DBMS PROJECT: GrocerEASE

Scope of Project:

In this modern and developing world, we aim to build an interface where people can get their grocery needs fulfilled. There may be several applications already present today but we aspire to build a system located locally in several regions so as to provide services to stakeholders quickly and efficiently. To achieve this type of efficiency our management consists of several departments which store products in different regions so as to minimize the time and effort for customers. We also support several delivery departments aiming for the same goal. There are many other features that will be highlighted further in our project.

Stakeholders :

1. User/Customer
2. Supplier
3. Owner
4. Employees
    a. Area Dept/ Shipment Dept
    b. Delivery Person
    c. Customer Care

Queries for Stakeholders :

- **Customer**
    a. Adds item into CART
    b. Place Order

   c. Review/Rating Product
   d. SignUp/Login
   e. Contact Customer care
   f. Searching product
   g. Payment Order
   h. View Delivery
   i. Return/Refund

- **Delivery Person**
   a. Contacts User/Customer
   b. Pickup Order (Return)
   c. Delivery
   d. Contact Area Manager
- **Supplier**
   a. Supplies Products
   b. Contact Area Manager
- **Customer Care**
   a. Connects with Customer
   b. Contact Area Manager
   c. View/Track Customer's Order
- **Owner**
   a. View Statistics and Reviews.
   b. Contact Area Manager
- **Area Manager**
   a. Contact Delivery Person
   b. Contact Supplier
   c. Contact Owner (Email only)
   d. Pays Supplier
   e. Place Order for Supplier
   f. Hier Supplier

Entity and Attribute

- **User**
  1. Name
     a. First name
     b. Last name
  2. Customer ID (Primary Key)
  3. Gender
  4. Email
  5. {Phone no}
  6. Address
     a. Street no
     b. City/village
     c. State
     d. PinCode

- **Product**
  a. Product ID (Primary Key)
  b. Name
  c. Availability/quantity
  d. Price
  e. Description
  f. Category
  g. MGF / Expiry date
  h. Supplier details
  i. Image (Media File)
  j. Discount Percentage

- **Supplier**
  a. Supplier id (Primary Key)
  b. Name
  c. {Phone no}
  d. Address
     i. Street no

        ii. City/village
        iii. State
        iv. PinCode
    e. {Product id}
    f. GST Number

- **Customer care**
    a. Customer ID (Primary Key)
    b. Name
        i. First name
        ii. Last name
    c. Gender
    d. DOB
    e. Email
    f. {Phone no}
    g. Address
        i. Street no
        ii. City/village
        iii. State
        iv. PinCode
- **Area department**
    a. Manager ID (Primary Key)
    b. Area Name
    c. {Phone no}
    d. Address
        i. Street no
        ii. City/village
        iii. State
        iv. PinCode
    e. Email
    f. Area managed
- **Delivery Dept**
    a. Employee ID(PRIMARY KEY)
    b. Dept name
    c. {Phone no}

- d. Address
  - i. Street no
  - ii. City/village
  - iii. State
  - iv. PinCode
- e. Email
- f. Area for Supplying

- **CartProductPool**
  - a. {Product ID (Foreign Key)}
  - b. Customer ID (Foreign Key)
  - c. Quantity
- **PreviousCartPool**
  - a. {Product ID (Foreign Key)}
  - b. Customer ID (Foreign Key)
  - c. Quantity
  - d. Date and time

Relationship

- User/Customer ***orders*** CartPool(User ID, Qty,Receipt,date and time, Amount,Order Id, Payment Mode)
- User/Customer **Delivery** Delivery Dept(Order Id, Dstatus, Delivery ID,Date and time,Delivery Dept id)
- User/Customer ***adds*** Cart (Product ID, Cart ID)
- CartPool **removes** PreviousCartPool
- User/Customer ***reviews*** Product (Product ID, Rating, Review,User ID)
- User/Customer ***Returnorder*** order (Order Id, Return Id, Date and time, amount,User ID)
- DeliveryDept **ReturnDelivery** Returns (Returns Id, Date and time, Rstatus, ReturnDel, Delivery Dept ID)
- Supplier ***wharehouseorder*** Area Dept,Product (Supplier ID, Area Dept ID, Product ID, Quantity, Price, Supplies ID,Date and time)

- PreviousCartPool **fillpreviouscart** CartproductPool
- User/Customer ***UCcontacts*** Customer Care (Status(resolved or not), Feedback, User ID, Customer Care ID,Date and time,Connect id)
- Areadept ***ACcontacts*** Supplier (Status(resolved or not), Area manager ID, Supplier ID, Date and time,Connect id)
- Deliverydept ***DUcontacts*** User (User ID, Status(Resolved or not),Date and time, Delivery ID,Connect ID)
- Deliverydept ***DAcontacts*** Area Dept(Status(resolved or not), Delivery ID, Area Dept ID,Connect ID)

Weak Entities:

Cartproductpool and **Previouscartpool** because they both depend on the user and if the user gets removed or deregistered then **Cartproductpool** and **Previouscartpool** also don't exist. (Also **Cartproductpool** and **Previouscartpool** don't have primary key)

TERNARY Relationship:

We have an **adds** relationship that is based upon **Cartproductpool** and **Products** and user since a user will search the Product he/she wants which is present in **Products** Entity and then insert them into **Cartproductpool** to order them. Therefore, it depends upon all 3 **Customer**, **Products**, **Cartproductpool** Hence it is a **Ternary** relationship.

Another **Ternary** relationship is **Warehouseorders** where the Area **department** orders products based upon Product availability from suppliers.

RELATIONAL SCHEMAS:

Customer (**Id**, Fname, Lname, Gender, Email, Houseno, CityorVillage, State, Pincode)

Phone_pool_customer (**Id, Phone_no**)

Products (**Id**, Pname, Available_Q, Price , Prescription, Category, Exp, Discount, Image)

Supplier (**Id**, Sname, Gstno, Scono, CityorVillage, State, Pincode)

Phone_pool_supplier (**Id, Phone_no**)

Customercare (**Id**, Fname, Lname, Gender, Email, Houseno, CityorVillage, State, Pincode)

Phone_no_pool_customercare (**Id, Phone_no**)

Areadept (**Id**, Deptname, Email, Scono, CityorVillage, State, Pincode, Area)

Warehouseorder (**Id**, Dnt, Sid, Aid, Pid,Quantity, Price)

Phone_no_pool_areadept (**Id,phone_no**)

Deliverydept (**Id**, Dname, Email, Scono, CityorVillage, State, Pincode)

Phone_no_pool_deliverydept (**Id,Phone_no**)

Previouscartpool (**Uid**, Pid, Qty, **Dnt**)

Cartproductpool (**Uid**, Pid,Qty)

Orders (**Id**, Dnt, Uid, Amount, Pmode, Recepit, Qty)

Orderdelivery (**Oid**, Did, Dstatus, Dnt)

Orderreturn (**<u>Id</u>**, Oid, Uid, Dnt, Refundamount)

Returndelivery (**<u>Id,</u>** Did, Rid, Dnt, Rstatus)

Supplierproductpool (**<u>Sid, Pid</u>**)

UCcontacts (**<u>Connectid</u>**, Qstatus, Ccid, Uid, Dnt, Feedback)

Reviews (Rating, Reviews, **<u>Pid, Uid</u>**)

SAcontact (**<u>Connectid</u>**, Qstatus, Sid, Aid, Dnt)

DAcontacts (**<u>Connectid</u>**, Qstatus, Did, Aid)

DUcontacts (**<u>Connectid</u>**, Qstatus, Did, Uid, Dnt)

SQL Queries :

- **List details of juice which cost more than 110**

```
select * from products
where substring_index(category,',',1) = 'juice'
and price > 110;
```

- **List top x Customers who bought most products**

```
select uid, count(pid)
from previouscartpool
group by uid
order by count(pid) limit  {x};
```

- **Calculate total quantity sold for a product x after a given day y (where x is id of product and y is date in YYYY-MM-DD format)**

```
select cast(DNT as date) uid, sum(qty)
from previouscartpool
```

```
where pid = {x} and DNT > {y};
```

- **List all unsuccessful and unsatisfied customers with customer care in year 2022**

```
select *, cast(DNT as Date)
from uccontacts
where qstatus = 0 and DNT > 2022-01-01;
```

- **List Details of Product whose rating is more than 4 and less than 10 products are available.**

```
select *
from products
where available_q < 10
and id in (select pid from reviews where rating > 4);
```

- **List all customers id with their total expenditure more than {X} amount.**

```
select uid, total_amount
from (select uid, sum(amount) as total_amount
from orders
group by uid) as xyz
where total_amount > {X};
```

- **Calculate AVG rating of each product according to reviews/rating given by customers.**

```
Create view avgrating as select pid, avg(rating) as
avg_rating from reviews group by pid;
select * from products,avgrating
where avgrating.pid = products.id;
```

- **List details of all inactive customers (customers who haven't bought anything till now)**

```
select * from customer
where id not in (select distinct uid from orders);
```

- **Calculate the total amount of sales in a particular state {X}**

```
select sum(amount)
from orders where uid in( select id from customer
where state = "{X}");
```

- **List top 10 suppliers who supplies most product**

```
select sid,count(*)
from supplierproductpool
group by sid limit 10;
```

- **Calculate total number of customer who give reviews and find average rating for all product from a region**

```
select count(distinct uid), avg(rating)
from reviews
where uid in (select id from customer
where state = "{X}");
```

Data Populations:

The main task was to generate a database which we have made from scratch by ourselves(A big part of data) as the data available on the internet was not fulfilling our needs. We have chosen the cardinality as per the need. Tables have cardinality varying from **20 - 500**. For example we have 20 suppliers and 500+ orders in our data. We formed our data and converted that into a **.csv** file for our convenience. Then the task was to populate the data in our database for that we used the **import wizard** facility available in the **mySQL workbench**. Then most of the work was already done. After that we gave it a final check to ensure the accuracy.

Views and Grants:

- **Creates a view for Customer to show only limited details of customer care employee.**

```
CREATE VIEW
        customercare_view
AS SELECT
        customercare.ID,
        Fname,
        Lname,
        Email,
        phone_no_pool_coustomercare.Phone_no
FROM
        customercare
JOIN
        phone_no_pool_coustomercare
ON
        customercare.ID = phone_no_pool_coustomercare.id;

create user {user}@localhost identified by 'password';
GRANT all on customercare_view to '{user}'@'localhost';
```

- **Create a view of all Favorite products for a Customer.**

```
CREATE VIEW
        favorite_products as
(SELECT
       p3.id,
       p3.pname,
       p3.Available_Q,
       p3.price,
       p3.pdescription,
       p3.Category,
       p3.EXP,
       p3.Discount,
       p3.image
   FROM
```

```
        (SELECT
            *,
            count(*) AS cnt,
            (count(*) * sum(qty)) AS num
        FROM
            previouscartpool AS p1,
            products AS p2
        WHERE
            p1.uid = 1
            AND p1.pid = p2.id
        GROUP BY
            p1.pid
        ORDER BY
            num DESC LIMIT 3) AS p3
    ORDER BY
        p3.id
);

create user {user}@localhost identified by 'password';
GRANT all on favorite_products to '{user}'@'localhost';
```

- **Creates a view for Customer to show only limited details of delivery dept employee.**

```
CREATE VIEW
        deliverydept_view as
SELECT
        deliverydept.ID,
        Dname,
        Email,
        phone_no_pool_deliverdept.Phone_no
FROM
        deliverydept
JOIN
        phone_no_pool_deliverdept
ON
        deliverydept.ID = phone_no_pool_deliverdept.id;
```

```
create user {user}@localhost identified by 'password';
GRANT all on deliverydept_view to '{user}'@'localhost';
```

Queries(new):

- **Find all my Orders where the Payment mode is ‘Online Wallet’**

```
SELECT
        *
FROM
        orders
WHERE
        uid = {x} and pmode = “Online wallet”;
```

- **List all vegetables or fruits which have up to 25% discount.**

```
SELECT
        *
FROM
        products
WHERE
        Category Like ”%Fruits” and Discount <= 25;
```

- **List all Product bought together in an order**

```
SELECT
        pid
FROM
        previouscartpool
WHERE
        Uid= {x} and DNT = {Y}
```

- **Bestsellers of every category. (Fruits, Vegetables, Package, Dairy)**

```
        p3.Discount,
        p3.image
    FROM
```

```
            (SELECT
              *,
              count(*) AS cnt,
              SUBSTRING_INDEX(category,
              ',',
              -1) AS cat,
              sum(qty),
              (count(*) * sum(qty)) AS num
            FROM
              previouscartpool AS p1,
              products AS p2
            WHERE
              p1.pid = p2.id
            GROUP BY
              p1.pid
            ORDER BY
              num DESC) AS p3
          GROUP BY
            p3.cat
          ORDER BY
            NULL
```

    ○

- **Change details of the Customer(maybe Email). EMBEDDED**

```
      UPDATE
            customer
      SET
            Email = {email}
      WHERE
            uid = {x}
       SELECT
            p3.id,
            p3.pname,
            p3.Available_Q,
            p3.price,
                p3.pdescription,
                p3.Category,
                p3.EXP,
```

- **Calculate my total Expenditure in App (plus total product bought) till date**

```
    SELECT
        sum(amount),
        sum(qty)
    FROM
        orders
    WHERE
        uid = {X}
```

- **Calculate the Amount require to Avail Free Delivery Cupon. EMBEDDED**

```
    SELECT
        sum((price*qty)) as num
    FROM
        products as p1,
        (select
            pid,
            qty
            FROM
                cartproductpool
            WHERE
                uid = {x}) AS p2
    WHERE
        p1.id=p2.pid;
```

- **List Product according to Current time. EMBEDDED**

```
    SELECT
        *
    FROM
        `products`
```

```
WHERE
      category Like {X} {according to time}
```

- **List suggested Products. EMBEDDED**

```
SELECT
      *
   FROM
      products
   WHERE
      SUBSTRING_INDEX(products.Category, ',', -1) IN (
         SELECT
            DISTINCT
SUBSTRING_INDEX(favorite_products.Category,
            ',',
            -1)
         FROM
            favorite_products
      )
   ORDER BY
      products.price DESC
```

- **List My favorites order (Home page)**

```
create view favorite_products as (SELECT
      p3.id,
      p3.pname,
      p3.Available_Q,
      p3.price,
      p3.pdescription,
      p3.Category,
      p3.EXP,
      p3.Discount,
      p3.image
   FROM
      (SELECT
                *,
                count(*) AS cnt,
                (count(*) * sum(qty)) AS num
             FROM
                previouscartpool AS p1,
                products AS p2
             WHERE
                p1.uid = 1
                AND p1.pid = p2.id
             GROUP BY
                p1.pid
             ORDER BY
                num DESC LIMIT 3) AS p3
          ORDER BY
             p3.id
       );
       Select * from favorite_products;
```

- **Add to Cart and Order Cart EMBEDDED**

```
SELECT
        sum((price*qty)),
        sum(qty) as num
FROM
        products as p1 ,
        (SELECT
                pid,
                qty
                FROM
                        cartproductpool
                WHERE
                        uid = {X}) AS p2
WHERE
        p1.id=p2.pid;
```

```
INSERT INTO orders
        (DNT, UID, AMOUNT, PMODE, RECEPIT, QTY)
VALUES
        (DNT, UID, AMOUNT, PMODE, RECEPIT, QTY);
DELETE FROM
        cartproductpool
WHERE
        uid = {x}
```

Indexing:

- **Bestsellers of every category. (Fruits, Vegetables, Package, Dairy)**
```
ALTER TABLE previouscartpool ADD INDEX
previouscartpool_idx_pid (pid);
ALTER TABLE products ADD INDEX products_idx_id (id);
```
- **List My favorites order (Home page)**
```
ALTER TABLE previouscartpool ADD INDEX
previouscartpool_idx_uid_pid (uid,pid);
ALTER TABLE products ADD INDEX products_idx_id (id);
```

- **List suggested Products**
```
ALTER TABLE `products` ADD INDEX products_idx_price
(price);
```
- **List all vegetables or fruits which have up to 25% discount.**
```
ALTER TABLE products ADD INDEX
products_idx_discount (Discount);
```

Triggers:

- **Inserting product into previouscartpool when products are deleted from cartproductpool**
```
delimiter $$
CREATE TRIGGER
```

```
        insert_product
    BEFORE DELETE ON
        cartproductpool
    FOR EACH ROW
    BEGIN
        INSERT INTO
            previouscartpool
        VALUES
            (OLD.Uid, old.pid, old.QTY, NOW());
    END; $$
```

- **Incrementing Number of Reviews as soon as new Review is added**

```
    delimiter $$
    CREATE TRIGGER
        Count_reviews
    AFTER INSERT ON
        reviews
    FOR EACH ROW
    BEGIN
        Declare count int default 0;
        SET count = count + 1;
    END; $$
```

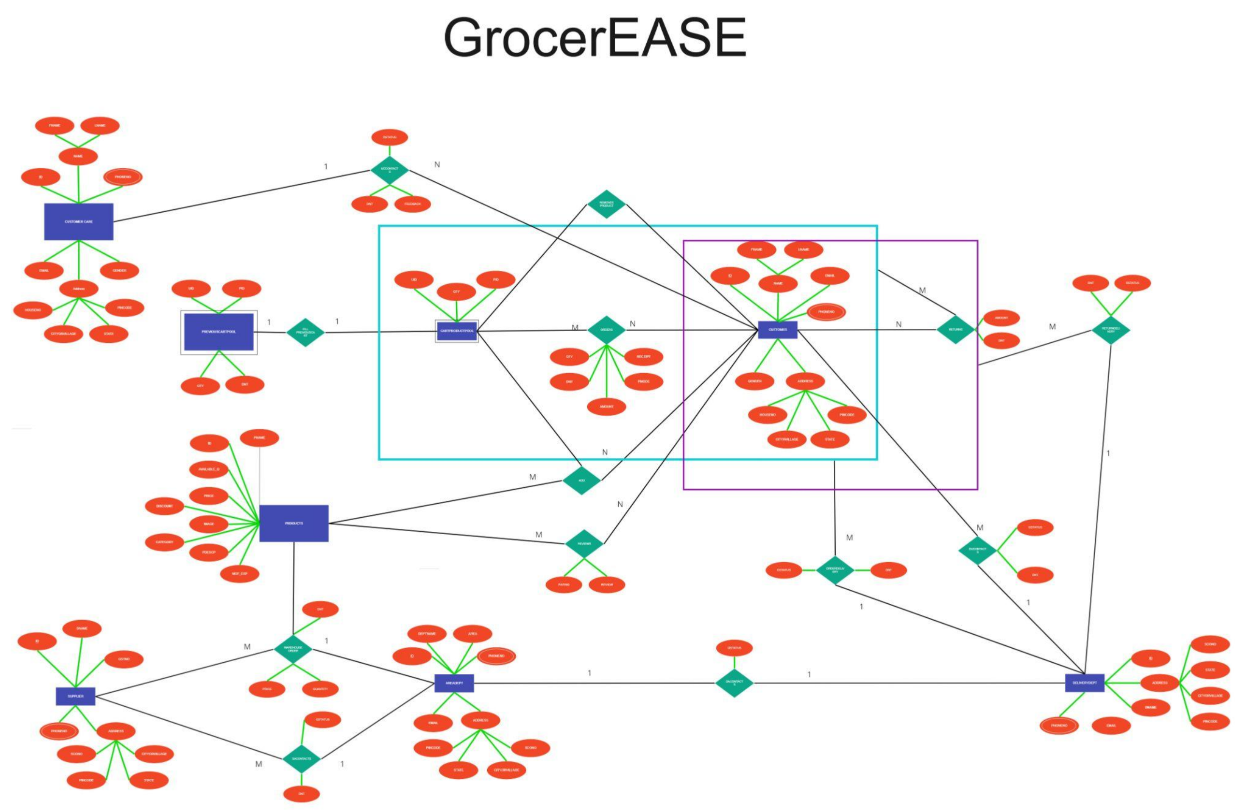
- **De-Registeration of an Area dept**

```
    delimiter $$
    CREATE TRIGGER
        delete_areadept
    BEFORE DELETE ON
        areadept
    FOR EACH ROW
    BEGIN
        DELETE FROM
            warehouseorder
        WHERE
            warehouseorder.AID = OLD.ID;
        DELETE FROM
            SAcontact
        WHERE
```

```
                SAcontact.AID = OLD.ID;
        DELETE FROM
                DAcontacts
        WHERE
                DAcontacts.AID = OLD.ID;
        DELETE FROM
                phone_no_pool_areadept
        WHERE
                phone_no_pool_areadept.ID = OLD.ID;
END; $$
```

Entity Relationship Diagram:

(PDF Also attached)



ER: https://miro.com/app/board/uXjVOJJR22g=/

References:

https://github.com/marcusklasson/GroceryStoreDataset (Product images)
https://www.mockaroo.com/schemas/389606 (For datageneration)