FUNCTIONS AND TRIGGERS

Najrin Sultana-1605042

Abrar Nadib-1605043

FUNCTIONS

**post\_product()**

**This function will post a product.**

**The function will return false if the user has to pay for the product.  
A notification will be sent the user to pay for the product through trigger**

**CREATE OR REPLACE FUNCTION public.post\_product(pcatagory\_id integer, pseller\_id integer, pprice integer, plocation\_id   
integer,pcondition character varying,pphone\_no bigint,pdescription character varying, pitem\_type character varying,   
pbrand character varying, pproduct\_name character varying )  
RETURNS bool   
LANGUAGE 'plpgsql'  
declare  
pis\_paid bool;  
amount integer;  
begin  
select catagory.add\_fee into amount from catagory where catagory.catagory\_id=pcatagory\_id;  
if amount=0 then   
pis\_paid:=null;  
else  
pis\_paid:=false;  
end if;  
INSERT INTO public.products(  
catagory\_id, seller\_id, price, location\_id, condition, phone\_no, posting\_date, posting\_time, is\_varified, is\_paid, description, item\_type, brand, product\_name)  
VALUES (pcatagory\_id, pseller\_id, pprice, plocation\_id, pcondition, pphone\_no, current\_date, localtime, false,pis\_paid,pdescription, pitem\_type, pbrand, pproduct\_name);  
return pis\_paid;  
end;**

#### register\_user()

##### This function will register the user if no user with the same email id exists.

###### CREATE OR REPLACE FUNCTION public.register\_user( page integer, pemail character varying, pgender character varying, pname character varying, ppassword character varying, pphone bigint) RETURNS boolean AS $BODY$ declare coun integer; BODY boolean; begin select count(\*) into coun from users where pemail=users.email; if(coun=0) then BODY:=true; INSERT INTO public.users( email, name, password, age, gender, phone) VALUES (pemail, pname, ppassword, page, pgender, pphone); else BODY:=false; end if; return BODY; end;

#### add\_favourite()

##### This function will add a product as favourite

###### CREATE OR REPLACE FUNCTION public.add\_favourite(u\_id in integer,p\_id in integer) RETURNS void begin if p\_id not in (select product\_id from favourites where favourites.user\_id=u\_id) then INSERT INTO public.favourites(user\_id, product\_id) VALUES (u\_id,p\_id); end if; end;

TRIGGER

FUNCTIONS

#### please\_pay()

##### This trigger function will send notification to user to pay.

###### CREATE FUNCTION public.please\_pay() RETURNS trigger begin if new.is\_paid=false then INSERT INTO public.notifications(user\_id, notice, "time", date, notification\_type) VALUES (new.seller\_id, 'Please pay for your product '||new.product\_name, localtime, current\_date, 'payment'); end if; return null; end;

#### product\_verified()

##### This trigger function will send notification to seller after admin verification.

###### CREATE FUNCTION public.product\_verified() RETURNS trigger begin if old.is\_varified=false and new.is\_varified=true then INSERT INTO public.notifications( user\_id, notice, "time", date, notification\_type) VALUES (new.seller\_id, ' your product '||new.product\_name||' is verified by admin', localtime, current\_date, 'verification'); end if; return null; end;

#### sent\_for\_verification\_paid()

##### This trigger function will send notification to seller after his payment.

###### CREATE FUNCTION public.sent\_for\_verification\_paid() RETURNS trigger declare amount integer; begin select add\_fee into amount from catagory where catagory.catagory\_id=( select products.catagory\_id from products where products.product\_id=new.adv\_id); if new.payment\_amount>=amount then UPDATE public.products SET is\_paid=true WHERE products.product\_id=new.adv\_id; INSERT INTO public.notifications(user\_id, notice, "time", date, notification\_type) VALUES ((select products.seller\_id from products where products.product\_id=new.adv\_id), 'Your product '||((select products.product\_name from products where products.product\_id=new.adv\_id))||'is sent for admin verification', localtime, current\_date, 'verification'); end if; return null; end;

#### sent\_for\_verification\_unpaid()

##### This trigger function will send notification to seller after posting add if he doesn't need to pay.

###### CREATE FUNCTION public.sent\_for\_verification\_unpaid() RETURNS trigger begin if new.is\_paid is null then INSERT INTO public.notifications(user\_id, notice, "time", date, notification\_type) VALUES (new.seller\_id, ' your product '||new.product\_name||' is sent for admin verification ', localtime, current\_date, 'verification'); end if; return null; end;

#### public.someone\_reported()

##### This trigger function will send notification to every admin if someone reports.

###### CREATE FUNCTION public.someone\_reported() RETURNS trigger declare r record; begin for r in(select user\_id from users where users.is\_admin=true) loop INSERT INTO public.notifications(user\_id, notice, "time", date, notification\_type) VALUES (r.user\_id, 'the product with id '||new.product\_id||' is reported', localtime, current\_date, 'report'); end loop; return null; end;

#### public.product\_deleted()

##### This trigger function will send notification to seller if his product is removed.

###### CREATE FUNCTION public.product\_deleted() RETURNS trigger begin INSERT INTO public.notifications(user\_id, notice, "time", date, notification\_type) VALUES (old.seller\_id, ' your product '||old.product\_name||' is removed by admin', localtime, current\_date, 'removal'); return null; end;

TRIGGERS

#### public.log\_reported

##### This trigger will fire someone\_reported() function.

###### CREATE TRIGGER log\_reported AFTER INSERT ON public.reports FOR EACH ROW EXECUTE PROCEDURE public.someone\_reported();

#### public.log\_sent\_for\_verification\_unpaid

##### This trigger will fire sent\_for\_verification\_unpaid() function

###### CREATE TRIGGER log\_sent\_for\_verification\_unpaid AFTER INSERT ON public.products FOR EACH ROW EXECUTE PROCEDURE public.sent\_for\_verification\_unpaid();

#### public.log\_product\_verified

##### This trigger will fire log\_product\_verified() function..

###### CREATE TRIGGER log\_product\_verified AFTER UPDATE OF is\_varified ON public.products FOR EACH ROW EXECUTE PROCEDURE public.product\_verified();

#### public.log\_sent\_for\_verification\_paid

##### This trigger will fire sent\_for\_verification\_paid() function .

###### CREATE TRIGGER log\_sent\_for\_verification\_paid AFTER INSERT ON public.paymnt\_history FOR EACH ROW EXECUTE PROCEDURE public.sent\_for\_verification\_paid();

#### public.log\_product\_deleted

##### This trigger will fire product\_deleted() function .

###### CREATE TRIGGER log\_product\_deleted AFTER DELETE ON public.products FOR EACH ROW EXECUTE PROCEDURE public.product\_deleted();

#### public.log\_please\_pay

##### This trigger will fire please\_pay() function.

###### CREATE TRIGGER log\_please\_pay AFTER INSERT ON public.products FOR EACH ROW EXECUTE PROCEDURE public.please\_pay();

THE END.