**Database Programming and Implementation (SQL)**

**This project aims to get the proper skills to use SQL trigger, procedure and function.**

1. **The Initial State of the Database**

Branch Table:

select \* from branch;

Table

Description automatically generated

Campaign table:

select \* from campaign;

Table

Description automatically generated

DiscountDetails:

select \* from discountDetails;

Table

Description automatically generated

Membership:

select \* from membership;

Graphical user interface

Description automatically generated

Product:

select \* from product;

Table

Description automatically generated

1. **Stored Programs**

-- task 2

Drop TRIGGER if exists CHECK\_MEMBERSHIP\_UPDATE;

DELIMITER //

CREATE TRIGGER CHECK\_MEMBERSHIP\_UPDATE

BEFORE UPDATE ON Membership

FOR EACH ROW

BEGIN

DECLARE msg VARCHAR(200);

DECLARE currentDate DATE default now();

Declare flag boolean;

-- when try to change membership level with the values other than 'Silver', 'Gold', 'Platinum'

set flag = new.MembershipLevel='Platinum' or new.MembershipLevel='Silver' or new.MembershipLevel='Gold';

if flag = false then

SET msg="Membership level can not have values except Platinum, Gold, Silver.";

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT =msg;

end if;

IF new.MemberExpDate < currentDate and old.MembershipLevel <> new.MembershipLevel THEN

-- when nothing changed except MembershipLevel

SET msg="Membership expired! Can't upgrade membership level.";

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT =msg;

else

if old.MembershipLevel='Platinum' and new.MembershipLevel <>'Platinum' then

SET msg="There is no further upgrade for the PLATINUM members";

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT =msg;

elseif new.MembershipLevel='Platinum' and old.MembershipLevel='Silver' then

SET msg="Only the GOLD member can be upgraded to the PLATINUM members";

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT =msg;

elseif new.MembershipLevel='Silver' and old.MembershipLevel<>'Silver' then

SET msg="Silver is the lowest membership level";

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT =msg;

end if;

END IF;

END //

DELIMITER ;

-- task 3

drop procedure if exists BrandNameCampaign;

delimiter //

create procedure BrandNameCampaign(in brandName varchar(255))

begin

declare v\_finished int default 0;

declare campaignNo int ;

declare campaignStart date;

declare campaignFinish date;

declare v\_productID int;

declare discountProduct cursor for

select productID from product where brand=brandName order by price desc limit 5;

declare continue handler for not found set v\_finished = 1;

select campaignID into campaignNo from campaign order by campaignID desc limit 1;

set campaignNo = campaignNo +1;

set campaignStart = DATE\_ADD(now(), INTERVAL 2 week);

set campaignFinish = DATE\_ADD(now(), INTERVAL 6 week);

insert into campaign Values (campaignNo, campaignStart, campaignFinish);

open discountProduct;

repeat

fetch discountProduct into v\_productID;

if not (v\_finished =1) then

insert into discountDetails Values (v\_productID, campaignNo,'Silver',10);

insert into discountDetails Values (v\_productID, campaignNo,'Gold',20);

insert into discountDetails Values (v\_productID, campaignNo,'Platinum',30);

end if;

until v\_finished

end repeat;

close discountProduct;

end //

delimiter ;

1. **Required Testing on the Sample Data - Task 4 Part (a)**

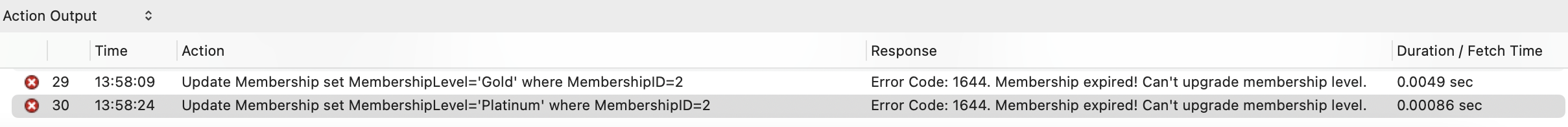
-- task4 Part(a)

-- task 2 testing

-- a.2.1 when try to upgrade the membeshipLevel for the expired member

Update Membership set MembershipLevel='Gold' where MembershipID=2;

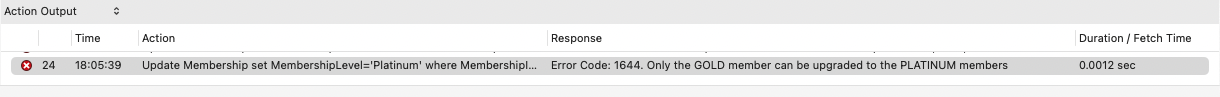
Update Membership set MembershipLevel='Platinum' where MembershipID=2;



-- failed since membership is expired

-- a.2.2 when try to change silver to platinum

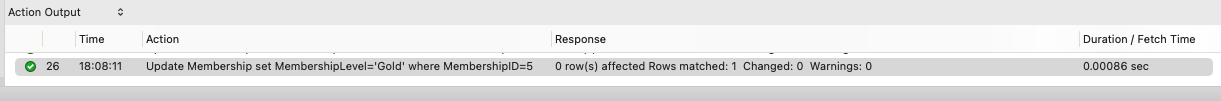
Update Membership set MembershipLevel='Platinum' where MembershipID=5;



-- failed since only the ‘Gold’ member can be upgraded to ‘Platinum’ member

-- a.2.3 when try to change silver to gold (vaild)

Update Membership set MembershipLevel='Gold' where MembershipID=5;



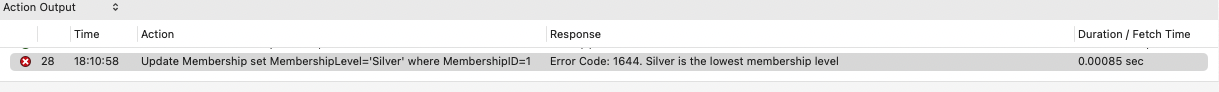
Graphical user interface, text, application, email

Description automatically generated

-- success because ‘Silver’ member can be upgraded to ‘Gold’ member

-- a.2.4 when try to change gold to silver

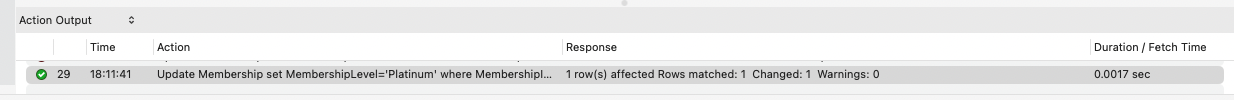
Update Membership set MembershipLevel='Silver' where MembershipID=1;



-- failed since ‘Silver’ is the lowest membership level

-- a.2.5 when try to change gold to platinum (vaild)

Update Membership set MembershipLevel='Platinum' where MembershipID=1;



Graphical user interface, text, application

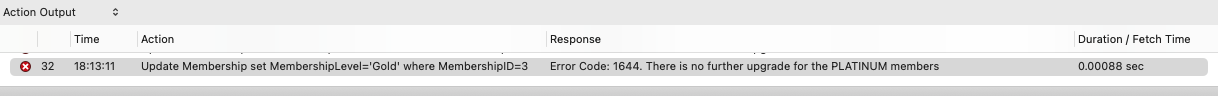
Description automatically generated

-- success because ‘Gold’ member can be upgraded to ‘Platinum’ member

-- a.2.6 when try to change platinum to silver or gold

Update Membership set MembershipLevel='Silver' where MembershipID=3;

Update Membership set MembershipLevel='Gold' where MembershipID=3;



-- failed since there are no further upgrades for ‘Platinum’ member

-- test 3 testing

-- a.3.1 when call the procedure with 'Blonde'

call BrandNameCampaign('Blonde');

Table

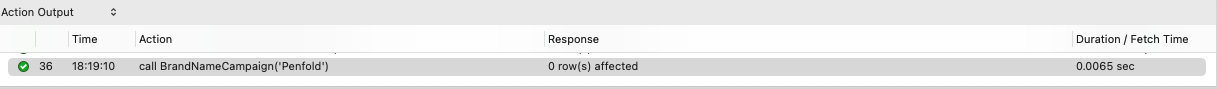
Description automatically generatedcampaign table

Table

Description automatically generatedDiscountDetails table

-- a.3.2 when call the procedure with 'Penfold'

call BrandNameCampaign('Penfold');



Table

Description automatically generatedcampaign table

Table

Description automatically generateddiscountDetails table

1. **More Extensive Testing - Task 4 Part (b)**

-- task 2 testing

-- b.2.1 Insert values to check what happens when trying to change the membership level of a row having today as an expiry date

insert into membership values (6, 'Noah', 'Kwon', 'ahweqwheq@gmail.com','Gold' ,date(now()));

Graphical user interface, application

Description automatically generated

-- test when today is the expiry date

-- purpose : prove that relation between member expiry date and the current time is well set up

Update Membership set MembershipLevel='Platinum' where MembershipID=6;

Graphical user interface, application, table

Description automatically generated

-- the test was successful since the membership is not expired yet.

-- b.2.2 Other variables like FirstName, LastName can be changed even when the expiry date is passed since there are no given restrictions for that

-- purpose: prove that there are no expiry date restrictions for updating other variables except membershipLevel

Update Membership set FirstName='Bree' where MembershipID=2;

Update Membership set eMail='test@email'where MembershipID=2; Table

Description automatically generated with medium confidence

-- the test was successful since there is no expiry date restriction when updating another variable

-- b.2.3 As an extension of test b.2.2 try to change two variables includes membership level when membership is expired

-- purpose: prove that membershipLevel can not be updated even with updating other variables when membership is expired

Update Membership set FirstName='quin', MembershipLevel='Gold' where MembershipID=2; 

-- b.2.4 more than two other variables include membershipLevel can be changed if the update is valid

-- purpose: prove that more than two other variables can be changed if the new values are valid

Update Membership set eMail='GoldTest@google.com', membershipLevel='Platinum' where MembershipID=4;



Graphical user interface

Description automatically generated with medium confidence

-- other variables can be changed without any expiry date restriction

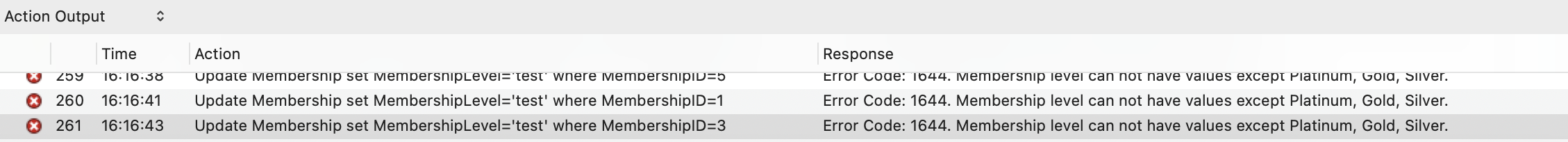
-- b.2.5 when trying to change MembershipLevel to an invalid value which is not 'Silver', 'Gold', 'Platinum'

-- purpose: prove that membershipLevel can not be any other values than ‘Silver’, ‘Gold’, ‘Platinum’

Update Membership set MembershipLevel='test' where MembershipID=5;

Update Membership set MembershipLevel='test' where MembershipID=1;

Update Membership set MembershipLevel='test' where MembershipID=3;



-- Test failed since there is no membership level other than 'Silver', 'Gold', 'Platinum'.

-- b.2.6 when try to change membershipLevel into same membershipLevel

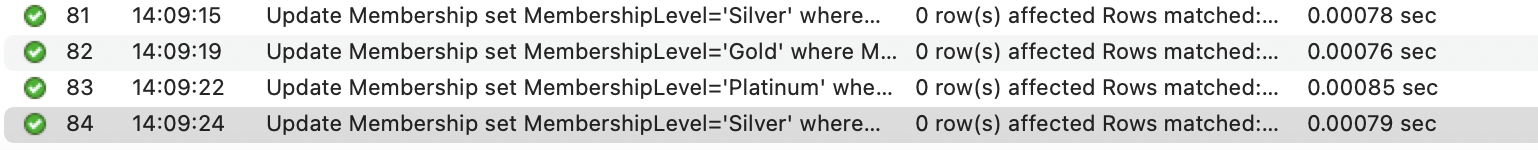
-- purpose: prove that changing membershipLevel into the same value is valid

Update Membership set MembershipLevel='Silver' where MembershipID=5;

Update Membership set MembershipLevel='Gold' where MembershipID=1;

Update Membership set MembershipLevel='Platinum' where MembershipID=3;

Update Membership set MembershipLevel='Silver' where MembershipID=2;



-- the test was successful since nothing the value has not changed at all logically

-- task 3 testing

-- b.3.1 insert more products which are brand is 'Penfold'

-- purpose: insert more 'Penfold' products to check if the sorting function in the procedure is working well. before there are only 4 default 'Penfold' products so they can not be tested.

insert into product values (6, 'Wine', 'Bottle',2021,550,'Penfold');

insert into product values (7, 'Wine', 'Bottle',2020,530,'Penfold');

insert into product values (8, 'Wine', 'Bottle',2018,320,'Penfold');

insert into product values (9, 'Wine', 'Bottle',2019,3800,'Penfold');

Table

Description automatically generated

-- 6~9 are new inserted rows

-- b.3.2 call function with 'Penfold'

-- purpose: test that this procedure can offer the sale campaign for the top 5 expensive 'Penfold' products

call BrandNameCampaign('Penfold');

Table

Description automatically generated

-- the campaign is successfully added

Table

Description automatically generated

-- The discountDetail is successfully added since the top 5 expensive ‘Penfold’ products are like as shown in the table below(9,1,6,7,2).

Table

Description automatically generated

-- b.3.3 insert 3 products which are brand is 'XXXX'

-- purpose: test that this procedure can be performed for the new brand products which were not in the default table

insert into product values (10, 'Wine', 'Bottle',2019,30,'XXXX');

insert into product values (11, 'beer', 'Bottle',2020,15,'XXXX');

insert into product values (12, 'Wine', 'can',2020,35,'XXXX');

Table

Description automatically generated

-- the products are successfully added

-- b.3.4 call function with 'XXXX'

-- purpose: test that this procedure can offer the sale campaign for the top 3 expensive 'XXXX' products

call BrandNameCampaign('XXXX');

Table

Description automatically generated

-- the campaign is successfully added

Table

Description automatically generated

-- The discountDetail is successfully added

-- b.3.5 call function with 'randomtest'

-- purpose: Test campaigns when there are no products with a given brand.

call BrandNameCampaign('randomtest');

Table

Description automatically generated

-- The campaign is successfully added

Table

Description automatically generated

-- However, because there is no product with the brand 'randomtest', the discountDetail rows are not created.