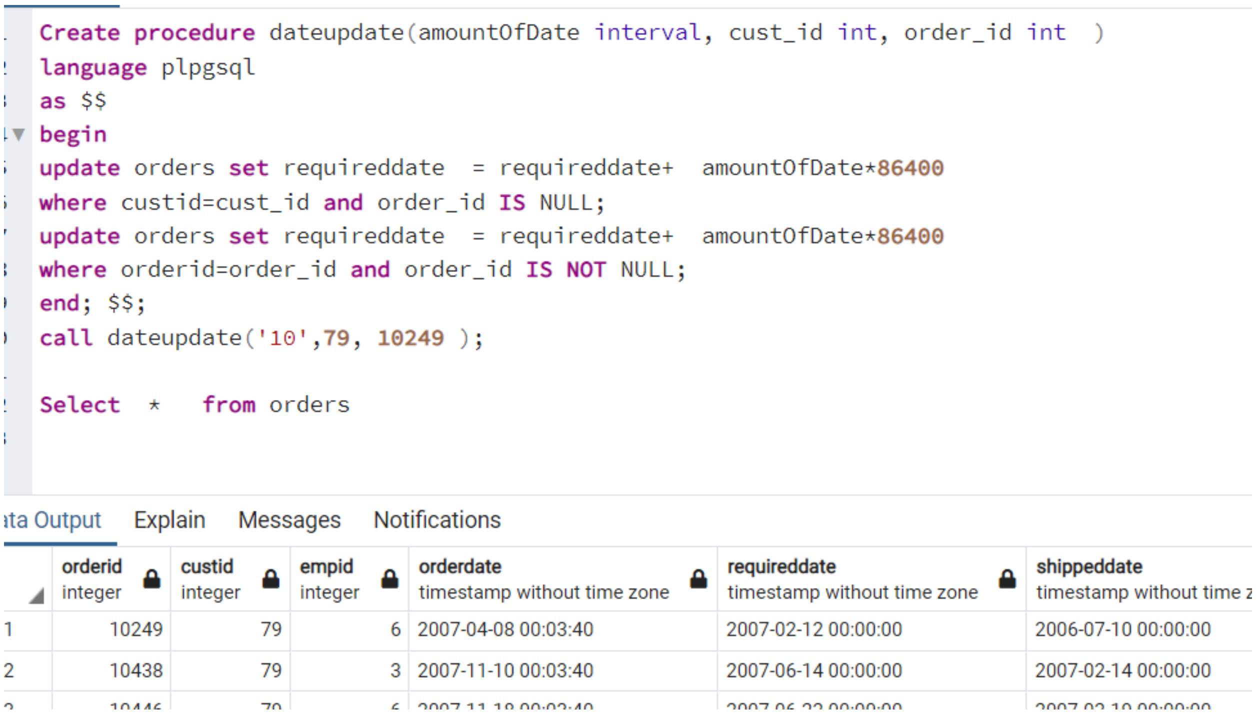
For homework 3 solution. I did not use commit to make permanent change into the dataset.

a) Create a procedure that adds X amount of days (given by the user) to the "requireddate" value based on custid or orderid. If orderid given is NULL, the procedure runs based on custid. The procedure takes three arguments.

Query code-

```
Create procedure dateupdate(amountOfDate interval, cust_id int, order_id int )

language plpgsql

as $$

begin

update orders set requireddate  = requireddate+  amountOfDate*86400

where custid=cust_id and order_id IS NULL;

update orders set requireddate  = requireddate+  amountOfDate*86400

where orderid=order_id and order_id IS NOT NULL;

end; $$;

call dateupdate('10',79, 10249 );

Select  *   from orders
```



```
Create procedure dateupdate(amountOfDate interval, cust_id int, order_id int  )
language plpgsql
as $$
begin
update orders set requireddate  = requireddate+  amountOfDate*86400
where custid=cust_id and order_id IS NULL;
update orders set requireddate  = requireddate+  amountOfDate*86400
where orderid=order_id and order_id IS NOT NULL;
end; $$;
call dateupdate('10',79, 10249 );

Select  *   from orders
```

	orderid integer	custid integer	empid integer	orderdate timestamp without time zone	requireddate timestamp without time zone	shippeddate timestamp without time z
1	10249	79	6	2007-04-08 00:03:40	2007-02-12 00:00:00	2006-07-10 00:00:00
2	10438	79	3	2007-11-10 00:03:40	2007-06-14 00:00:00	2007-02-14 00:00:00
3	10446	79	6	2007-11-18 00:03:40	2007-06-22 00:00:00	2007-02-10 00:00:00

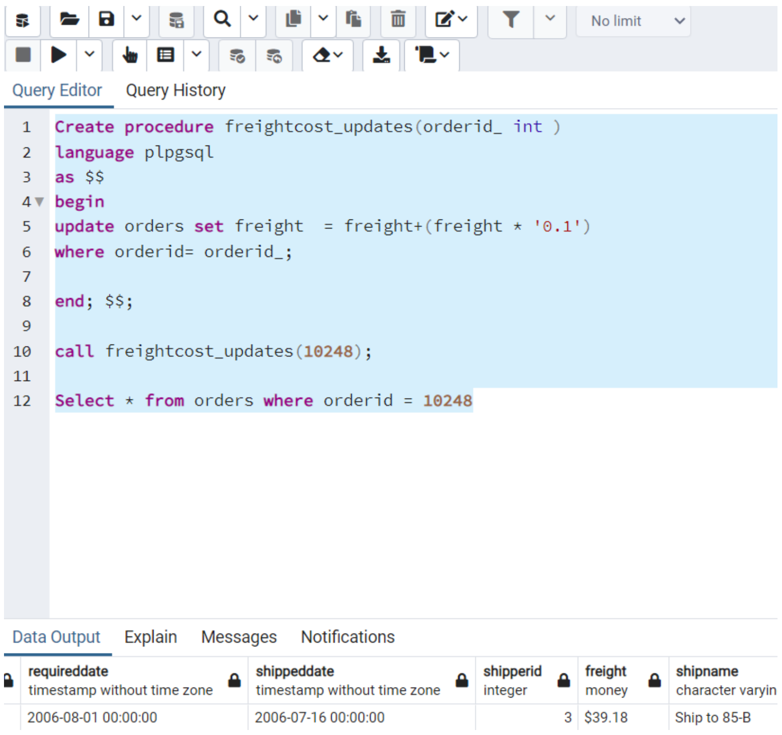
Here, updating 10 days and taking input number of days, cust id and order id.

b) Create a procedure that adds 10 % to the freight money

since it does not mention if it was a task to update all rows or any specific rows, here procedure I have written it takes order id as input and add 10% to the freight cost.

Query code-

```
Create procedure freightcost_updates(orderid_ int )
language plpgsql
as $$
begin
update orders set freight  = freight+(freight * '0.1')
where orderid= orderid_;
end; $$;
call freightcost_updates(10248);
Select * from orders where orderid = 10248
```



Query Editor Query History

```
Create procedure freightcost_updates(orderid_ int )
language plpgsql
as $$
begin
update orders set freight  = freight+(freight * '0.1')
where orderid= orderid_;

end; $$;

call freightcost_updates(10248);

Select * from orders where orderid = 10248
```

Data Output Explain Messages Notifications

```
requireddate                    shippeddate                     shipperid  freight  shipname
timestamp without time zone     timestamp without time zone     integer    money    character varyin
2006-08-01 00:00:00             2006-07-16 00:00:00                     3  $39.18   Ship to 85-B
```

C) Create a procdeure that rounds the freight costs to nearest 10 €

Procedure that it takes order id as input and round 10 to freight cost:

Query code-

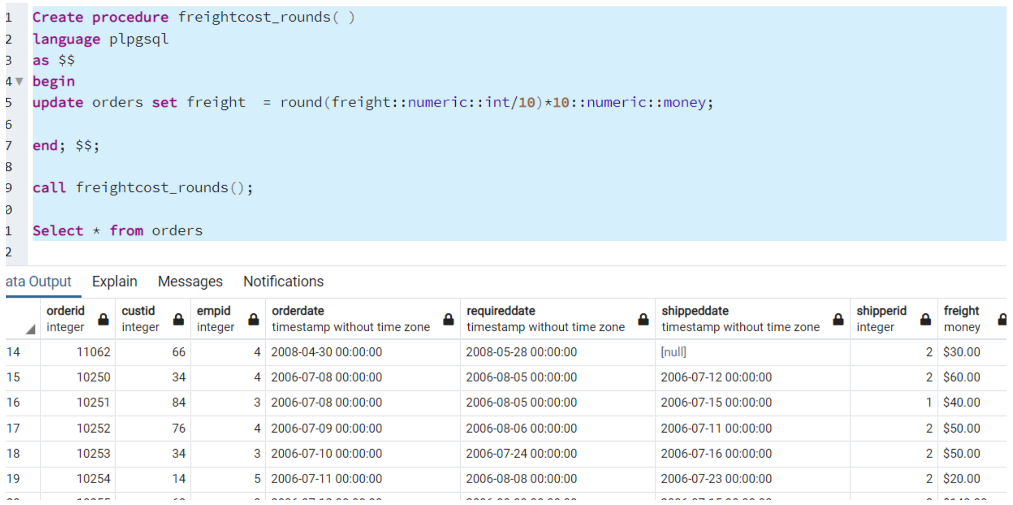
```
Create procedure freightcost_round(orderid_ int )
language plpgsql
as $$
begin
update orders set freight  = round(freight::numeric::int/10)*10::numeric::money
where orderid= orderid_;
end; $$;
call freightcost_round(10248);
Select * from orders where orderid = 10248
```

Procedure to target all freight rows and round it nearest to 10:

Query code-

```
Create procedure freightcost_rounds( )
```

```
language plpgsql
as $$
begin
update orders set freight  = round(freight::numeric::int/10)*10::numeric::money;
end; $$;
call freightcost_rounds();
Select * from orders
```



```
Create procedure freightcost_rounds( )
language plpgsql
as $$
begin
update orders set freight  = round(freight::numeric::int/10)*10::numeric::money;

end; $$;

call freightcost_rounds();

Select * from orders
```

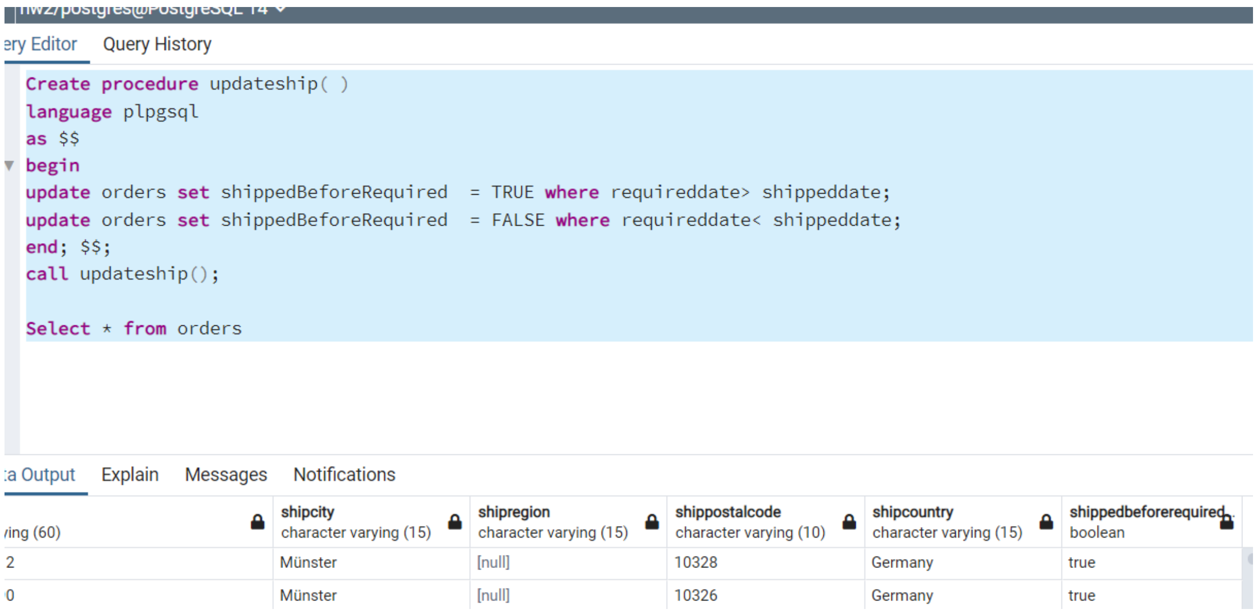
ata Output Explain Messages Notifications

	orderid integer	custid integer	empid integer	orderdate timestamp without time zone	requireddate timestamp without time zone	shippeddate timestamp without time zone	shipperid integer	freight money
14	11062	66	4	2008-04-30 00:00:00	2008-05-28 00:00:00	[null]	2	$30.00
15	10250	34	4	2006-07-08 00:00:00	2006-08-05 00:00:00	2006-07-12 00:00:00	2	$60.00
16	10251	84	3	2006-07-08 00:00:00	2006-08-05 00:00:00	2006-07-15 00:00:00	1	$40.00
17	10252	76	4	2006-07-09 00:00:00	2006-08-06 00:00:00	2006-07-11 00:00:00	2	$50.00
18	10253	34	3	2006-07-10 00:00:00	2006-07-24 00:00:00	2006-07-16 00:00:00	2	$50.00
19	10254	14	5	2006-07-11 00:00:00	2006-08-08 00:00:00	2006-07-23 00:00:00	2	$20.00

d) Add a new column 'shippedBeforeRequired' to Orders table (using ALTER command) of boolean type. Create a procedure that sets 'shippedBeforeRequired' to true if shippeddate is smaller than requrieddate and false if vice-versa

Query code-

```
alter table orders ADD COLUMN shippedBeforeRequired boolean

Create procedure updateship( )
language plpgsql
as $$
begin
update orders set shippedBeforeRequired  = TRUE where requireddate> shippeddate;
update orders set shippedBeforeRequired  = FALSE where requireddate< shippeddate;
end; $$;
call updateship();

Select * from orders
```



```
Create procedure updateship( )
language plpgsql
as $$
begin
update orders set shippedBeforeRequired  = TRUE where requireddate> shippeddate;
update orders set shippedBeforeRequired  = FALSE where requireddate< shippeddate;
end; $$;
call updateship();

Select * from orders
```

ving (60)	shipcity character varying (15)	shipregion character varying (15)	shippostalcode character varying (10)	shipcountry character varying (15)	shippedbeforerequired boolean
2	Münster	[null]	10328	Germany	true
0	Münster	[null]	10326	Germany	true