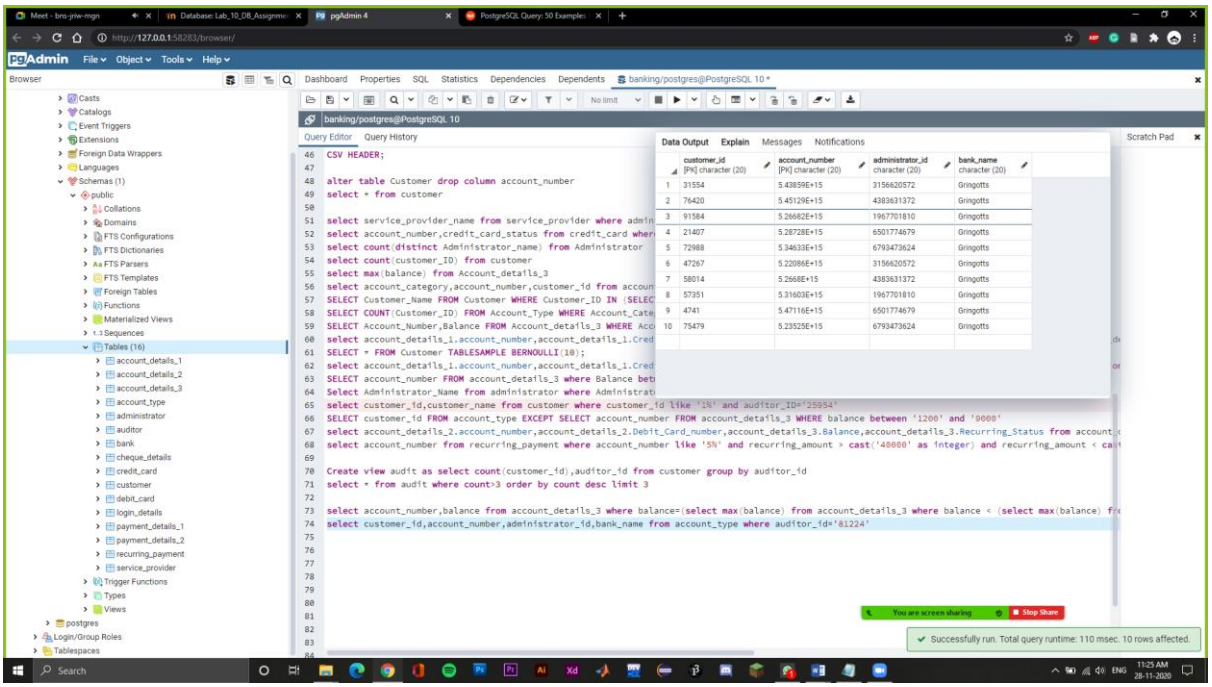
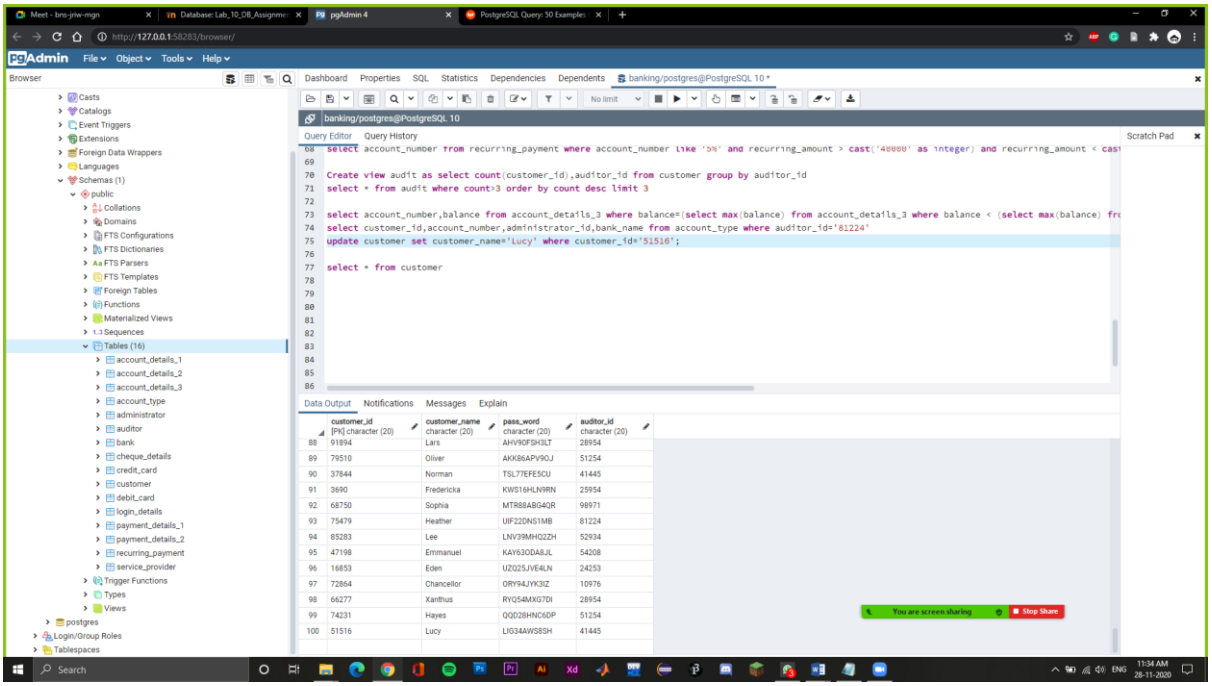
select customer_id,account_number,administrator_id,bank_name from account_type where auditor_id='81224'

->Showing customer id, account number, administrator id, bank name when auditor id is 81224.



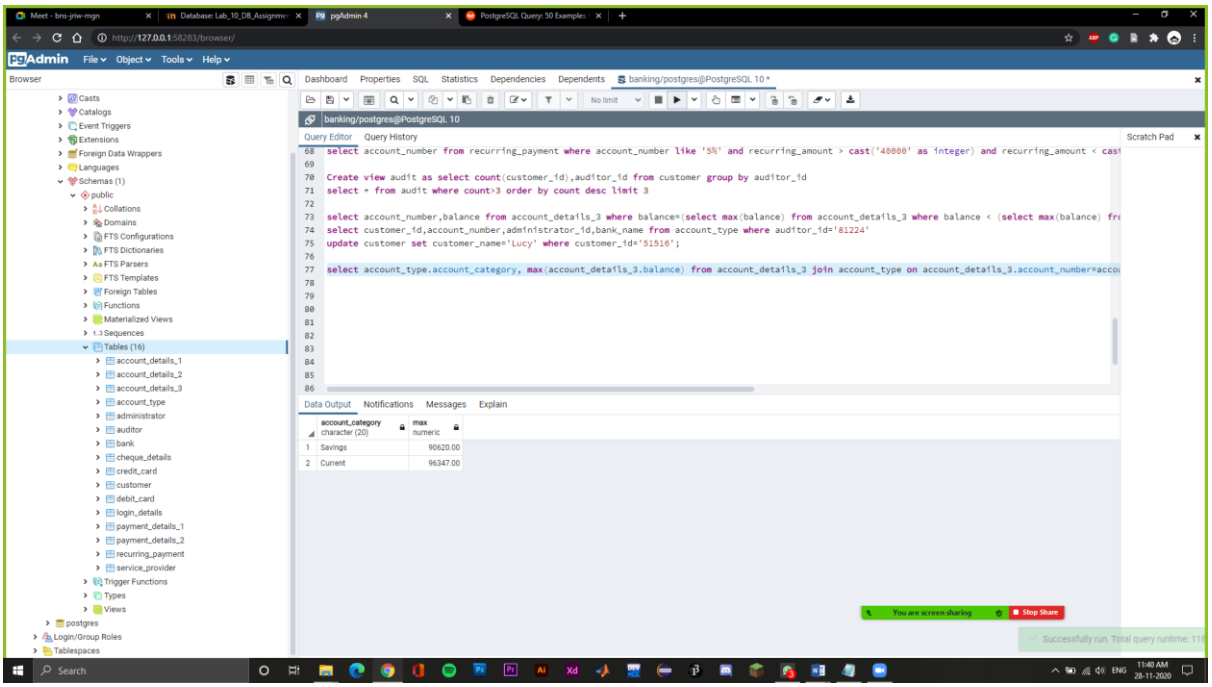
update customer set customer_name='Lucy' where customer_id='51516';

->Update customer name as lucy whose customer id is 51516



select account_type.account_category, max(account_details_3.balance) from account_details_3 join account_type on account_details_3.account_number=account_type.account_number group by account_type.account_category

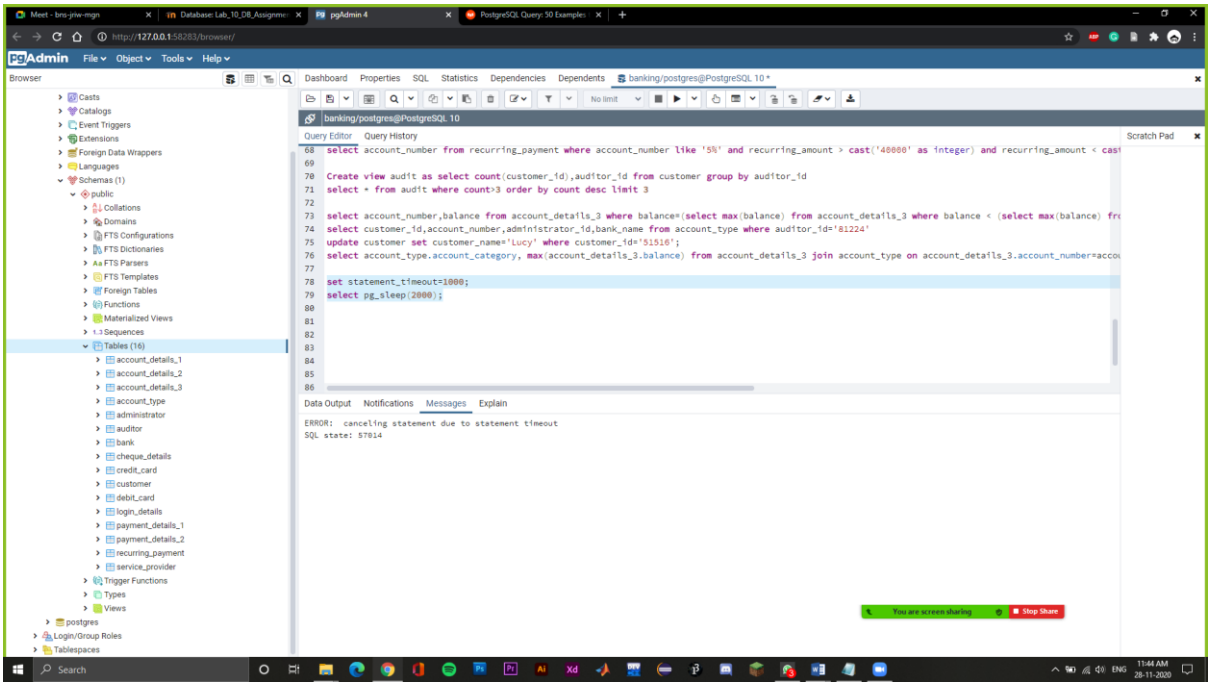
->Show highest balance of each account category



set statement_timeout=1000;

select pg_sleep(2000);

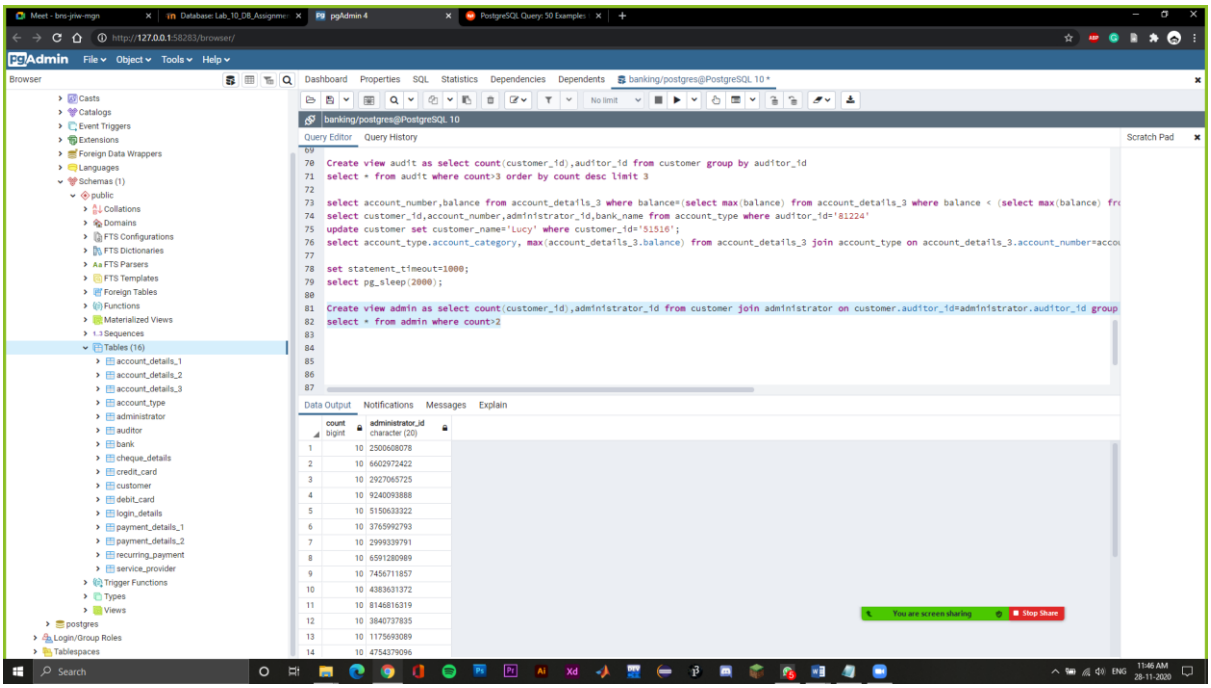
->Set max query time as 1000 ms



Create view admin as select count(customer_id),administrator_id from customer join administrator on customer.auditor_id=administrator.auditor_id group by administrator_id

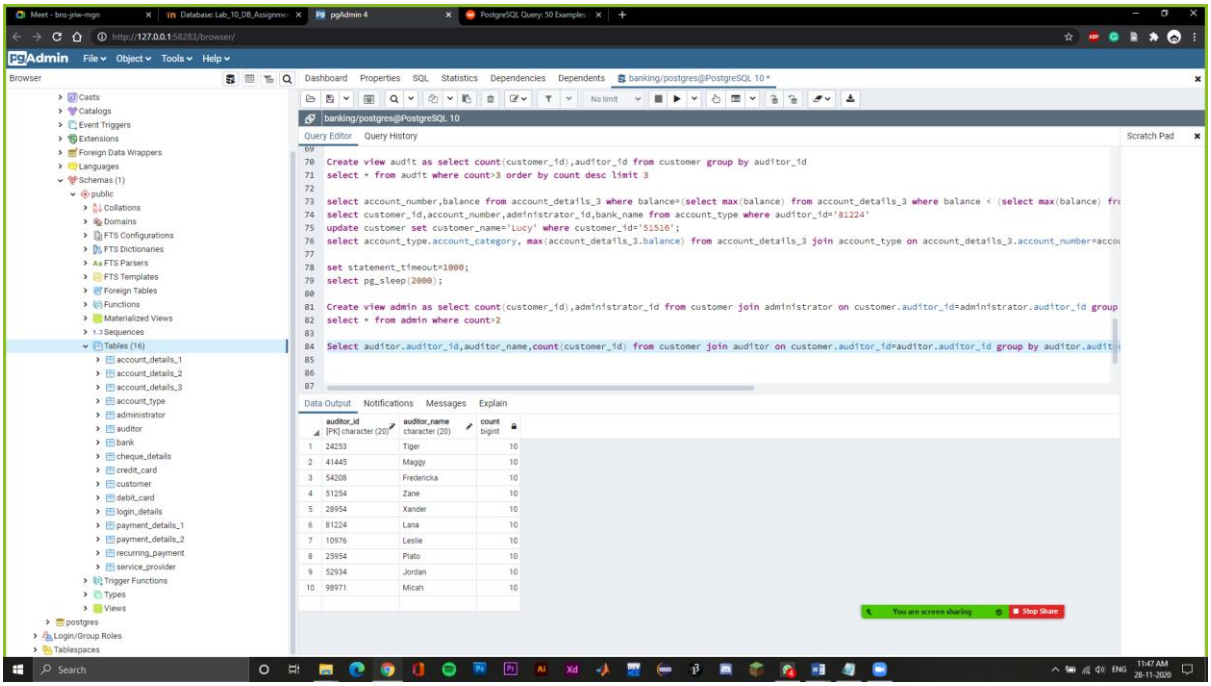
select * from admin where count>2

->Created a view named admin which shows administrator id and count of customer id it serves and below that we have printed down where count is greater than 2.



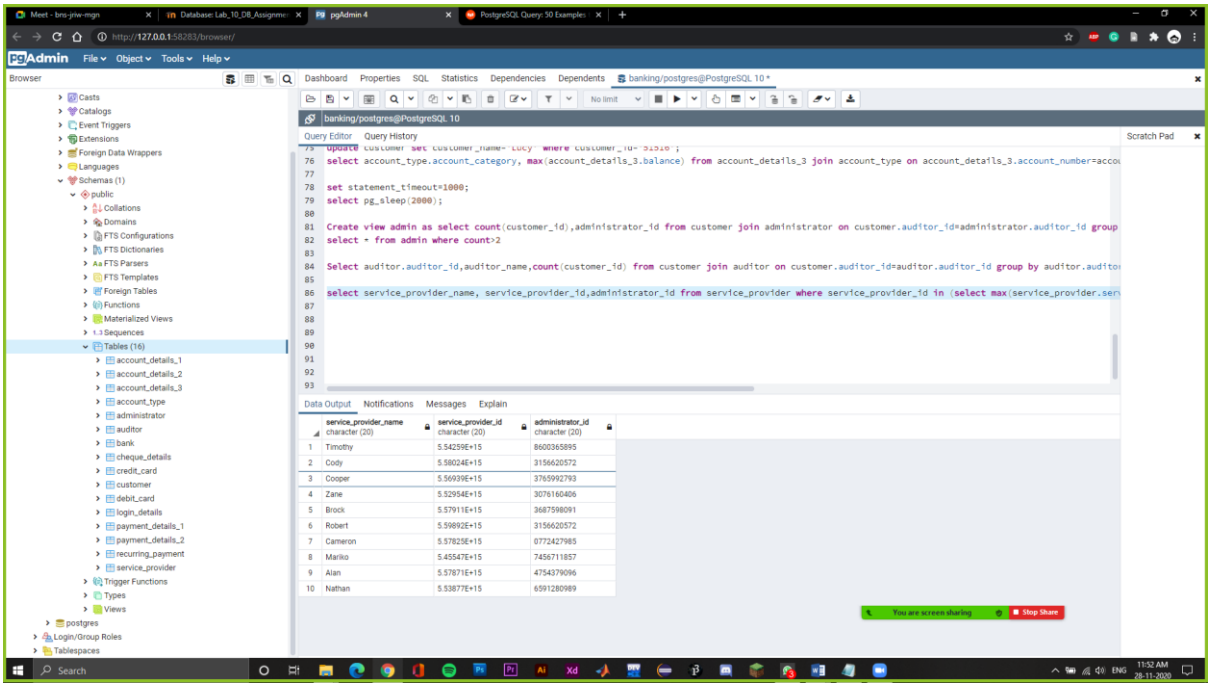
Select auditor.auditor_id,auditor_name,count(customer_id) from customer join auditor on customer.auditor_id=auditor.auditor_id group by auditor.auditor_id

->Showing auditor id, auditor name and count of the customer id by performing join operation.



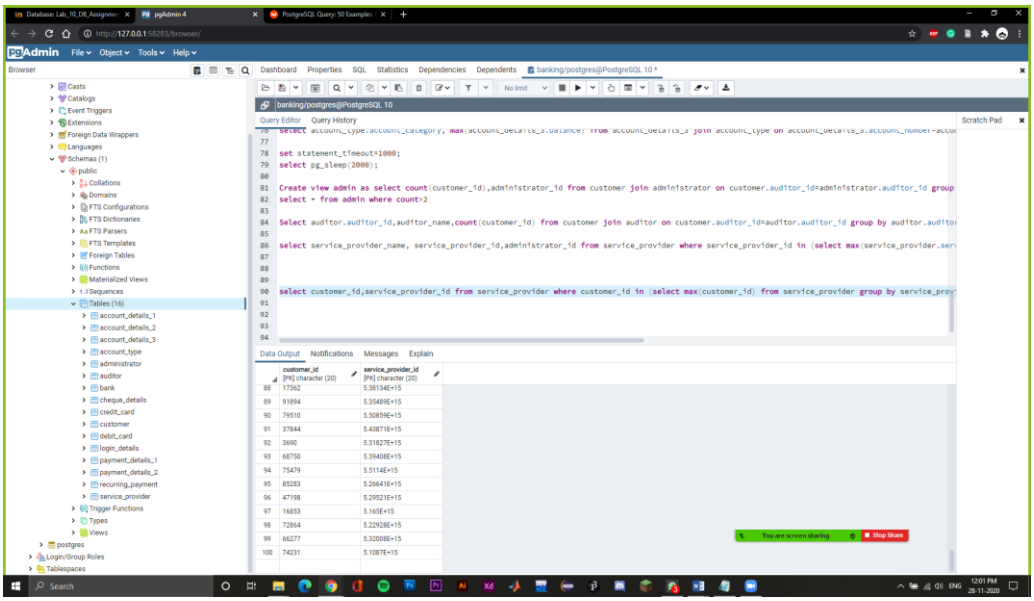
select service_provider_name, service_provider_id,administrator_id from service_provider where service_provider_id in (select max(service_provider.service_provider_id) from service_provider group by auditor_id)

->Print service provider name, its ID, corresponding administrator ID where the service provider ID is maximum corresponding to its auditor.



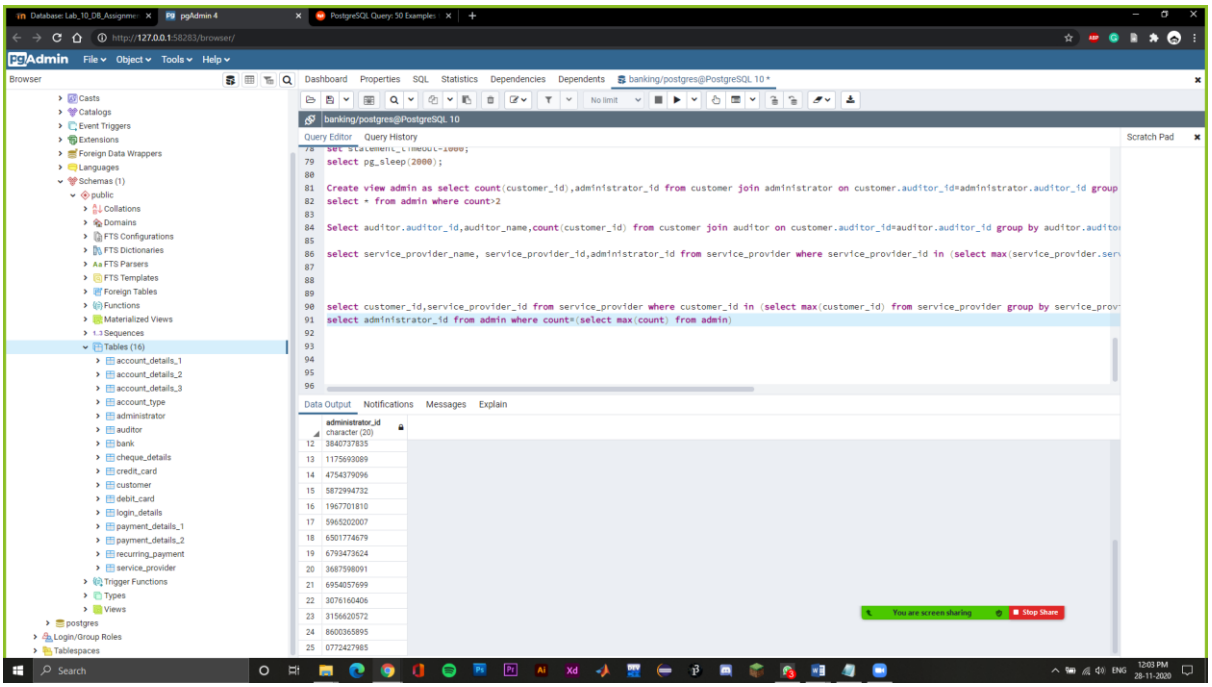
select customer_id,service_provider_id from service_provider where customer_id in (select max(customer_id) from service_provider group by service_provider_id)

->Select Maximum Customer ID from the pool of customers who are being serviced by a certain service provider and show the corresponding service provider ID as well



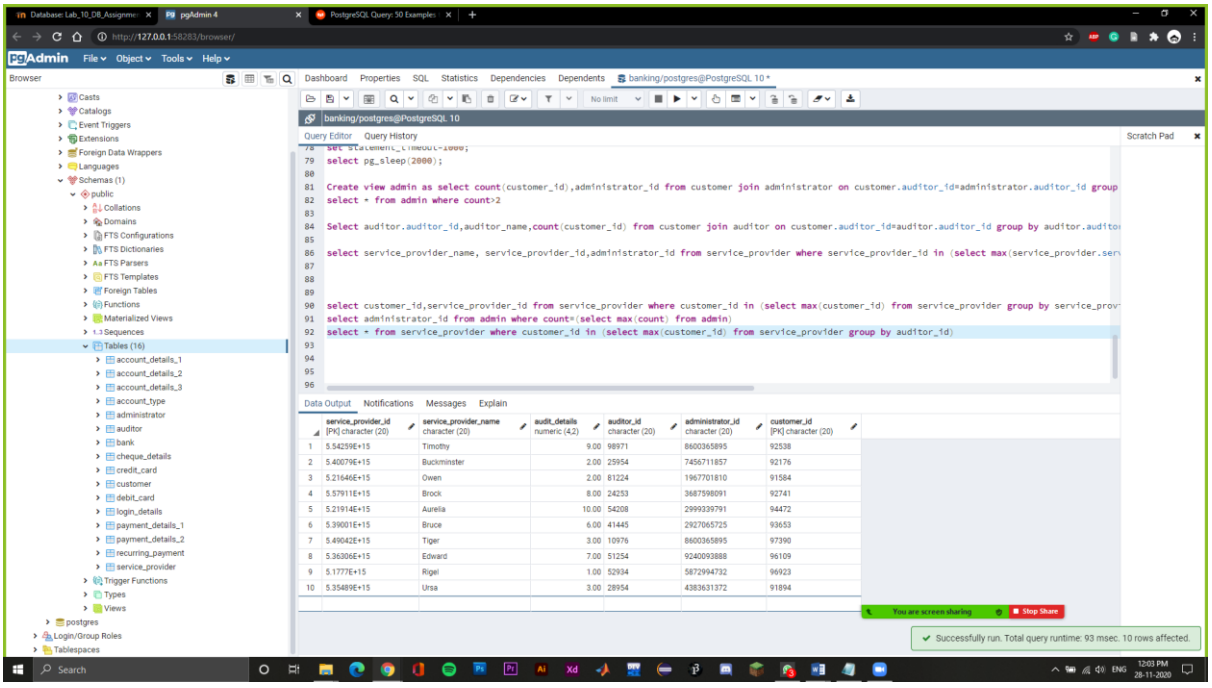
select administrator_id from admin where count=(select max(count) from admin)

->Print ID's of the administrator for those who serve the maximum number of customers.



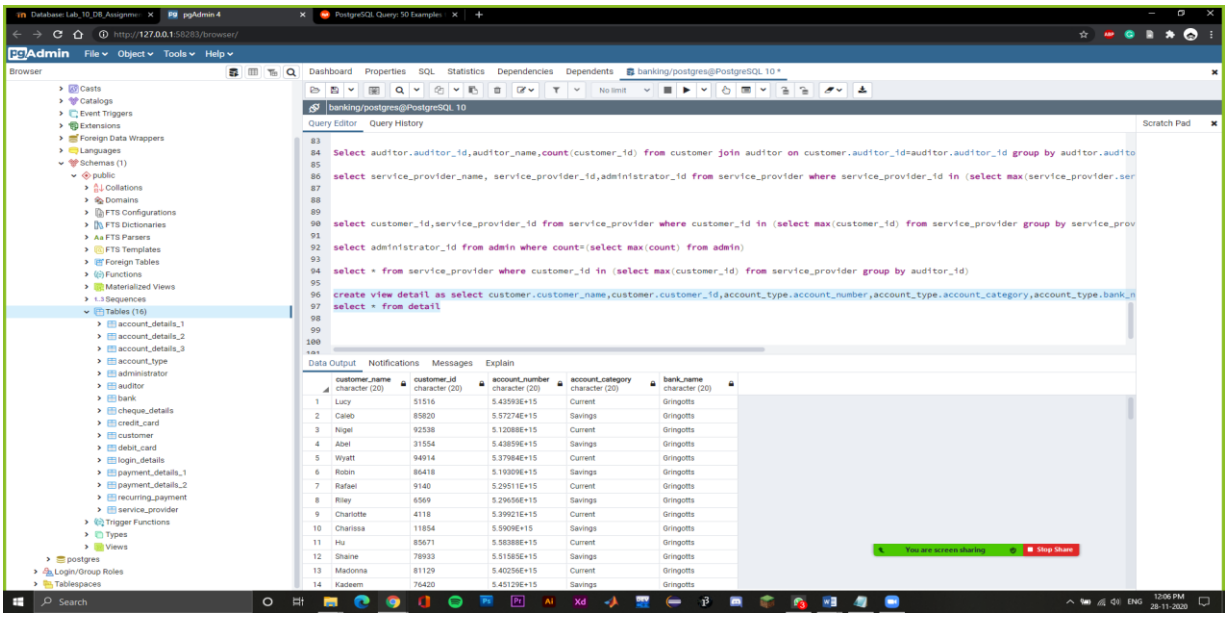
select * from service_provider where customer_id in (select max(customer_id) from service_provider group by auditor_id)

->Select maximum Customer ID along with its details corresponding to an Auditor



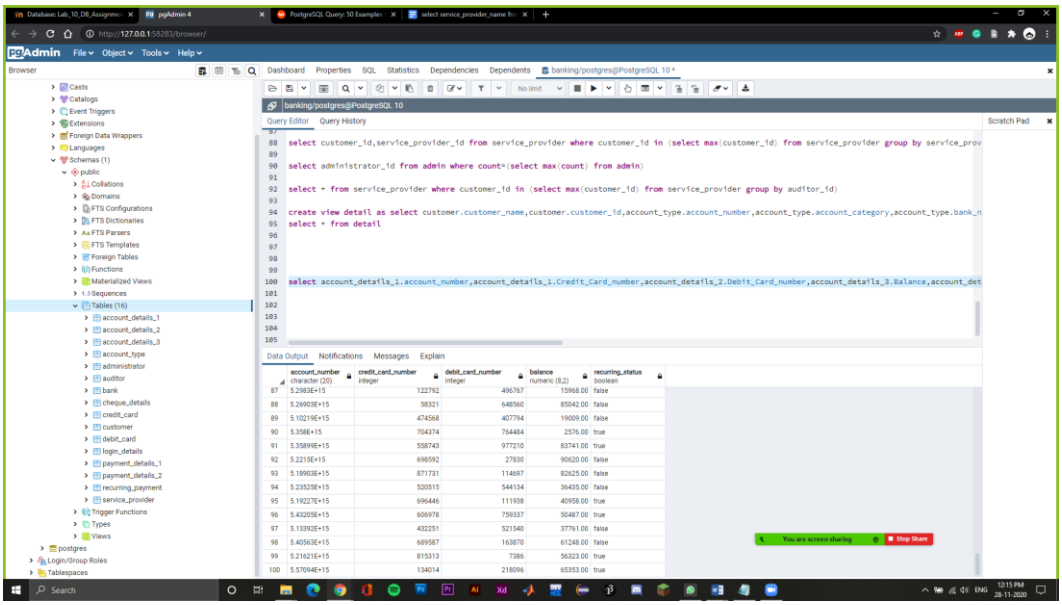
create view detail as select customer.customer_name,customer.customer_id,account_type.account_number,account_type.account_category,account_type.bank_name from customer join account_type on customer.customer_id=account_type.customer_id
select * from detail

->Create a view which shows all the information of a customer like customer name, customer id, account number, account category and bank name.



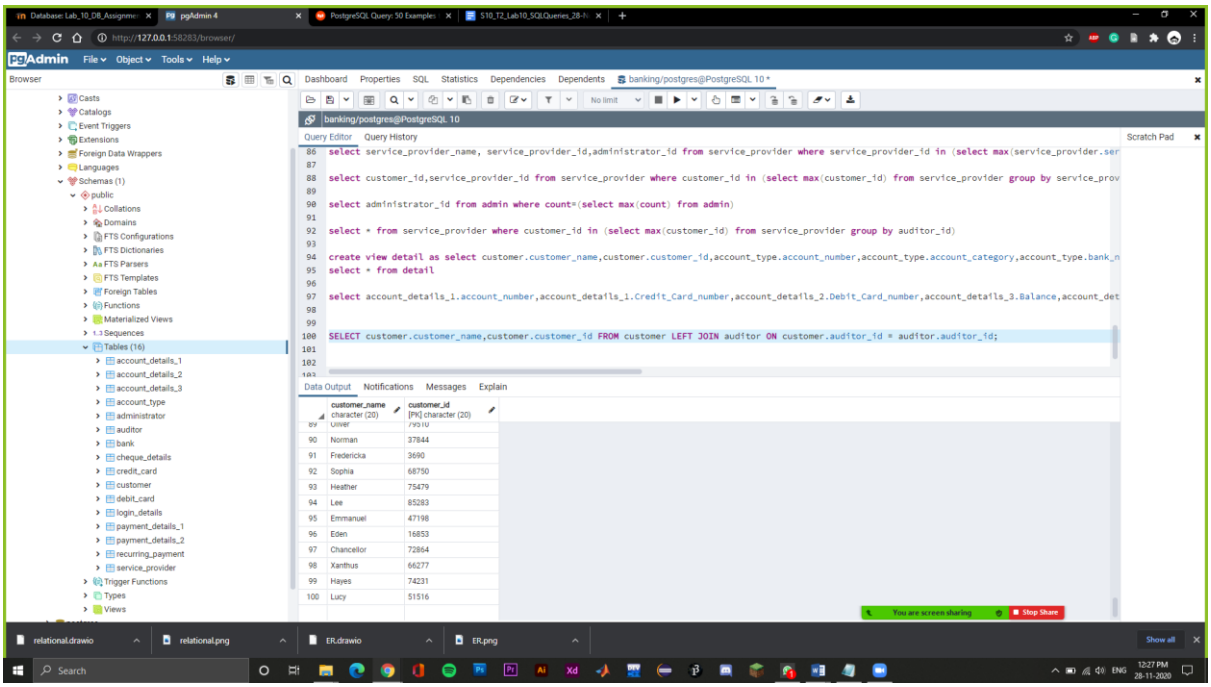
Select account_details_1.account_number, account_details_1.Credit_Card_number, account_details_2.Debit_Card_number, account_details_3.Balance, account_details_3.Recurring_Status from account_details_1 join account_details_3 on account_details_1.account_number=account_details_3.account_number join account_details_2 on account_details_1.account_number=account_details_2.account_number

->Join account_details_1,account_details_2 and account_details_3 based on their account number and display their details



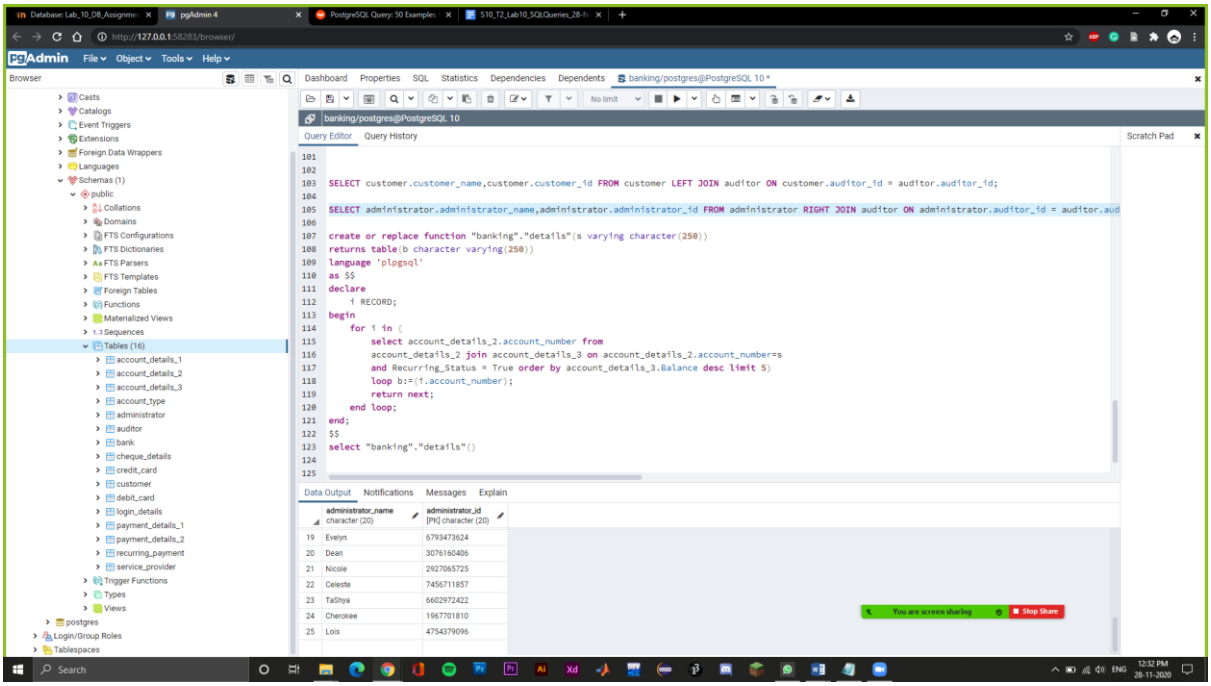
SELECT customer.customer_name,customer.customer_id FROM customer LEFT JOIN auditor ON customer.auditor_id = auditor.auditor_id;

->Shows left join of customer name on customer id.



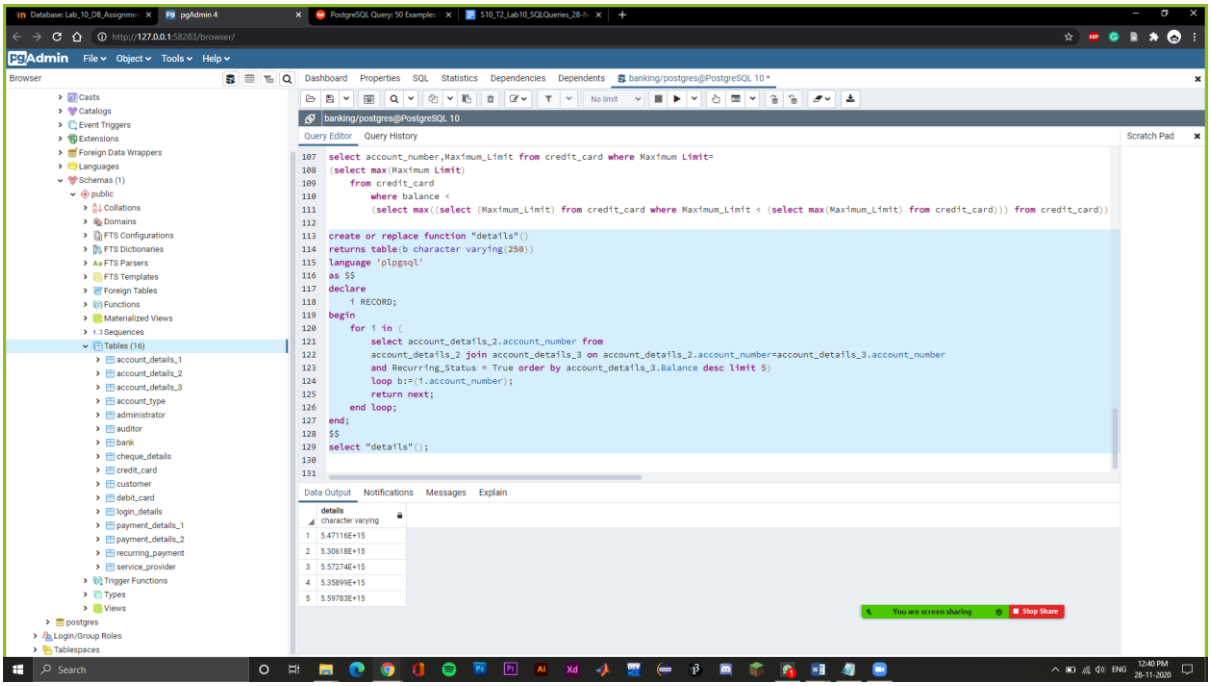
SELECT administrator.administrator_name,administrator.administrator_id FROM administrator RIGHT JOIN auditor ON administrator.auditor_id = auditor.auditor_id;

->Shows the right join of administrator name on administrator id.



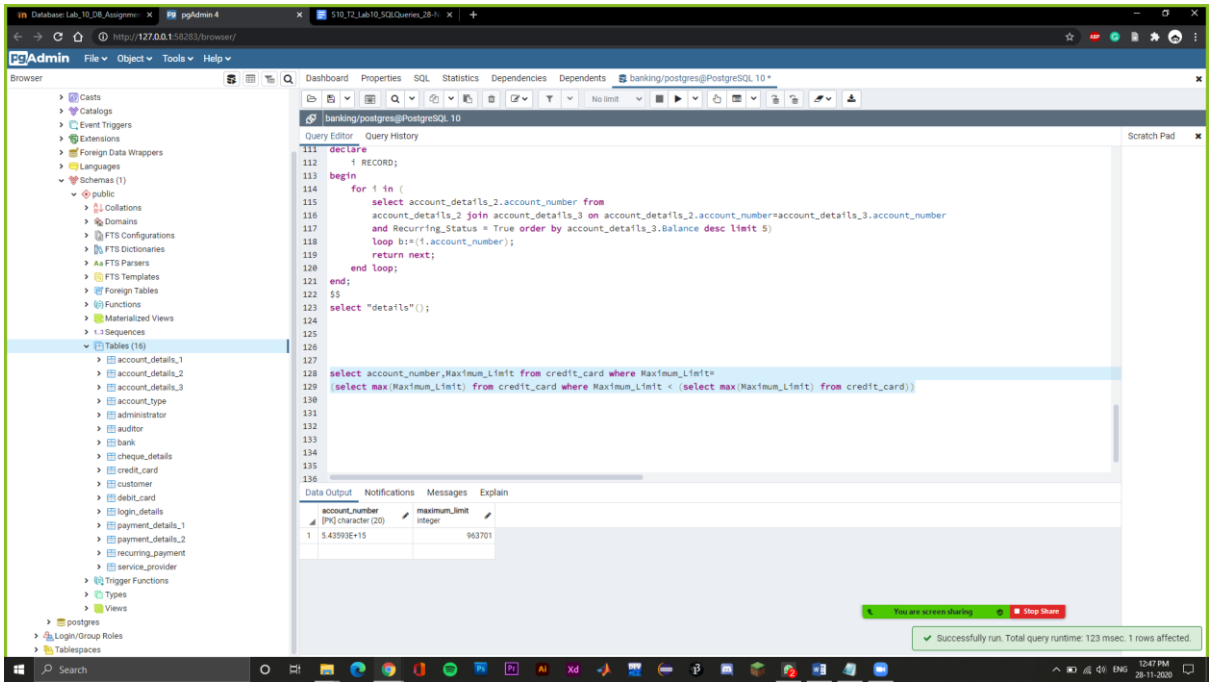
```
create or replace function "details"()
returns table(b character varying(250))
language 'plpgsql'
as $$
declare
	i RECORD;
begin
	for i in (
		select account_details_2.account_number from
		account_details_2 join account_details_3 on
account_details_2.account_number=account_details_3.account_number
		and Recurring_Status = True order by account_details_3.Balance desc limit 5)
		loop b:=(i.account_number);
		return next;
	end loop;
end;
$$
select "details"();
```

->Create a function to display the top 5 account numbers having the highest balance from the join of account_details_2 and account_details_3 where recurring status is true



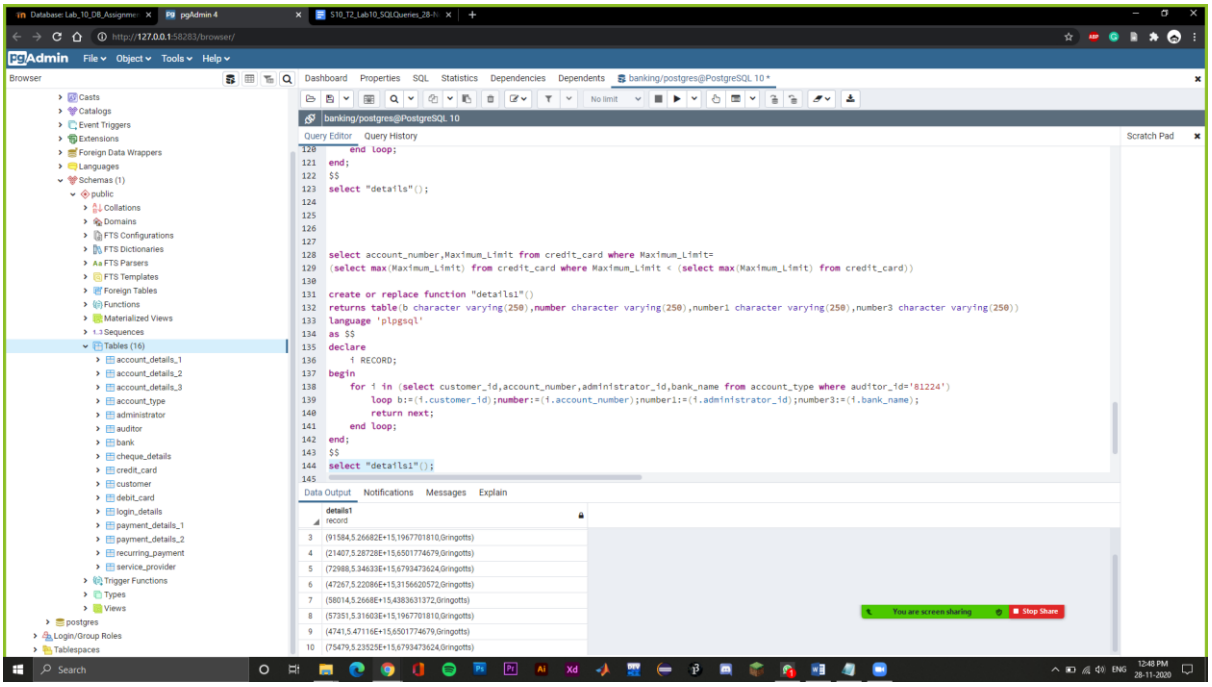
select account_number,Maximum_Limit from credit_card where Maximum_Limit=
(select max(Maximum_Limit) from credit_card where Maximum_Limit < (select max(Maximum_Limit) from credit_card))

->Select the account number with the second highest Credit Limit



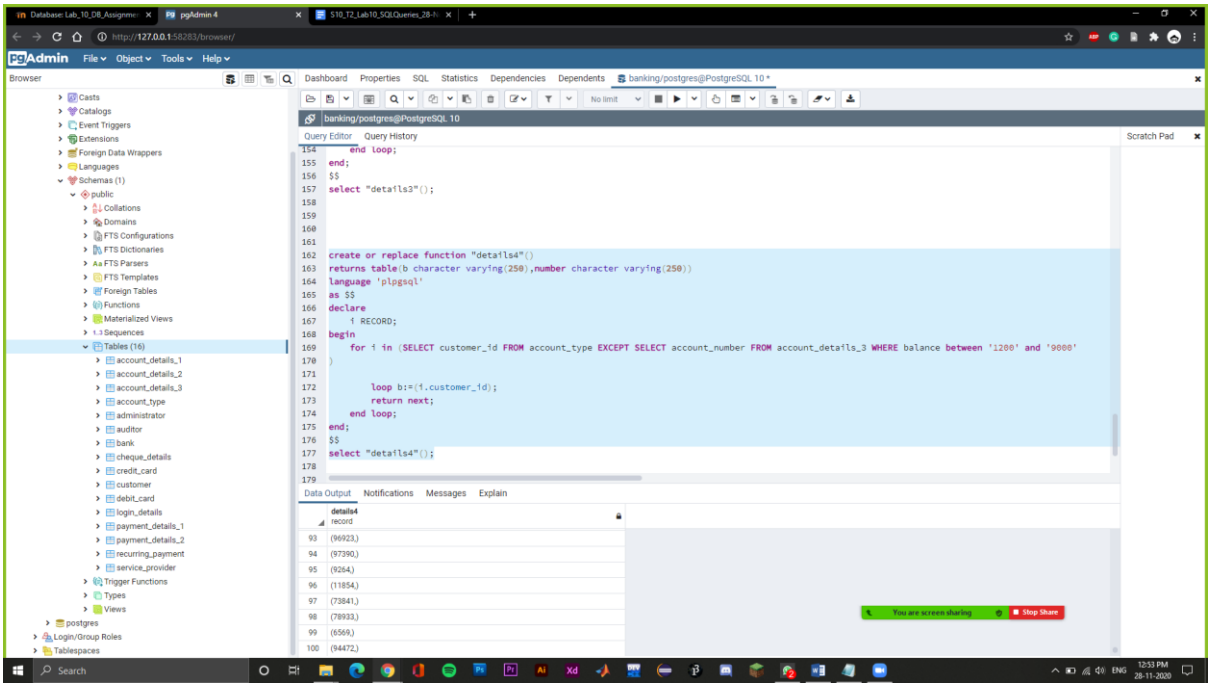
```
create or replace function "details1"()
returns table(b character varying(250),number character varying(250),number1 character varying(250),number3
character varying(250))
language 'plpgsql'
as $$
declare
        i RECORD;
begin
        for i in (select customer_id,account_number,administrator_id,bank_name from account_type where
auditor_id='81224')
                loop
b:=(i.customer_id);number:=(i.account_number);number1:=(i.administrator_id);number3:=(i.bank_name);
                return next;
        end loop;
end;
$$
select "details1"();
```

->Create a function to display customer ID, account number, administrator ID and bank Name from account_type when the auditor ID is '81224'



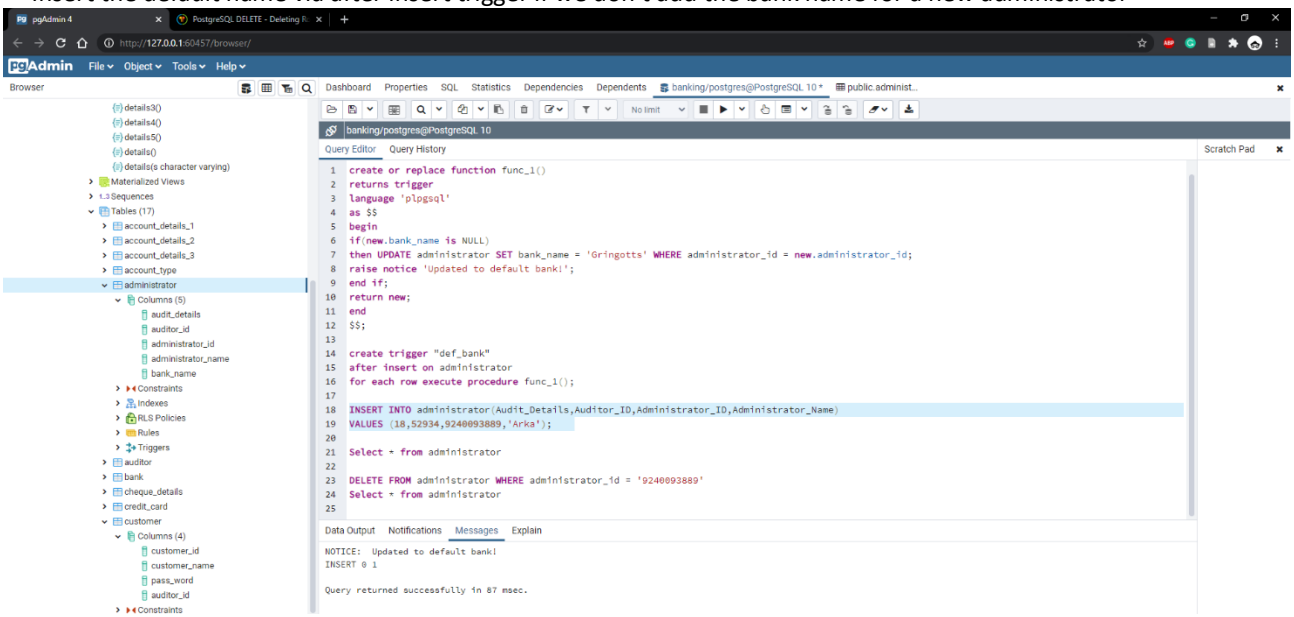
```
create or replace function "details4"()
returns table(b character varying(250),number character varying(250))
language 'plpgsql'
as $$
declare
	i RECORD;
begin
	for i in (SELECT customer_id FROM account_type EXCEPT SELECT account_number FROM account_details_3
WHERE balance between '1200' and '9000'
)

		loop b:=(i.customer_id);
		return next;
	end loop;
end;
$$
select "details4"();
```

->Create a stored procedure of customer IDs except those who have their account balance between 1200 and 9000.

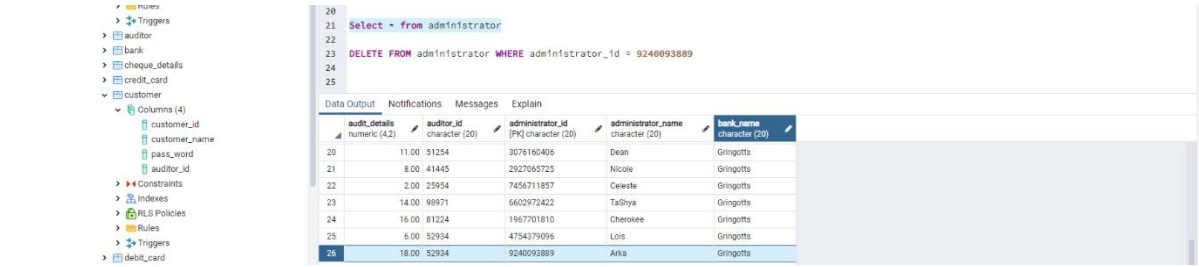


```
create or replace function func_1()
returns trigger
language 'plpgsql'
as $$
begin
if (new.bank_name is NULL)
then UPDATE administrator SET bank_name = 'Gringotts' WHERE administrator_id = new.administrator_id;
raise notice 'Updated to default bank!';
end if;
return new;
end
$$;
create trigger "def_bank"
after insert on administrator
for each row execute procedure func_1();
INSERT INTO administrator(Audit_Details,Auditor_ID,Administrator_ID,Administrator_Name)
VALUES (18,52934,9240093889,'Arka');
```

-> Insert the default name via after insert trigger if we don't add the bank name for a new administrator



Select * from administrator



```
create or replace function func_3()
returns trigger
language 'plpgsql'
as $$
begin
if(new.balance < 100)
then raise notice 'Balance must be in 3 digits';
return old;
end if;
return new;
end
$$;
create trigger "bal"
before insert on account_details_3
for each row execute procedure func_3();
INSERT INTO account_details_3(Account_Number,Balance,Recurring_Status)
VALUES ('5.43999E+15', 74, false);
```

->If balance before insert is in 2 digits then raise minimum balance notice