Use this document to capture the screenshots requested in the instructions for assignment 3.0.

**Name**:

**Date:**

**The submission template uses the following format :**

* Document question number and your code describing what each query is doing.
* Paste in your code.
* Paste in a screenshot of the results running your code.

1. **Document question number and your code describing what each query is doing.**

Paste in your code.

create or replace function next\_in\_stock\_dvd(cust\_id numeric) returns text

as $$

declare

next\_dvd text;

begin

select d.dvdtitle into next\_dvd

from dvd d join rentalqueue rq on rq.dvdid=d.dvdid

join member m on m.memberid=rq.memberid

where m.memberid=cust\_id and d.dvdquantityonhand>0

order by rq.dateaddedinqueue limit 1;

return next\_dvd;

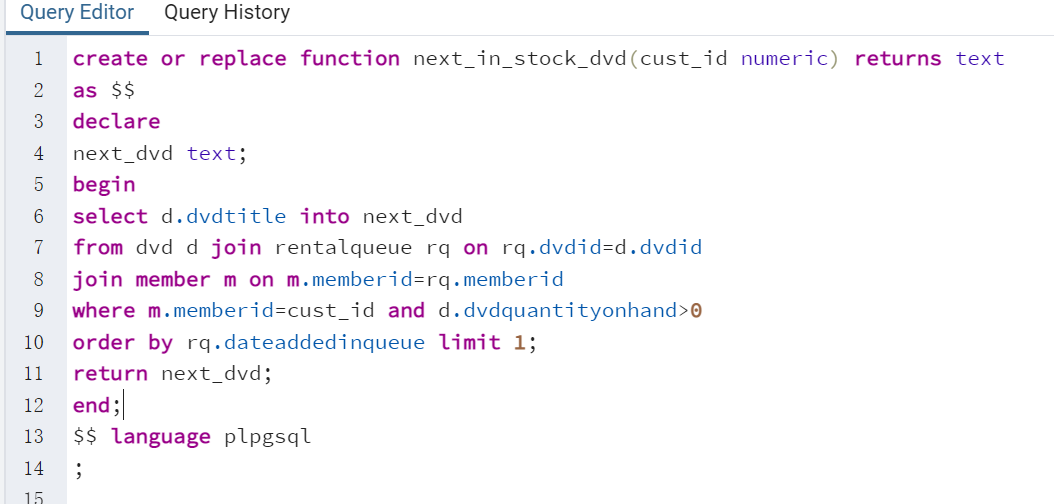
end;

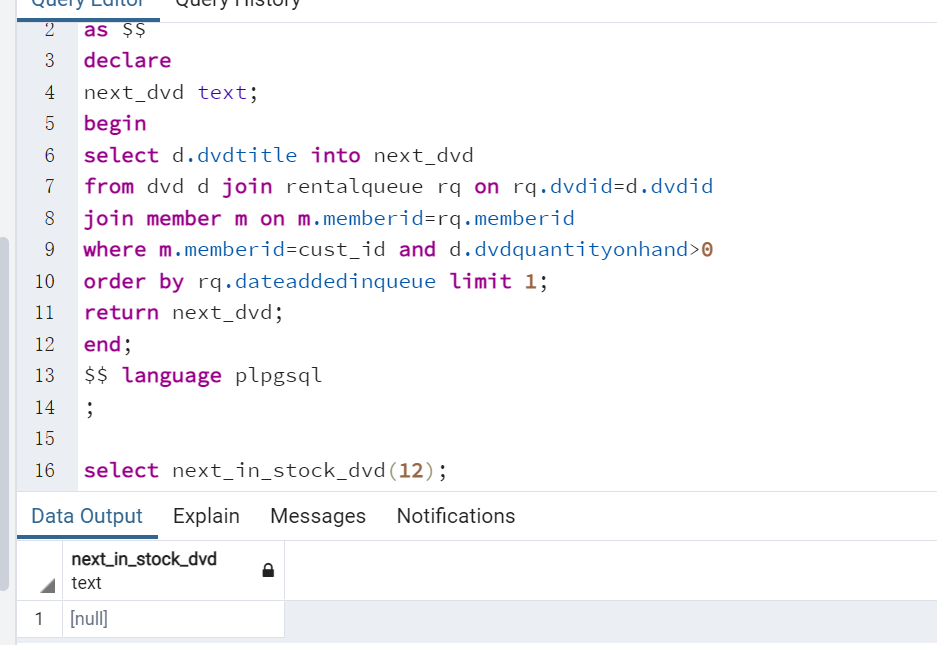
$$ language plpgsql

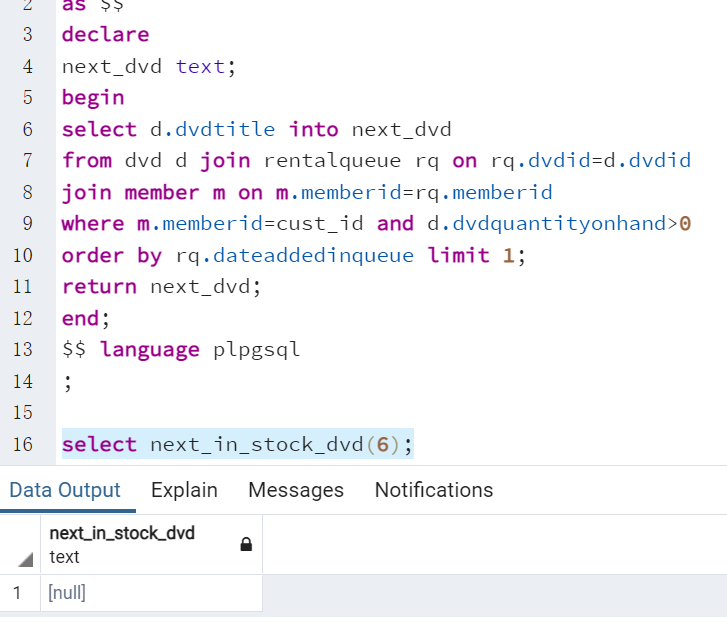
;

select next\_in\_stock\_dvd(12);

Paste in a screenshot of the results running your code.







1. **Document question number and your code describing what each query is doing.**

Paste in your code.

create or replace function additional\_dvds(cust\_id numeric) returns integer

as $$

declare

additional integer;

begin

with tmp

as

(

select t1.memberid, t1.onetime\_permit, t1.has\_rent,

t1.membershiplimitpermonth, case when t2.plan\_rent is null then 0 else t2.plan\_rent end as plan\_rent,

case when t3.his\_has\_rent is null then 0 else t3.his\_has\_rent end as his\_has\_rent

from

(

select m.memberid,

left(ms.membershiptype,1)::integer as onetime\_permit, ms.membershiplimitpermonth,

count(r.dvdid) as has\_rent

from member m

left join membership ms on m.membershipid=ms.membershipid

left join rental r on r.memberid=m.memberid

left join payment p on p.memberid=m.memberid

where r.rentalreturneddate is null and r.rentalrequestdate<p.amountpaiduntildate

group by m.memberid, ms.membershiptype, ms.membershiplimitpermonth

union

select m.memberid,

left(ms.membershiptype,1)::integer as onetime\_permit, ms.membershiplimitpermonth,

count(r.dvdid) as has\_rent

from member m

left join membership ms on m.membershipid=ms.membershipid

left join rental r on r.memberid=m.memberid

left join payment p on p.memberid=m.memberid

where r.rentalrequestdate is null

group by m.memberid, ms.membershiptype, ms.membershiplimitpermonth

) t1

full join

(

select m.memberid,

count(rq.dvdid) as plan\_rent

from member m

join membership ms on m.membershipid=ms.membershipid

join payment p on p.memberid=m.memberid

join rentalqueue rq on rq.memberid=m.memberid

where rq.dateaddedinqueue<p.amountpaiduntildate

group by m.memberid

) t2

on t1.memberid=t2.memberid

full join

(

select m.memberid,

left(ms.membershiptype,1)::integer as onetime\_permit, ms.membershiplimitpermonth,

count(r.dvdid) as his\_has\_rent

from member m

left join membership ms on m.membershipid=ms.membershipid

left join rental r on r.memberid=m.memberid

left join payment p on p.memberid=m.memberid

where r.rentalrequestdate <p.amountpaiduntildate and r.rentalreturneddate<p.amountpaiduntildate

group by m.memberid, ms.membershiptype, ms.membershiplimitpermonth

) t3 on t1.memberid=t3.memberid

where t1.has\_rent<=t1.onetime\_permit

)

select case when onetime\_permit-has\_rent<membershiplimitpermonth-(his\_has\_rent+has\_rent)

and has\_rent+plan\_rent<onetime\_permit

then onetime\_permit-has\_rent

else membershiplimitpermonth-(his\_has\_rent+has\_rent)

end into additional

from tmp

where his\_has\_rent+has\_rent+plan\_rent<membershiplimitpermonth

and memberid=$1

;

return additional;

end;

$$ language plpgsql

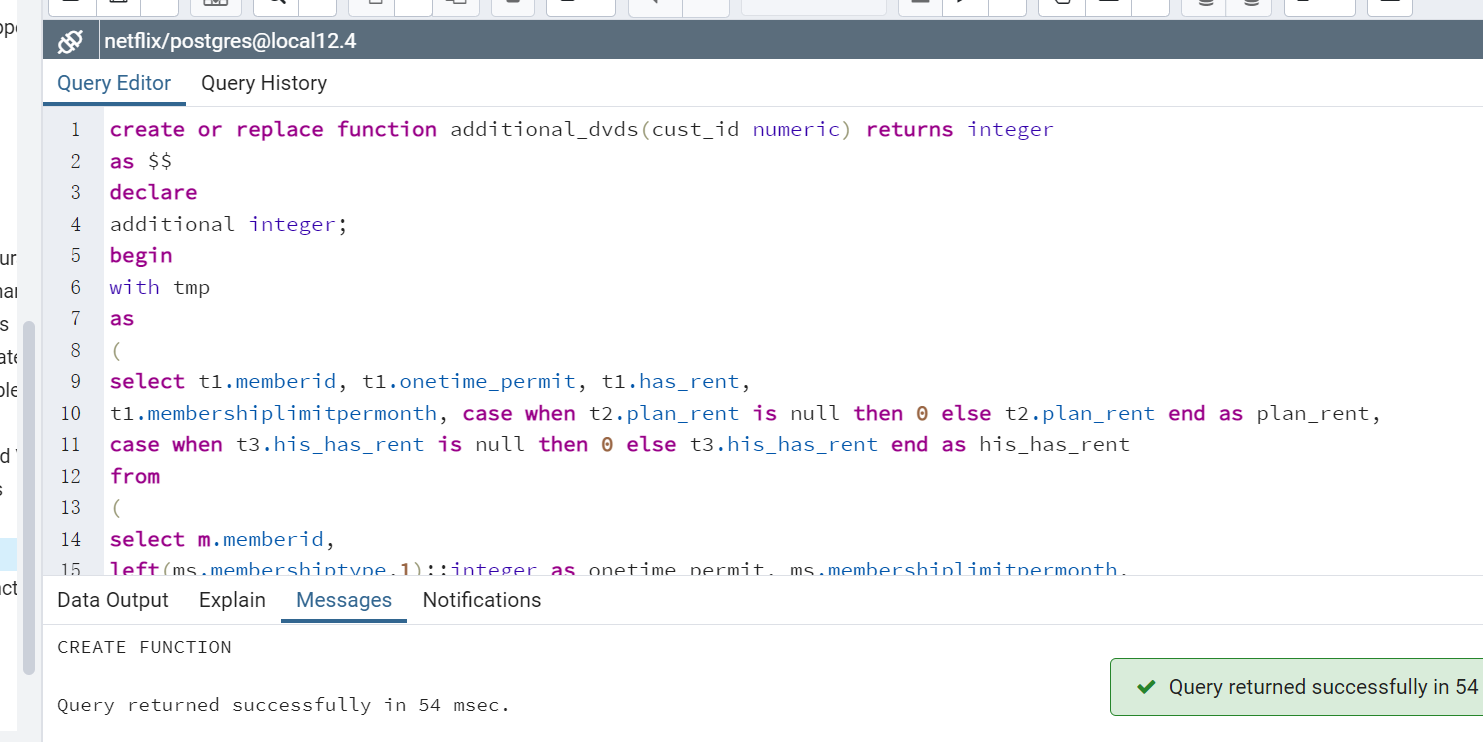
;

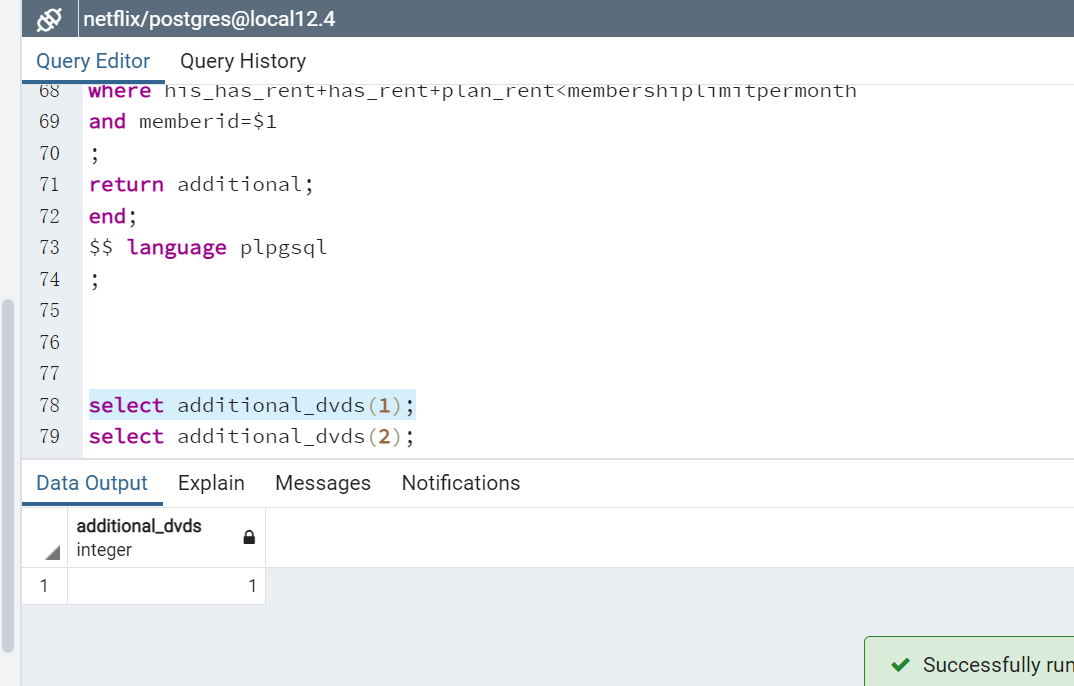
select additional\_dvds(1);

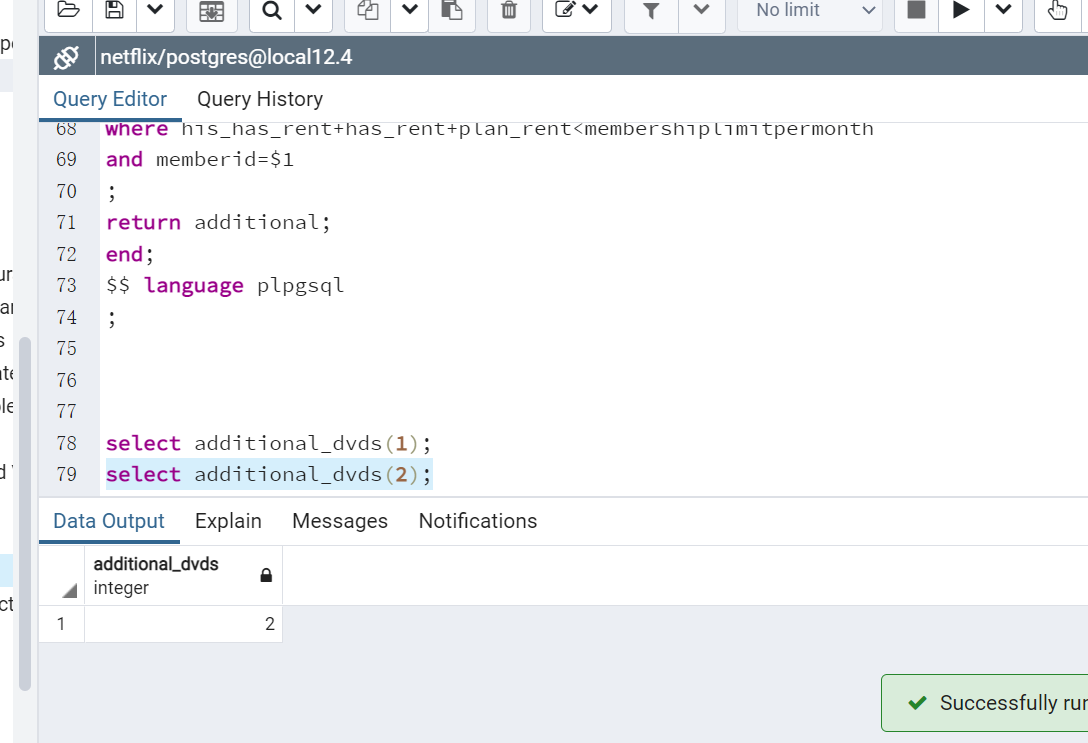
select additional\_dvds(2);

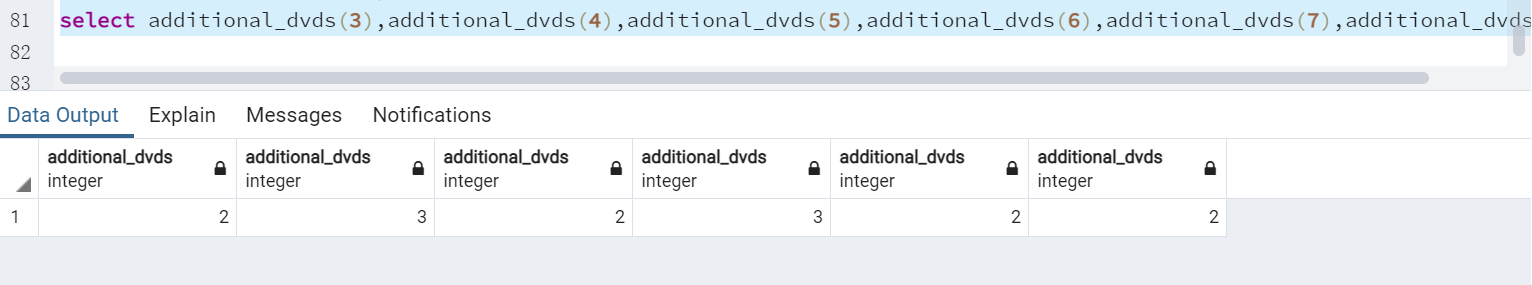
select additional\_dvds(3),additional\_dvds(4),additional\_dvds(5),additional\_dvds(6),additional\_dvds(7),additional\_dvds(8);

Paste in a screenshot of the results running your code.









1. **Document question number and your code describing what each query is doing.**

Paste in your code.

create or replace procedure rent\_dvd(in cust\_id numeric)

as $$

declare

additional integer;

additional\_dvd text;

next\_rent numeric;

dvd\_id numeric;

next\_pay numeric;

lost\_price numeric;

begin

select additional\_dvds(cust\_id) into additional;

select next\_in\_stock\_dvd(cust\_id) into additional\_dvd;

select count(\*) into next\_rent from rental;

select dvdid into dvd\_id from dvd where dvdtitle=additional\_dvd;

if dvd\_id is not null then

call delete\_dvd(cust\_id, dvd\_id);

insert into rental values (next\_rent+1,cust\_id, dvd\_id,now(),null,null);

select membershipdvdlostprice into lost\_price from membership m join member mm on mm.memrbershipid=m.membershipid;

select max(paymentid) into next\_pay from payment;

insert into payment values(next\_pay+1,cust\_id,lost\_price, now(),now());

end if;

end;

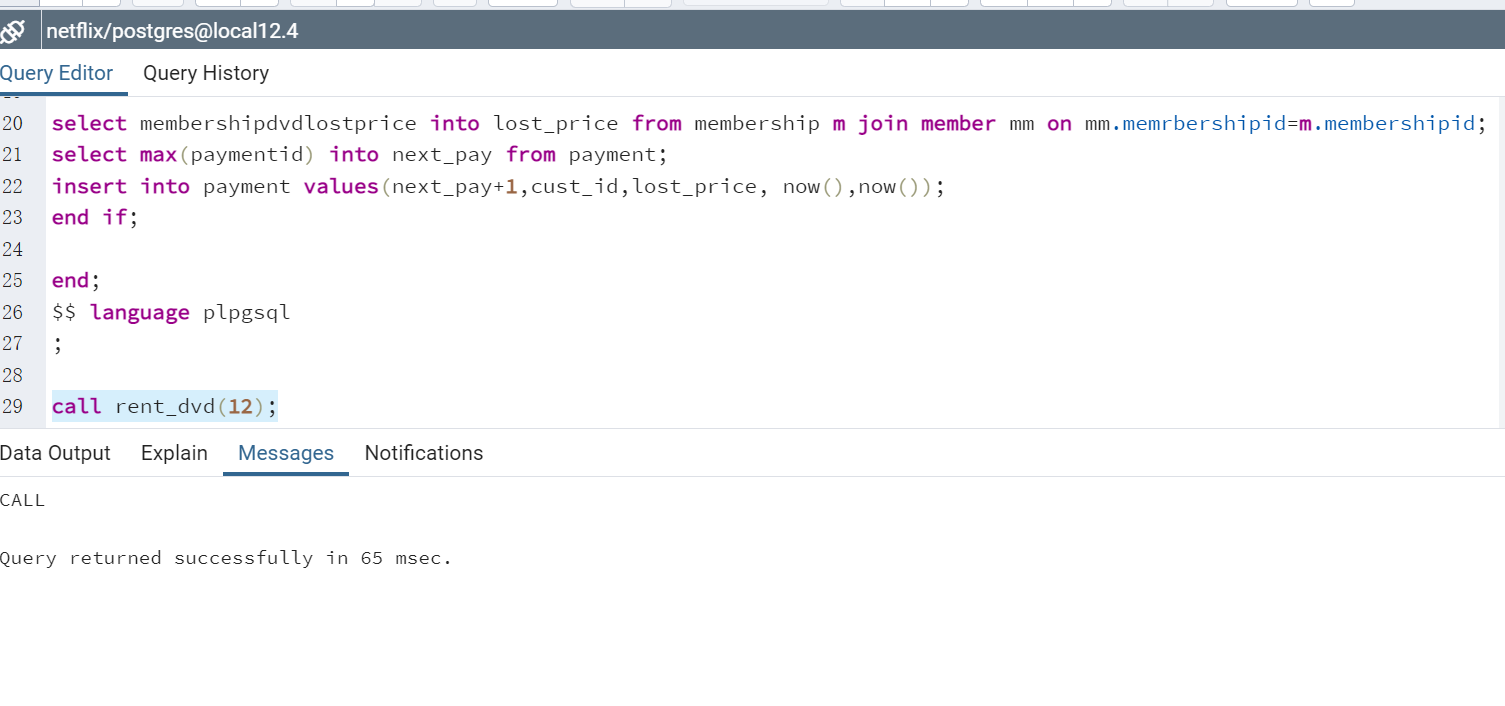
$$ language plpgsql

;

call rent\_dvd(12);

Paste in a screenshot of the results running your code.





Use the **Ask the Facilitators Discussion Board** if you have any questions regarding the how to approach this assignment.

Save your assignment as ***lastnameFirstname\_assign3.0.docx*** and submit it in the *Assignments* section of the course.

For help uploading files please refer to the *Technical Support* page in the syllabus.