Lab 7 database

1.create a function that:

a.Increments given values by 1 and returns it.  
create function *increment*(val integer)  
 returns int as  
$$  
begin  
 return val + 1;  
end;  
$$  
 language plpgsql;

b.Returns sum of two numbers.  
create function *twosum*(val1 integer, val2 integer)  
 returns int as  
$$  
begin  
 return val1 + val2;  
end;  
$$  
 language plpgsql;

c.Returns true or false if numbers are divisible by 2.  
  
create function *divisibleByTwoOrNot*(variadic arr integer[])  
 returns boolean as  
$$  
declare  
 i integer;  
begin  
 foreach i in array arr  
 loop  
 if (i % 2 = 1) then  
 return false;  
 end if;  
 end loop;  
 return true;  
  
end;  
$$  
 language plpgsql;

d.Checks some password for validity.  
create function *check\_password*(pass text)  
 returns boolean as  
$$  
declare  
 passed boolean;  
begin  
 if (*length*(pass) > 7) then  
 return true;  
  
 else  
 return false;  
 end if;  
end;  
  
$$ language plpgsql;

e.Returns two outputs, but has one input.  
create function *twoOutput*(val int, out first integer, out second integer)  
as  
$$  
begin  
 first = val \* val;  
 second = val + val;  
end;  
$$  
 language plpgsql;

2. Create a trigger that:

2a. Return timestamp of the occured action within the database.  
  
create function *returnTimestamp*()  
 returns trigger as  
$$  
BEGIN  
 raise notice '%', *current\_timestamp*;  
 return new;  
  
  
end;  
$$  
 language plpgsql;  
  
  
create trigger occur  
 after insert or delete or update  
 on actions for each row  
execute function *returnTimestamp*();

2b. Computes the age of a person when persons’ date of birth is inserted.

create or replace function *compute*() returns trigger as  
$$  
begin  
 raise notice '%', *date\_part*('year', *now*()) - *date\_part*('year', new.dateBirth);  
 return new;  
  
end;  
$$  
 language plpgsql;  
  
create trigger computeBirth  
 before insert on people for each row  
 execute function *compute*();

2c. Adds 12% tax on the price of the inserted item.

create function *addTax*() returns trigger as  
$$  
begin  
 update products set price = price \* 1.12 where price = price;  
 return new;

end;  
$$  
 language plpgsql;

create trigger addTaxForPrice  
 after insert  
 on products  
execute function *addTax*();

2d. Prevents deletion of any row from only one table.

create function *prevent*() returns trigger as  
$$  
begin  
 raise exception 'You cannot delete';  
  
  
end;  
$$  
 language plpgsql;  
  
  
  
create trigger preventDelete  
 before delete  
 on person for each row  
execute function *prevent*();

2e. Launches functions 1.d and 1.e

create function *callFunctions*() returns trigger as  
$$  
begin  
 raise notice '%', *check\_password*(new.password);  
 raise notice '%', *twooutput*(new.number);  
 return new;  
  
end;  
  
$$  
 language plpgsql;  
  
create trigger callTwoFunctions  
 before insert  
 on numberAndPassword for each row  
execute function *callFunctions*();

3.Create procedures that:  
  
--a) Increases salary by 10% for every 2 years of work experience and provides  
--10% discount and after 5 years adds 1% to the discount.

create or replace procedure *updateEmployee*(experience integer)  
 language plpgsql  
as  
$$  
declare  
 count int;  
begin  
 count = experience / 2;  
 for i in 0..count  
 loop  
 update employees set salary = salary \* 1.1 where salary = salary;  
 end loop;  
 count = experience / 5;  
 for i in 0..count  
 loop  
 update employees set discount = discount \* 1.01;  
 end loop;  
  
 commit;  
  
  
end;  
$$;

b) After reaching 40 years, increase salary by 15%. If work experience is more  
--than 8 years, increase salary for 15% of the already increased value for work  
--experience and provide a constant 20% discount.

create or replace procedure *updateEmployee2*(experience integer)  
 language plpgsql  
as  
$$  
declare  
 count int;  
begin  
 count = experience / 40;  
 for i in 0..count  
 loop  
 update employees set salary = salary \* 1.15 where salary = salary;  
 end loop;  
 count = experience / 5;  
 if(experience > 8) then update employees set salary = salary \* 1.15, discount = 20 where salary = salary ;  
 end if;  
  
 commit;  
  
  
end;  
$$;