DBMS- Mini Project

Title of the Project –
LOYALTY POINTS MANAGEMENT SYSTEM

Name: Sathvik A
SRN: PES1UG20CS493
Section: I

V Semester

Short Description and Scope of the Project

- Loyalty management software helps businesses create, manage, and analyze customer loyalty programs.
- This product allows businesses to identify either repeat customers or potential repeat customers, and then send them incentives such as discounts or rewards points so they return to that business
- To qualify for inclusion in the Loyalty Management category, a product must:
    - Enroll valuable, loyal, or interested customers in a loyalty program
    - Allow for the configuration of one or more loyalty programs such as point-based, spend-based, tiered, paid, etc.
    - Provide tools to communicate with loyal customers and distribute discounts, cashback, or other rewards
    - Capture customer data including purchase history, membership status, and engagement with the loyalty program

Scope:

The project scope defines the description of the work that is required in delivering the rental house management system.

The following are the scopes of work during the course of the project:

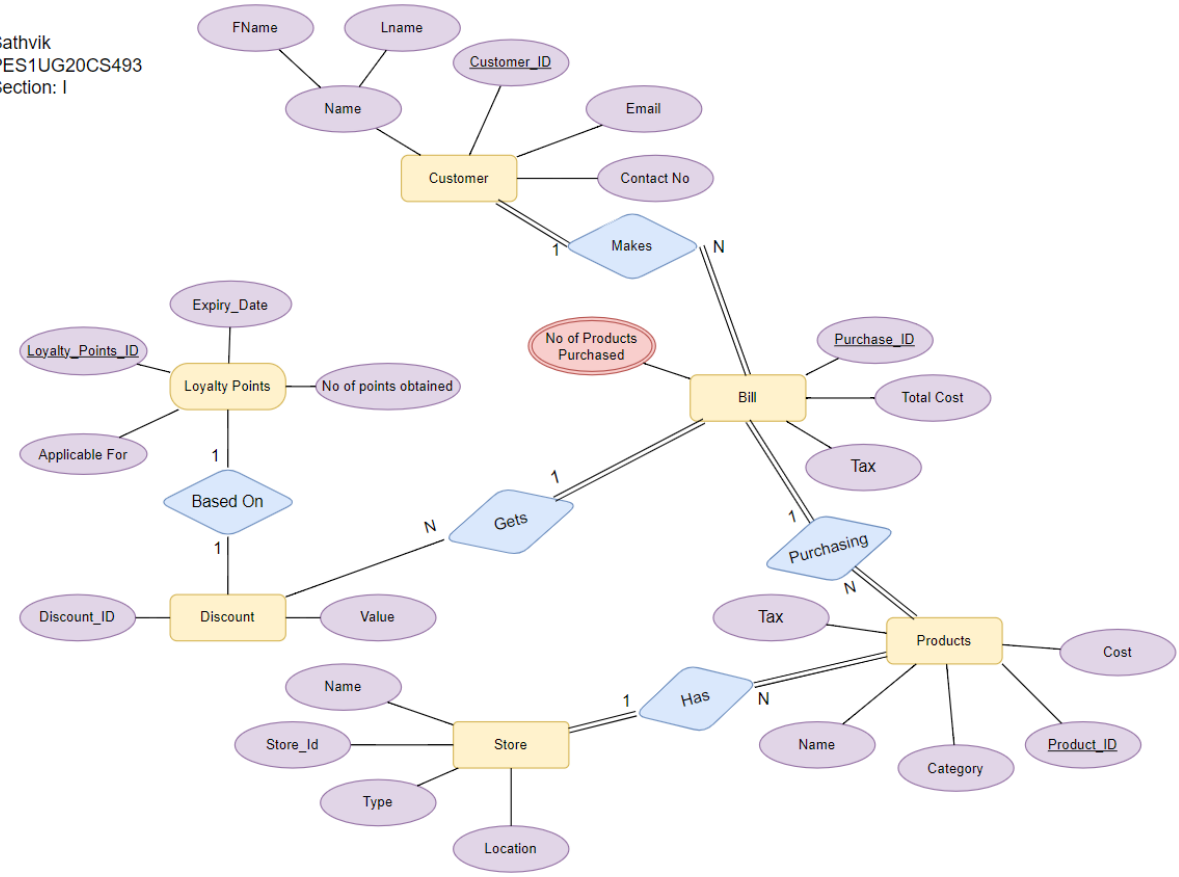
- Study and understand the requirement of this project
- ER diagram and relational schema
- A minimalistic UI/front-end for both ADMIN and users
- Creating and populating the database
- Using the SQL queries and show the reflections in the front-end developed

Modules Used:

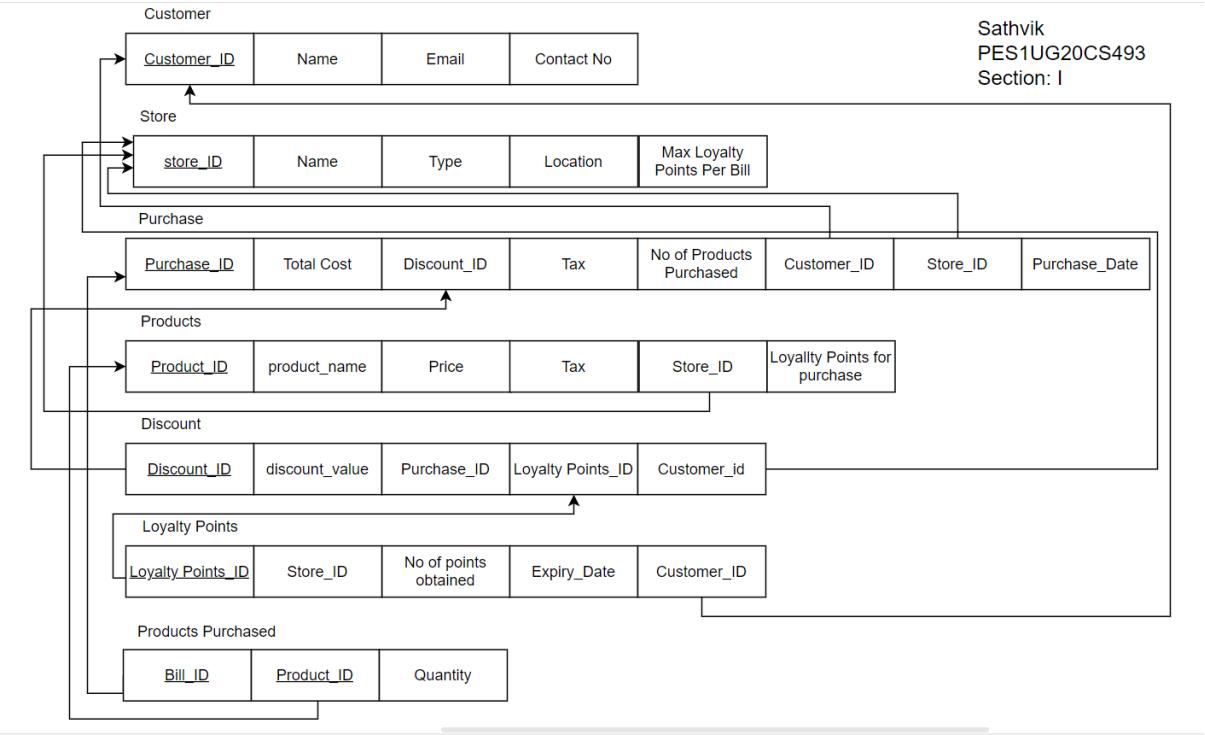
- Create, Read, Update, Delete to perform CRUD operations as an ADMIN
- Register: To register the customer
- Purchase: To make a purchase from a store
- View: To show the Loyalty Points for a customer
- Modification: To provide the required modification

ER Diagram

Sathvik
PES1UG20CS493
Section: I



Relational Schema



DDL statements - Building the database

Creation of database:

```
create database project;
```

Creation of tables: Customer, Store, Purchase, Product, Loyalty_Points, Discount, Products_Purchased

Customer Table:

```
create table customer(
    customer_id int AUTO_INCREMENT,
    constraint customer_id_pk primary key(customer_id),
    name varchar(30),
    email varchar(30),
    contact_no char(10)
);
```

Store Table:

```
create table store(
    store_id int AUTO_INCREMENT,
    constraint store_pk primary key(store_id),
    name varchar(30),
    type varchar(30),
    location varchar(50),
    no_of_years_for_expiry float,
    max_points_per_bill int
);
```

Purchase Table:

```
create table purchase(
    bill_id int AUTO_INCREMENT,
    total_cost float,
    discount float,
    tax float,
    no_of_products_purchased int,
    customer_id int,
    store_id int,
    purchase_date date default current_timestamp,
    constraint store_fk foreign key(store_id) references store(store_id),
    constraint Loyal foreign key(customer_id) references
customer(customer_id),
    constraint purchase_pk primary key(bill_id));
```

Product Table:

```
create table product(
    product_id int AUTO_INCREMENT,
    product_name varchar(30),
    price float,
    tax float,
    store_id int,
    loyalty_Points_on_purchase int,
    constraint store_id_fk foreign key(store_id) references store(store_id),
    constraint product_pk primary key(product_id)
);
```

Loyalty_Points Table:

```
create table loyalty_Points(
    points_id int AUTO_INCREMENT,
    store_id int,
    customer_id int,
    points int,
    date_of_expiry date,
    constraint customer_id_Loyal_fk foreign key(customer_id) references
customer(customer_id),
    constraint store_id_loyal_fk foreign key(store_id) references
store(store_id),
    constraint loyalty_pk primary key(points_id)
);
```

Discount Table:

```
create table discount(
    discount_id int,
    bill_id int,
    customer_id int,
    discount_value float,
    loyalty_Points_used int,
    constraint customer_id_discout_fk foreign key(customer_id) references
customer(customer_id),
    constraint Purchase_discout_fk foreign key(bill_id) references
purchase(bill_id),
    constraint discount_pk primary key(discount_id)
);
```

Products_Purchased Table:

```
create table products_purchased(

    bill_id int,
    product_id int,
    quantity int,
    constraint bill_id_fk foreign key(bill_id) references purchase(bill_id),
    constraint product_id_fk foreign key(product_id) references
product(product_id),
    constraint products_purchased_pk primary key(bill_id, product_id)
);
```

Populating the Database

Populating the Customer Table

```
LOAD DATA INFILE "D:\Sem 5\\DBMS\\Project\\Data\\Customer.csv"
INTO TABLE Customer
COLUMNS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
ESCAPED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;
```

Populating the Store Table

```
LOAD DATA INFILE "D:\Sem 5\\DBMS\\Project\\Data\\Store.csv"
INTO TABLE Store
COLUMNS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
ESCAPED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;
```

Populating the Product Table

```
LOAD DATA INFILE "D:\Sem 5\\DBMS\\Project\\Data\\Products.csv"
INTO TABLE product
COLUMNS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
ESCAPED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;
```

Populating the Purchase Table

```
LOAD DATA INFILE "D:\Sem 5\\DBMS\\Project\\Data\\Purchases.csv"
INTO TABLE purchase
COLUMNS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
ESCAPED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;
```

Populating the products_purchased Table

```
LOAD DATA INFILE "D:\Sem 5\\DBMS\\Project\\Data\\No_of_products.csv"
INTO TABLE products_purchased
COLUMNS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
ESCAPED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;
```

Populating the Loyalty_points Table

```
LOAD DATA INFILE "D:\Sem 5\\DBMS\\Project\\Data\\Loyalty_points.csv"
INTO TABLE loyalty_points
COLUMNS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
ESCAPED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;
```

Populating the Discount Table

```
LOAD DATA INFILE "D:\Sem 5\\DBMS\\Project\\Data\\Discount.csv"
INTO TABLE Discount
COLUMNS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
ESCAPED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;
```

The Data is imported from CSV file
Example:

ID	Name	Type	Location	max_point	No of Years for Expiry
1	Goodness	Jewellery Store	Antwerp	20000	3
2	Tech Assau	Electronics Store	San Francisco	10000	1
3	Sunrise Ma	Grocery Supermarket	Bern	2000	0.25
4	Read Gooc	Book Store	Portland	2000	0.5
5	Fashion Fie	Clothing Store	Milan	10000	0.25
6	Sweet Toc	Confectionery Store	Paris	3000	0.5
7	Primo Spac	Department Store	London	5000	0.25
8	Get Movin	Automobile retail	Molsheim	100000	5
9	Gas-N-Go	Filling Station	Tehran	5000	0.25
10	Mediserv	Pharmacy	Dublin	1000	0.5

Join Queries:

To view the loyalty Points obtained by a customer in all the stores

```
def view_loyalty_points():
    cursor.execute("SELECT CUSTOMER_ID, NAME FROM CUSTOMER")
    customers = cursor.fetchall()
    to_update = st.selectbox("Select the Name to update", customers)
    customer_id = to_update[0]
    cursor.execute(
        f"""SELECT POINTS, NAME
        FROM LOYALTY_POINTS L NATURAL JOIN STORE S
        WHERE L.STORE_ID=S.STORE_ID AND CUSTOMER_ID='{customer_id}' and points > 0""")
    data = cursor.fetchall()

    if len(data) == 0:
        st.write("No loyalty points found")
    else:
        st.write("Loyalty Points")
        data = [{"Points": x[0], "Store Name": x[1]} for x in data]
        hide_dataframe_row_index = """
                <style>
                .row_heading.level0 {display:none}
                .blank {display:none}
                </style>
            """
        st.markdown(hide_dataframe_row_index, unsafe_allow_html=True)

        st.table(data)
```

(3, 'Robert Shoemake')

Loyalty Points

Points	Store Name
200	Sunrise Mart
60	Sunrise Mart
60	Sunrise Mart

Aggregate Functions:

```
List the total number of customers who have purchased a product from a
particular store.
```

Enter Query

```
SELECT COUNT(DISTINCT C.customer_id )
FROM Customer C, Purchase P, Store S
WHERE C.customer_id = P.customer_id AND P.store_id = S.store_id AND S.name = 'Goodness Gems';
```

COUNT(DISTINCT C.customer_id)
2

Set Operation:

Enter Query

```
SELECT DISTINCT C.name
FROM Customer C, Purchase P, Store S, Products_purchased PP
WHERE C.customer_id = P.customer_id AND P.store_id = S.store_id AND S.name = 'Fashion Fiesta'
AND P.bill_id= PP.bill_id AND PP.quantity > 10
UNION
SELECT DISTINCT C.name
FROM Customer C, Purchase P, Store S
WHERE C.customer_id = P.customer_id AND P.store_id = S.store_id AND S.name = 'Fashion Fiesta';
```

name
Matt Foreman

Functions and Procedures:

Functions:

Function to calculate the total cost of the bill

```
DELIMITER ##
create function total_cost(bill int) returns float
begin
    declare total float;
    select sum(price*quantity) into total from products_purchased natural join
product where bill_id = bill and product_id = product_id;
    return total;
end;
##
DELIMITER ;
```

Enter Query

```
SELECT total_cost(2)
```

total_cost(2)
20,000.0000

Function to calculate the total tax of the bill

```
DELIMITER ##
create function total_tax(bill int) returns float
begin
    declare total float;
    select sum(tax*quantity*price/100) into total from products_purchased
natural join product where bill_id = bill;
    return total;
end;
##
DELIMITER ;
```

Enter Query

```
SELECT total_tax(2)
```

total_tax(2)
4,000.0000

Function to calculate the total loyalty points obtained per the bill

```
DELIMITER ##
create function total_loyalty_points(bill int) returns int
begin
    declare total int;
    select sum(loyalty_Points_on_purchase*quantity) into total from
products_purchased natural join product where bill_id = bill;

    return total;
end;
##
DELIMITER ;
```

Enter Query

```
SELECT total_loyalty_points(2)
```

total_loyalty_points(2)
3000

Procedures:

1. add_loyalty_points(bill int)

- Procedure to add loyalty points to the customer for a purchase
- Calculate the total loyalty points obtained per the bill using the function total_loyalty_points
- Insert the loyalty points into the loyalty_Points table
- Set the date of expiry of the loyalty points using the trigger set_date_of_expiry
- Calculate date of expiry by adding the no_of_years_for_expiry to current timestamp

```
DELIMITER ##
create procedure add_loyalty_points(bill int)
begin
    declare total_loyalty_points_purchase int;
    declare customer int;
    declare no_of_years_for_expiryd date;
    declare store int;
    declare expiry_dated date;
    select total_loyalty_points(bill) into total_loyalty_points_purchase;
    select customer_id, store_id into customer, store from purchase where
bill_id = bill;
    select no_of_years_for_expiry into no_of_years_for_expiryd from store
where store_id = store;
    set expiry_dated = date_add(current_timestamp, interval
no_of_years_for_expiryd year);
    insert into loyalty_Points(customer_id, store_id, points, date_of_expiry)
values(customer, store, total_loyalty_points_purchase, expiry_dated);
end;
##
DELIMITER ;
```

2. discount_from_loyalty_points(bill int, customer int)

- Procedure to calculate the discount from loyalty points from previous purchases
- The Loyalty points are used in the order of their expiry
- Each loyalty point is worth 1 rupee of discount

```
DELIMITER ##
create procedure discount_from_loyalty_points(bill int, customer int)
begin
    declare discount_valu float;
    declare loyalty_points_use int;
    declare points_iden int;
    declare total_cost_ float;
    select total_cost(bill) into total_cost_;
    select points_id, points into points_iden, loyalty_points_use from
loyalty_Points where customer_id = customer and date_of_expiry >
current_timestamp and points > 0 limit 1;
    if points_iden is null then
        set discount_valu = 0;
        set loyalty_points_use = 0;
    else
        if loyalty_points_use > total_cost_ then
        set loyalty_points_use = total_cost_;
        end if;

    end if;
    set discount_valu = loyalty_points_use;
    update loyalty_Points set points = points - loyalty_points_use where
points_id = points_iden and date_of_expiry > current_timestamp and points > 0;
    insert into discount(bill_id, customer_id, discount_value,
loyalty_points_used) values (bill, customer, discount_valu,
loyalty_points_use);
end;
##
DELIMITER ;
```

3. total_cost_after_discount_and_taxes

Procedure to calculate the total cost of the bill after discount and taxes

```
DELIMITER ##
create procedure total_cost_after_discount_and_taxes(bill int)
begin
    declare total_cost float;
    declare total_tax float;
    declare discountt float;
    select total_cost(bill) into total_cost;
    select total_tax(bill) into total_tax from purchase where bill_id = bill;
    select discount_value into discountt from discount where bill_id = bill;
    select total_cost + total_tax into total_cost;
    update purchase set total_cost = total_cost where bill_id = bill;
    update purchase set tax = total_tax where bill_id = bill;
    update purchase set discount = discountt where bill_id = bill;
end;
##
DELIMITER ;
```

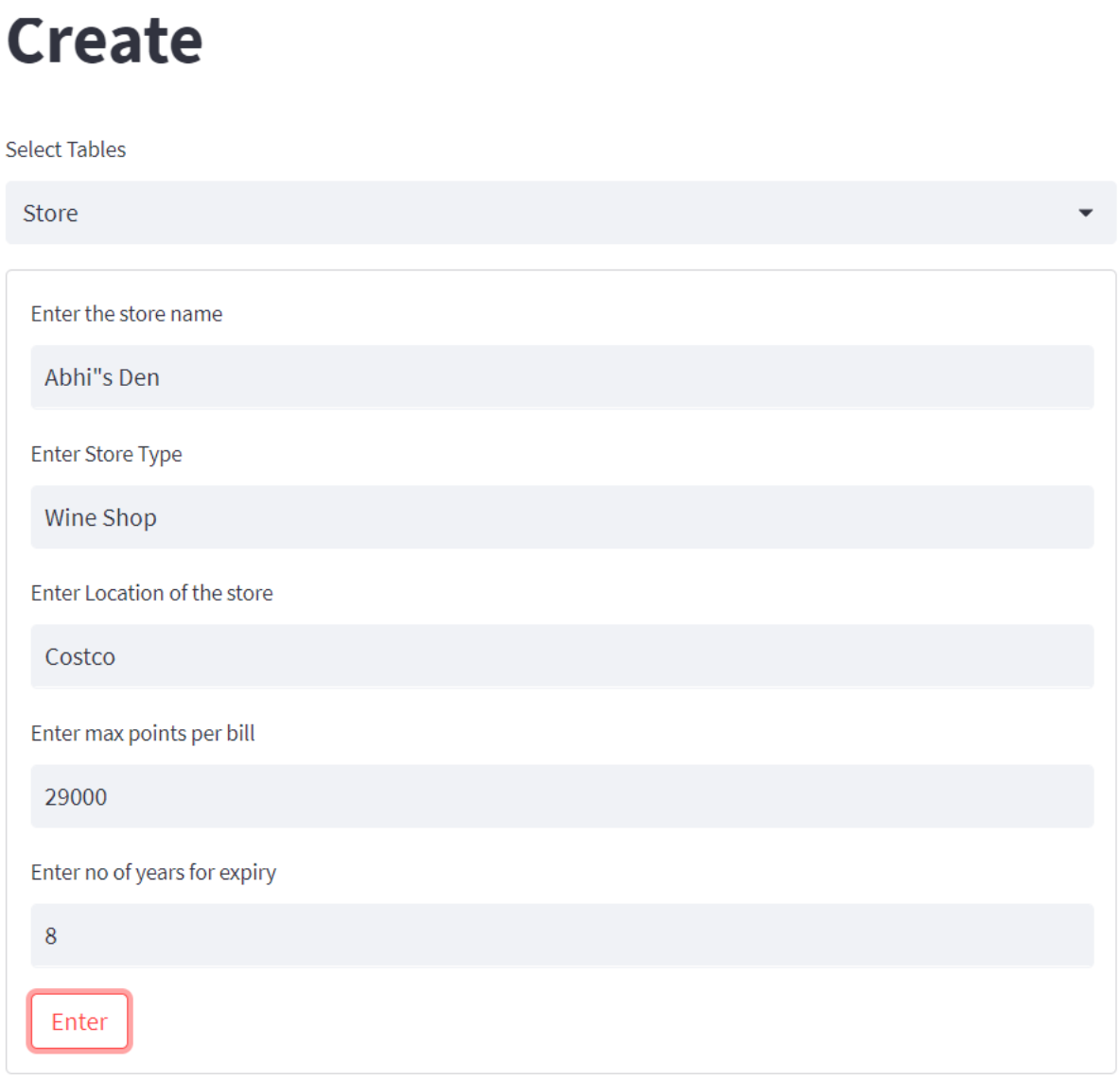
Triggers:

Trigger to limit the loyalty points obtained by a customer in the bill to max_points_per_bill, unique for a store

```
DELIMITER ##
create trigger check_max_points_per_bill
before insert on loyalty_Points
for each row
begin
    declare max_points int;
    select max_points_per_bill into max_points from store where store_id =
new.store_id;
    if new.points > max_points then
        set new.points = max_points;
    else
        set new.points = new.points;
    end if;
end;
##
DELIMITER ;
```

Frontend:

Create a store



Reading the store table:

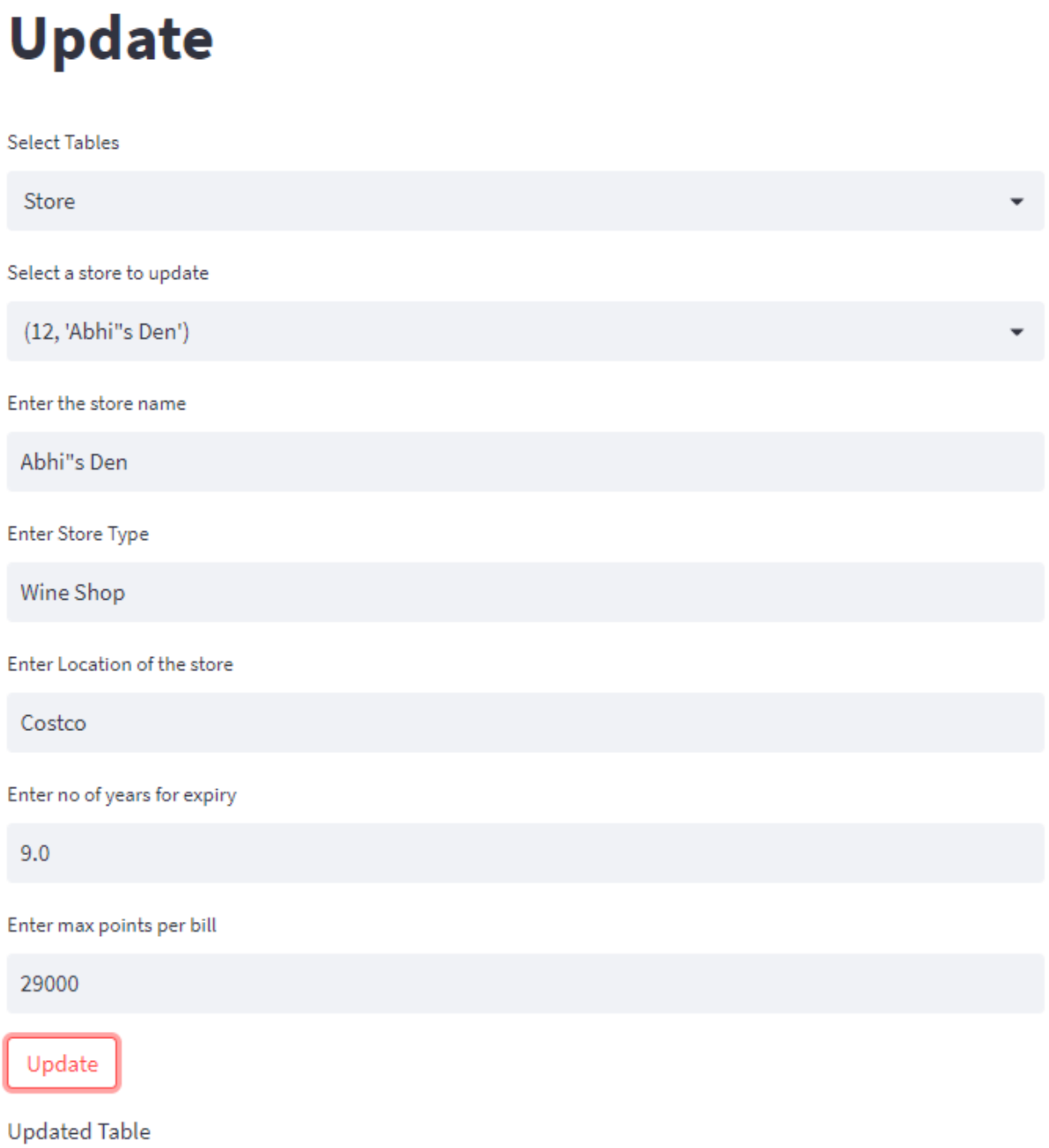
Read

Select Tables

Store

ID	Store Name	Store Type	Store Location	Max Points Per Bill	No of Years for Expiry
1	Goodness Gems	Jewellery Store	Antwerp	0.2000	20000
2	Tech Assault	Electronics Store	San Francisco	0.7000	10000
3	Sunrise Mart	Grocery Supermarket	Bern	0.4000	2000
4	Read Good	Book Store	Portland	0.8000	2000
5	Fashion Fiesta	Clothing Store	Milan	0.8000	10000
6	Sweet Tooth	Confectionery Store Chain	Paris	0.5000	3000
7	Primo Space	Department Store	London	0.4000	5000
8	Get Moving Motors	Automobile retail	Molsheim	0.2000	100000
10	Mediserv	Pharmacy	Dublin	0.5000	1000
11	Srinu Batla Kottu	Battla Kottu	London	0.2500	200000
12	Abhi"s Den	Wine Shop	Costco	8.0000	29000

Updating the table:



Updated Table

ID	Store Name	Store Type	Store Location	Max Points Per Bill	No of Years for Expiry
1	Goodness Gems	Jewellery Store	Antwerp	0.2000	20000
2	Tech Assault	Electronics Store	San Francisco	0.7000	10000
3	Sunrise Mart	Grocery Supermarket	Bern	0.4000	2000
4	Read Good	Book Store	Portland	0.8000	2000
5	Fashion Fiesta	Clothing Store	Milan	0.8000	10000
6	Sweet Tooth	Confectionery Store Chain	Paris	0.5000	3000
7	Primo Space	Department Store	London	0.4000	5000
8	Get Moving Motors	Automobile retail	Molsheim	0.2000	100000
10	Mediserv	Pharmacy	Dublin	0.5000	1000
11	Srinu Batla Kottu	Battla Kottu	London	0.2500	200000
12	Abhi"s Den	Wine Shop	Costco	9.0000	29000

Deleting Table:

Delete

Select Tables

Store

Store_ID

12

Delete

Table After Deleting

ID	Store Name	Store Type	Store Location	Max Points Per Bill	No of Years for Expiry
1	Goodness Gems	Jewellery Store	Antwerp	0.2000	20000
2	Tech Assault	Electronics Store	San Francisco	0.7000	10000
3	Sunrise Mart	Grocery Supermarket	Bern	0.4000	2000
4	Read Good	Book Store	Portland	0.8000	2000
5	Fashion Fiesta	Clothing Store	Milan	0.8000	10000
6	Sweet Tooth	Confectionery Store Chain	Paris	0.5000	3000
7	Primo Space	Department Store	London	0.4000	5000
8	Get Moving Motors	Automobile retail	Molsheim	0.2000	100000
10	Mediserv	Pharmacy	Dublin	0.5000	1000
11	Srinu Batla Kottu	Battla Kottu	London	0.2500	200000

Making a purchase:

Purchase

Select a customer to update

(3, 'Robert Shoemake')

Select Customer

Select a store to purchase from

Sweet Tooth

Select Store

Selected store Sweet Tooth

Product Name	Price
macaron	200.0000
Crepes	300.0000

Select the product

Crepes

Enter the quantity

3 − +

Add to cart

`3`

`3`

Bill ID: `84`

Total cost: `1500.0`

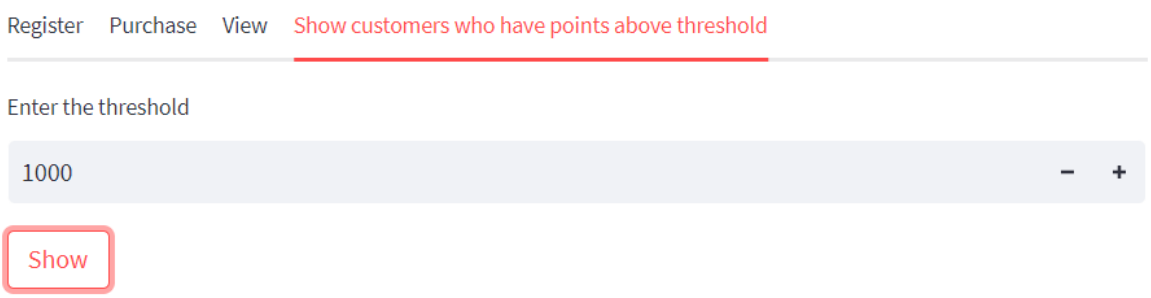
Total tax: `171.0`

Total loyalty points: `135`

d	total_cost	discount	tax	no_of_products_purchased	customer_id	store_id	purchase_dat
4	1,671.0000	0.0000	171.0000	6	3	6	2022-11-29

Modification:

Procedure to view the loyalty Points obtained by a customers with points above a threshold in all the stores



Customer_ID	Points
1	168000
2	1500
4	69000
7	10048
8	40000

Front end function

```
def view_loyalty_points_above_threshold():
    threshold = st.number_input(
        "Enter the threshold", min_value=0, max_value=1000000, value=0)
    if st.button("Show"):
        threshold = cursor.callproc(
            "display_loyalty_points_based_on_threshold", [threshold])
        data = []
        for result in cursor.stored_results():
            data.append(result.fetchall())
        results = data[0]

        if len(results) == 0:
            st.write("No customers")
        else:
            data = [{"Customer_ID":x[0], "Points": x[1]} for x in results]

            df = pd.DataFrame(data)
            hide_dataframe_row_index = """
                <style>
                .row_heading.level0 {display:none}
                .blank {display:none}
                </style>
            """
            st.markdown(hide_dataframe_row_index, unsafe_allow_html=True)
            st.table(df)
```

Procedure:

```
-- Procedure to select the total number of loyalty points obtained by
customers based on threshold on points
DELIMITER ##
create procedure display_loyalty_points_based_on_threshold(threshold float)
begin
    select CUSTOMER_ID, SUM(POINTS) as TOTAL
    from LOYALTY_POINTS NATURAL JOIN CUSTOMER
    group by CUSTOMER_ID
    HAVING SUM(POINTS)> threshold;
end;
##
DELIMITER ;
```