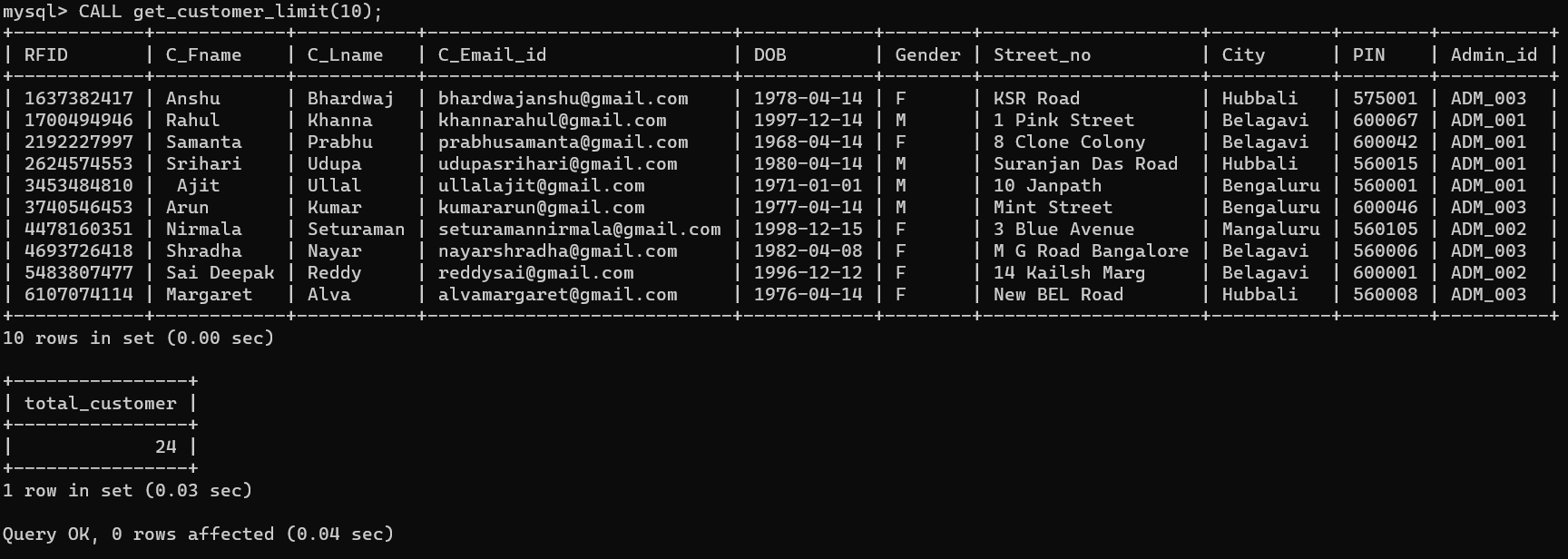
    select \* from CUSTOMER limit var1;

    select count(RFID) as total\_customer from CUSTOMER;

    END $$

DELIMITER ;

Output :



1. Create a procedure to retrieve all the customers managed by given admin (Admin id should be given as input).

Query :

DELIMITER $$

CREATE PROCEDURE get\_customer\_admin(in Adm varchar(10))

    BEGIN

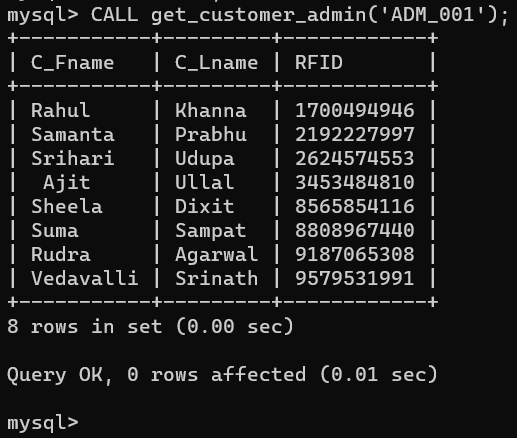
    select C\_Fname, C\_Lname, RFID

    from CUSTOMER WHERE CUSTOMER.Admin\_id = Adm;

end $$

DELIMITER ;

Output :



**Triggers** **and** **Cursors**

1. Create a trigger to check and update the validity of newly inserting bills.

Query :

DELIMITER $$

CREATE TRIGGER buying\_date

BEFORE INSERT

ON BILL

FOR EACH ROW

BEGIN

    IF new.Present\_date > new.Last\_valid\_date THEN SET new.Validity = 'Invalid';

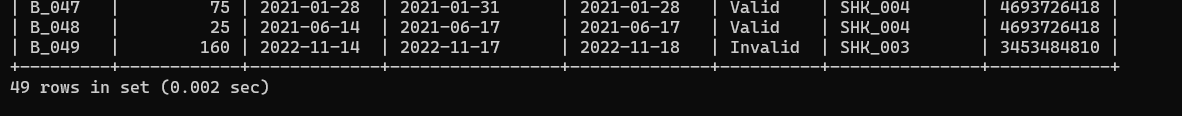
    END IF;

END$$

DELIMITER ;

INSERT INTO BILL VALUES('B\_049', 160, '2022-11-114', '2022-11-17', '2022-11-18', 'Valid', 'SHK\_003', 3453484810);

Output :



1. Create the trigger to show error when age of the newly inserting dependent is less than 10.

Query :

DELIMITER $$

CREATE TRIGGER dependent\_age

BEFORE INSERT

ON DEPENDENT

FOR EACH ROW

BEGIN

    DECLARE error\_msg VARCHAR(300);

    SET error\_msg = ("Age of the dependent should be 10 or more");

    IF new.Age < 10 THEN

    SIGNAL SQLSTATE '45000'

    SET MESSAGE\_TEXT = error\_msg;

    END IF;

END $$

DELIMITER ;

INSERT INTO DEPENDENT VALUES(3453484810, 'Aniket', '2000-11-3', 'M', '9', 'Son');

Output :

dependent_age

1. Use cursor to add valid bills to new table.

Query :

CREATE TABLE VALID\_BILLS(

    Bill\_id VARCHAR(10),

    Total\_cost INT(5),

    Issued\_date DATE,

    Last\_valid\_date DATE,

    Present\_date DATE,

    Validity VARCHAR(20),

    Shopkeeper\_id VARCHAR(10),

    RFID BIGINT(10)

);

DELIMITER $$

CREATE PROCEDURE get\_valid\_bills()

BEGIN

    DECLARE done int default 0;

    DECLARE b\_total\_cost INTEGER;

    DECLARE b\_RFID bigint;

    DECLARE b\_bill\_id, b\_validity, b\_shopkeeper\_id varchar(20);

    DECLARE b\_issued\_date, b\_last\_valid\_date, b\_present\_date date;

    DECLARE cur cursor for select \* from BILL WHERE BILL.VALIDITY = 'Valid';

    DECLARE continue handler for not found set done = 1;

    OPEN cur;

    label: LOOP

    fetch cur into b\_bill\_id, b\_total\_cost, b\_issued\_date, b\_last\_valid\_date, b\_present\_date, b\_validity, b\_shopkeeper\_id, b\_RFID;

    if done = 1 then leave label;

    end if;

    INSERT INTO VALID\_BILLS VALUES(b\_bill\_id, b\_total\_cost, b\_issued\_date, b\_last\_valid\_date, b\_present\_date, b\_validity, b\_shopkeeper\_id, b\_RFID);

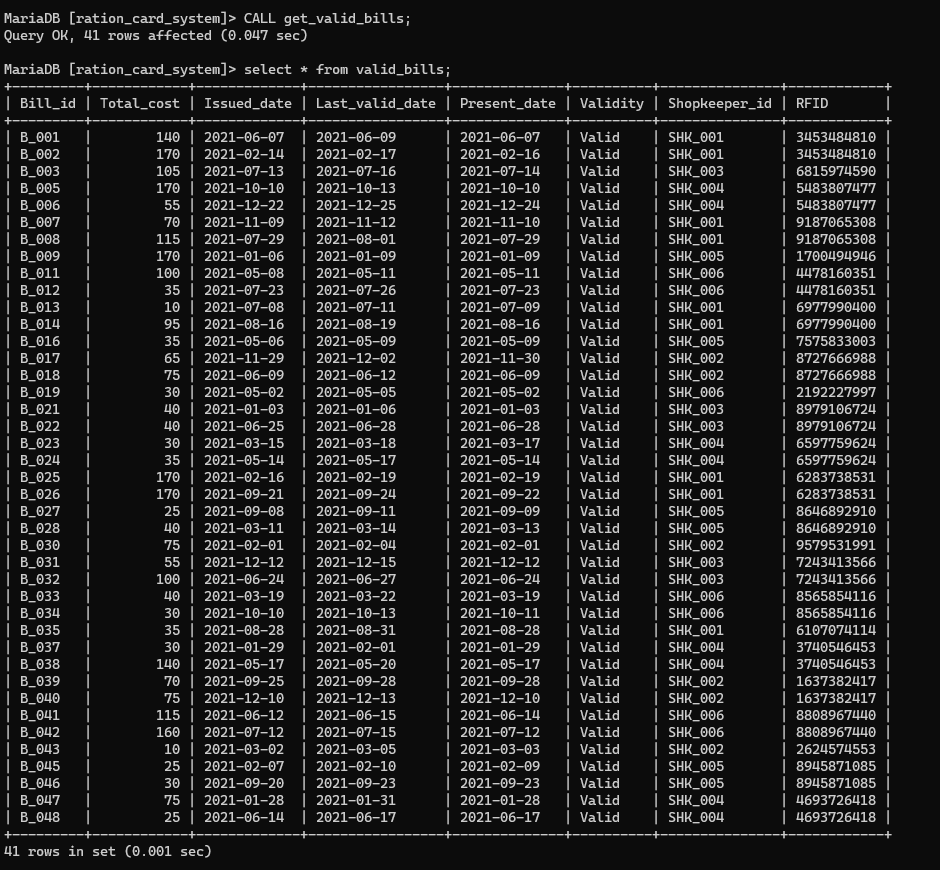
    end loop;

    close cur;

end$$

DELIMITER ;

Output :



1. Use cursor to add name and email id of customers to new table.

Query :

CREATE TABLE CUSTOMER\_EMAIL(

    Fname varchar(10),

    Lname varchar(10),

    Email\_id VARCHAR(50)

);

DELIMITER $$

CREATE PROCEDURE create\_email\_list ()

BEGIN

    DECLARE done INTEGER DEFAULT 0;

    DECLARE Fname varchar(10) ;

    DECLARE Lname varchar(10) ;

    DECLARE emailAddress varchar(100) ;

    DEClARE curEmail CURSOR FOR SELECT C\_Fname, C\_Lname, C\_Email\_id FROM CUSTOMER;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN curEmail;

    LABLE: LOOP

    FETCH curEmail INTO Fname, Lname, emailAddress;

    IF done = 1 THEN

    LEAVE LABLE;

    END IF;

    INSERT INTO CUSTOMER\_EMAIL VALUES(Fname, Lname, emailAddress);

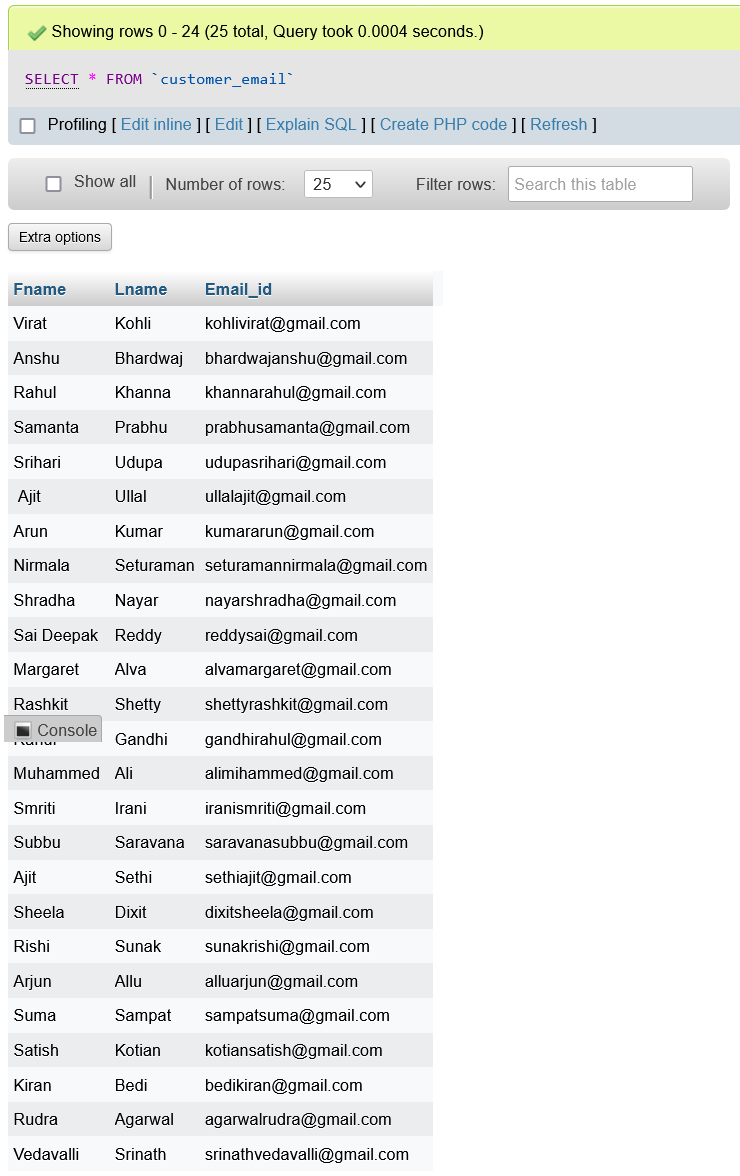
    END LOOP;

    CLOSE curEmail;

END$$

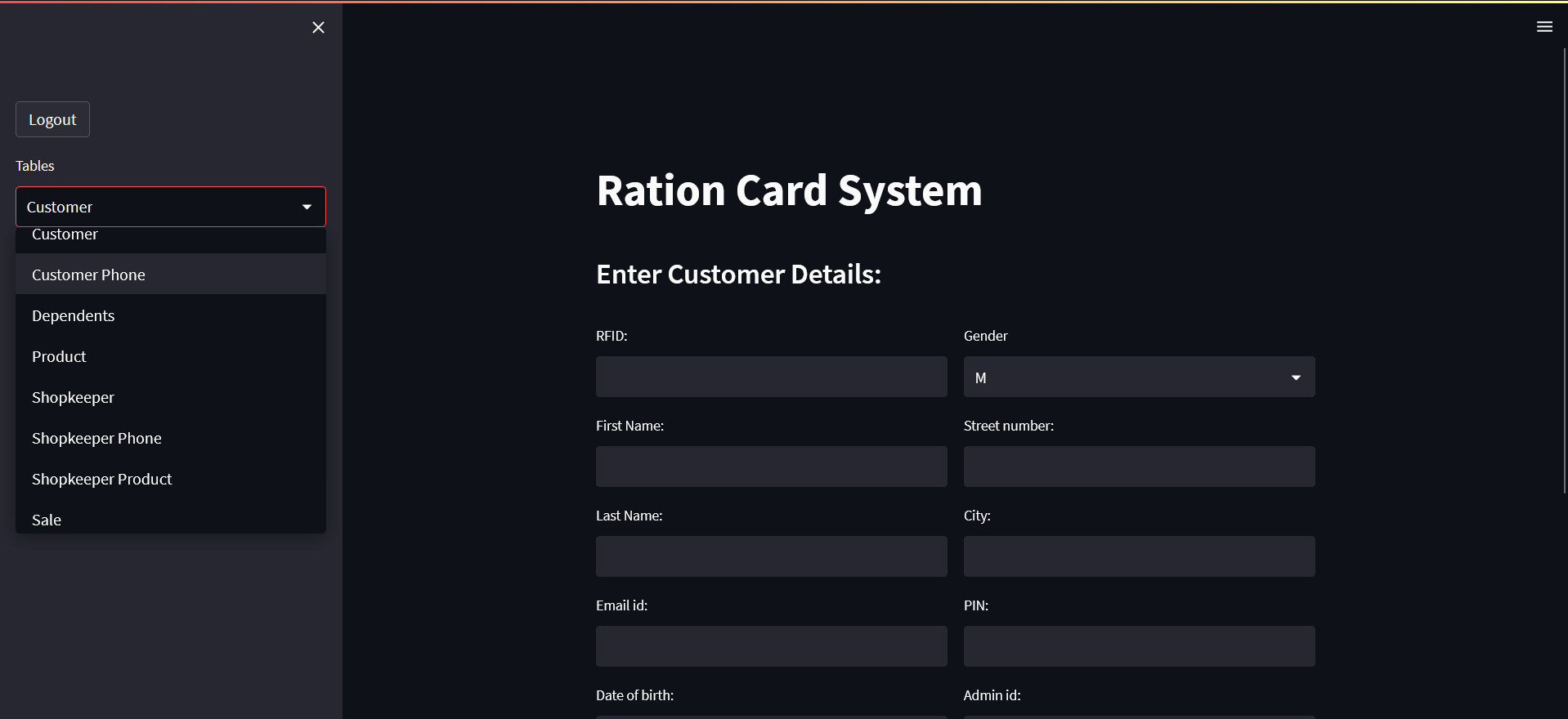
DELIMITER ;

Output :

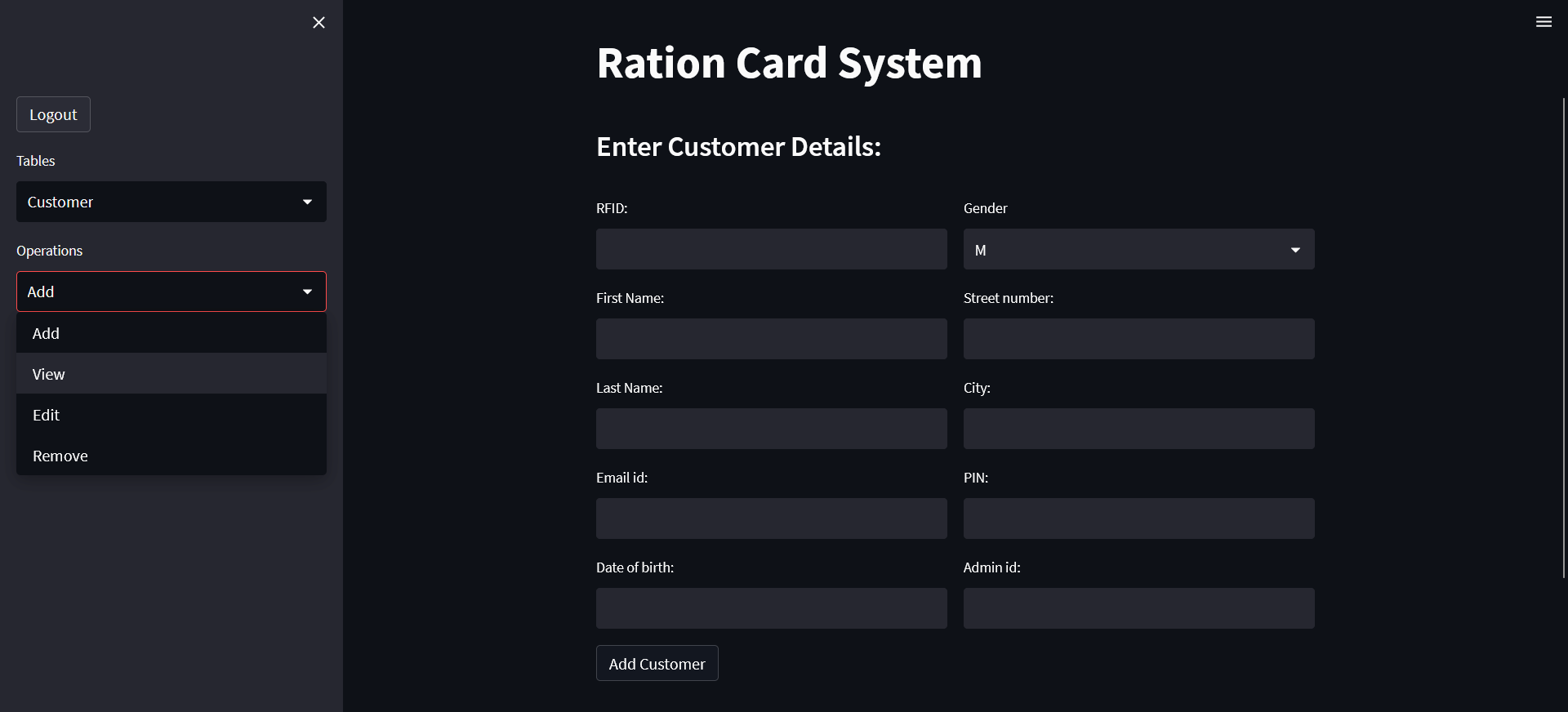


**FRONT END**

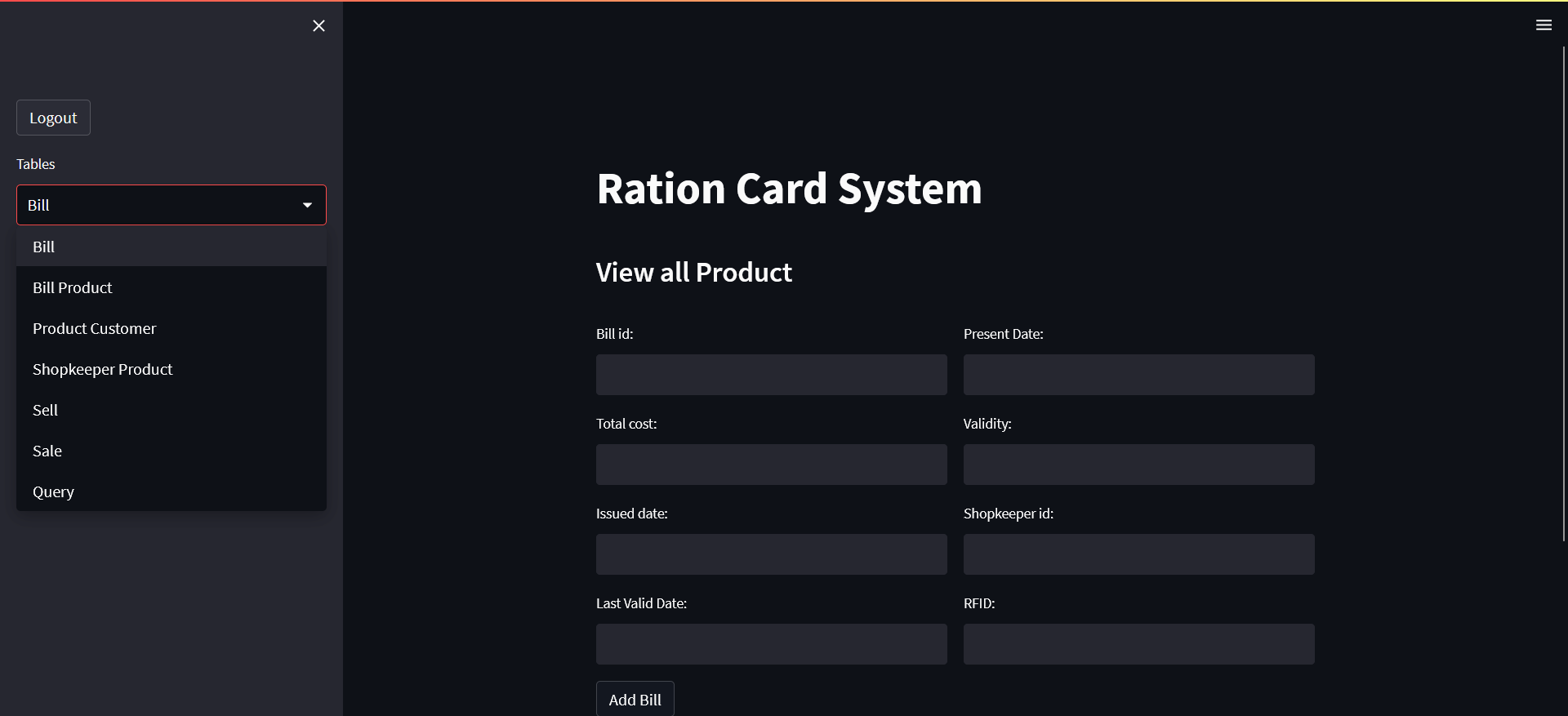
1. Tables that admin have the access.



1. Operations that admin can perform on these tables.



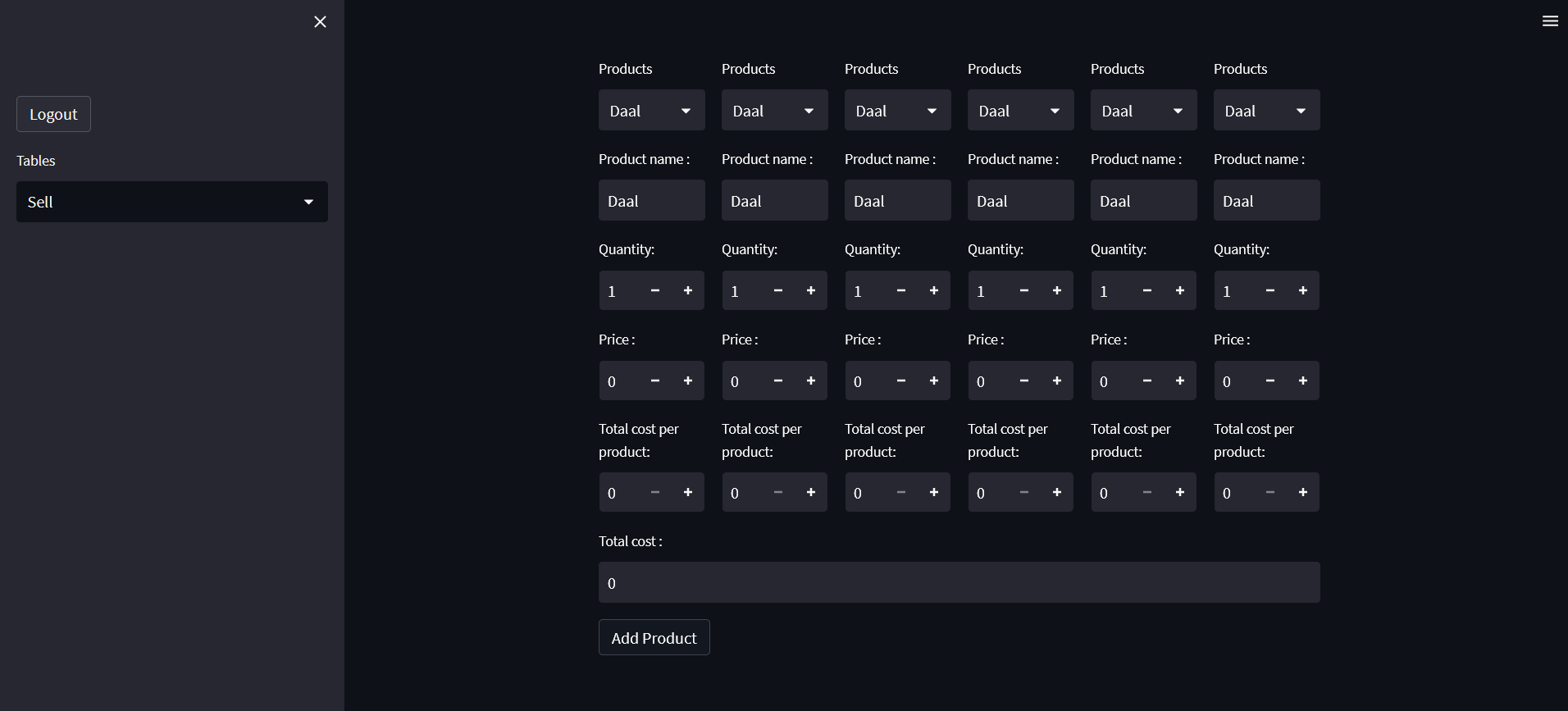
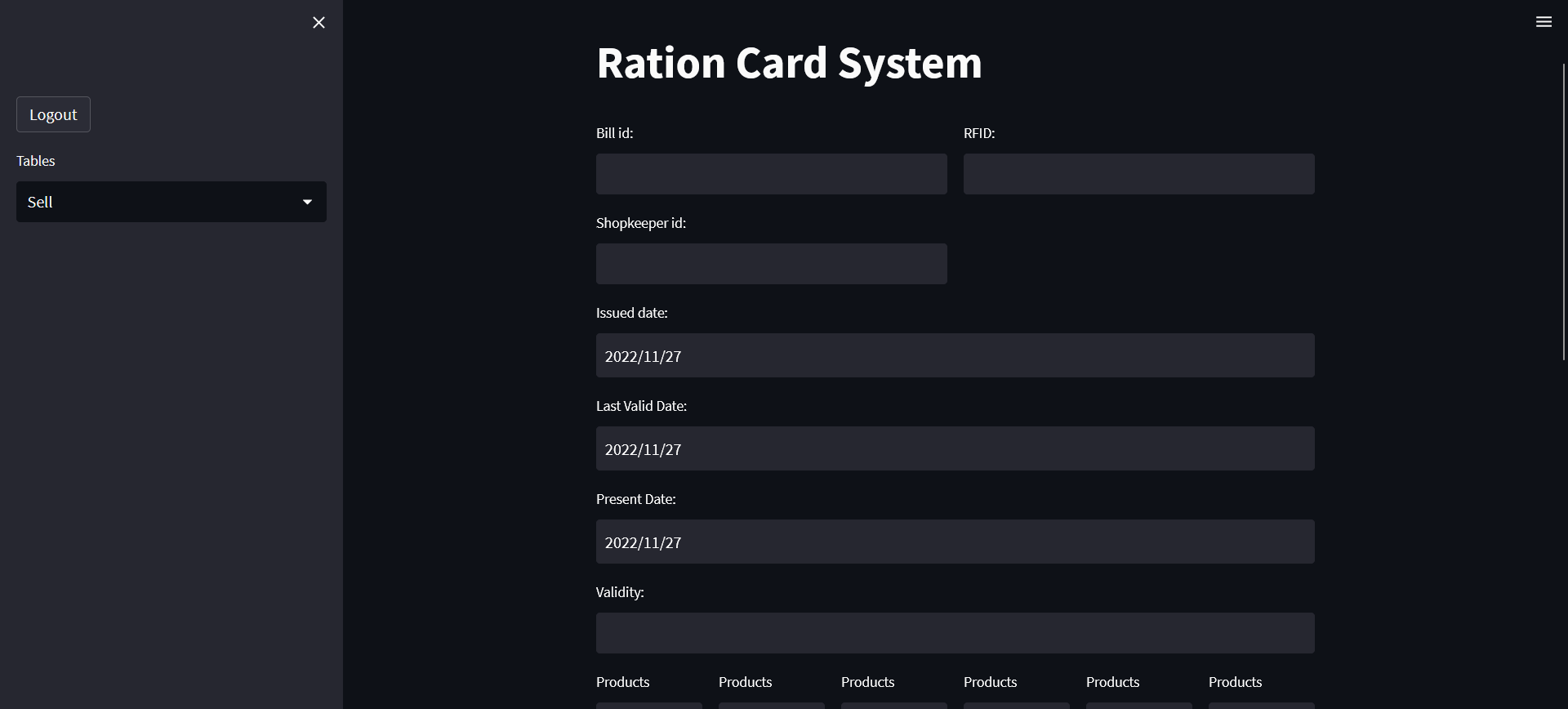
1. Tables that shopkeeper have the access.



1. Operations that shopkeeper can perform on these tables.



1. Selling products.



1. Pie Chart for Sales between given Dates.

