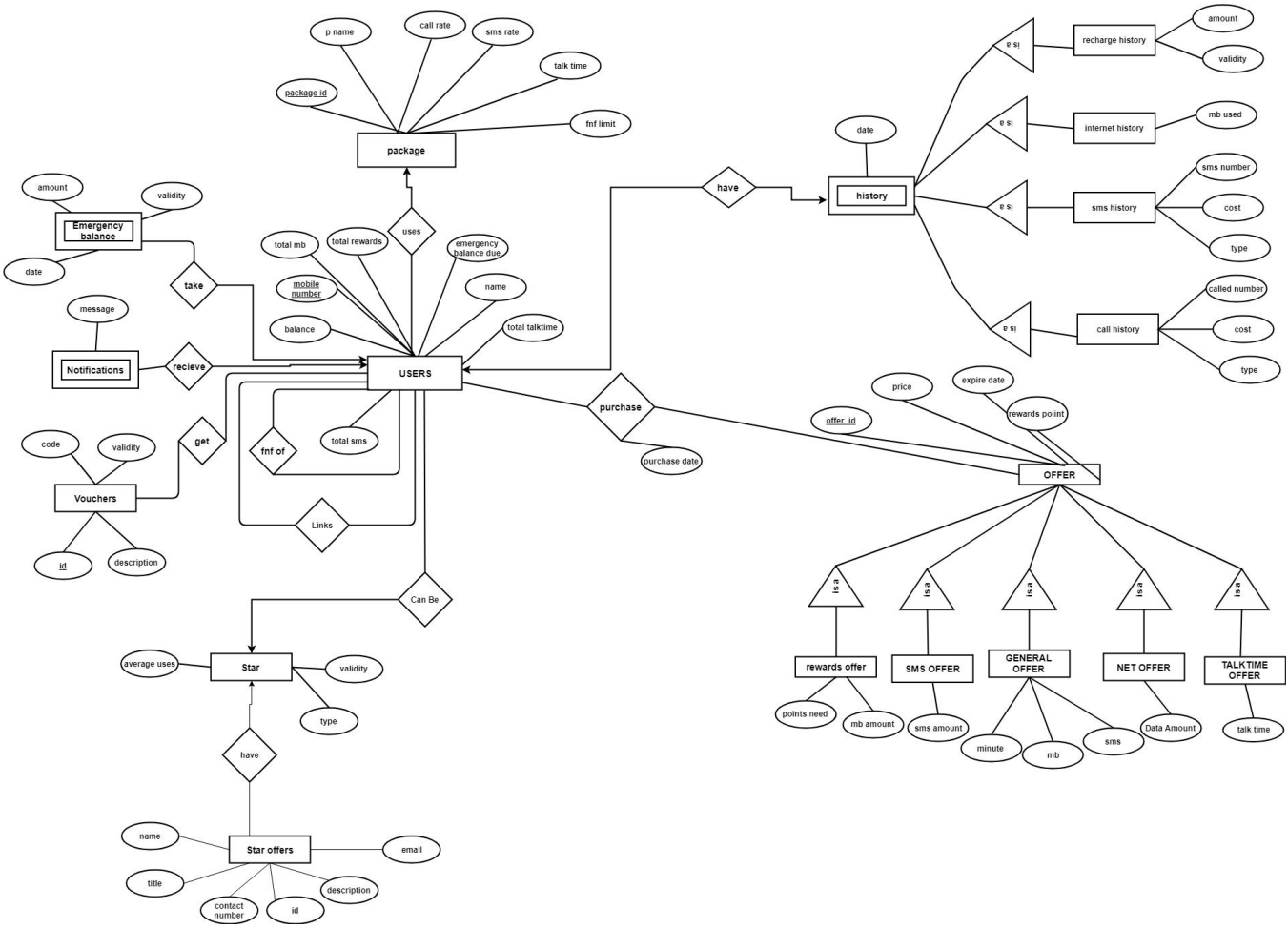
MYGP DATABASE PROJECT

ERD:



Entity Classes :

1. Users
2. Package
3. Offer
4. Reward Offers
5. SMS Offers
6. Internet Offers
7. Talk Time Offers
8. General Offers(Flexi Plan Offers)
9. History
10. Recharge History
11. Internet History

12. SMS History
13. Talk Time History
14. Star
15. Star Offers
16. Notifications
17. Emergency Balance
18. Vouchers

Relations:

1. Users-Package (uses - many to one)
2. Users-Offer (purchase - many to many)
3. Users-History (have - one to one)
4. Users-Users (fnf of - many to many)
5. Users-Users (link - many to many)
6. Users-Star (can be- many to one)
7. Users-Vouchers (get - many to many)
8. Users-Notifications (receive - one to many)
9. Users-Emergency Balance (take - one to many)

Simple Queries :

1. Find the users information(name, balance, total data, total talk time, total sms, total reward points, package name, star status) of the given mobile number

SQL Code:

```
select user_name, balance, total_mb, total_talk_time, total_offer_sms, total_reward_point,
        (select package_name from package where package_id = us.package_id)
package_name, (select type from star where star_id = us.star_id) star_status
from users us
where mobile_number = '01755840785'
```

2. Find the call history(called number, date, cost, type) of the user with the given mobile number

SQL Code:

```
select call_number, to_char(h_date, 'YYYY-MM-DD HH12:MI:SS PM') as date, cost, type
from call_history
where user_id = '01755840785'
```

3. Find the internet history(data used, date) of the user with the given mobile number

SQL Code:

```
select mb_used, to_char(h_date, 'YYYY-MM-DD HH12:MI:SS PM') as date
from internet_history
where user_id = '01755840785'
```

4. Find the sms history(sms number, date, cost, type) of the user with the given mobile number

SQL Code:

```
select sms_number, to_char(h_date, 'YYYY-MM-DD HH12:MI:SS PM') as date, cost, type
from sms_history
where user_id = '01755840785'
```

5. Find the recharge history(amount, validity, recharge time) of the user with mobile number

SQL Code:

```
select amount, validity, to_char(h_date, 'YYYY-MM HH12:MI:SS PM') as recharged_time
from recharge_history
where   user_id = '01755840785'
```

6. Show the reward offers(offer id, mb amount, points need, validity)

SQL Code:

```
select offer_id, mb_amount, points_need, validity
from reward_offer
```

7. Show the talk time offers(offer id, talk time, validity, price, reward points)

SQL Code:

```
select offer_id, talk_time, validity, price, reward_points
```

```
from talk_time_offer
```

8. Show the SMS offers(offer id, sms amount, validity, price, reward points)

SQL Code:

```
select offer_id, sms_amount, validity, price, reward_points
from sms_offer
```

9. Show the internet offers(offer id, data amount, validity, price, rewards points)

SQL Code:

```
select offer_id, data_amount, validity, price, reward_points
from internet_offer
```

10. Show the package information(package name, call rate, data rate, fnf limit, pulse)

SQL Code:

```
select package_name, call_rate, data_rate, fnf_limit
from package
```

11. Show star offers for the given star(offer name, title , description, eligible star)

SQL Code:

```
select offer_name, title, description, (select type from star where star_id = so.star_id)
star_status
from stars_offer so
where star_id = (
  select star_id from users
  where mobile_number = '01755840785')
```

12. Show the notifications of the given mobile number(notifications message)

SQL Code:

```
select message from notifications where user_id = '01755840785'
```

13. Show the fnf numbers of the given mobile number(fnf number)

SQL Code:

```
select fnf_to from fnf where fnf_by = '01755840785'
```

14. Show the linked numbers of the given mobile number(linked numbers)

SQL Code:

```
select linked_to from link where linked_by = '01755840785'
```

15. Show the offer ID.

SQL Code:

```
select offer_id from offers order by offer_id;
```

Complex Queries :

1. Find the offer_id ,price and number of times purchased, of top five offers that have been maximally purchased in the last 3 months.

SQL Code:

```
select offer_id , price, (select count(*) from purchase_offer where offer_id = io.offer_id)
purchase_count from internet_offer io
where io.offer_id in
        (select offer_id
         from (select offer_id,
                (select count(*) from purchase_offer
                 where offer_id = o.offer_id and (now()-purchase_date) < interval '1
day'*90 ) cnt
                 from offers o
                 group by o.offer_id
                 order by cnt desc fetch first 5 row only) first_five
        )
union
select offer_id , price, (select count(*) from purchase_offer where offer_id = sm.offer_id)
purchase_count   from sms_offer sm
where sm.offer_id in
        (select offer_id
         from (select offer_id,
```

```
            (select count(*) from purchase_offer
             where offer_id = o.offer_id and (now()-purchase_date) < interval '1
day'*90 ) cnt
             from offers o
             group by o.offer_id
             order by cnt desc fetch first 5 row only) first_five
      )
union
select offer_id , price, (select count(*) from purchase_offer where offer_id = tm.offer_id)
purchase_count   from talk_time_offer tm
where tm.offer_id in
      (select offer_id
       from (select offer_id,
            (select count(*) from purchase_offer
             where offer_id = o.offer_id and (now()-purchase_date) < interval '1
day'*90 ) cnt
             from offers o
             group by o.offer_id
             order by cnt desc fetch first 5 row only) first_five
      )
union
select offer_id , price, (select count(*) from purchase_offer where offer_id = go.offer_id)
purchase_count   from general_offer go
where go.offer_id in
      (select offer_id
       from (select offer_id,
            (select count(*) from purchase_offer
             where offer_id = o.offer_id and (now()-purchase_date) < interval '1
day'*90 ) cnt
             from offers o
             group by o.offer_id
             order by cnt desc fetch first 5 row only) first_five
      )
union
select offer_id , price, (select count(*) from purchase_offer where offer_id = ro.offer_id)
purchase_count   from reward_offer ro
where ro.offer_id in
      (select offer_id
       from (select offer_id,
            (select count(*) from purchase_offer
             where offer_id = o.offer_id and (now()-purchase_date) < interval '1
day'*90 ) cnt
             from offers o
```

```
                group by o.offer_id
                order by cnt desc fetch first 5 row only) first_five
        )
order by purchase_count desc ;
```

2. Find the users who have made at least 1 call to his/her current fnf numbers in last 6 months. Show the output along with the fnf numbers to which they called and total number of calls made

SQL Code:

```
select fn.fnf_by,   fn.fnf_to,
       (select count(*)
        from call_history
        where user_id = fn.fnf_by
          and type = 'outgoing'
          and (now() - h_date) < interval '1 day'*180
          and call_number in (
          select fnf_to
          from fnf
          where fnf_by = fn.fnf_by
        )
       )
from fnf fn
where 1<=(select count(*)
        from call_history
        where user_id = fn.fnf_by
          and type = 'outgoing'
          and (now() - h_date) < interval '1 day'*180
          and call_number in (
          select fnf_to
          from fnf
          where fnf_by = fn.fnf_by
        )
       );
```

3. Show the user_name and total number of sms he or she has sent in the past 90 days, of the users who have used an offer more than or equal 3 times in the past 60 days

SQL Code:

```
select user_name, (
  coalesce((select count(*)
  from sms_history where
       user_id = u.mobile_number and now() - h_date <= interval   '1 day' * 90
                        and type = 'outgoing'), 0)) sms_sent
from users u
where 3 <= (coalesce((select max(tab.max_count) from (select count(*) max_count from
purchase_offer where user_id = u.mobile_number
                          and now() - purchase_date <= interval '1 day' * 60
                          group by offer_id) tab),0))
```

4. Show the user_name and mobile number of the users who use djuice package and have used al least 1000 taka in the past 60 days

SQL Code:

```
select user_name, mobile_number   from users u
where package_id   = (select package_id
                        from package where package_name = 'DJUICE'
) and (
     (coalesce((select sum(cost) from call_history
                where user_id = u.mobile_number and now()-h_date <=
                                                        interval '1 day' * 60),0)+
      coalesce((select sum(cost) from sms_history where user_id = u.mobile_number
                                                        and now() - h_date <= interval '1
day' * 60),0)+
      coalesce((select sum(s.price) from purchase_offer o join sms_offer s on o.offer_id =
s.offer_id
                where o.user_id = u.mobile_number and now()- purchase_date <=
interval '1 day' * 60), 0) +
      coalesce((
                 select sum(i.price) from purchase_offer o join internet_offer i on
o.offer_id = i.offer_id
                 where o.user_id = u.mobile_number and now()- purchase_date <=
interval '1 day' * 60), 0) +
      coalesce((
                 select sum(t.price) from purchase_offer o join talk_time_offer t on
o.offer_id = t.offer_id
                 where o.user_id = u.mobile_number and now()- purchase_date <=
interval '1 day' * 60), 0)+
```

```
      coalesce((

                  select sum(g.price) from purchase_offer o join general_offer g on
o.offer_id = g.offer_id
                  where o.user_id = u.mobile_number and now()- purchase_date <=
interval '1 day' * 60), 0)) >= 1000
  )
```

5. Show the users name, mobile number and star status, who have taken the offer that have been purchased maximum number of times

SQL Code:

```
select user_name, mobile_number, (
  select type
  from star
  where star_id = u.star_id
) star_status
from users u   where mobile_number in ( select max_users_table.uss from
  (select distinct user_id uss
   from purchase_offer
   where offer_id = (select max_table.offer_id
                     from (
                           select offer_id, count(purchase_offer.offer_id) cc
                           from purchase_offer
                           group by offer_id
                           order by cc desc
                             fetch first row only
                          ) max_table)
  ) max_users_table)
```

6. Show the user name and the total used amount in the last month, who are PLATINUM PLUS users and fnf of at least 2 users and linked by at least two users

SQL Code:

```
select user_name, (
  select (coalesce((select sum(cost) from call_history where user_id = u.mobile_number
                                                        and now()-h_date <=
interval '1 day' * 60),0)+
          coalesce((select sum(cost) from sms_history where user_id = u.mobile_number
```

```
                                                    and now() - h_date <=
interval '1 day' * 30),0)+
            coalesce((select sum(s.price) from purchase_offer o join sms_offer s on
o.offer_id = s.offer_id
                        where o.user_id = u.mobile_number and now()- purchase_date <=
interval '1 day' * 30), 0) +
            coalesce((select sum(i.price) from purchase_offer o join internet_offer i on
o.offer_id = i.offer_id
                        where o.user_id = u.mobile_number and now()- purchase_date <=
interval '1 day' * 30), 0) +
            coalesce(( select sum(t.price) from purchase_offer o join talk_time_offer t on
o.offer_id = t.offer_id
                         where o.user_id = u.mobile_number and now()- purchase_date <=
interval '1 day' * 30), 0)+
            coalesce((select sum(g.price) from purchase_offer o join general_offer g on
o.offer_id = g.offer_id
                        where o.user_id = u.mobile_number and now()- purchase_date <=
interval '1 day' * 30), 0))
) total_used_amount
from users u
where u.star_id = (
  select star_id from star
  where type = 'PLATINUM_PLUS'
) and 2 <= (
  coalesce((select count(*) from fnf where fnf_to = u.mobile_number), 0)
  ) and 2 <= (
  coalesce((select count(*) from link where linked_to = u.mobile_number),0))
```

7. Give the total count of internet offer, sms offer and talk-time offers purchased by each star type in last 2 months along with the star status

SQL Code:

```
select str.type ,
        (select count(*)
         from purchase_offer
         where offer_id >= 12000 and offer_id < 13000
           and user_id in (
           select mobile_number
           from users
           where star_id = (
             select star_id
             from star
```

```
            where type = str.type
          )
        ) and (now()-purchase_date) < interval '1 day'*60
      ) total_internet_offer,
      (select count(*)
       from purchase_offer
       where offer_id >= 11000 and offer_id < 12000
         and user_id in (
         select mobile_number
         from users
         where star_id = (
           select star_id
           from star
           where type = str.type
         )
       ) and (now()-purchase_date) < interval '1 day'*60
      ) total_talk_time_offer,
      (select count(*)
       from purchase_offer
       where offer_id >= 10000 and offer_id < 11000
         and user_id in (
         select mobile_number
         from users
         where star_id = (
           select star_id
           from star
           where type = str.type
         )
       ) and (now()-purchase_date) < interval '1 day'*60
      ) total_sms_offer
from star str
```

8. Users who are under the bondhu package and a star, give their star status, total count of the call list and total sms made in the last 3 months

SQL Code:

```
select mobile_number, user_name ,
        (select type from star where star_id = us.star_id) star_status,
        (select count(*) from call_history ch
        where ch.user_id = us.mobile_number and (now() - h_date) < interval '1 day'*90)
total_call_made,
        (select count(*) from sms_history sh
```

```
        where sh.user_id = us.mobile_number and (now() - h_date) < interval '1 day'*90)
total_sms_made
from users us
where package_id = (
  select package_id from package
  where package_name = 'BONDHU'
  ) and star_id > 0
```

Functions and Procedures :

1. Recharge Account: Recharge the users account with the given mobile number and amount. If mobile number is not valid it shows an invalid message.

SQL Code:

```
create or replace function recharge_account(in mob_number varchar(11), in amount
numeric) returns varchar(100) as $$
declare
  old_amount numeric;
  eb_due numeric;
  cur_timestamp timestamptz;
  validity numeric;
  end_date varchar(100);
  message varchar(100);
begin
  if not is_valid_number(mob_number) then
    return 'Mobile number is not valid';
  end if;
  select balance, emergency_balance_due into old_amount, eb_due from users where
mobile_number = mob_number;
  cur_timestamp := now();
  validity := find_recharge_last_date(amount);
  end_date := to_char(cur_timestamp + interval '1 day' * validity, 'yyyy-mm-dd hh12:mi:ss
am');
  insert into recharge_history values (cur_timestamp, mob_number, amount, validity);
  if eb_due > 0 then
    if amount > eb_due then
      amount := amount-eb_due;
      eb_due := 0;
    else
      eb_due := eb_due-amount;
      amount := 0;
```

```
    end if;
  end if;

  update users set (balance,emergency_balance_due) = (old_amount + amount, eb_due)
where mobile_number = mob_number;
  insert into notifications
  values (mob_number, 'you have successfully recharged ' ||
                        amount || ' taka in your account balance. your current account
balance is '
    || old_amount+ amount || ' taka. your balance will be expired on '|| end_date);
  message := 'Successfully recharged in the account! :D';
  return message;
  exception when others then
    message := 'Cannot recharge :(';
    return message;
end;
$$ language plpgsql;
```

2. Purchase Internet Offer : Purchase an internet offer with the given mobile number and internet offer id. If mobile number or internet id is not valid it shows an invalid message.

SQL Code:

```
create or replace function purchase_internet_offer(in offer_no numeric, in user_no
varchar(11)) returns varchar(100) as $$
  declare
    purchased_timestamp timestamptz;
    costt numeric(8,2);
    end_date varchar(100);
    valid numeric;
    reward_point numeric;
    data_total numeric;
    old_reward_point numeric;
    old_account_balance numeric;
    old_data numeric;
    message varchar(100);
  begin
    if not is_valid_number(user_no) then
      return 'Mobile number is not valid';
    end if;
```

```
    if not is_valid_data_offer(offer_no) then
      return 'Data offer id is not valid';
    end if;
    purchased_timestamp := now();
    select price, validity, reward_points, data_amount   from internet_offer where   offer_id
= offer_no
    into costt, valid, reward_point, data_total;
    select total_reward_point, balance, total_mb from users where mobile_number =
user_no
    into   old_reward_point, old_account_balance, old_data;
    raise notice ' %   %', old_account_balance, costt;

    if old_account_balance >= costt then
      end_date := to_char(purchased_timestamp + interval '1 day' * valid, 'yyyy-mm-dd
hh12:mi:ss am');
      insert into purchase_offer values (user_no, offer_no,purchased_timestamp);
      insert into notifications values (user_no, 'you have successfully purchased '
      ||data_total || ' mb. '||costt || ' taka has been deducted from your account balance.'
      ||' the data amount will expire on '|| end_date);
      update users set (total_reward_point, total_mb, balance) =
      (old_reward_point+reward_point, old_data + data_total, old_account_balance-costt)
      where mobile_number = user_no;
      message := 'Successfully purchased the data offer! :D';
    else
      message := 'Have not sufficient balance :(';
    end if;
    return message;
   exception
    when others then
    message :=  'Cannot purchase the data offer :(';
    return message;
  end;
  $$ language plpgsql;
```

3. Purchase Talk-Time Offer: Purchase a talk time offer with the given mobile number and internet id. If mobile number or talk time offer id is not valid it shows an invalid message

SQL Code:

```
create or replace function purchase_talk_time_offer(in offer_no numeric, in user_no
varchar(11)) returns varchar(100) as $$
  declare
```

```
    purchased_timestamp timestamptz;
    cost numeric(8,2);
    end_date varchar(100);
    valid numeric;
    reward_point numeric;
    talk_time_total numeric;
    old_reward_point numeric;
    old_account_balance numeric;
    old_talk_time numeric;
    message varchar(100);
  begin
    if not is_valid_number(user_no) then
      return 'Mobile number is not valid';
    end if;
    if not is_valid_talk_time_offer(offer_no) then
      return 'Talk Time offer id is not valid';
    end if;
    purchased_timestamp := now();
    select price, validity, reward_points, talk_time   from talk_time_offer where   offer_id =
offer_no
    into cost, valid, reward_point, talk_time_total;
    select total_reward_point, balance, total_talk_time from users where mobile_number =
user_no
    into old_reward_point, old_account_balance, old_talk_time;
    if old_account_balance >= cost then
      end_date := to_char(purchased_timestamp + interval '1 day' * valid, 'yyyy-mm-dd
hh12:mi:ss am');
      insert into purchase_offer values (user_no, offer_no,purchased_timestamp);
      insert into notifications values (user_no, 'you have successfully purchased '
      ||talk_time_total || ' minutes. '||cost || ' taka has been deducted from your account
balance.'
      ||' the talk time bundle will expire on '|| end_date);
      update users set (total_reward_point, total_talk_time, balance) =
      (old_reward_point+reward_point, old_talk_time + talk_time_total,
old_account_balance-cost)
      where mobile_number = user_no;
      message :=  'Successfully purchased the talk time offer! :D';
    else
      message := 'Have not sufficient balance :(';
    end if;
    return message;

    exception   when others then
```

```
    message := 'Cannot purchase the talk time offer :(';
    return message;
  end;
  $$ language plpgsql;
```

4. Purchase SMS Offer :Purchase a SMS offer with the given mobile number and sms offer id. If mobile number or sms offer id is not valid it shows an invalid message

SQL Code:

```
create or replace function purchase_sms_offer(in offer_no numeric, in user_no varchar(11))
returns varchar(100) as $$
  declare
    purchased_timestamp timestamptz;
    cost numeric(8,2);
    end_date varchar(100);
    valid numeric;
    reward_point numeric;
    sms_total numeric;
    old_reward_point numeric;
    old_account_balance numeric;
    old_sms numeric;
    message varchar(100);
  begin
    if not is_valid_number(user_no) then
      return 'Mobile number is not valid';
    end if;
    if not is_valid_sms_offer(offer_no) then
      return 'SMS offer id is not valid';
    end if;
    purchased_timestamp := now();
    select price, validity, reward_points, sms_amount  from sms_offer where  offer_id =
offer_no
    into cost, valid, reward_point, sms_total;
    select total_reward_point, balance, total_offer_sms from users where mobile_number =
user_no
    into old_reward_point, old_account_balance, old_sms;
    if old_account_balance >= cost then
      end_date := to_char(purchased_timestamp + interval '1 day' * valid, 'yyyy-mm-dd
hh12:mi:ss am');
```

```
        insert into purchase_offer values (user_no, offer_no,purchased_timestamp);
        insert into notifications values (user_no, 'you have successfully purchased '
        ||sms_total || '. '||cost || ' taka has been deducted from your account balance.'
        ||' the sms bundle will expire on '|| end_date);
        update users set(total_reward_point, total_offer_sms, balance) =
        (old_reward_point+reward_point, old_sms + sms_total, old_account_balance-cost)
        where mobile_number = user_no;
        message := 'Successfully purchased the sms offer! :D';
    else
    message := 'Have not sufficient balance :(';
    end if;
    return message;

    exception
    when others then
    message :=  'Cannot purchase the sms offer :(';
    return message;
  end;
  $$ language plpgsql;
```

5. Purchase Reward Offer :Purchase a reward offer with the given mobile number and reward offer id. If mobile number or reward offer id is not valid it shows an invalid message

SQL Code:

```
create or replace function purchase_reward_offer(in offer_no numeric, in user_no
varchar(11)) returns varchar(100) as $$
declare
    purchased_timestamp timestamptz;
    end_date varchar(100);
    valid numeric;
    reward_point numeric;
    data_total numeric;
    old_reward_point numeric;
    old_data numeric;
    message varchar(100);
 begin
  if not is_valid_number(user_no) then
    return 'Mobile number is not valid';
  end if;
  if not is_valid_reward_offer(offer_no) then
    return 'Reward offer id is not valid';
  end if;
```

```
    purchased_timestamp := now();
    select   validity, points_need, mb_amount   from reward_offer where   offer_id =
offer_no
    into valid, reward_point, data_total;
    select total_reward_point, total_mb from users where mobile_number = user_no
    into old_reward_point, old_data;

    if old_reward_point >= reward_point then
      end_date := to_char(purchased_timestamp + interval '1 day' * valid, 'yyyy-mm-dd
hh12:mi:ss am');
      if ((select count(*) from offers where offer_id = offer_no) = 1) then
        raise notice 'is present in the table';
      end if;
      insert into purchase_offer values (user_no, offer_no,purchased_timestamp);
      insert into notifications values (user_no, 'you have successfully purchased '
      ||data_total || ' mb. '|| reward_point || ' reward points have been deducted from your
total reward points.'
      ||' the data amount will expire on '|| end_date);
      update users set (total_reward_point, total_mb) =
      (old_reward_point-reward_point, old_data + data_total)
      where mobile_number = user_no;
      message := 'Successfully purchased a data offer using reward points! :D';
    else
      message :=   'Have not sufficient reward points :(';
    end if;
    return message;

  exception
   when others then
    message :=   'Cannot purchase the data offer using the reward points :(';
    return message;
  end;

$$ language plpgsql;
```

6. Purchase General Offer :Purchase a general offer with the given mobile number, minutes, data amount, total sms, validity. If mobile number is not valid it shows an invalid message

SQL Code:

```
create or replace function purchase_general_offer
```

```
  (in user_no varchar(11), in min numeric, in data numeric, in sms numeric, in valid
numeric) returns varchar(100)
as $$
  declare
    purchased_timestamp timestamptz;
    cost numeric(8,2);
    end_date varchar(100);
    old_account_balance numeric;
    old_talk_time numeric;
    old_data numeric;
    old_sms numeric;
    offer_no numeric;
    messsage varchar(100);
  begin
    if not is_valid_number(user_no) then
      return 'Mobile number is not valid';
    end if;
    purchased_timestamp := now();
    select balance, total_talk_time, total_mb, total_offer_sms from users where
mobile_number = user_no into
    old_account_balance, old_talk_time, old_data, old_sms;
    cost := count_general_offer_price(sms, data, min, valid);

    if old_account_balance >= cost then
      end_date := to_char(purchased_timestamp + interval '1 day' * valid, 'yyyy-mm-dd
hh12:mi:ss am');
      offer_no = nextval(pg_get_serial_sequence('general_offer', 'custom_id'));
      insert into general_offer(offer_id, price, validity, munite, mb_amount, sms_amount)
      values (offer_no, cost,valid, min, data, sms);
      insert into purchase_offer values(user_no, offer_no, purchased_timestamp);
      insert into notifications values (user_no, 'you have successfully purchased '
      ||min || ' minutes. ' || data || ' mb. ' || sms || ' sms. ' ||cost || ' taka has been deducted
from your account balance.'
      ||' the offer will expire on '|| end_date);
      update users set (total_mb, total_talk_time, total_offer_sms, balance) =
      (old_data+data, old_talk_time + min,old_sms+sms,   old_account_balance-cost)
      where mobile_number = user_no;
      messsage := 'Successfully purchased a general offer! :D';
    else
      messsage :=  'Have not sufficient balance :(';
    end if;
    return messsage;
```

```
    exception
    when others then
      messsage :=  'Cannot purchase the general offer :(';
      return messsage;
  end;
  $$ language plpgsql;
```

7. Make FNF : Add an fnf to the given user mobile number if the user is eligible to add.

SQL Code:

```
create or replace function make_fnf(in number_by varchar(11), in number_to varchar(11))
returns varchar(100) as $$
declare
  message varchar(100);
  max_fnf_limit numeric;
  current_fnf numeric;
begin
  select count(*) from fnf where fnf_by = number_by into current_fnf;
  select fnf_limit from package where package_id = (select package_id from users
    where mobile_number = number_by) into max_fnf_limit;
  if (current_fnf = max_fnf_limit) then
    return 'You cannot further make fnf as maximum limit has been reached';
  end if;
  if not is_valid_number(number_by) then
    return 'Mobile number is not valid';
  elseif not is_valid_number(number_to) then
    return 'FNF number is not valid';
  end if;
    insert into fnf values(number_by, number_to);
    message := 'Successfully made the fnf';
    return message;
end;
$$ language plpgsql;
```

8. Make link :Link a mobile number to the given user mobile number if the user is eligible to add.

SQL Code:

```
create or replace function make_link(in number_by varchar(11), in number_to varchar(11))
returns varchar(100) as $$
declare
  message varchar(100);
  current_link numeric;

begin
  select count(*) from link where linked_by = number_by into current_link;
  if current_link = 3 then
    return 'You cannot make further links as maximum limit has been reached';
  end if;
  if not is_valid_number(number_by) then
    return 'Mobile number is not valid';
  elseif not is_valid_number(number_to) then
    return 'Link number is not valid';
  end if;
  insert into link values (number_by, number_to);
  message := 'Successfully made the link! :D';
  return message;
end;
$$ language plpgsql;
```

9. Migrate Package : Migrate the package into the given package to the given mobile number.

SQL Code:

```
create or replace function migrate_package(in mob_number varchar(11), in p_name
varchar(40)) returns varchar(100) as $$
declare
  p_id numeric;
  message varchar(100);
begin
  if not is_valid_number(mob_number) then
    return 'Mobile number is not valid';
  elsif ((select count(*) from package where package_name = p_name) = 0) then
    return 'Invalid package name';
  end if;
  select package_id from package where package_name = p_name into p_id;
```

```
  update users set package_id = p_id where mobile_number = mob_number;
  message := 'Successfully migrated into the package! :D';
  return message;
  exception
  when others then
  return 'Cannot migrate into the package :(';
end;
$$ language plpgsql;
```

10. Take Emergency Balance : Recharge emergency balance to the given mobile number with the given balance account if the user is eligible to take.

SQL Code:

```
create or replace function take_emergency_balance(in mob_number varchar(11), in amount
numeric) returns varchar(100) as $$
declare
  prev_bal numeric;
  prev_due numeric;
  message varchar(100);
begin
  if not is_valid_number(mob_number) then
    return 'Mobile number is not valid';
  end if;
  select balance, emergency_balance_due from users where mobile_number =
mob_number
                                                    into prev_bal, prev_due;
  if prev_bal > 0.5 or amount>50 then
    message := 'You are not eligible to take emergency balance :(';
  else
    update users set (balance, emergency_balance_due) = (prev_bal+amount,
prev_due+amount) where mobile_number=mob_number;
    insert into emergency_balance values (mob_number, amount, now(), 30);
    message := 'Successfully taken the emergency balance :D';
    insert  into notifications values(mob_number, 'You have successfully taken ' || amount
|| ' taka emergency balance.');
  end if;
  return message;
  exception
  when others then
  return 'Something error occurred :(';
```

```
end;
$$ language plpgsql;
```

11. Transfer Balance : Transfer balance from one account to another account if eligible with the given emergency balance.

SQL Code:

```
create or replace function transfer_balance( in user_number varchar(11), in
user_to_transfer  varchar(11), in amount numeric)
  returns text as $$
declare
  prev_bal_user numeric;
  user_due numeric;
  prev_bal_tt numeric;
begin
  if not is_valid_number(user_number) then
    return 'User number is not valid';
  elseif not is_valid_number(user_to_transfer) then
    return 'Transfer number is not valid';
  end if;
  select balance,emergency_balance_due into prev_bal_user,user_due from users where
mobile_number = user_number;
  select balance into prev_bal_tt from users where mobile_number = user_to_transfer;

  if prev_bal_user+2 < amount or user_due>0 then
    return 'Insufficient balance :(';
  else
    update users set balance = (prev_bal_user-amount) where mobile_number =
user_number;
    update users set balance = (prev_bal_tt+amount) where mobile_number =
user_to_transfer;
    return 'Balance transfer successful! :D';
  end if;
end;
$$ language plpgsql;
```

Trigger Descriptions :

1. Check Star Trigger :

It triggers after purchasing any offer on purchase_offer table and checks the validity of the user that he/she has become a star or not.

SQL Code:

```
create or replace function check_star_function() returns trigger as $$
declare
  total_usage_last3months numeric;
  prc numeric;
  curr_date timestamptz;
  str_id numeric;
  usage_tobe_star numeric;
  r cursor for
    select * from star;
  off_id cursor for
    select OFFER_ID from PURCHASE_OFFER
    where USER_ID = new.user_id and (curr_date - PURCHASE_DATE) < interval '1 day' *
90;
begin
  curr_date := now();
  total_usage_last3months := 0;

  for b in off_id
  loop
    select price into prc from internet_offer where offer_id = b.offer_id;
    if prc is not null then   total_usage_last3months := total_usage_last3months + prc; end
if;
    select price into prc from talk_time_offer where offer_id = b.offer_id;
    if prc is not null then   total_usage_last3months := total_usage_last3months + prc; end
if;
    select price into prc from sms_offer where offer_id = b.offer_id;
    if prc is not null then   total_usage_last3months := total_usage_last3months + prc; end
if;
    select price into prc from general_offer where offer_id = b.offer_id;
    if prc is not null then   total_usage_last3months := total_usage_last3months + prc; end
if;
  end loop;

  for a in r
  loop
    str_id := a.star_id;
    usage_tobe_star := a.average_uses;
    if total_usage_last3months >= usage_tobe_star then
```

```
        update users set (STAR_ID,star_date) = (str_id, curr_date) where MOBILE_NUMBER =
new.user_id;
        exit ;
      end if;
   end loop;
   return new;
end;
$$ language plpgsql;

create trigger check_star after insert on purchase_offer
for each row
   execute procedure check_star_function();
```

2. Insert internet pk to offer:

It triggers after inserting an internet offer on internet offer table and insert the id to the offer table

SQL Code:

```
create or replace function insert_pk_to_offer() returns trigger as $$
begin
   insert into offers values (new.offer_id);
   return new;
end;
$$ language plpgsql;

create trigger insert_internet_pk_to_parent after insert on internet_offer
for each row
   execute procedure insert_pk_to_offer();
```

3. Insert sms pk to offer:

It triggers after inserting a sms offer on sms offer table and insert the id to the offer table

SQL Code:

```
create or replace function insert_pk_to_offer() returns trigger as $$
begin
   insert into offers values (new.offer_id);
   return new;
end;
```

```
$$ language plpgsql;

create trigger insert_sms_pk_to_parent after insert on sms_offer
for each row
   execute procedure insert_pk_to_offer();
```

4. Insert talk time pk to offer:

It triggers after inserting an talk time offer on talk time offer table and insert the id to the offer table

SQL Code:

```
create or replace function insert_pk_to_offer() returns trigger as $$
begin
   insert into offers values (new.offer_id);
   return new;
end;
$$ language plpgsql;

create trigger insert_talk_time_pk_to_parent after insert on talk_time_offer
for each row
   execute procedure insert_pk_to_offer();
```

5. Insert reward pk to offer:

It triggers after inserting an reward offer on reward offer table and insert the id to the offer table

SQL Code:

```
create or replace function insert_pk_to_offer() returns trigger as $$
begin
   insert into offers values (new.offer_id);
   return new;
end;
$$ language plpgsql;

create trigger insert_reward_pk_to_parent after insert on reward_offer
for each row
```

```
  execute procedure insert_pk_to_offer();
```

6. Insert general pk to offer:

It triggers after inserting an general offer on general offer table and insert the id to the offer table

SQL Code:

```
create or replace function insert_pk_to_offer() returns trigger as $$
begin
  insert into offers values (new.offer_id);
  return new;
end;
$$ language plpgsql;

create trigger insert_general_pk_to_parent after insert on general_offer
for each row
  execute procedure insert_pk_to_offer();
```

By:

Abdur Rashid Tushar (1605070)

Iqbal Hossain Raju (1605080)