-- 1.Create a function that:  
-- a. Increments given values by 1 and returns it.  
CREATE FUNCTION *inc*(val integer) RETURNS integer AS  
$$  
BEGIN  
 RETURN val + 1;  
END;  
$$  
 LANGUAGE plpgsql;  
  
select *inc*(4)  
-- b. Returns sum of 2 numbers.  
CREATE FUNCTION *sum\_of\_two\_integers*(a integer, b integer)  
 RETURNS integer AS  
$$  
BEGIN  
 RETURN a + b;  
END;  
$$  
 language plpgsql;  
select *sum\_of\_two\_integers*(3, 4)  
-- c. Returns true or false if numbers are divisible by 2.  
CREATE or replace FUNCTION *is\_even*(val integer) returns boolean as  
$$  
begin  
 return 1 - val % 2;  
end;  
$$  
 LANGUAGE plpgsql;  
select *is\_even*(6);  
-- d. Checks some password for validity.  
  
  
CREATE or replace FUNCTION *check\_password*(password text) returns boolean as  
$$  
  
BEGIN  
 if (password like 'A%') then  
 return true;  
 --Minimum eight characters, at least one letter and one number  
 else  
 return false;  
 end if;  
end;  
$$ LANGUAGE plpgsql;  
  
select *check\_password*('Aakytzhan123');  
  
drop FUNCTION *two\_outputs*;  
-- e. Returns two outputs, but has one input.  
  
  
CREATE or replace FUNCTION *two\_outputs*(val integer, out a integer, out b integer)  
as  
$$  
  
begin  
 a = val + 2;  
 b = val \* 10;  
end;  
$$ language plpgsql;  
  
  
  
select *two\_outputs*(2);  
  
  
create schema task4  
  
 create table my\_table  
 (  
 id integer primary key not null,  
 name varchar not null,  
 date\_of\_birth timestamp not null,  
 age integer not null,  
 salary integer,  
 workexperience integer,  
 discount integer  
 );  
  
create table my\_table\_history  
(  
 id serial primary key not null,  
 operation\_name varchar not null,  
 operation\_datetime timestamp not null  
  
);  
insert into my\_table  
values (1, 'Bakytzhan', '2003-05-11', 19, 500000, 1, 0);  
insert into my\_table  
values (2, 'Bauka', '2001-07-01', 21, 800000, 3, 10);  
insert into my\_table  
values (3, 'Maksat', '2003-07-11', 19, 600000, 2, 15);  
drop function *get\_date\_of\_birth*;  
create or replace function *get\_date\_of\_birth*(current\_year integer)  
 returns table  
 (  
 name\_of\_people varchar,  
 year\_of\_birth integer  
 )  
as  
$get\_date\_of\_birth$  
  
begin  
 RETURN QUERY SELECT name, current\_year - age from my\_table;  
end;  
$get\_date\_of\_birth$ language plpgsql;  
  
select \*  
from *get\_date\_of\_birth*(2022)  
--  
-- 2. Create a trigger that:  
-- a.Return timestamp of the occured action within the database.  
;  
create or replace function *schema\_update*() returns trigger as  
$schema\_update$  
declare  
 operation\_date timestamp;  
begin  
 if (tg\_op = 'DELETE') then  
 insert into my\_table\_history (operation\_name, operation\_datetime)  
 values (tg\_op, *now*());  
 return old;  
  
 ELSE  
 INSERT INTO my\_table\_history (operation\_name, operation\_datetime)  
 VALUES (TG\_OP, *now*());  
  
 RETURN NEW;  
 end if;  
  
end;  
$schema\_update$ LANGUAGE plpgsql;  
  
-- select current\_timestamp  
drop trigger date\_trigger on my\_table;  
create trigger date\_trigger  
 after INSERT OR UPDATE OR DELETE  
 on my\_table  
 for each row  
execute function *schema\_update*();  
  
alter table my\_table  
 enable trigger date\_trigger;  
select \*  
from my\_table;  
insert into my\_table  
values (15, 'bA', '2003-03-04', 12, 12333, 1, 3);  
  
SELECT \*  
FROM my\_table\_history  
  
-- b.Computes the age of a person when persons’ date of birth is inserted.  
  
create table persons\_age  
(  
 age integer  
);  
  
create or replace function *insert\_birthday*() returns trigger as  
$insert\_birthday$  
  
begin  
 raise notice '%', *date\_part*('year', *now*()) - *date\_part*('year', new.date\_of\_birth);  
 return new;  
end;  
$insert\_birthday$ language plpgsql;  
  
create trigger birth\_date\_trigger  
 before insert  
 on my\_table  
 for each row  
execute procedure *insert\_birthday*();  
  
alter table my\_table  
 enable trigger birth\_date\_trigger;  
  
select \*  
from my\_table;  
  
select \*  
from persons\_age;  
  
insert into my\_table  
values (53, 'Damir', '2011-03-12', 12, 100000, 1, 3);  
-- c.Adds 12% tax on the price of the inserted item.  
create table market  
(  
 tovar\_id integer primary key,  
 tovar varchar(40),  
 total\_price integer not null  
);  
  
  
create or replace function *add\_tax*() returns trigger as  
$$  
begin  
 update market set total\_price = total\_price \* 1.12 where total\_price = total\_price;  
 return new;  
end;  
$$ language plpgsql;  
  
create trigger add\_tax\_trigger  
 after insert  
 on market  
execute procedure *add\_tax*();  
  
alter table market  
 enable trigger add\_tax\_trigger;  
insert into market  
values (1, 'Burger', 100);  
insert into market  
values (2, 'Cola', 200);  
select \*  
from market  
-- d.Prevents deletion of any row from only one table.  
create function *prevent\_deleting*() returns trigger as  
$$  
begin  
 raise exception 'Delete prevented';  
end;  
$$ language plpgsql;  
  
create trigger prevent\_deleting\_trigger  
 before delete  
 on market  
 for each row  
execute function *prevent\_deleting*();  
alter table market  
 enable trigger prevent\_deleting\_trigger  
delete  
from market  
where tovar\_id = 2;  
  
  
-- e.Launches functions 1.d and 1.e.  
  
create table accounts  
(  
  
 id integer,  
 username varchar(10),  
 password varchar(10)  
);  
  
create or replace function *launch\_functions*() returns trigger as  
$$  
begin  
 raise notice '%', *check\_password*(new.password);  
 raise notice '%', *two\_outputs*(new.id);  
 return new;  
end;  
$$ LANGUAGE plpgsql;  
create trigger launch\_trigger  
 before insert  
 on accounts  
 for each row  
execute procedure *launch\_functions*();  
alter table accounts  
 enable trigger launch\_trigger;  
insert into accounts  
values (1, 'Bakha', 'Aakytzhan');  
-- 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 *increase\_and\_provide\_features*(experience integer)  
 language plpgsql  
as  
 $$  
 declare cnt int;  
begin  
 cnt = experience / 2;  
 for i in 0..cnt  
 loop  
 update my\_table set discount = discount \* 1.01;  
 end loop;  
 commit ;  
end;  
 $$;  
  
  
--  
-- 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 procedure *increase\_and\_provide*(experience integer)  
language plpgsql  
as  
 $$  
 declare  
 cnt int;  
begin  
 cnt = experience/40;  
 for i in 0..cnt  
 loop  
 update my\_table set salary = salary \* 1.15 where salary = salary;  
 end loop;  
 cnt = experience/5;  
 if(experience > 8) then update my\_table set salary = salary \* 1.15, discount = 20 where salary = salary;  
 end if;  
 commit ;  
end;  
 $$;  
-- Consider the following schema for the task4: