Customer Loyalty Marketplace Application

CSC 540 Database Management concepts and Systems Project Report

Team 31 Members:

Blake Patterson (bpatter2)

Vinay Kumar Reddy Perolla (vperoll)

Bapiraju Vamsi Tadikonda (btadiko)

Bharath Katabathuni (bkataba)

Problem Statement:

We have to develop a database system that manages loyalty programs for different brands. This program controls various operations related to loyalty programs, brands, wallets, etc which are related to each other within the database.

Admin can access the complete data in the database while the brands can access the data of the customers of a particular brand's customers. Within the particular brand, the customer profile is built based on the transactions in the brand and rewards will be assigned accordingly based on a few assigned rules in the database. Through these reward points, the customer will be able to redeem points to avail rewards.

Intended Classes of Users:

- **Customer - Customers are able to view their transaction history as well as their rewards history.**
- **Administrator (Admin) - Administrator has access to the entire database.**
- **Brands - have access to few loyalty programs within the system**

All Entities and design of Schemas:

1. **UserType**
```
create table userTypes
(
 uTypeID  integer GENERATED BY DEFAULT ON NULL AS IDENTITY,
 userType varchar(20) NOT NULL,
 constraint pk_userTypes primary key (uTypeID)
);
```

    Primary Key: uTypeID
    Foreign Keys: None
    Other Constraints:
    - **not null:** userType
    - unique: uTypeID

2. **loginUser**
```
create table loginUser
```

```
(
 username varchar(45) NOT NULL,
 passwd varchar(45) NOT NULL,
 uTypeID integer NOT NULL,
 userID varchar(45) NOT NULL,
 constraint unique_loginUser unique(username),
 constraint pk_loginUser primary key (userID),
 constraint fk_loginUser foreign key(uTypeID) references userTypes(uTypeID)
);
```

Primary Key: userID
Foreign Keys: uTypeID
Other Constraints:

- **not null:** username, passwd, uTypeID, userID
- unique: username

3. **Brands**:

```
create table brands
(
 bname     varchar(45) NOT NULL ,
 address  varchar(90),
 joinDate date NOT NULL ,
 brandID  varchar(45) NOT NULL ,
 constraint pk_brands primary key (brandID),
 constraint fk_brands foreign key(brandID) references loginUser(userID)
);
```

Primary Key: brandID
Foreign Keys: userID
Other Constraints:

- **not null:** bname, joinDate, brandID

4. **Customers**:

```
create table customers
(
 address  varchar(90),
 cname     varchar(45) NOT NULL ,
 phoneno  number(15),
 walletID integer GENERATED BY DEFAULT ON NULL AS IDENTITY,
 customerID varchar(45) NOT NULL,
 constraint unique_customers unique (walletID),
 constraint pk_customers primary key (customerID),
```

 constraint fk_customers foreign key(customerID) references loginUser(userID)
);

Primary Key: customerID
Foreign Keys: userID
Other Constraints:

- **not null:** cname, customerID
- unique: walletID

5. **LoyaltyProgram**:
create table loyaltyprogram(
 pstate number(1,0) DEFAULT 0,
 lcode varchar(45) NOT NULL,
 brandID varchar(45) NOT NULL,
 lname varchar(45) NOT NULL,
 IsTier number(1,0) DEFAULT 0,
 constraint pk_lprogram primary key(brandID),
 constraint fk_lprogram foreign key(brandID) references brands(brandID) ON DELETE CASCADE
);

Primary Key: brandID
Foreign Keys: brandID
Other Constraints:

- **not null:** lcode, brandID, lname
- pstate can be either 0 or 1

6. **TierProgram**:
create table tierprogram(
 brandID varchar(45) NOT NULL,
 levelno integer NOT NULL,
 multiplier integer NOT NULL,
 pointsReq integer NOT NULL,
 lname varchar(45) NOT NULL,
 constraint unique_tierprogram unique(brandID,levelno),
 constraint pk_tierprogram primary key(brandID,lname),
 constraint fk_tierprogram foreign key(brandID) references loyaltyprogram(brandID) ON DELETE CASCADE
);

Primary Key: brandID,levelno
Foreign Keys: brandID
Other Constraints:

- **not null:** brandID,levelno, multiplier, pointsReq, lname

        - Unique : brandID,levelno

7. **RewardTypes**:

```
create table rewardTypes
(
 rTypeID integer NOT NULL,
 rTypeName varchar(45) NOT NULL,
 constraint pk_rewardTypes primary key (rTypeID)
);
```

    Primary Key: rTypeID
    Foreign Keys: None
    Other Constraints:
        - **not null:** rTypeID, rTypeName

8. **Rewards**:

```
create table rewards
(
 brandID  varchar(45) NOT NULL,
 rewardID varchar(45) NOT NULL,
 rTypeID  integer NOT NULL,
 quantity integer NOT NULL,
 amount   integer,
 constraint pf_rewards primary key(brandID, rewardID),
 constraint fk_rewards_rtype foreign key(rTypeID) references rewardTypes(rTypeID),
 constraint fk_rewards_brand foreign key(brandID) references loyaltyprogram(brandID)
ON DELETE CASCADE
);
```

    Primary Key: brandID, rewardID
    Foreign Keys: rTypeID, brandID
    Other Constraints:
        - **not null:** brandID, rewardID,rTypeID, quantity

9. **ActivityTypes**:

```
create table activityTypes
(
 aTypeID integer NOT NULL,
 aTypeName varchar(20) NOT NULL,
 constraint pk_activityTypes primary key (aTypeID)
);
```

    Primary Key: aTypeID
    Foreign Keys: None
    Other Constraints:
        - **not null:** aTypeID, aTypeName

10. **Activities**:

```
create table activities
create table activities
(
 brandID  varchar(45) NOT NULL ,
 activityID varchar(45) NOT NULL,
 aTypeID integer NOT NULL,
 constraint pf_activities primary key(brandID, activityID),
 constraint fk_activities foreign key(aTypeID) references activityTypes(aTypeID),
 constraint fk_activities_brands foreign key(brandID) references loyaltyprogram(brandID)
ON DELETE CASCADE
);
```

Primary Key: brandID, activityID
Foreign Keys: aTypeID, brandID
Other Constraints:

- **not null:** brandID, activityID, aTypeID

11. **RERules**:

```
CREATE TABLE RERules
(
 rule_code integer NOT NULL ,
 versionno integer NOT NULL ,
 activityID varchar(45) NOT NULL ,
 brandID   varchar(45) NOT NULL ,
 points    integer NOT NULL ,
 constraint pk_RERules primary key(rule_code, versionno, brandID),
 constraint fk_RERules foreign key(activityID,brandID) references
activities(activityID,brandID) ON DELETE CASCADE
);
```

Primary Key: rule_code, versionno, brandID
Foreign Keys: activityID,brandID
Other Constraints:

- **not null:** rule_code, versionno, brandID, activityID, points

12. **RRRules**:

```
CREATE TABLE RRRules
(
 rule_code integer NOT NULL ,
 versionno   integer NOT NULL ,
 rewardID  varchar(45) NOT NULL ,
 brandID   varchar(45) NOT NULL ,
 points    integer NOT NULL ,
 constraint pk_RRRules primary key(rule_code, versionno, brandID),
 constraint fk_RRRules_rewards foreign key(rewardID,brandID) references
rewards(rewardID,brandID) ON DELETE CASCADE
);
```

Primary Key: rule_code, versionno, brandID
Foreign Keys: rewardID,brandID
Other Constraints:

- **not null:** rule_code, versionno, brandID, rewardID, points

13. **Wallets**:

```
CREATE TABLE wallets
(
 transID integer GENERATED BY DEFAULT ON NULL AS IDENTITY,
 brandID varchar(45) NOT NULL,
 customerID varchar(45) NOT NULL,
 activityCode varchar(45) NOT NULL,
 activityDate TIMESTAMP NOT NULL,
 activityType varchar(45) NOT NULL,
 activityPoints integer NOT NULL,
 /*   ACTIVITY_NAME VARCHAR2(20) NOT NULL,*/
 constraint check_performActivity_activity_code CHECK (activityType in ('RE','RR')),
 constraint fk_performActivity_customer foreign key(customerID) references
customers(customerID) ON DELETE CASCADE,
 constraint fk_performActivity_lprogram foreign key(brandID) references
loyaltyprogram(brandID) ON DELETE SET NULL
);
```

Primary Key: None
Foreign Keys: customerID, brandID
Other Constraints:

- **not null:** brandID, customerID, activityCode, activityDate, activityType, activityPoints
- Unique : transID
- CHECK activityType in ('RE','RR')

14. CustomerTierStatus

```
CREATE TABLE CustomerTierStatus
(
 brandID varchar(45) NOT NULL,
 customerID varchar(45) NOT NULL,
 totalPoints integer DEFAULT 0,
 balancePoints integer DEFAULT 0,
 TierStatus varchar(45) NOT NULL,
 constraint pk_CustomerTierStatus primary key(brandID,customerID),
 constraint fk_CustomerTierStatus_customer foreign key(customerID) references
customers(customerID) ON DELETE CASCADE,
 constraint fk_CustomerTierStatus_tierprogram foreign key(brandID,TierStatus)
references tierprogram(brandID,lname) ON DELETE CASCADE
);
```

Primary Key: brandID,customerID
Foreign Keys: customerID, brandID,lname
Other Constraints:

- **not null:** brandID, customerID, TierStatus

15. Claims

```
CREATE TABLE Claims
(
 claimID integer GENERATED BY DEFAULT ON NULL AS IDENTITY,
 brandID varchar(45) NOT NULL,
 rewardID varchar(45) NOT NULL,
 customerID varchar(45) NOT NULL,
 claimed number(1,0) DEFAULT 0,
 expiry_date date,
 constraint fk_Claims_customer foreign key(customerID) references
customers(customerID) ON DELETE SET NULL,
 constraint fk_Claims_rewards foreign key(brandID,rewardID) references
rewards(brandID,rewardID) ON DELETE CASCADE
);
```

Foreign Keys: customerID, brandID,rewardID
Other Constraints:

- **not null:** brandID, rewardID, customerID

Relational Schemas:

A. userTypes: (**userType, <u>uTypeID</u>**)
B. loginUser: (**passwd, username,<u>userID</u>, <u>uTypeID</u>**)
C. Brands: (**bname, address, joinDate, <u>brandID</u>, <u>userID</u>**)
D. customers:(**address, cname, phoneno,walletID, <u>customerID</u>, <u>userID</u>**)
E. loyaltyprogram:(**pstate, lcode,lname,lsTier, <u>brandID</u>**)
F. tierprogram:(**<u>brandID, lname,</u>levelno,multiplier,pointsReq**)
G. rewardTypes:(**rTypeName, <u>rTypeID</u>**)
H. rewards:(<u>brandID</u>, <u>rewardID</u>,<u>rTypeID</u>,quantity,amount)
I. activityTypes: (**aTypeName , <u>aTypeID</u>**)
J. activities:(**<u>brandID</u>, <u>activityID</u>, <u>aTypeID</u>**)

K. RERules:(**points, rule_code, versionno, activityID, brandID**)
L. RRRules:(**points, rule_code, versionno, rewardID, brandID)**
M. Wallets: (transID,**customerID,brandID, activityCode,activityDate,activityType, activityPoints**)
N. CustomerTierStatus: (brandID,TierStatus,customerID, totalPoints,balancePoints)
O. Claims:(claimID,claimed,expiry_date,**customerID,brandID,rewardID**)

Documentation of Relational Schemas:

- **EntitySets to Relations :**

  we have used the exact attributed as mentioned in the ER diagram

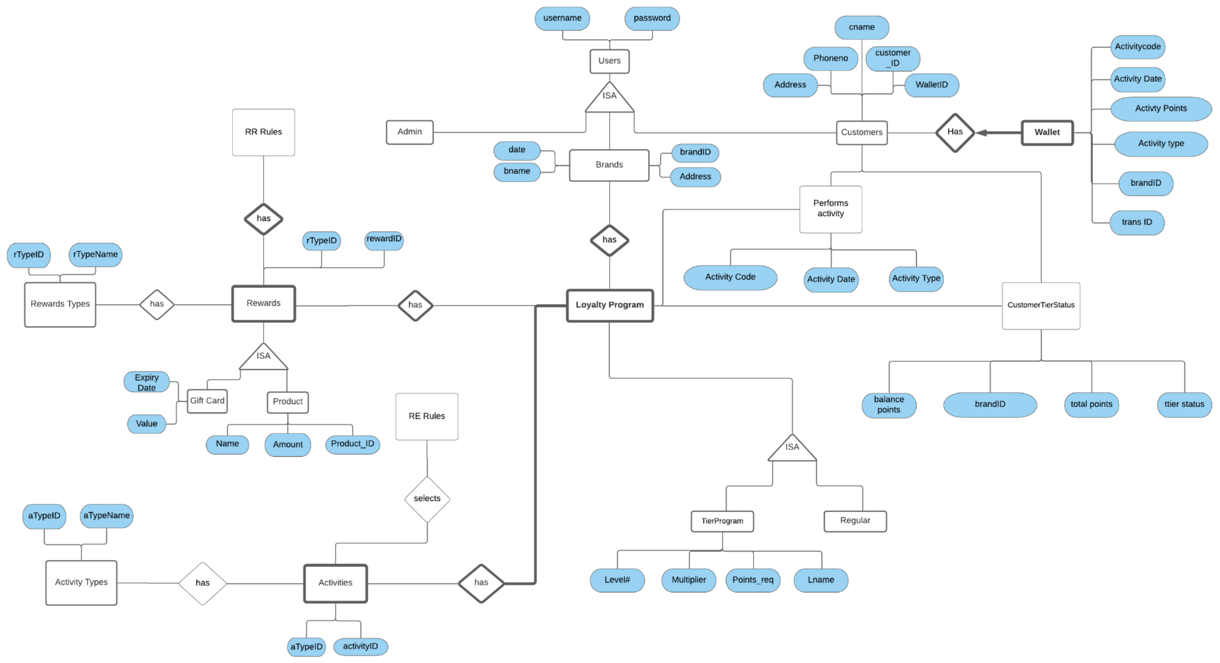
- **Relationship to Schemas:**

  few relations like login, enroll and others are converted into relations

- **Subclass entity set to relations:**

  keys of the subclass have keys of the loyalty programs sets entities

E/R Diagram:



Documentation of ER Model:

1. Here the loyalty program has brands, customer, rewards, activities
2. Admin handles all the users, brands, customer data and all the other functionality
3. Customers can enroll in any number of brands at the same time but can perform only one activity at a time.
4. Each customer has his unique wallet for a particular brand which has reward points earned and transaction history
5. Each customer can view the wallet and can redeem the points
6. Each brand has unique login application
7. Each loyalty program has RR and RE rules with them
8. RR,RR has rules which enables the activity
9. Each activity has a name and activity performed by the customer

Triggers:

1. **Tg_add_user_id : This is for generating user id**

```
create or replace trigger tg_add_user_id BEFORE INSERT ON loginUser
  FOR EACH ROW
  BEGIN
    IF :NEW.uTypeID = 1 THEN
      SELECT 'A' || lpad(sq_usertypes_admin_id.NEXTVAL, 4, '0') INTO :NEW.userID FROM dual;
    ELSIF :NEW.uTypeID = 2 THEN
      SELECT 'B' || lpad(sq_usertypes_brand_id.NEXTVAL, 4, '0') INTO :NEW.userID FROM dual;
```

```
        ELSE
            SELECT 'C' || lpad(sq_usertypes_customer_id.NEXTVAL, 4, '0') INTO :NEW.userID FROM dual;
        END IF;
    END;
/
```

2. InitializeCustomerStatus: This is for default status after customer enrolling

```
CREATE OR REPLACE TRIGGER InitializeCustomerStatus BEFORE INSERT ON CustomerTierStatus
FOR EACH ROW
DECLARE
    tier_name VARCHAR2(10) default null;
    is_tiered NUMBER(1,0);
    is_valid NUMBER(1,0) default 0;
BEGIN
    SELECT L.pstate INTO is_valid FROM loyaltyprogram L WHERE L.brandID = :new.brandID;
    IF(is_valid = 0) THEN
        RAISE_APPLICATION_ERROR(-20021,'Can not enroll to this loyalty program as it is not
validated',False);
    END IF;
    IF (:new.TierStatus IS NULL) THEN
        SELECT L.IsTier INTO is_tiered FROM loyaltyprogram L WHERE L.brandID = :new.brandID;
        IF(is_tiered = 1) THEN
            SELECT T.lname INTO tier_name FROM tierprogram T WHERE T.brandID = :new.brandID and
T.pointsReq = 0;
        END IF;
    END IF;
    SELECT tier_name INTO :new.TierStatus FROM dual;
END;
/
```

3. UpdateCustomerStatusInLoyaltyProgram : This is For updating customer status

```
CREATE OR REPLACE TRIGGER UpdateCustomerStatusInLoyaltyProgram AFTER INSERT ON wallets
FOR EACH ROW
DECLARE
    current_level VARCHAR(45) default null;
    totalPoints integer;
BEGIN
    IF(:new.activityType='RE') THEN
        SELECT CTS.TierStatus, CTS.totalPoints INTO current_level, totalPoints FROM CustomerTierStatus
CTS WHERE CTS.customerID = :new.customerID and CTS.brandID = :new.brandID;
        totalPoints := totalPoints + :new.activityPoints;
        --dbms_output.Put_line(totalPoints); --display
        IF(current_level IS NOT NULL) THEN
            select g.lname into current_level from (select * from tierprogram t where t.brandID = :new.brandID
and t.lname is not null and t.pointsReq <= totalPoints ORDER BY t.pointsReq DESC) g where rownum = 1;
        END IF;
        --dbms_output.Put_line(totalPoints); --display
        --dbms_output.Put_line(current_level); --display
        update CustomerTierStatus CTS set CTS.TIERSTATUS = current_level,CTS.balancePoints =
CTS.balancePoints+ :new.activityPoints,CTS.TOTALPOINTS = CTS.totalPoints+ :new.activityPoints where
CTS.customerID = :new.customerID and CTS.brandID = :new.brandID;
        --dbms_output.Put_line(totalPoints); --display
        --dbms_output.Put_line(current_level); --display
```

```
   ELSIF(:new.activityType='RR') THEN
      update CustomerTierStatus CTS set CTS.balancePoints = CTS.balancePoints-:new.activityPoints
where CTS.customerID = :new.customerID and CTS.brandID = :new.brandID;
   END IF;
END;
/
```

4. AddPointsForWalletActivity : This is For updating the wallets points

```
CREATE OR REPLACE TRIGGER AddPointsForWalletActivity BEFORE INSERT ON wallets FOR EACH
ROW
DECLARE
   balance_points NUMBER(10) default 0;
   qty NUMBER(10) default 0;
   point_req NUMBER(10) default 0;
   multiplier INT default 1;
   is_tiered NUMBER(1,0);
   tier_name VARCHAR2(10) default null;
   expiry_date date;
BEGIN
   IF (:new.activityType='RE') THEN
      select r.points into point_req from RErules r INNER JOIN (select brandID,rule_code,max(versionno) as
vv from RErules group by brandID,rule_code) b ON r.brandID = b.brandID AND r.rule_code = b.rule_code
AND r.versionno = b.vv
      where r.BrandID =:new.BrandID and r.activityID = :new.activityCode;
      SELECT L.IsTier INTO is_tiered FROM loyaltyprogram L WHERE L.brandID = :new.brandID;
      IF(is_tiered = 1) THEN
         SELECT CTS.TierStatus INTO tier_name FROM CustomerTierStatus CTS WHERE CTS.customerID
= :new.customerID and CTS.brandID = :new.brandID;
         SELECT T.MULTIPLIER INTO multiplier FROM tierprogram T WHERE T.lname = tier_name;
      END IF;
      dbms_output.Put_line(point_req);
      dbms_output.Put_line(multiplier);
      dbms_output.Put_line(is_tiered);
      dbms_output.Put_line(tier_name); --display
      :new.activityPoints := point_req * multiplier;
   ELSIF (:new.activityType='RR') THEN
      select r.points into point_req from RRrules r INNER JOIN (select brandID,rule_code,max(versionno) as
vv from RRrules group by brandID,rule_code) b ON r.brandID = b.brandID AND r.rule_code = b.rule_code
AND r.versionno = b.vv
      where r.BrandID =:new.BrandID and r.rewardID = :new.activityCode;
      select CTS.TierStatus, CTS.balancePoints INTO tier_name, balance_points FROM
CustomerTierStatus CTS WHERE CTS.customerID = :new.customerID and CTS.brandID = :new.brandID;
      select rwd.quantity into qty from rewards rwd where rwd.brandID = :new.brandID and rwd.rewardID =
:new.activityCode;
      IF(qty <1) THEN
         RAISE_APPLICATION_ERROR(-20020,'Reward can not be redeemed, no more reward instances
available.',False);
         return;
      ELSIF(balance_points < point_req) THEN
        RAISE_APPLICATION_ERROR(-20020,'Reward can not be redeemed, insufficient points to redeem
reward.',False);
         return;
      ELSE
```

```
            SELECT point_req INTO :new.activityPoints FROM dual;
            update rewards rwd set rwd.quantity = rwd.quantity-1 where rwd.brandID = :new.brandID and
rwd.rewardID = :new.activityCode;
            insert into claims values
(DEFAULT,:new.brandID,:new.activityCode,:new.customerID,0,(SYSDATE)+90);
        END IF;
    END IF;
END;
/
```

Functional Dependencies:

1. userTypes: (userType, uTypeID)
   Key => uTypeID
   FDS : { uTypeID => userType }
   Therefore the relation holds true for BCNF.

2. loginUser: (passwd, username,userID, uTypeID)
   Key => username
   FDS : { userID=> passwd, username, uTypeID, userID
   username => userID }
   Therefore the relation holds true for BCNF
3. Brands: (bname, address, joinDate, brandID, userID)
   Key => brandID
   FDS : { uTypeID => bname, address, joinDate, brandID }
   Therefore the relation holds true for BCNF
4. customers:(address, cname, phoneno,walletID, customerID, userID)
   Key => customerID
   Candidate Key => walletID
   FDS : { customerID => address, cname, phoneno,walletID, customerID
   walletID => customerID }
   Therefore the relation holds true for BCNF
5. loyaltyprogram:(pstate, lcode,lname,IsTier, brandID)
   Key => brandID
   Candidate Key => lcode
   FDS : { brandID => pstate, lcode,lname,IsTier, brandID
   lcode => brandID }
   Therefore the relation holds true for BCNF
6. tierprogram:(brandID, lname,levelno,multiplier,pointsReq)
   Key => brandID, lname
   Candidate Key => brandID, levelno
   FDS : { brandID, lname => levelno,multiplier,pointsReq
   brandID, levelno=> brandID, lname }
   Therefore the relation holds true for BCNF
7. rewardTypes:(rTypeName, rTypeID)

Key => rTypeID
FDS : { rTypeID => rTypeName }
Therefore the relation holds true for BCNF

8. rewards:(brandID, rewardID,rTypeID,quantity,amount)
   Key => brandID, rewardID
   FDS : { brandID, rewardID => rTypeID,quantity,amount }
   Therefore the relation holds true for BCNF
9. activityTypes: (aTypeName , aTypeID)
   Key => aTypeID
   FDS : { aTypeID => aTypeName }
   Therefore the relation holds true for BCNF
10. activities:(brandID, activityID, aTypeID)
    Key => brandID, activityID
    FDS : { brandID, activityID => aTypeID }
    Therefore the relation holds true for BCNF
11. RERules:(points, rule_code, versionno, activityID, brandID)
    Key => rule_code, versionno, brandID
    FDS : { rule_code, versionno, brandID => activityID, points }
    Therefore the relation holds true for BCNF
12. RRRules:(points, rule_code, versionno, rewardID, brandID)
    Key => rule_code, versionno, brandID
    FDS : { rule_code, versionno, brandID => activityID, points }
    Therefore the relation holds true for BCNF
13. Wallets: (transID,customerID,brandID, activityCode,activityDate,activityType, activityPoints)
    Key => transID
    FDS : { transID => everything }
    Therefore the relation holds true for BCNF
14. CustomerTierStatus: (brandID,TierStatus,customerID, totalPoints,balancePoints)
    Key => brandID, customerID
    FDS : { brandID, customerID => totalpoints,balancePoints,TierStatus }
    Therefore the relation holds true for BCNF
15. Claims:(claimID,claimed,expiry_date,customerID,brandID,rewardID)
    Key => claimID
    FDS : { uTypeID => everything }
    Therefore the relation holds true for BCNF

Validation Function:

1. Check if activity types exist
2. Check if reward type exists
3. Check if satisfies RE and RR rules

Application side Constraints:

All constraints have been implemented in the DBMS.

Part B Queries

1. List all customers that are not part of Brand02’s program.

```
select * from customers where customerID not in (select distinct customerID from
CustomerTierStatus where brandID='B0002')
```

2. List customers that have joined a loyalty program but have not participated in any activity in that program (list the customerid and the loyalty program id).

```
select lcode, customerID from loyaltyprogram l, (select brandID, customerID from
CustomerTierStatus where totalpoints=0) x
where x.brandID = l.brandID
```

3. List the rewards that are part of Brand01 loyalty program.

```
select rt.rtypename from rewards r, rewardTypes rt
where rt.rtypeID = r.rtypeID and r.brandID='B0001';
```

4. List all the loyalty programs that include “refer a friend” as an activity in at least one of their reward rules.

```
select lcode, lname from loyaltyprogram where
brandID in (select brandID from activities where atypeID = (select atypeId from activitytypes
where atypename = 'refer a friend'))
```

5. For Brand01, list for each activity type in their loyalty program, the number instances that have occurred.

```
select brandID, activityCode, Count(*) from wallets
where brandID = 'B0001'
group by brandID, activityCode
```

6. List customers of Brand01 that have redeemed at least twice.

```
select customerID from wallets
where brandID='B0001' and activityType='RR'
group by customerID
```

```
having count(*) >2
```

7. All brands where total number of points redeemed overall is less than 500 points

```
select BRANDID, sum(ACTIVITYPOINTS) from wallets
where ACTIVITYTYPE='RR'
group by BRANDID
having sum(ACTIVITYPOINTS)<500
```

8. For Customer C0003, and Brand02, number of activities they have done in the period of 08/1/2021 and 9/30/2021.

```
select * from wallets
where brandID='B0002' and customerID='C0003' and activityDate between TO_DATE
('2021-08-01', 'YYYY-MM-DD') AND TO_DATE('2021-09-30', 'YYYY-MM-DD')
```