--create database shopping

-- TABLE

create table category (

cateid serial primary key,

catename varchar (15)

);

create table product (

prodid serial primary key,

prodname varchar (40),

prodperunit varchar (20),

prodprice money,

prodstock smallint,

prodstockavailable smallint,

prodcateid serial

);

create table customer (

custid varchar (5) primary key,

custname varchar (40),

custaddress varchar (60),

custcity varchar (15),

custcountry varchar (15)

);

create table orders (

ordername varchar (25) primary key,

ordercreated timestamp,

ordershipname varchar (40),

ordershipaddress varchar (60),

ordershipcity varchar (15),

ordershipcountry varchar (15),

orderqty smallint,

ordersubtotal money,

ordersubdiscount money,

ordertax money,

ordertotal money,

orderstatus varchar (15),

orderidtemp integer,

ordercustid varchar (5)

);

create table lineitems (

liteid serial primary key,

liteprice money,

liteqty smallint,

litediscount real,

liteprodid integer,

liteordername varchar (25)

);

create sequence cateid\_seq;

create sequence custid\_seq;

create sequence liteid\_seq;

-- MIGRATION

create extension dblink;

create foreign data wrapper postgres;

create server localhost foreign data wrapper postgres options

(hostaddr '127.0.0.1', dbname 'northwind');

create user mapping for postgres server localhost options

(user 'postgres', password '123');

select dblink\_connect ('localhost');

-- 1. loadcategory

create or replace function loadcategory()

--membuat fungsi atau menggganti yg sdh ada

returns table (cateid integer, catename varchar(15)) as $$

--menentukan tipe data kembalian. void = tidak mengembalikan apapun. table = table

begin

truncate table category;

insert into category (cateid, catename)

select \* from dblink ('localhost', 'select category\_id, category\_name from categories')

as data (cateid integer, catename varchar(15));

end; $$ language plpgsql;

select \* from loadcategory();

select \* from category;

drop function loadcategory();

-- 2. loadcustomer

create or replace function loadcustomer()

returns void as $$

begin

truncate table customer;

insert into customer (custid, custname, custaddress, custcity, custcountry)

select \* from dblink ('localhost', 'select customer\_id, company\_name, address, city, country from customers')

as data (custid varchar (5), custname varchar (40), custaddress varchar (60), custcity varchar (15), custcountry varchar (15));

end; $$ language plpgsql;

select \* from loadcustomer();

select \* from customer;

drop function loadcustomer();

-- 3. loadproduct

create or replace function loadproduct()

returns void as $$

begin

truncate table product;

insert into product (prodid, prodname, prodperunit, prodprice, prodstock, prodstockavailable)

select \* from dblink ('localhost', 'select product\_id, product\_name, quantity\_per\_unit, unit\_price, units\_in\_stock, discontinued from products')

as data (prodid integer, prodname varchar (40), prodperunit varchar (20), prodprice money, prodstock smallint, prodstockavailable smallint);

end; $$ language plpgsql;

select \* from loadproduct();

select \* from product;

drop function loadproduct();

-- 4. loadorders

CREATE SEQUENCE seq\_ord\_number

INCREMENT 1

MINVALUE 1

MAXVALUE 9223372036854775807

START 1

create or replace function loadOrders()

returns void as $$

begin

truncate table orders;

insert into orders (ordername, ordershipname, ordershipaddress, ordershipcity, ordershipcountry, orderstatus, orderidtemp)

select CONCAT('ORD',to\_char(now(),'YYYYMMDD'),'',lpad(''||nextval('seq\_ord\_number')::text,4,'0')),

ship\_name, ship\_address, ship\_city, ship\_country,

case when ship\_region is null then 'New'

when ship\_region = 'AK' then 'Cancelled'

when ship\_region = 'OR' then 'Closed'

else 'Shipping' end as orderstatus,

order\_id

from dblink ('localhost', 'select ship\_name, ship\_address, ship\_city, ship\_country, ship\_region, order\_id from orders')

as data (ship\_name varchar (40), ship\_address varchar (60), ship\_city varchar (15), ship\_country varchar (15), ship\_region varchar (15), order\_id integer);

end; $$ language plpgsql;

select \* from loadOrders();

select \* from orders;

drop function loadOrders();

--kecuali orderqty-ordertotal

-- 5. loadlineitems

create or replace procedure loadlineitems()

language plpgsql as $$

begin

truncate table lineitems;

insert into lineitems (liteprice, liteqty, litediscount, liteprodid)

select \* from dblink ('localhost', 'select unit\_price, quantity, discount, product\_id from order\_details')

as data (liteprice money, liteqty smallint, litediscount real, liteprodid integer);

commit; end; $$

call loadlineitems();

select \* from lineitems;

drop procedure loadlineitems;

-- 6. getordername()

create or replace function getordername (order\_date timestamp, order\_id integer)

returns text as $$

declare order\_date\_formatted text;

begin

truncate table orders;

order\_date\_formatted := to\_char(order\_date, 'YYYYMMDD');

return CONCAT('ORD', order\_date\_formatted, '', lpad(order\_id::text, 4, '0'));

end; $$ language plpgsql;

--contoh: select \* from getordername ('2023-02-22 10:00:00', 123);

drop function getordername ();

-- 7. cleartabletarget()

create or replace function ClearTableTarget()

returns void as $$

begin

delete from --nama table--;

end; $$ language plpgsql;

select ClearTableTarget();

--contoh:

create or replace function ClearTableTarget()

returns void as $$

begin

delete from category;

end; $$ language plpgsql;

select ClearTableTarget();

select\*from category

drop function ClearTableTarget();

-- 8. updatesumorder()

create or replace procedure updatesumorder(p\_ordername text) as $$

begin

update orders set ordertotal = ( select sum(liteqty \* liteprice \* (1 - litediscount))

from lineitems where ordername = p\_ordername )

where ordername = p\_ordername;

end; $$ language plpgsql;

call updatesumorder('OrderNumber001');

select \* from orders order by ordername;

drop procedure updatesumorder();

-- ALTER TABLE (belum dipakai)

--prod

alter table product add constraint fk\_prodcateid

foreign key (prodcateid) references category(cate\_id);

--order

alter table orders add constraint fk\_ordercustid

foreign key (ordercustid) references customer(cust\_id);

--lineitems

alter table lineitems add constraint fk\_liteprodid

foreign key (liteprodid) references product(prodid);

alter table lineitems add constraint fk\_liteordername

foreign key (liteordername) references orders(ordername);

create or replace function loadorders()

returns void as $$

begin

truncate table orders;

insert into orders (ordername, ordershipname, ordershipaddress, ordershipcity,

ordershipcountry, orderqty, ordersubtotal, ordersubdiscount, ordertax, ordertotal)

select

concat('ORD', to\_char(now(), 'yyyymmdd'), '', lpad(''||nextval('seq\_ord\_number')::text, 4, '0')),

--::konversi data ke text, 4 string

ship\_name, ship\_address, city, ship\_country,

coalesce(sum(li.liteqty), 0), coalesce(sum(li.liteprice)::money, 0),

coalesce(sum(li.litediscount)::money, 0), (sum(li.liteprice \* 0.1)::money, 0),

coalesce(sum((li.liteprice - li.litediscount) \* 1.1)::money, 0)

--coalesce menghitung total qty dari seluruh item. sum menjumlah semua nilai kolom liteqty

from dblink('localhost', 'select ship\_name, ship\_address, ship\_city, ship\_country, order\_id from orders')

as data (ship\_name varchar(40), ship\_address varchar(60), ship\_city varchar(15), ship\_country varchar(15), orderidtemp integer)

left join dblink('localhost', 'select liteid, liteqty, liteprice, litediscount, order\_id from lineitems')

as li (liteid integer, liteqty integer, liteprice money, litediscount money, orderidtemp integer)

on orderidtemp = li.orderidtemp

group by ship\_name, ship\_address, ship\_city, ship\_country, orderidtemp;

end; $$ language plpgsql;