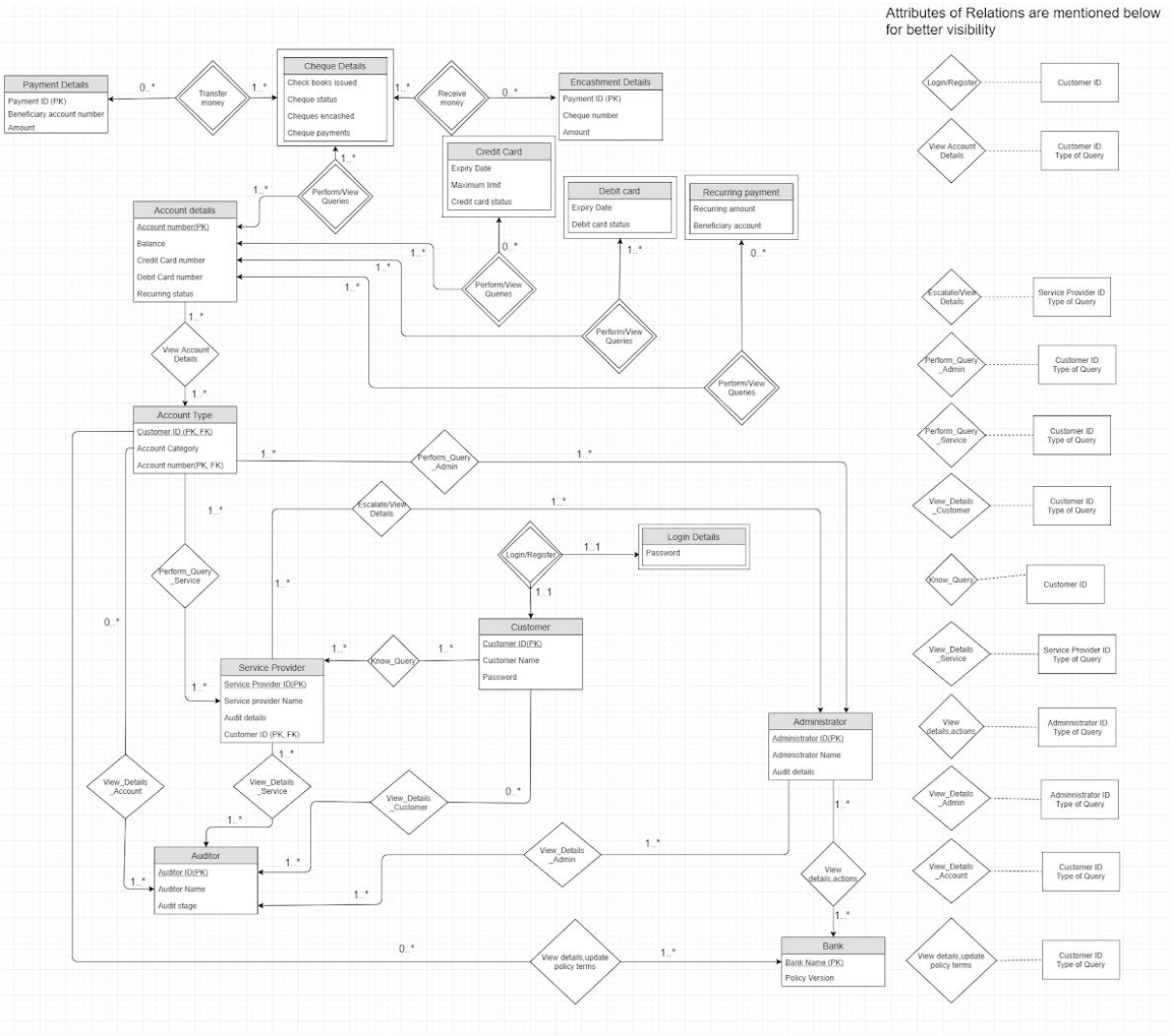
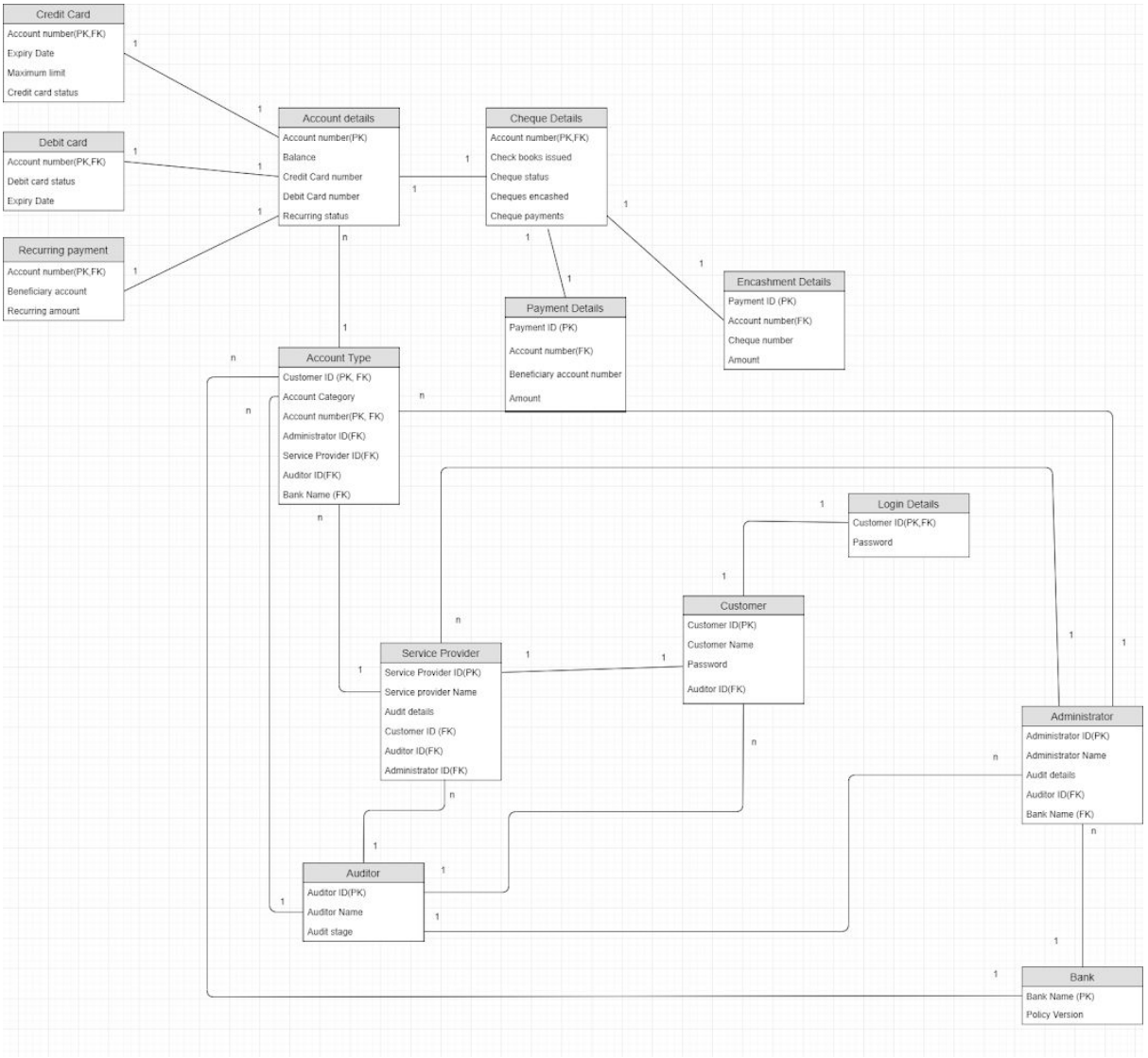


Updated ER diagram without normalization



Updated relational schema without normalization

IMPORTING CSV DATA

Auditor CSV insertion

```
COPY auditor(Auditor_ID,Auditor_Name,Audit_stage)
FROM 'C:\Users\Public\auditor.csv'
DELIMITER '|'
CSV HEADER;
```

Customer CSV insertion

```
COPY customer(Customer_ID,Customer_Name,Account_number,Pass_word,Auditor_ID)
FROM 'C:\Users\Public\Customer.csv'
DELIMITER ','
CSV HEADER;
```

Login details CSV insertion

```
COPY login_details(Customer_ID,Pass_word)
FROM 'C:\Users\Public\login.csv'
DELIMITER ','
CSV HEADER;
```

Bank

```
COPY Bank(Bank_name,Policy_version)
FROM 'C:\Users\Public\bank.csv'
DELIMITER ','
CSV HEADER;
```

Administrator

```
COPY
Administrator(Audit_Details,Auditor_ID,Administrator_ID,Administrator_Name,BANK_NAME)
FROM 'C:\Users\Public\administrator.csv'
DELIMITER ','
CSV HEADER;
```

Service_Provider

```
COPY Service_Provider(
Service_Provider_ID,Service_Provider_Name,Audit_Details,Auditor_ID,Administrator_ID,Cu
stomer_ID)
FROM 'C:\Users\Public\service_provider.csv'
DELIMITER ','
CSV HEADER;
```

Account_Details_1

```
COPY Account_Details_1(Account_Number,Credit_Card_number)
FROM 'C:\Users\Public\account_details_1.csv'
DELIMITER ','
CSV HEADER;
```

Account_Details_2

```
COPY Account_Details_2(Account_Number,Debit_Card_number)
FROM 'C:\Users\Public\account_details_2.csv'
DELIMITER ','
CSV HEADER;
```

Account_Details_3

```
COPY Account_Details_3(Account_Number,Balance,Recurring_Status)
FROM 'C:\Users\Public\account_details_3.csv'
DELIMITER ','
```

```
CSV HEADER;
```

Account_type

```
COPY Account_type(
Customer_ID,Account_Category,Account_number,Administrator_ID,Service_Provider_ID,Au
ditor_ID,Bank_Name)
FROM 'C:\Users\Public\Account_type.csv'
DELIMITER ','
CSV HEADER;
```

Credit_Card

```
COPY Credit_card(Account_Number,Expiry_Date,Maximum_Limit,Credit_Card_Status)
FROM 'C:\Users\Public\credit_card.csv'
DELIMITER ','
CSV HEADER;
```

Debit_card

```
COPY Debit_card(Account_Number,Expiry_Date,Debit_Card_Status)
FROM 'C:\Users\Public\debit_card.csv'
DELIMITER ','
CSV HEADER;
```

Recurring_payment

```
COPY Recurring_payment(Account_Number,Beneficiary_account,Recurring_amount)
FROM 'C:\Users\Public\recurring_payment.csv'
DELIMITER ','
CSV HEADER;
```

Cheque_details

```
COPY
Cheque_details(Account_Number,Cheques_encashed,Cheque_payments,Check_books_iss
ues,Cheque_Status)
FROM 'C:\Users\Public\cheque_details.csv'
DELIMITER ','
CSV HEADER;
```

Payment_Details_1

```
COPY Payment_Details_1(Payment_ID,Account_Number, Cheque_Number)
FROM 'C:\Users\Public\payment_details_1.csv'
DELIMITER ','
CSV HEADER;
```

Payment_Details_2

```
Payment_Details_2(Payment_ID,Account_Number, Amount)
FROM 'C:\Users\Public\payment_details_2.csv'
DELIMITER ','
CSV HEADER;
```

PostgreSQL CODE to create tables: (UPDATED)

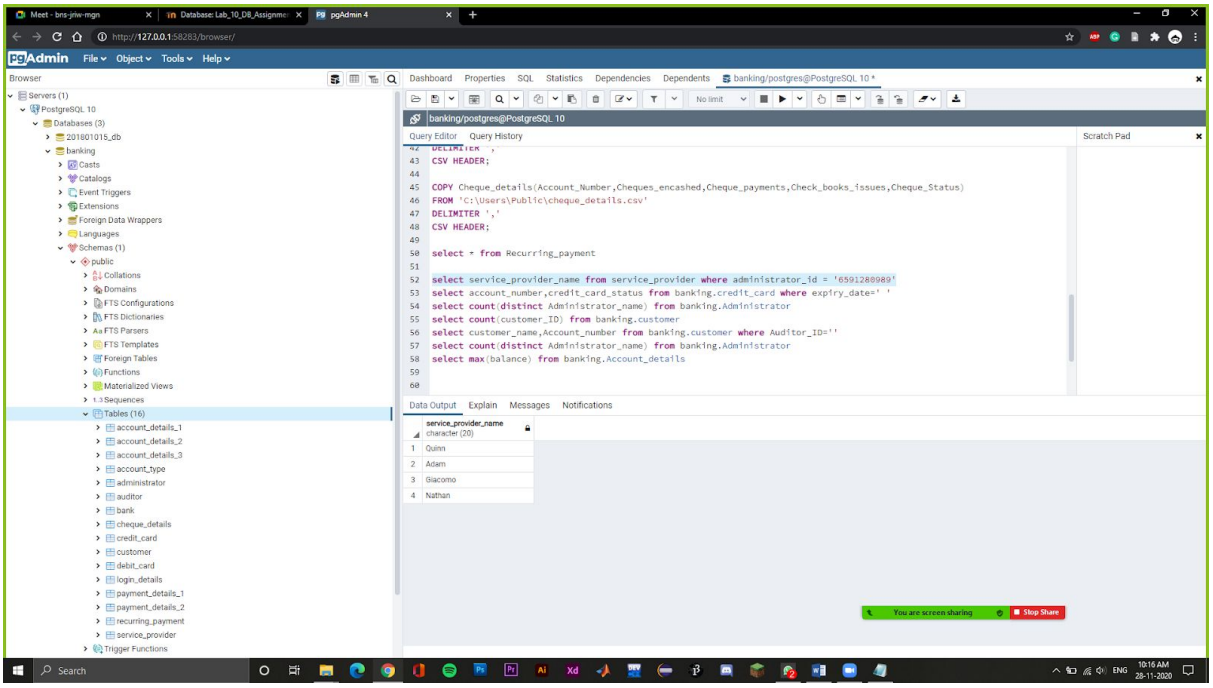
```
CREATE TABLE Auditor(
Auditor_ID CHAR(20) NOT NULL,
Auditor_Name CHAR(20),
Audit_stage INT,
PRIMARY KEY (Auditor_ID)
)

CREATE TABLE Customer(
Customer_ID CHAR(20) NOT NULL,
Customer_Name CHAR(20),
Account_number CHAR(20),
Pass_word CHAR(20),
Auditor_ID CHAR(20),
PRIMARY KEY (Customer_ID),
FOREIGN KEY (Auditor_ID) REFERENCES Auditor)

CREATE TABLE Login_Details(
Customer_ID CHAR(20) NOT NULL,
Pass_word CHAR(20),
PRIMARY KEY(Customer_ID),
FOREIGN KEY (Customer_ID) REFERENCES Customer
ON DELETE CASCADE)

CREATE TABLE Bank(
Bank_Name CHAR(20) NOT NULL,
Policy_Version numeric(4,2),
PRIMARY KEY(Bank_Name)
)

CREATE TABLE Administrator(
Audit_Details NUMERIC(4,2),
Auditor_ID CHAR(20),
Administrator_ID CHAR(20) NOT NULL,
Administrator_Name CHAR(20),
BANK_NAME CHAR (20),
PRIMARY KEY(Administrator_ID),
FOREIGN KEY (Bank_Name) REFERENCES Bank,
FOREIGN KEY (Auditor_ID) REFERENCES Auditor
)

CREATE TABLE  Service_Provider(
Service_Provider_ID CHAR(20) NOT NULL,
Service_Provider_Name CHAR(20),
Audit_Details NUMERIC(4,2),
Auditor_ID CHAR(20),
Administrator_ID CHAR(20),
Customer_ID CHAR(20),
PRIMARY KEY(Service_Provider_ID, Customer_ID),
FOREIGN KEY (Customer_ID ) REFERENCES customer,
FOREIGN KEY (Administrator_ID) REFERENCES Administrator,
FOREIGN KEY (Auditor_ID) REFERENCES Auditor
)

CREATE TABLE Account_Details_1(
Account_Number CHAR(20) NOT NULL,
Credit_Card_number INTEGER,
PRIMARY KEY (Account_Number)
)
CREATE TABLE Account_Details_2(
Account_Number CHAR(20) NOT NULL,
Debit_Card_number INTEGER,
PRIMARY KEY (Account_Number)
```

```
)
CREATE TABLE Account_Details_3(
Account_Number CHAR(20) NOT NULL,
Balance NUMERIC(8,2),
Recurring_Status BOOLEAN,
PRIMARY KEY (Account_Number)
)

CREATE TABLE Account_type(
Customer_ID CHAR(20) NOT NULL,
Account_Category CHAR(20),
Account_number CHAR(20),
Administrator_ID CHAR(20),
Service_Provider_ID CHAR(20),
Auditor_ID CHAR(20),
Bank_Name CHAR(20),
PRIMARY KEY (Customer_ID, Account_number),
FOREIGN KEY (Account_number) REFERENCES Account_Details_1 ON DELETE CASCADE,
FOREIGN KEY (Administrator_ID) REFERENCES Administrator ON DELETE CASCADE,
FOREIGN KEY (Service_Provider_ID, Customer_ID) REFERENCES Service_Provider ON DELETE CASCADE,
FOREIGN KEY (Auditor_ID) REFERENCES Auditor ON DELETE CASCADE,
FOREIGN KEY (Bank_Name) REFERENCES Bank ON DELETE CASCADE)

CREATE TABLE Credit_card(
Account_Number CHAR(20) NOT NULL,
Expiry_Date DATE,
Maximum_Limit INTEGER,
Credit_Card_Status BOOLEAN,
PRIMARY KEY (Account_Number),
FOREIGN KEY (Account_Number) REFERENCES Account_Details_1
ON DELETE CASCADE)

CREATE TABLE Debit_card(
Account_Number CHAR(20) NOT NULL,
Expiry_Date DATE,
Debit_Card_Status BOOLEAN,
PRIMARY KEY (Account_Number),
FOREIGN KEY (Account_Number) REFERENCES Account_Details_2
ON DELETE CASCADE)

CREATE TABLE Recurring_payment(
Account_Number CHAR(20) NOT NULL,
Beneficiary_account CHAR(20),
Recurring_amount INTEGER,
PRIMARY KEY (Account_Number),
FOREIGN KEY (Account_Number) REFERENCES Account_Details_3
ON DELETE CASCADE)

CREATE TABLE Cheque_details(
Account_Number CHAR(20) NOT NULL,
Cheques_encashed INTEGER,
Cheque_payments NUMERIC(8,2),
Check_books_issues INTEGER,
Cheque_Status BOOLEAN,
PRIMARY KEY (Account_Number),
FOREIGN KEY (Account_Number) REFERENCES Account_Details_1
ON DELETE CASCADE)

CREATE TABLE Payment_Details_1(
        Payment_ID CHAR(20) NOT NULL,
        Account_Number CHAR(20) NOT NULL,
        Cheque_Number CHAR(20),
        PRIMARY KEY (Payment_ID),
        FOREIGN KEY (Account_Number) REFERENCES Cheque_Details
        ON DELETE CASCADE
)
```

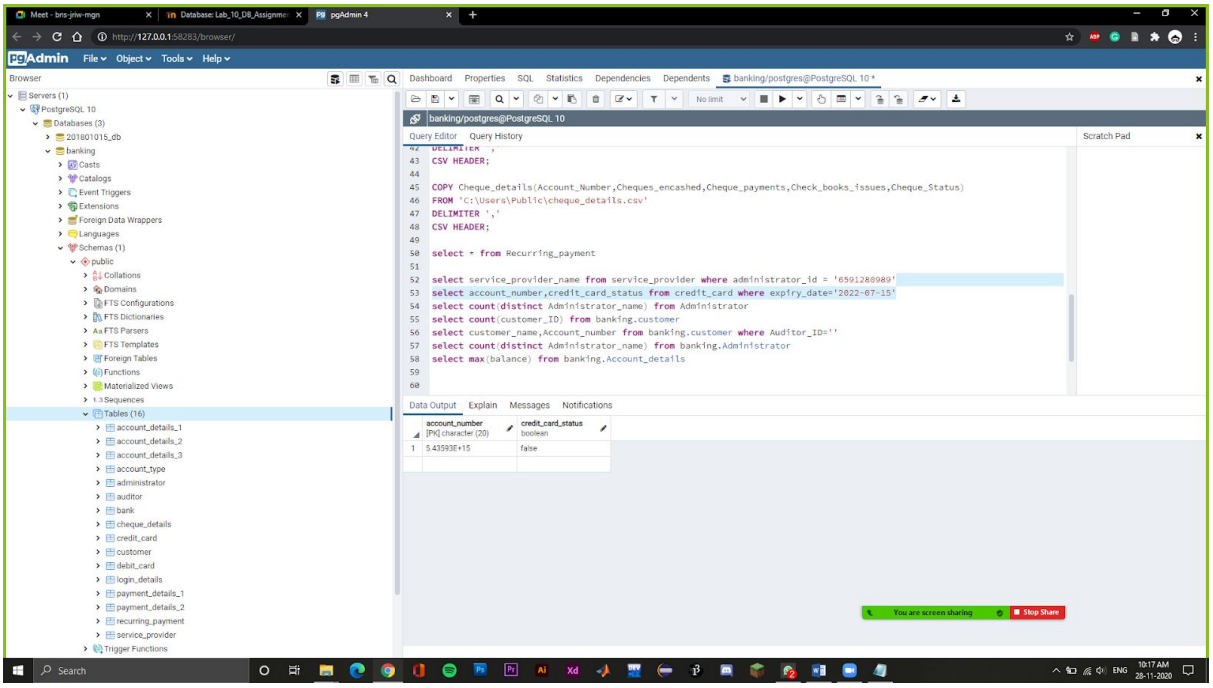
```
CREATE TABLE Payment_Details_2(
        Payment_ID CHAR(20) NOT NULL,
        Account_Number CHAR(20) NOT NULL,
        Amount INTEGER,
        PRIMARY KEY (Payment_ID),
        FOREIGN KEY (Account_Number) REFERENCES Cheque_Details
        ON DELETE CASCADE
)
```

40 different types of queries on our database are:

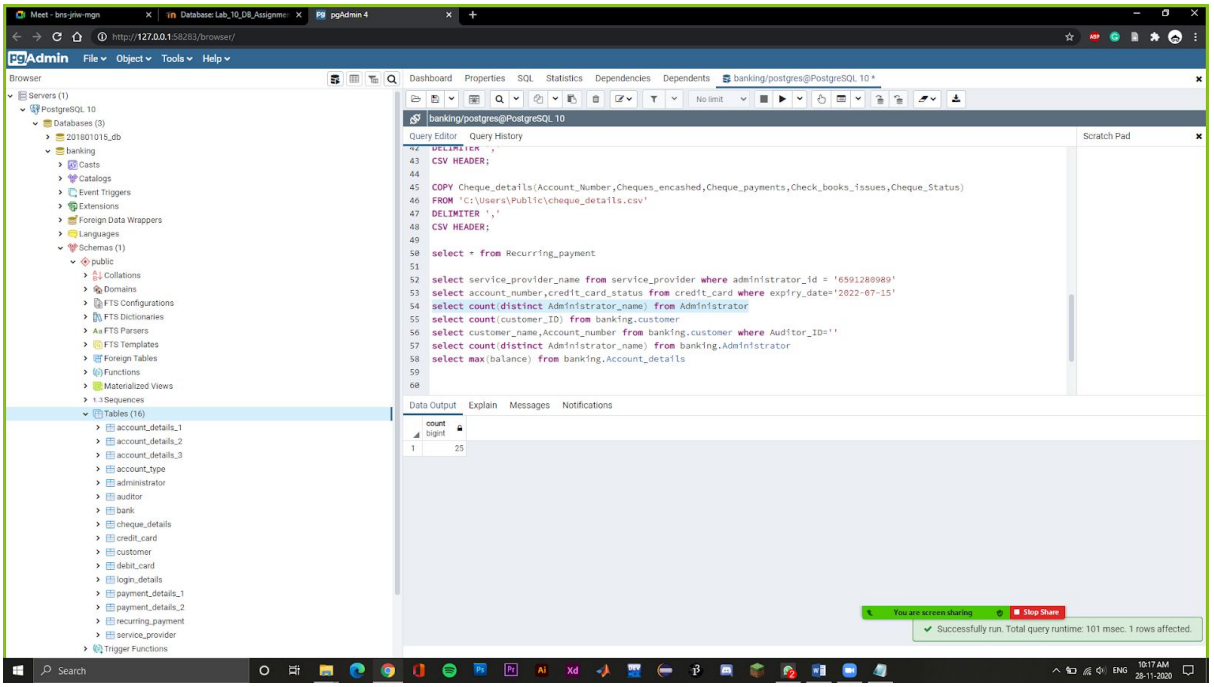
select service_provider_name from service_provider where administrator_id = '6591280989'
-> Here, we are selecting the name of the service provider for the administrator id 6591280989.



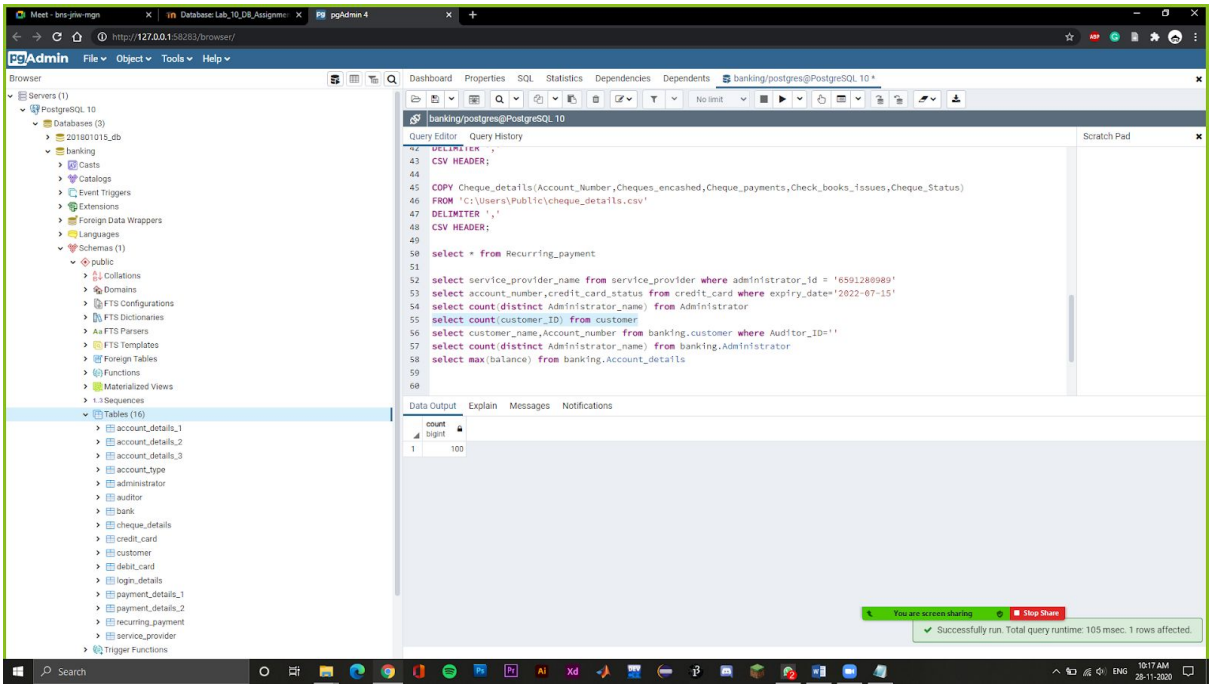
select account_number,credit_card_status from credit_card where expiry_date='2022-07-15'
-> Here, we are giving the account number and the credit card status for the credit card whose expiry date is 15-07-2022.



select count(distinct Administrator_name) from Administrator
-> Here, we are giving the total number of distinct administrators.

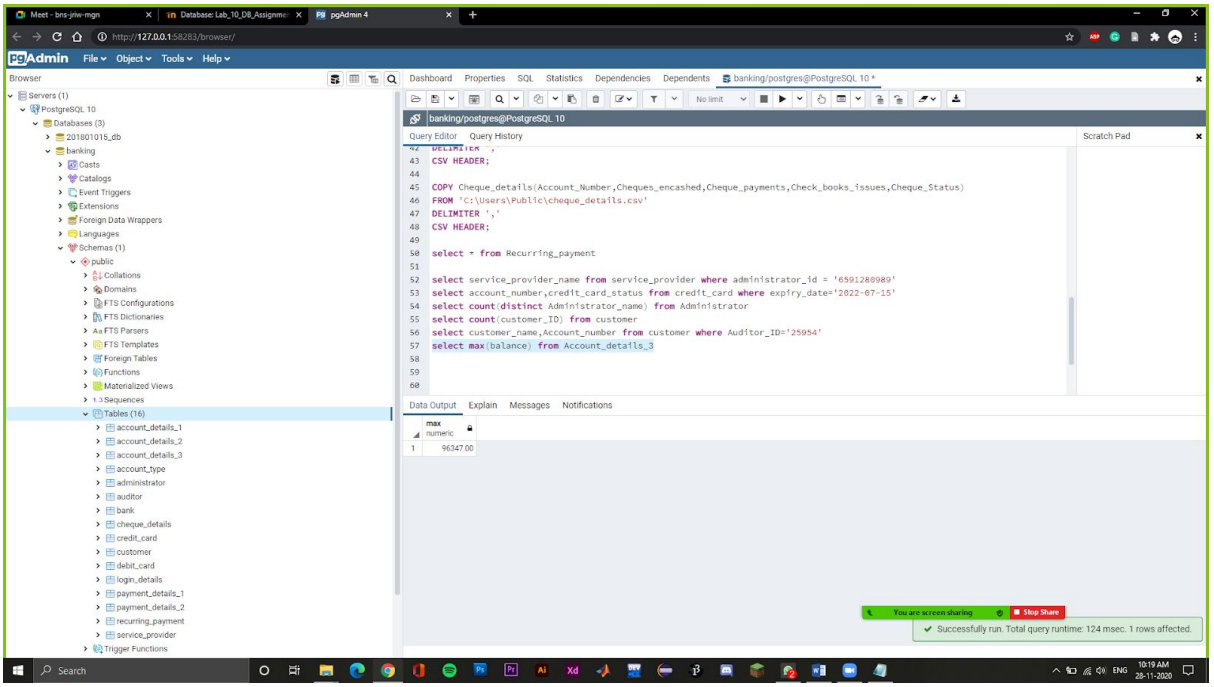


select count(customer_ID) from customer
-> Here, we are calculating the total number of customers.



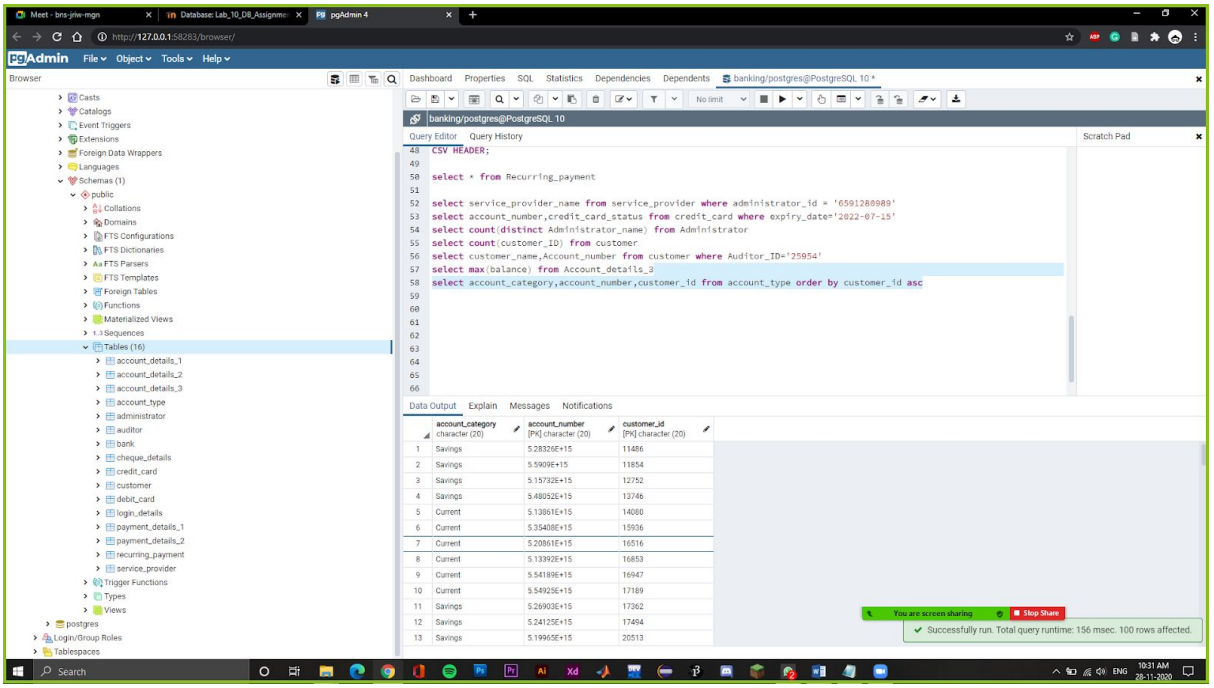
select max(balance) from Account_details_3

-> Here, we are printing the maximum balance of all the bank accounts in the bank.



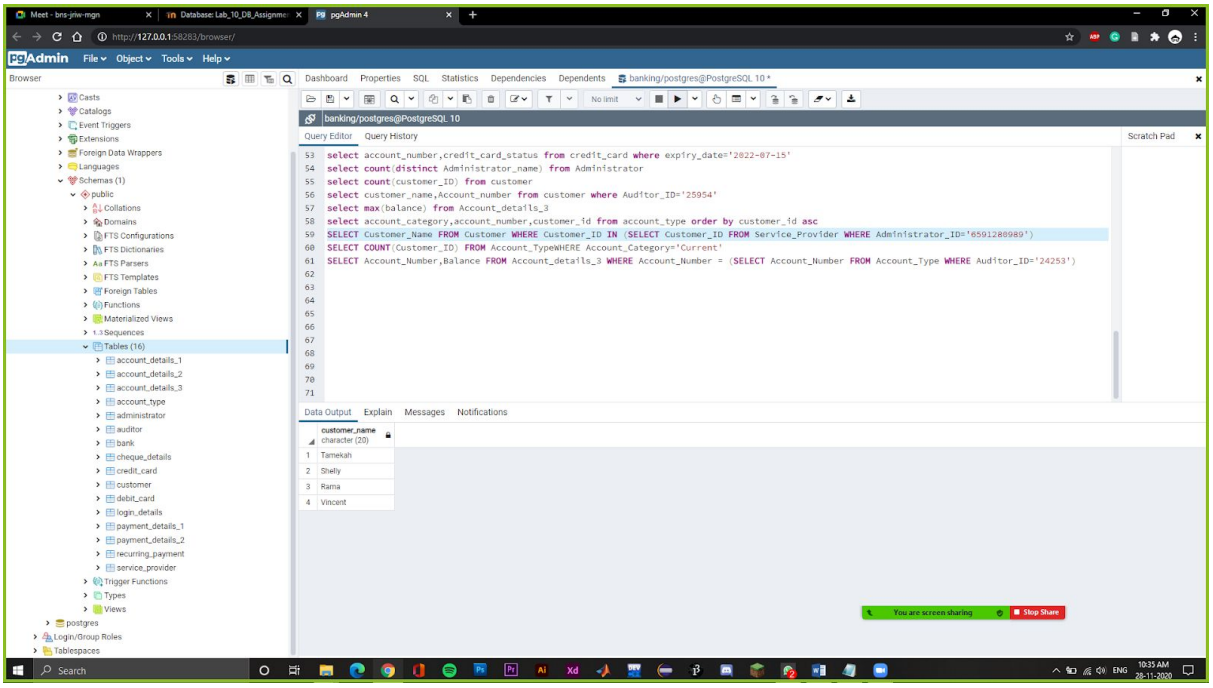
select account_category,account_number,customer_id from account_type order by customer_id asc

-> Here, we are printing the account category, account number,customer id and arranging them on the basics of customer id from smallest to largest order.



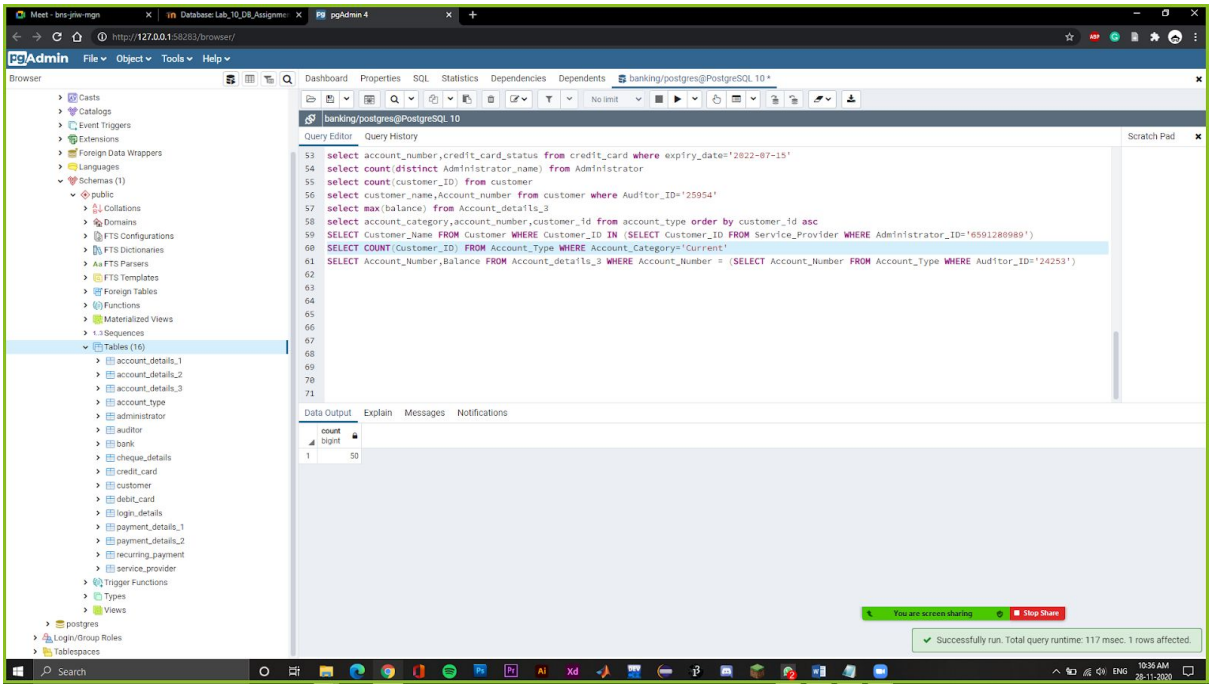
SELECT Customer_Name FROM Customer WHERE Customer_ID IN (SELECT Customer_ID FROM Service_Provider WHERE Administrator_ID='6591280989')

-> Show customer name who have administrator id as 6591280989



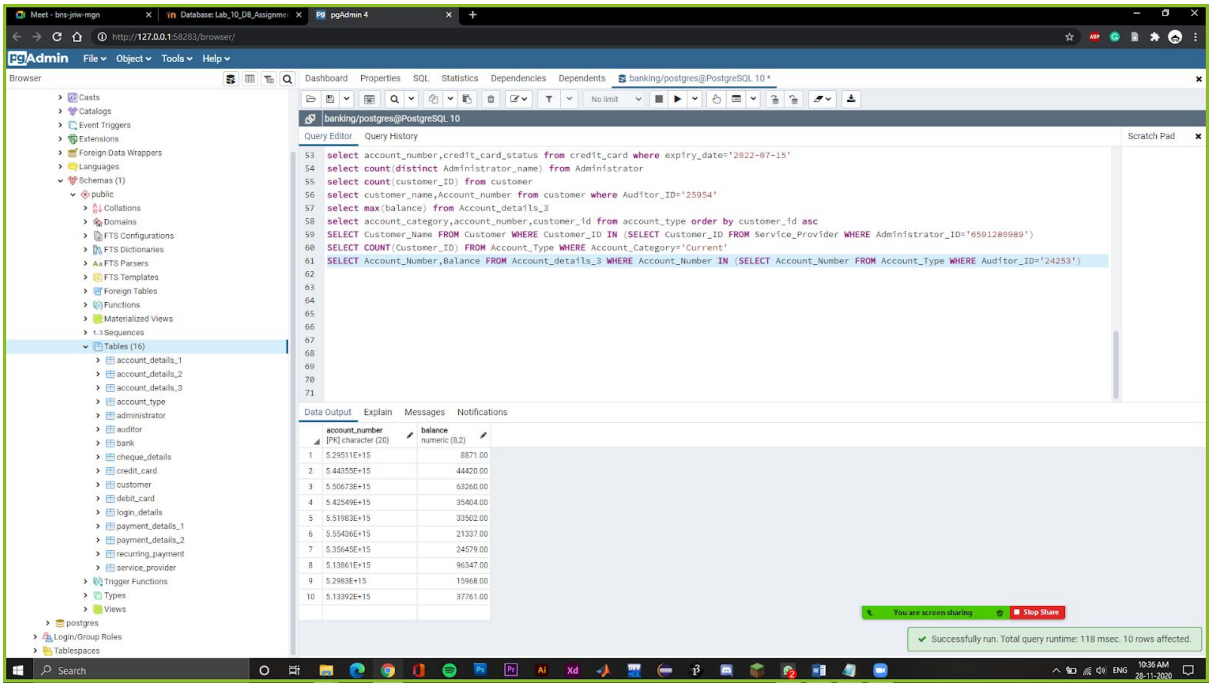
SELECT COUNT(Customer_ID) FROM Account_Type WHERE Account_Category='Current'

Show number of customers who have account category as current.



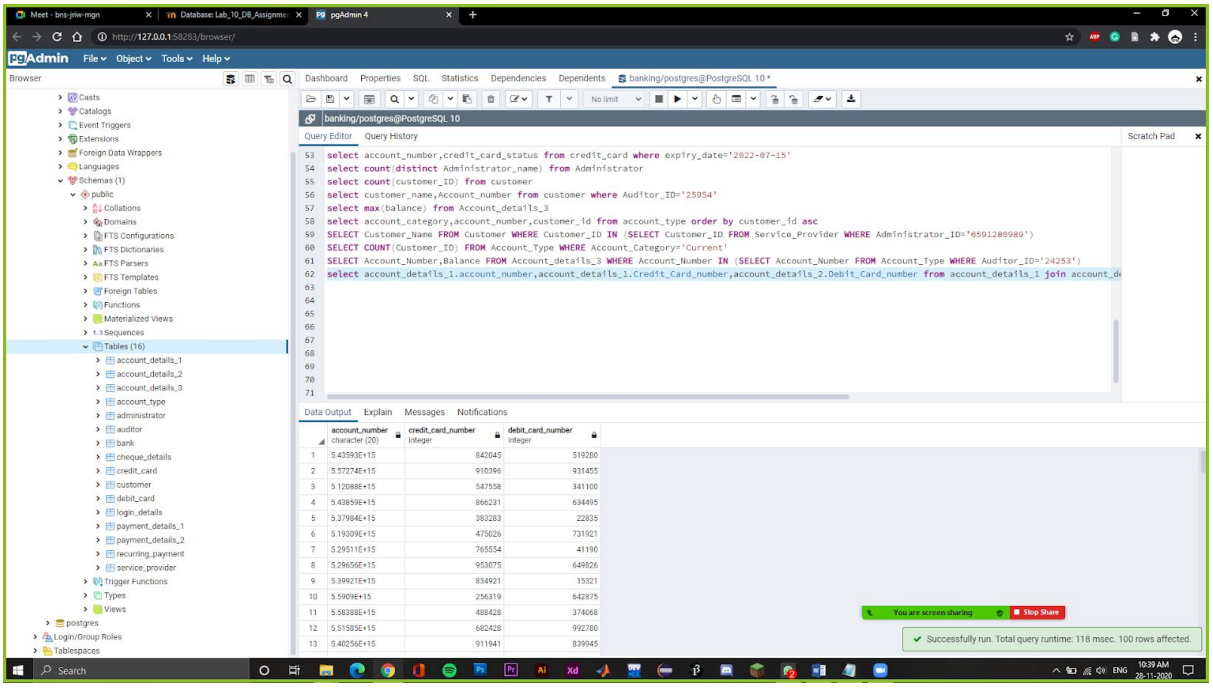
SELECT Account_Number,Balance FROM Account_details_3 WHERE Account_Number IN (SELECT Account_Number FROM Account_Type WHERE Auditor_ID='24253')

-> Show account number and balance of customers who have auditor id as 24253

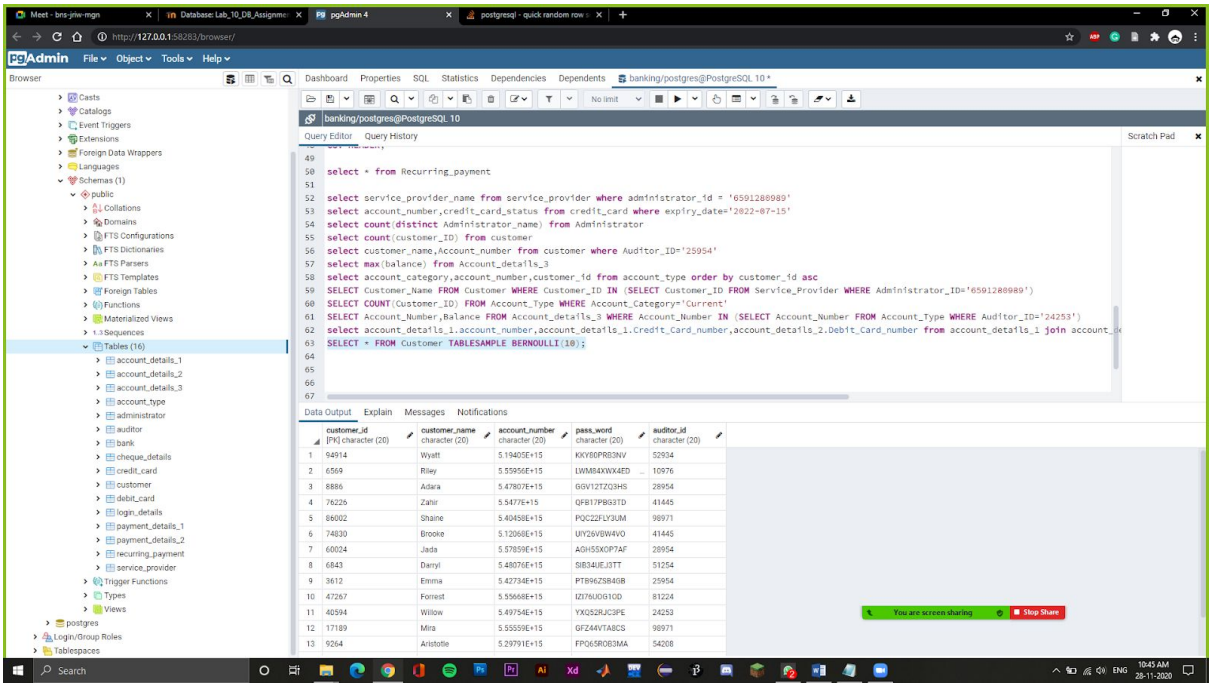


Select account_details_1.account_number,account_details_1.Credit_Card_number,account_details_2.Debit_Card_number from account_details_1 join account_details_2 on account_details_1.account_number=account_details_2.account_number

Show account number, credit card number and debit card number of all the customers

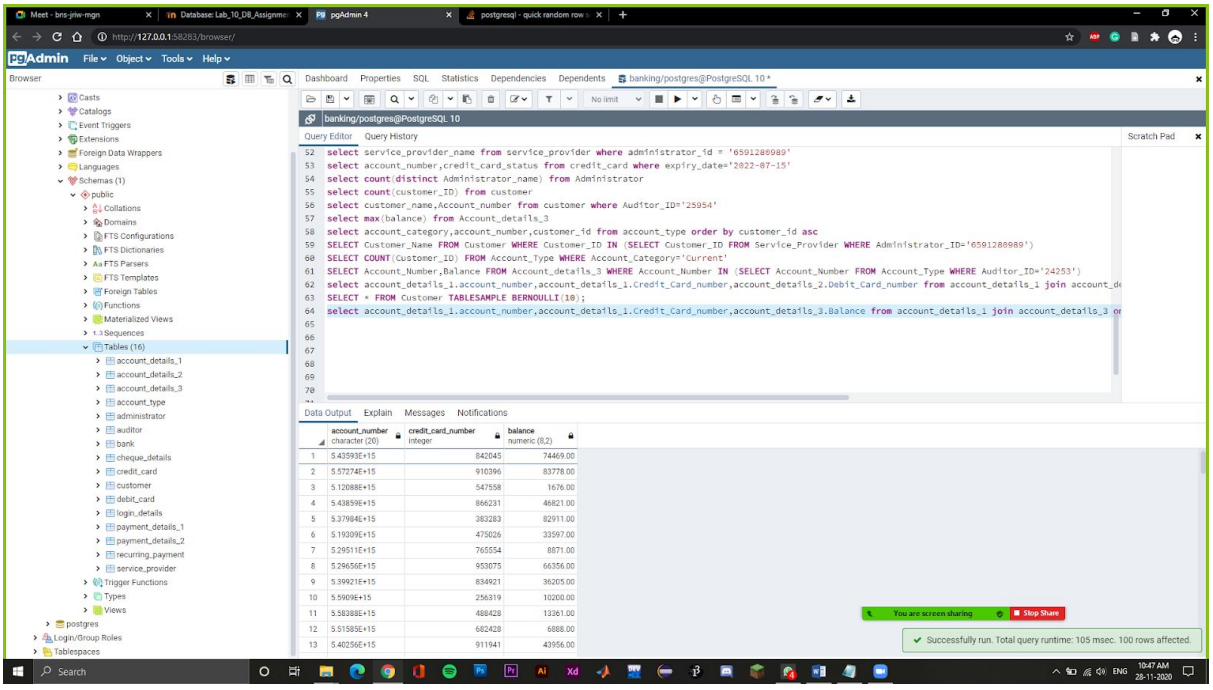


SELECT * FROM Customer TABLESAMPLE BERNOULLI(10);



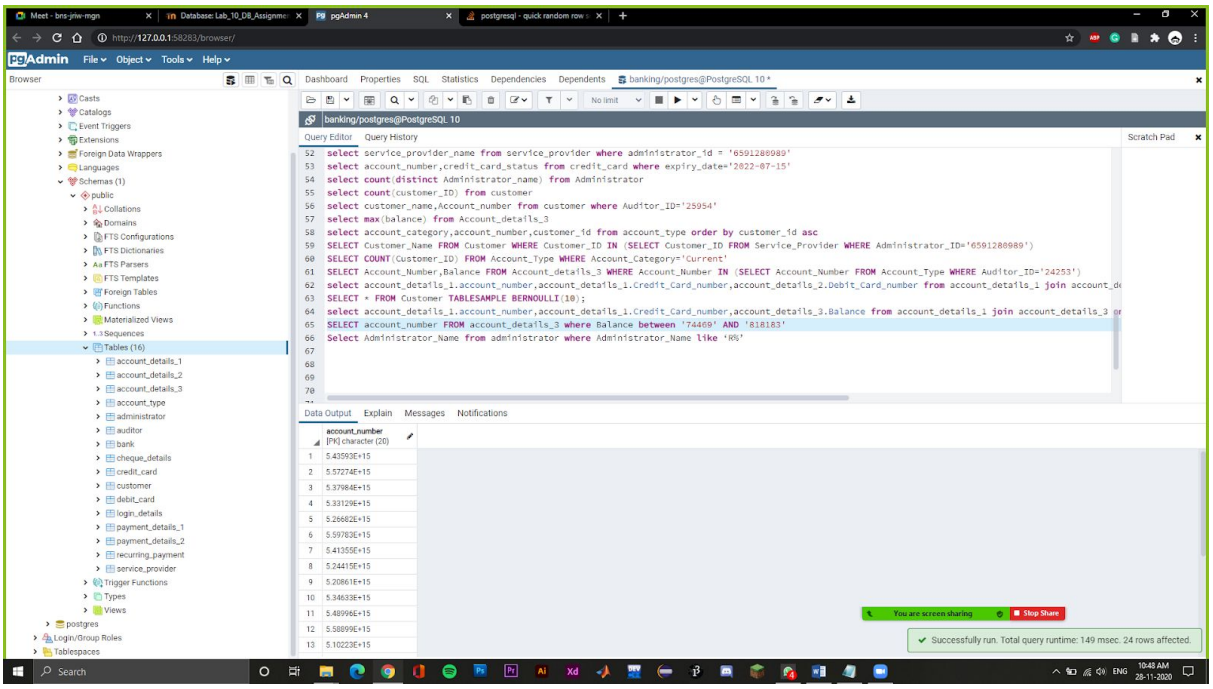
select account_details_1.account_number,account_details_1.Credit_Card_number,account_details_3.Balance from account_details_1 join account_details_3 on account_details_1.account_number=account_details_3.account_number

Show account number, credit card number and balance of all the customers



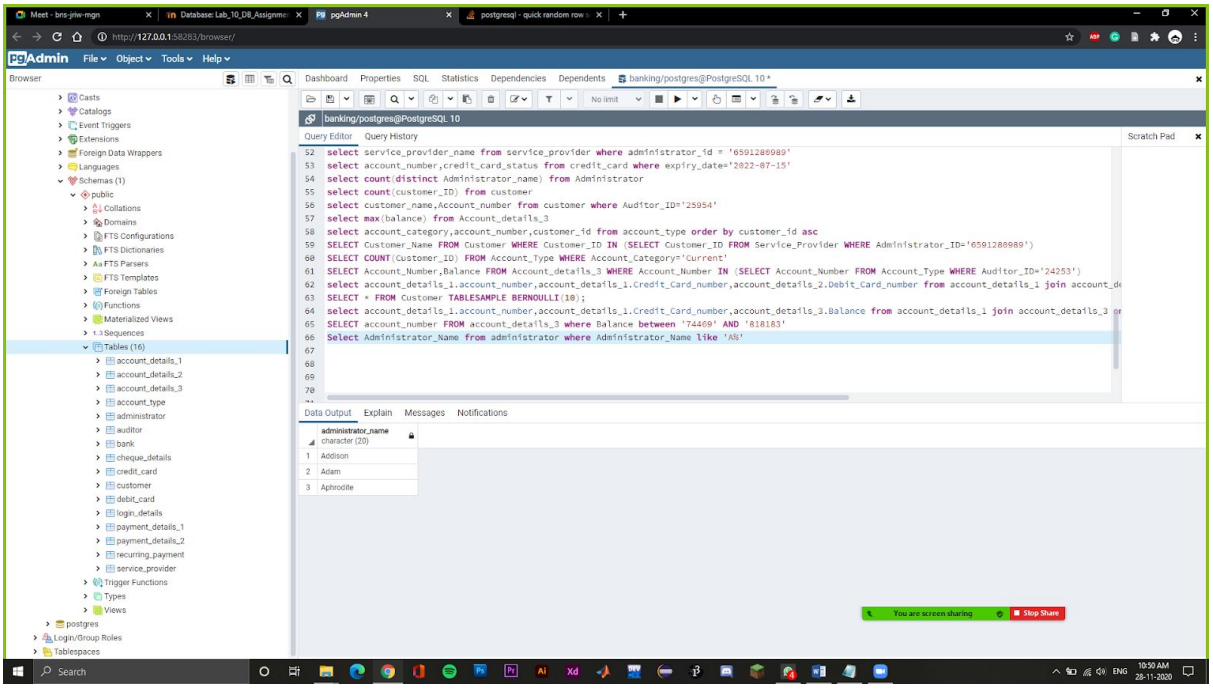
SELECT account_number FROM account_details_3 where Balance between '74469' AND '818183'

-> Showing the account number which has a balance between 74469 and 818183.



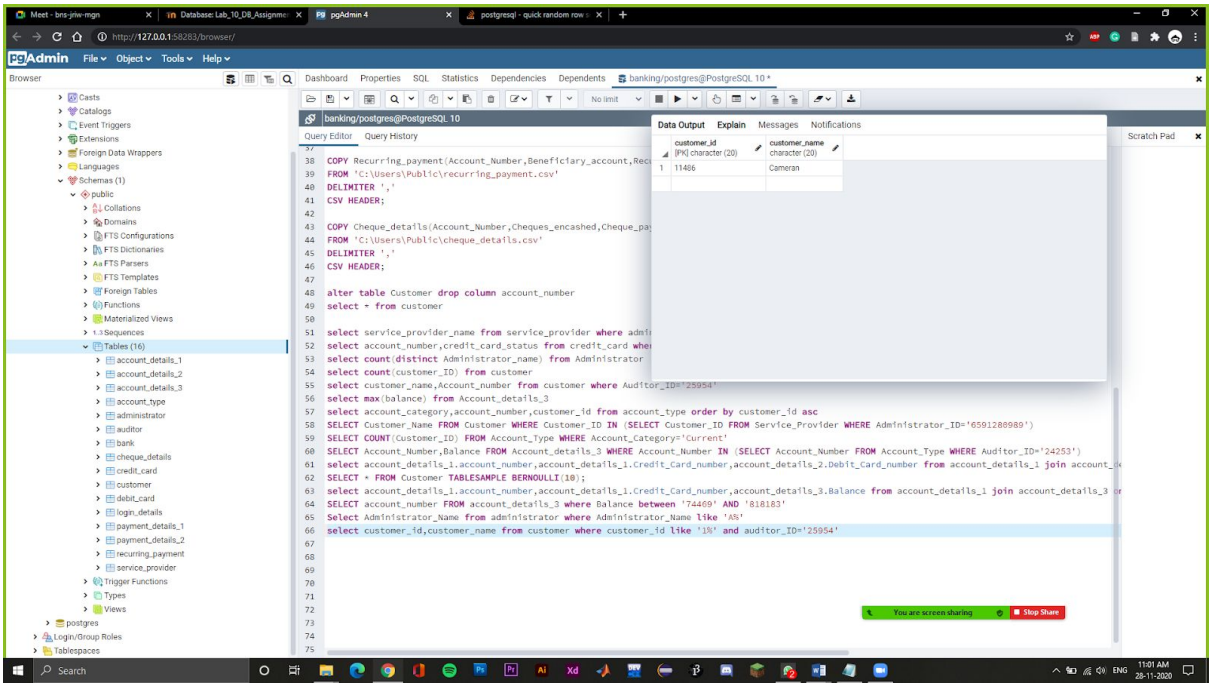
Select Administrator_Name from administrator where Administrator_Name like 'A%'

-> Showing the administrator name starting from A.



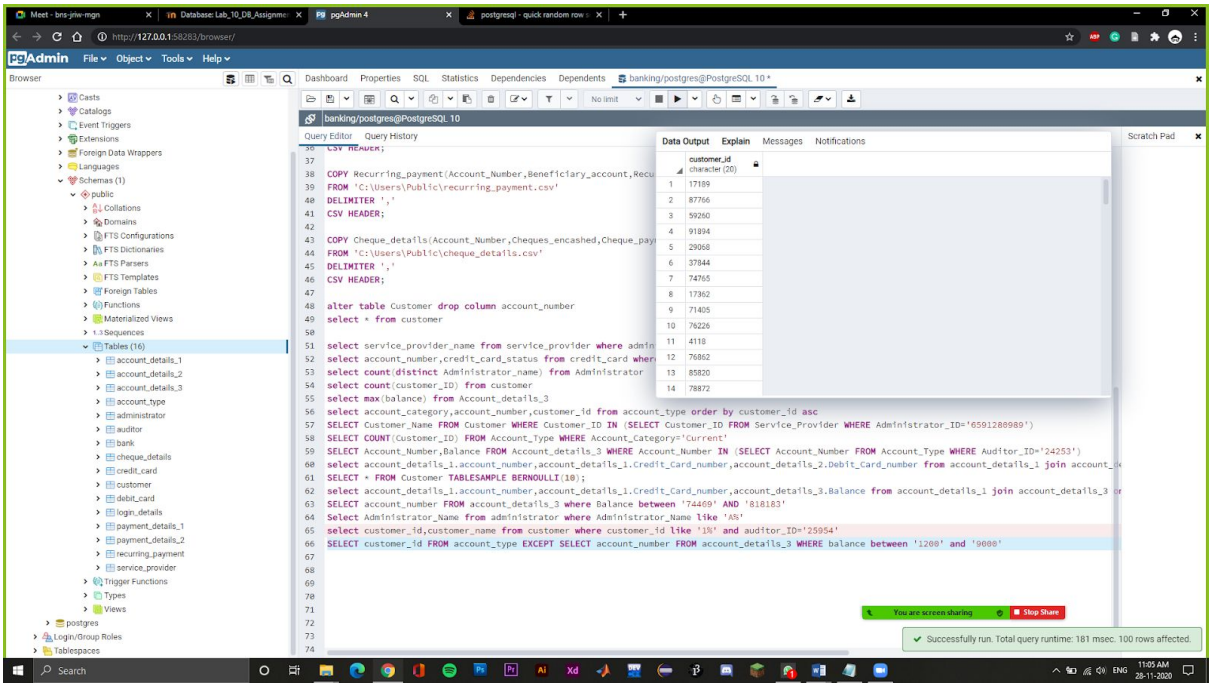
select customer_id,customer_name from customer where customer_id like '1%' and auditor_ID='25954'

-> Showing customer id and customer name which has auditor id 25954 and customer id starts from 1.



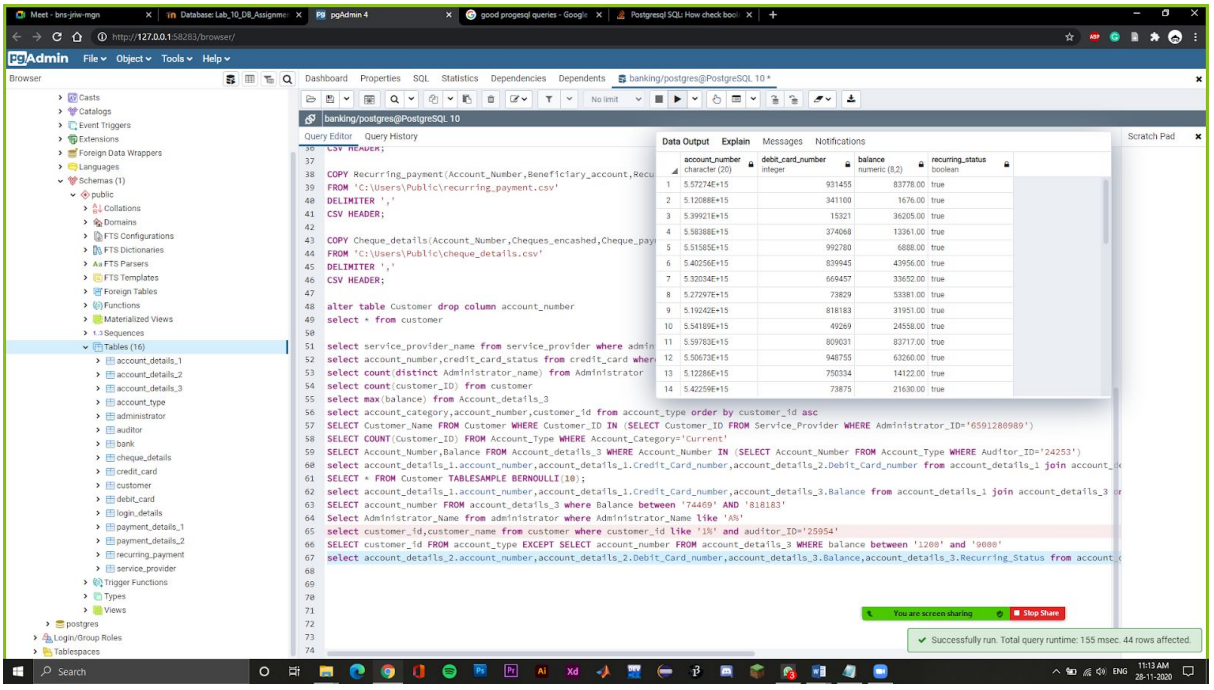
SELECT customer_id FROM account_type EXCEPT SELECT account_number FROM account_details_3 WHERE balance between '1200' and '9000'

-> Show customer id except for customer having balance between 1200 and 9000

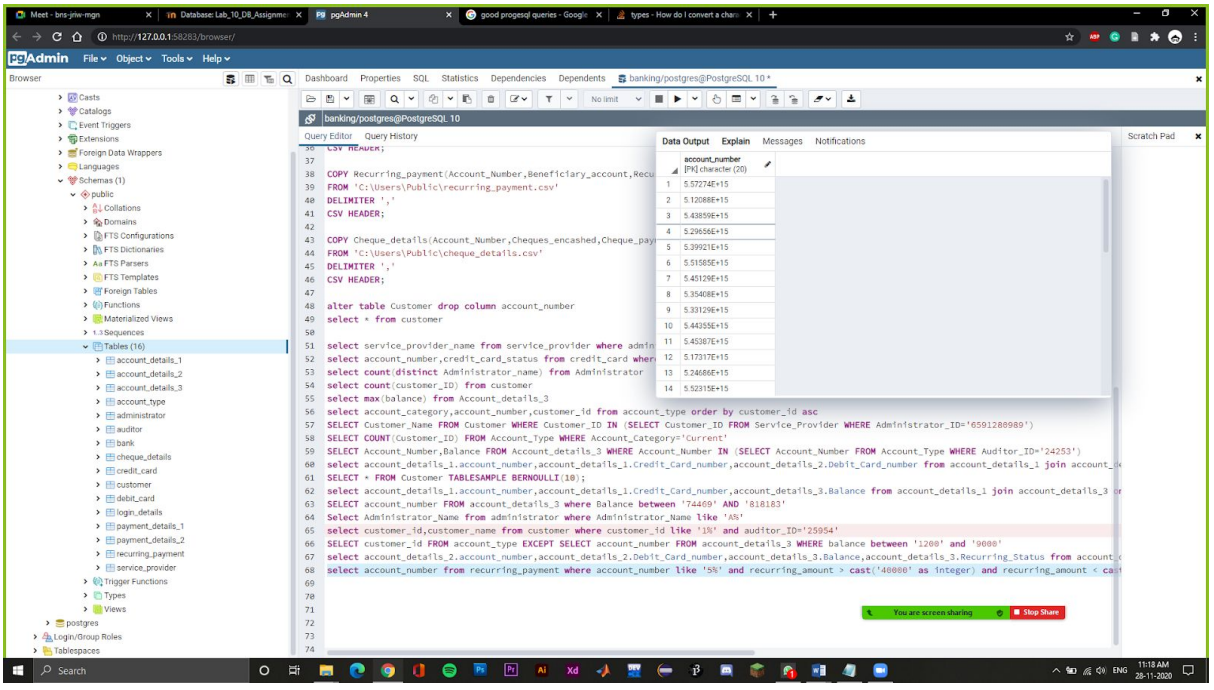


select account_details_2.account_number,account_details_2.Debit_Card_number,account_details_3.Balance,account_details_3.Recurring_Status from account_details_2 join account_details_3 on account_details_2.account_number=account_details_3.account_number and Recurring_Status = True

-> Show account number, debit card number, Balance and Recurring status of all the customers who have recurring customer as true

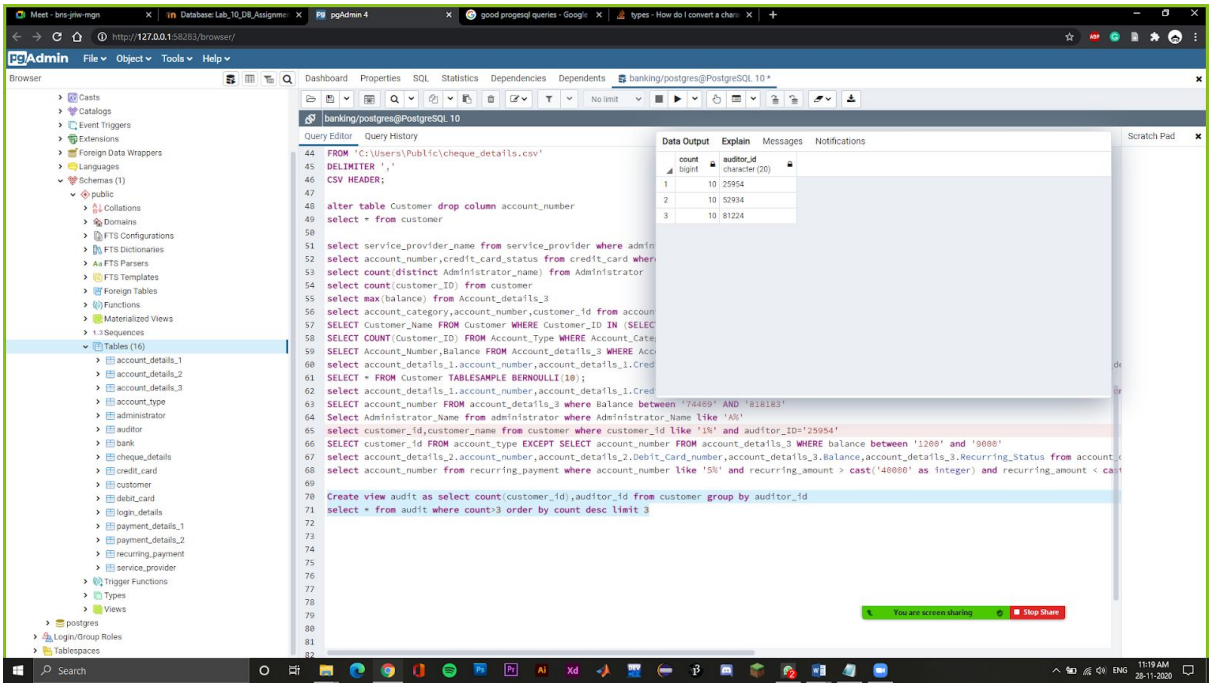


select account_number from recurring_payment where account_number like '5%' and recurring_amount > cast('40000' as integer) and recurring_amount < cast('80000' as integer)
-> Showing account number which has recurring payment between 40000 and 80000.



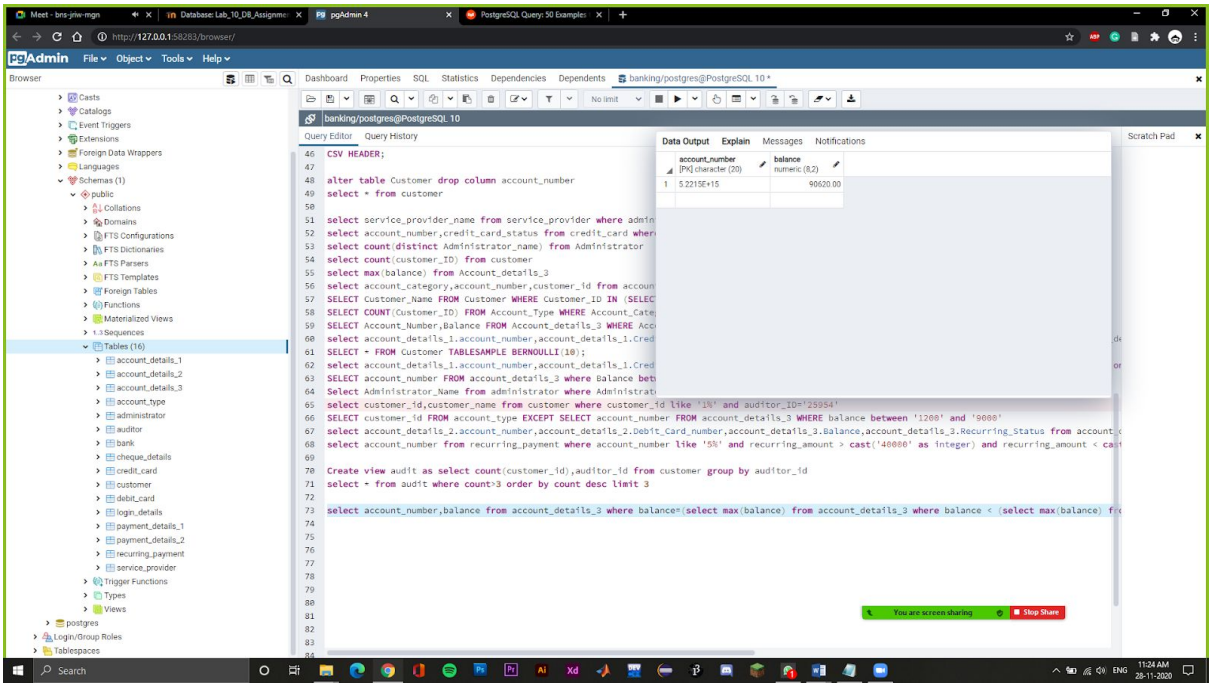
Create view audit as select count(customer_id),auditor_id from customer group by auditor_id
select * from audit where count>3 order by count desc limit 3

Show top 3 auditors who are auditing highest number of customer accounts

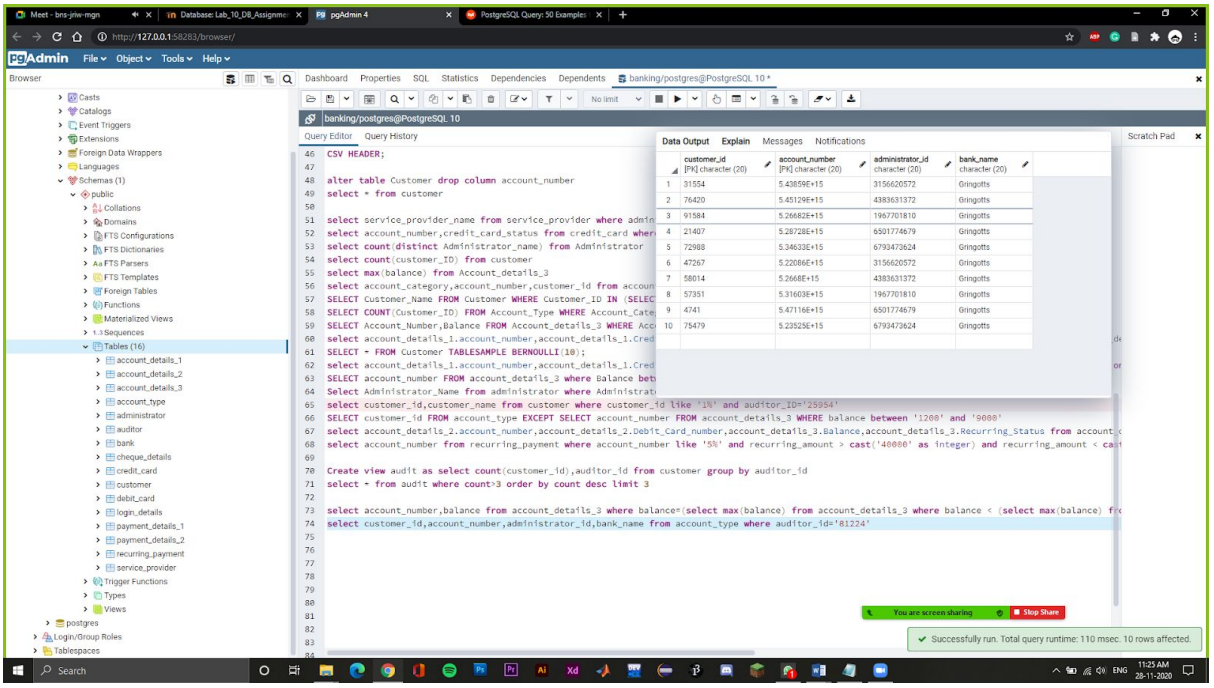


select account_number,balance from account_details_3 where balance=(select max(balance) from account_details_3 where balance < (select max(balance) from account_details_3))

Show second highest balance among all the accounts present

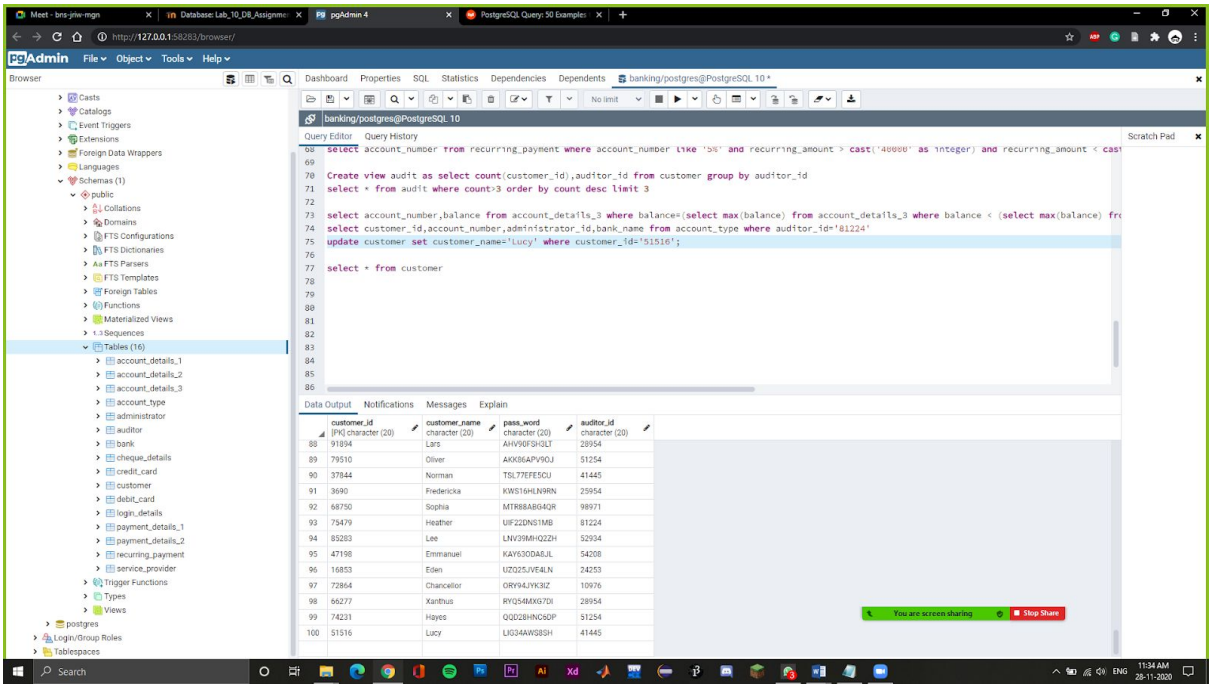


select customer_id,account_number,administrator_id,bank_name from account_type where auditor_id='81224'
-> Showing customer id, account number, administrator id, bank name when auditor id is 81224.



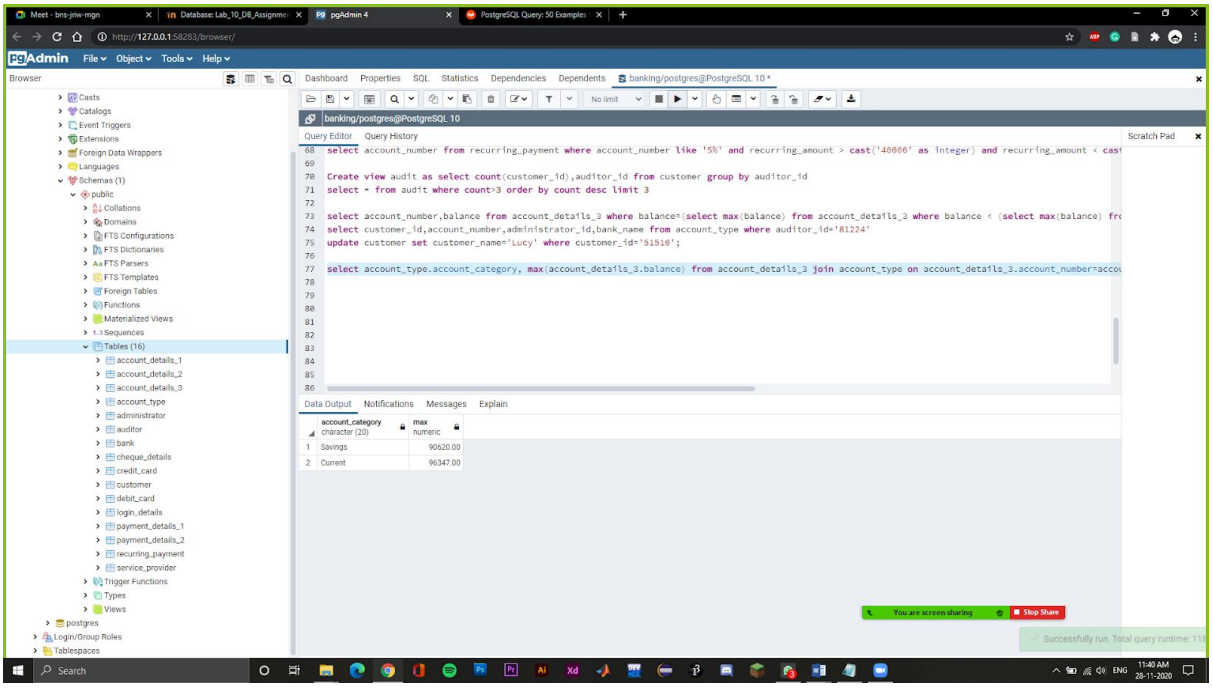
update customer set customer_name='Lucy' where customer_id='51516';

Update customer name as lucy whose customer id is 51516



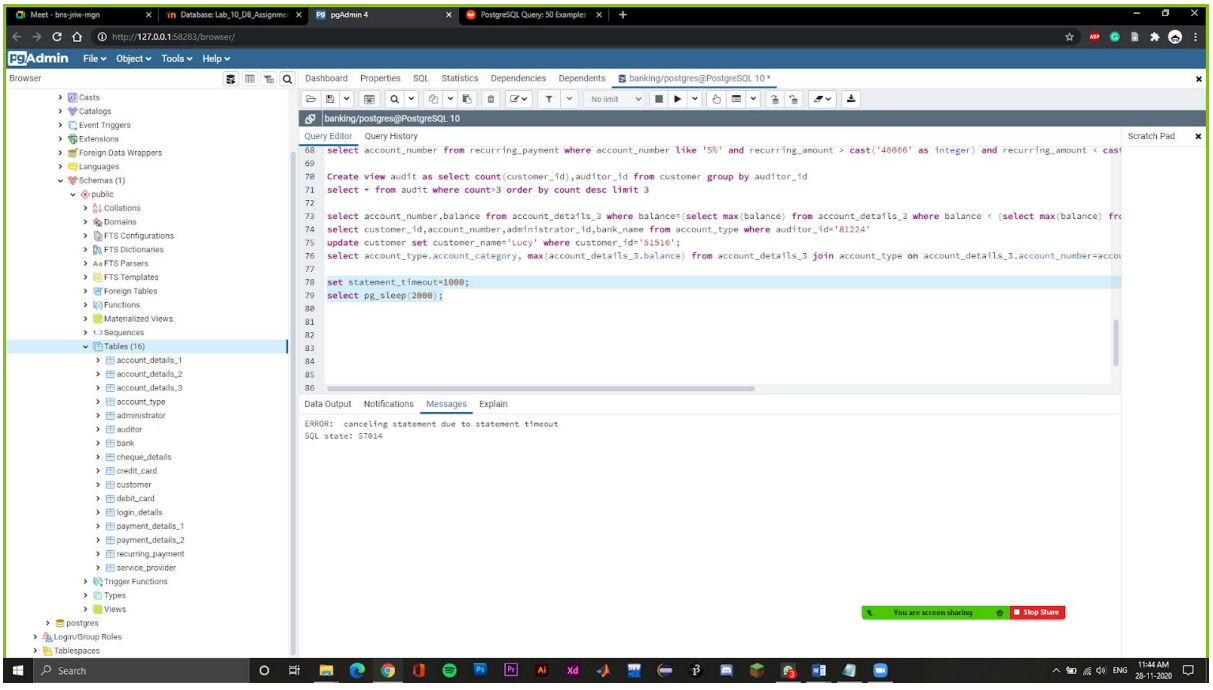
select account_type.account_category, max(account_details_3.balance) from account_details_3 join account_type on account_details_3.account_number=account_type.account_number group by account_type.account_category

Show highest balance of each account category



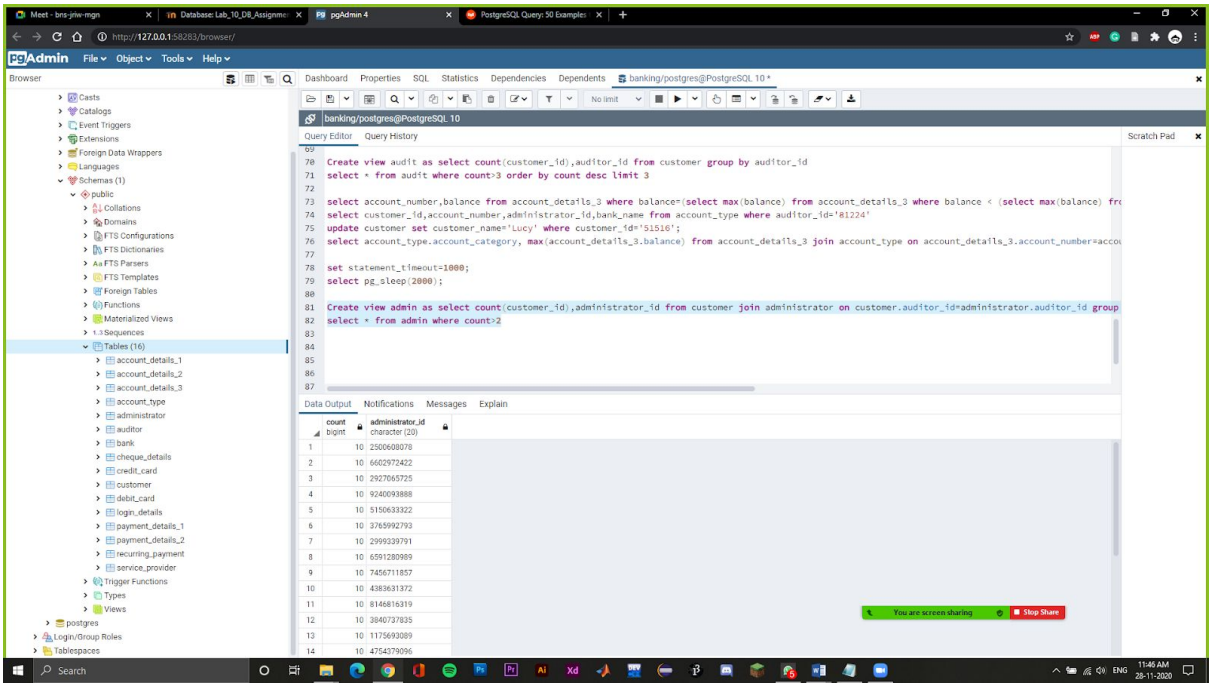
set statement_timeout=1000;
select pg_sleep(2000);

Set max query time as 1000 ms



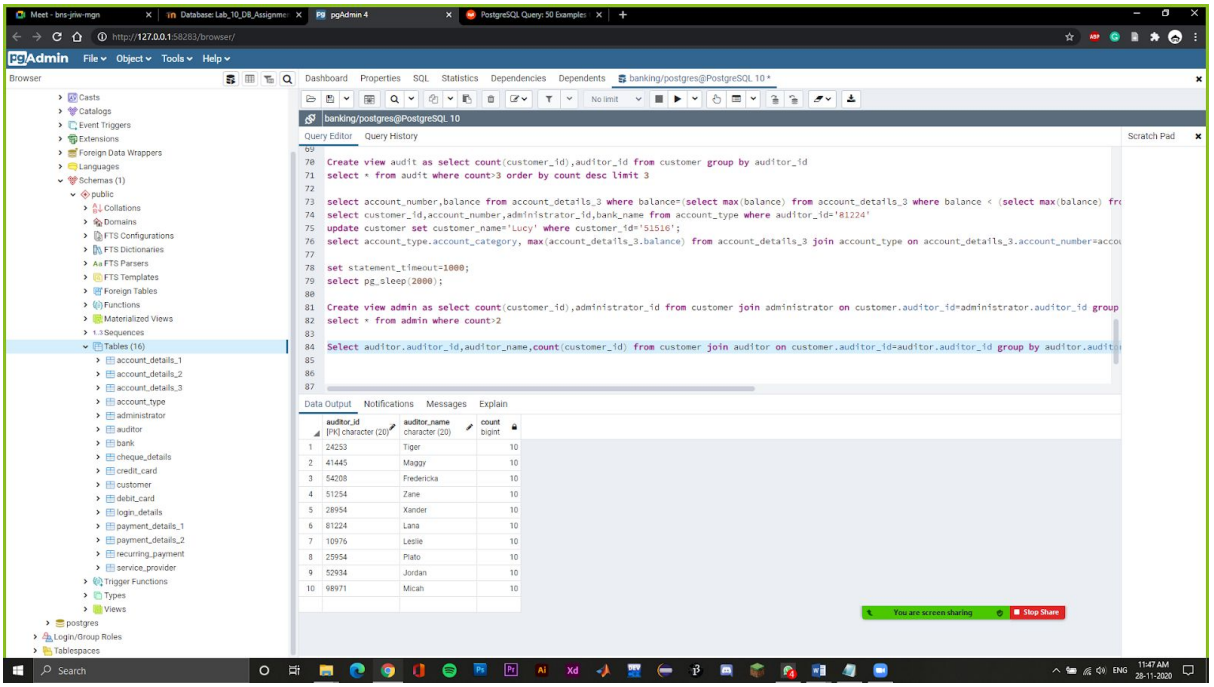
Create view admin as select count(customer_id),administrator_id from customer join administrator on customer.auditor_id=administrator.auditor_id group by administrator_id select * from admin where count>2

-> Created a view which has administrator id along with total of the customer id which the administrator has server and printing them which has count greater than 2.

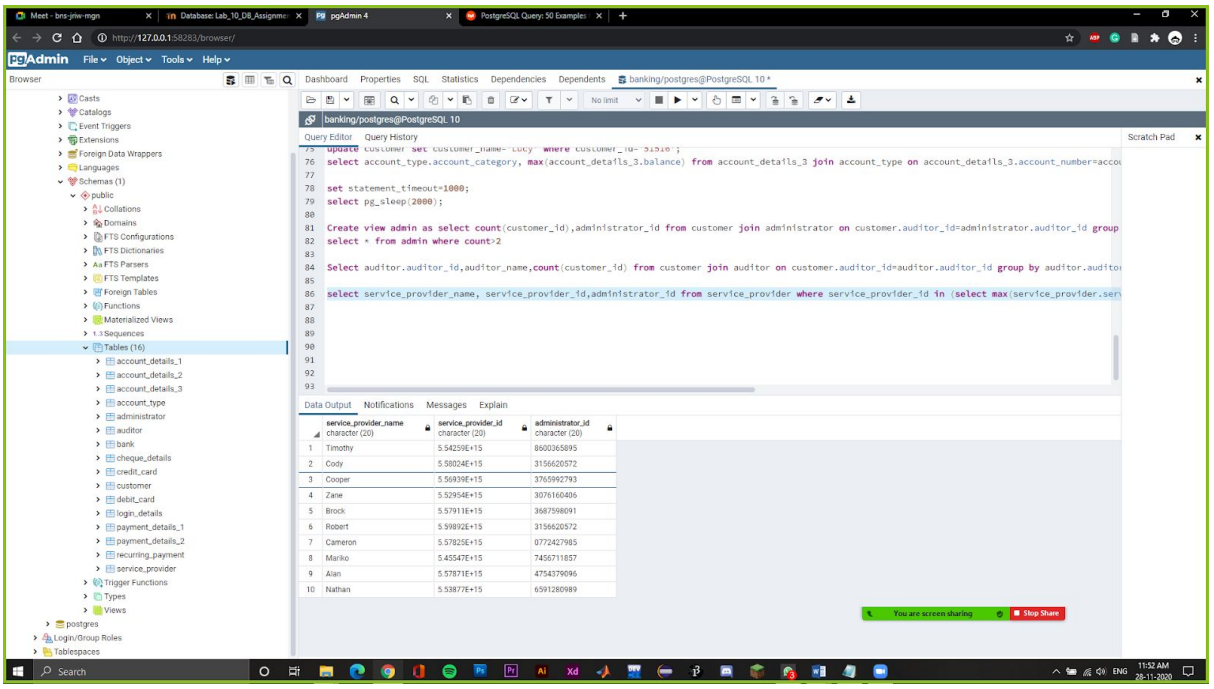


Select auditor.auditor_id,auditor_name,count(customer_id) from customer join auditor on customer.auditor_id=auditor.auditor_id group by auditor.auditor_id

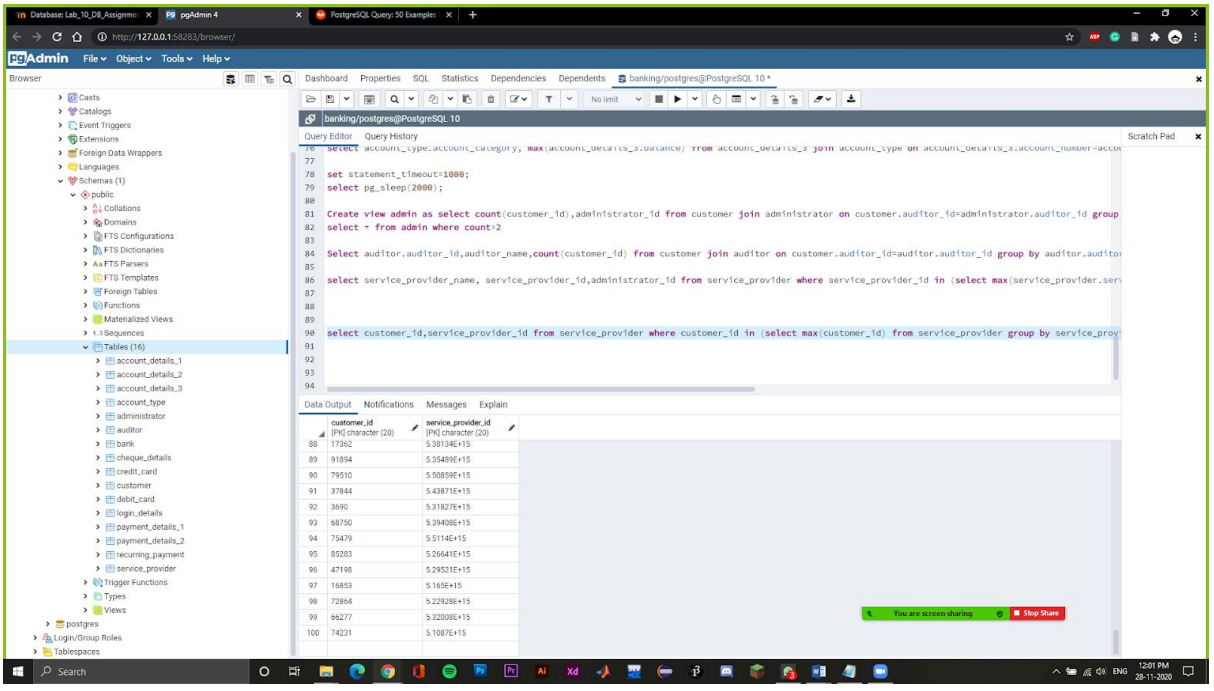
-> Showing auditor id, auditor name and count of the customer id by performing join operation.



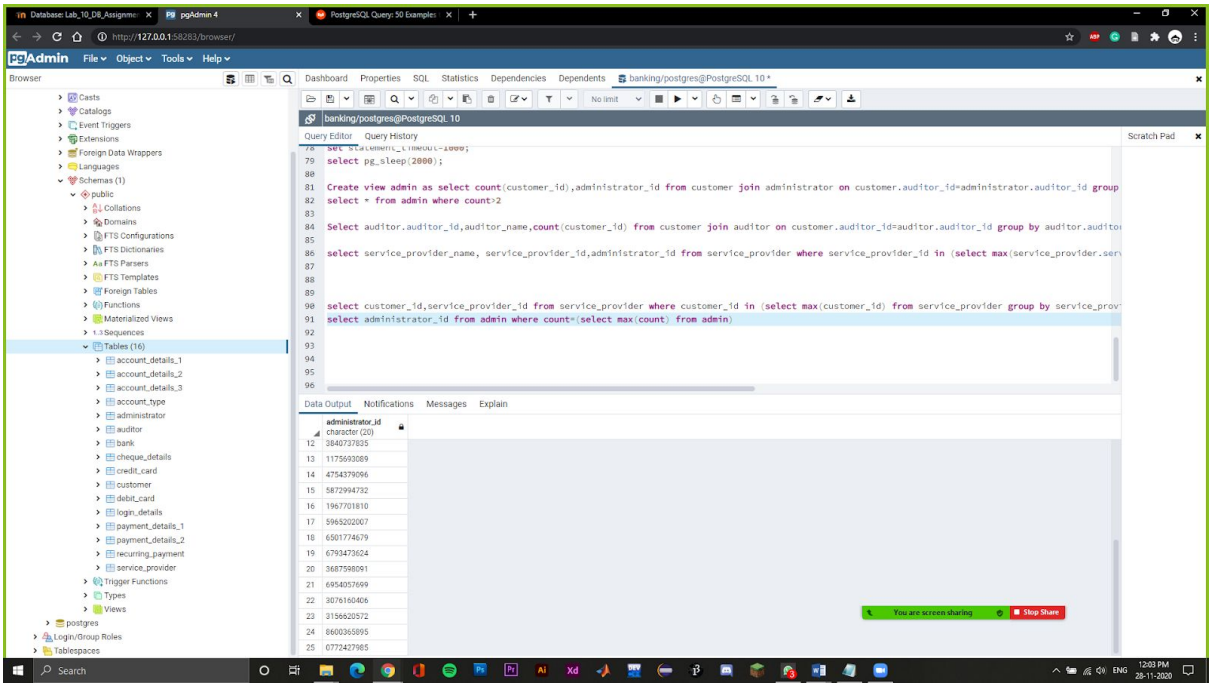
select service_provider_name, service_provider_id,administrator_id from service_provider where service_provider_id in (select max(service_provider.service_provider_id) from service_provider group by auditor_id)



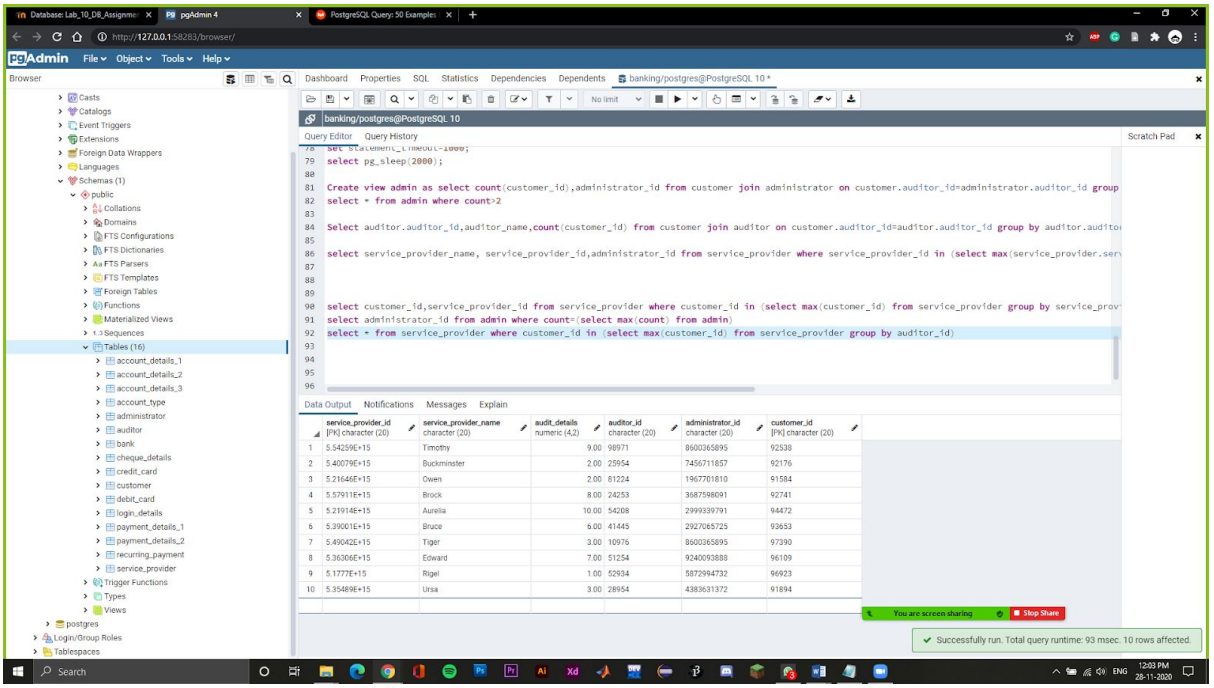
select customer_id,service_provider_id from service_provider where customer_id in (select max(customer_id) from service_provider group by service_provider_id)



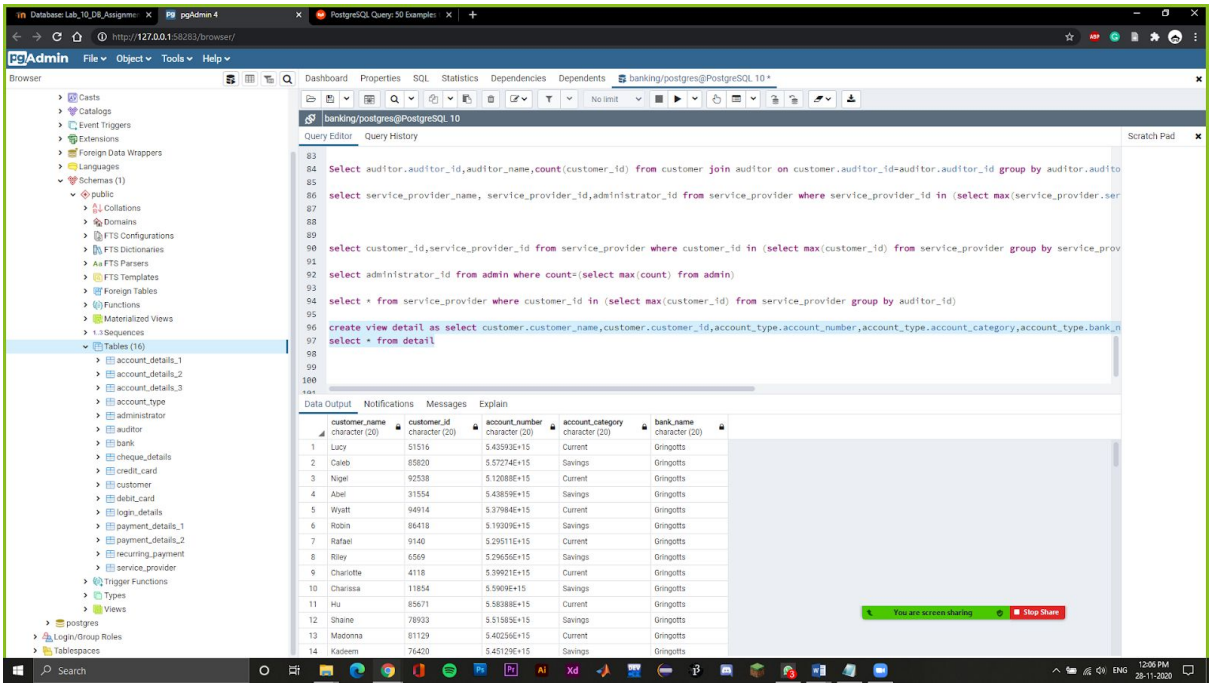
select administrator_id from admin where count=(select max(count) from admin)



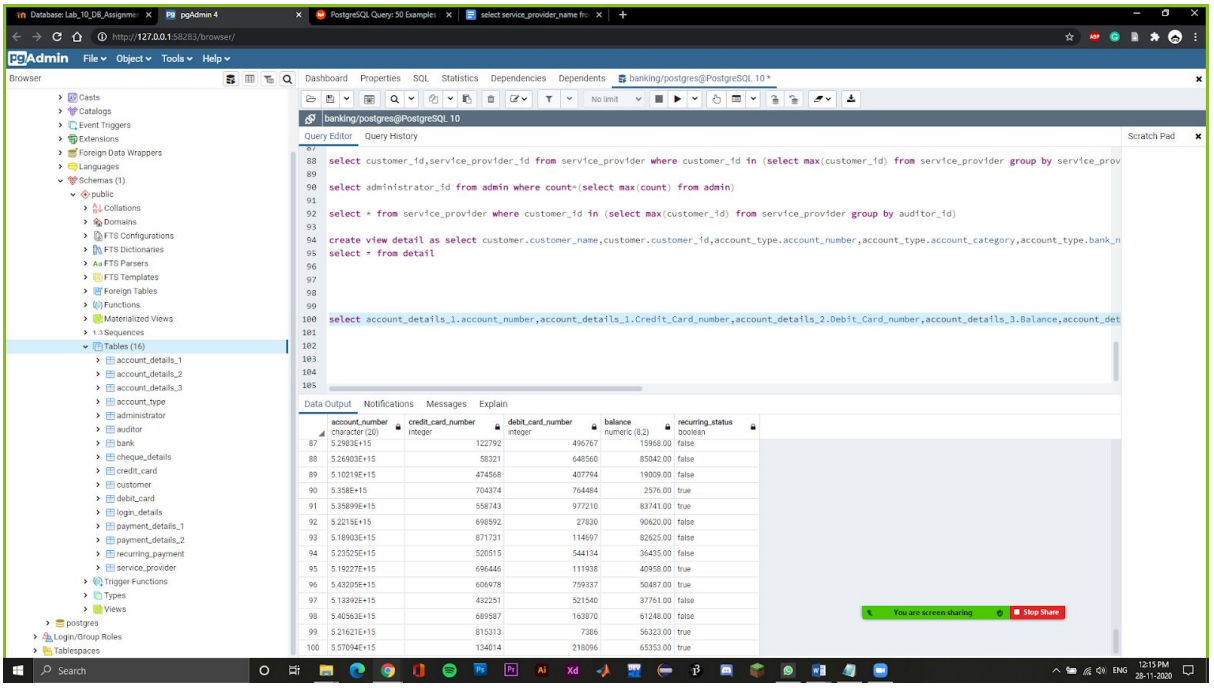
select * from service_provider where customer_id in (select max(customer_id) from service_provider group by auditor_id)



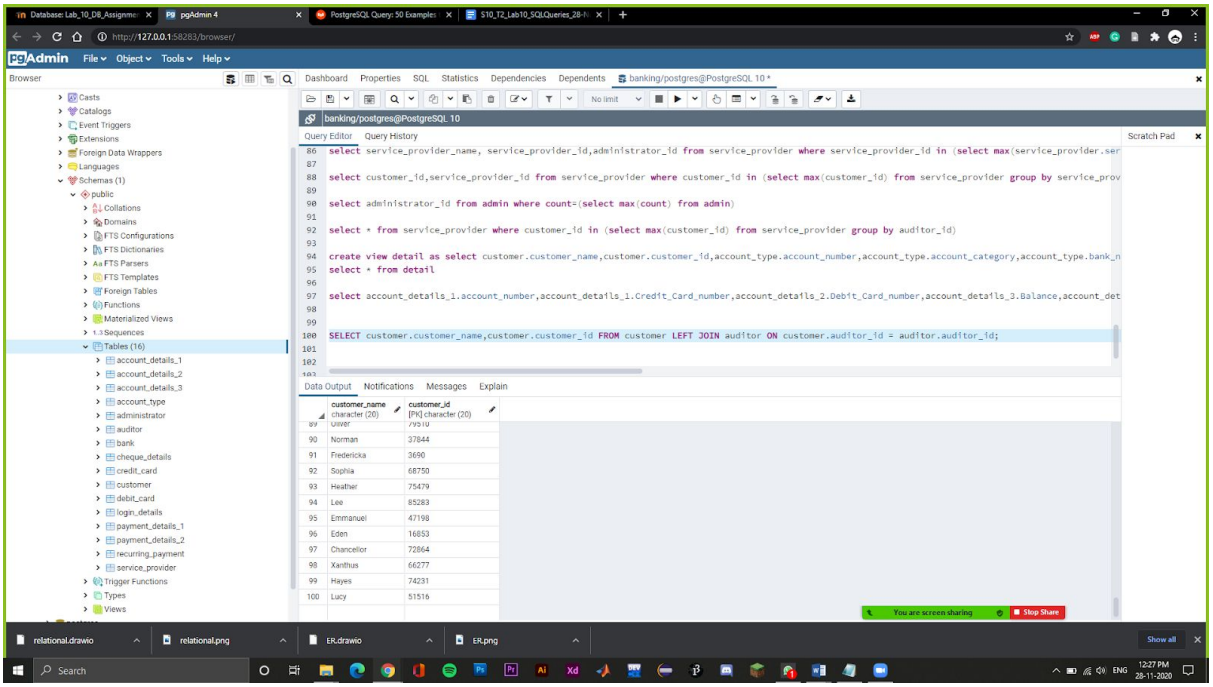
create view detail as select customer.customer_name,customer.customer_id,account_type.account_number,account_type.account_category,account_type.bank_name from customer join account_type on customer.customer_id=account_type.customer_id
select * from detail



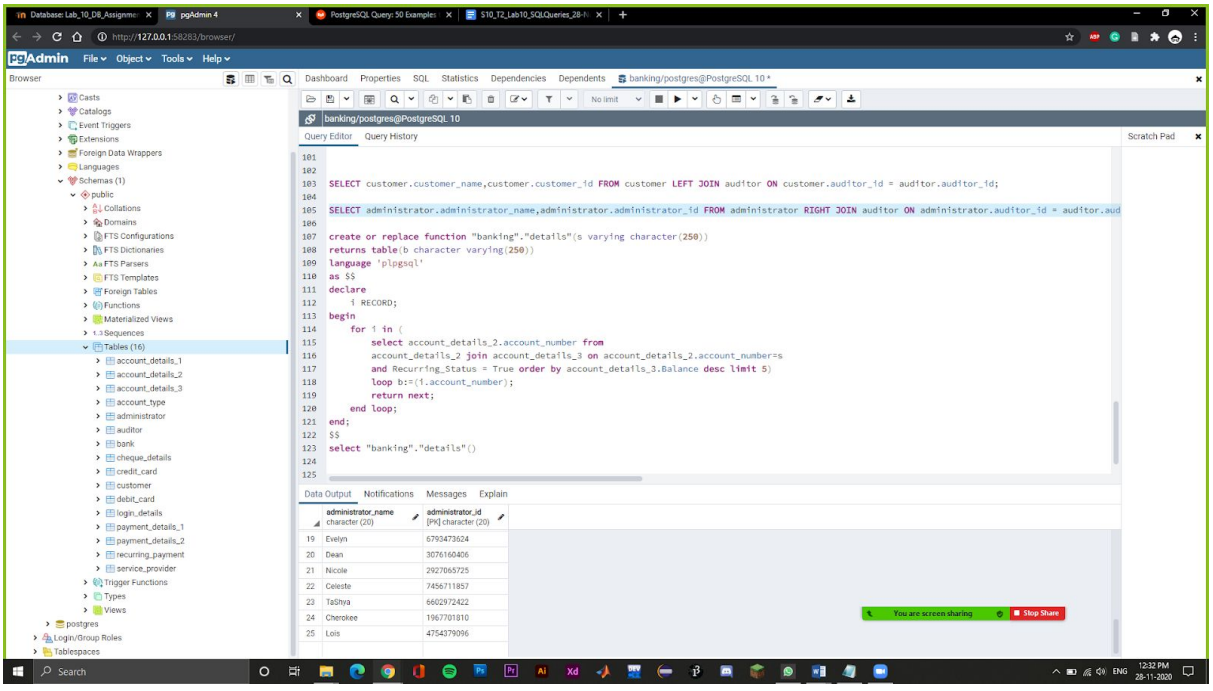
select account_details_1.account_number,account_details_1.Credit_Card_number,account_details_2.Debit_Card_number,account_details_3.Balance,account_details_3.Recurring_Status from account_details_1 join account_details_3 on account_details_1.account_number=account_details_3.account_number join account_details_2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



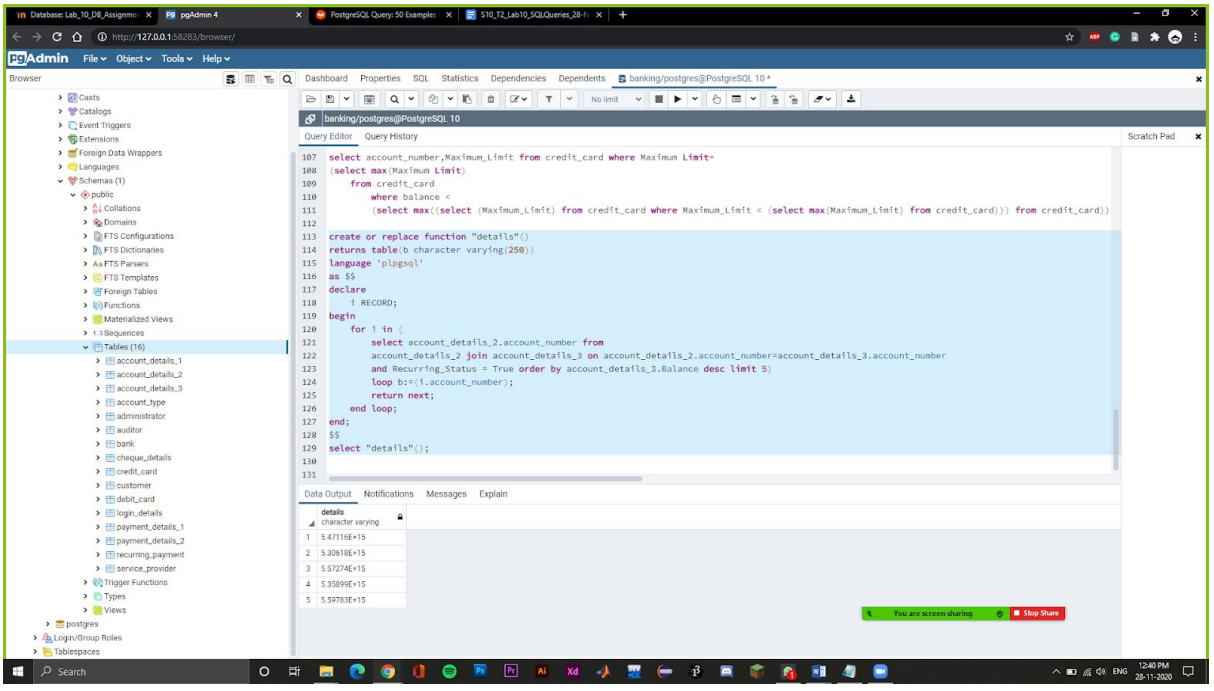
SELECT customer.customer_name,customer.customer_id FROM customer LEFT JOIN auditor ON customer.auditor_id = auditor.auditor_id;



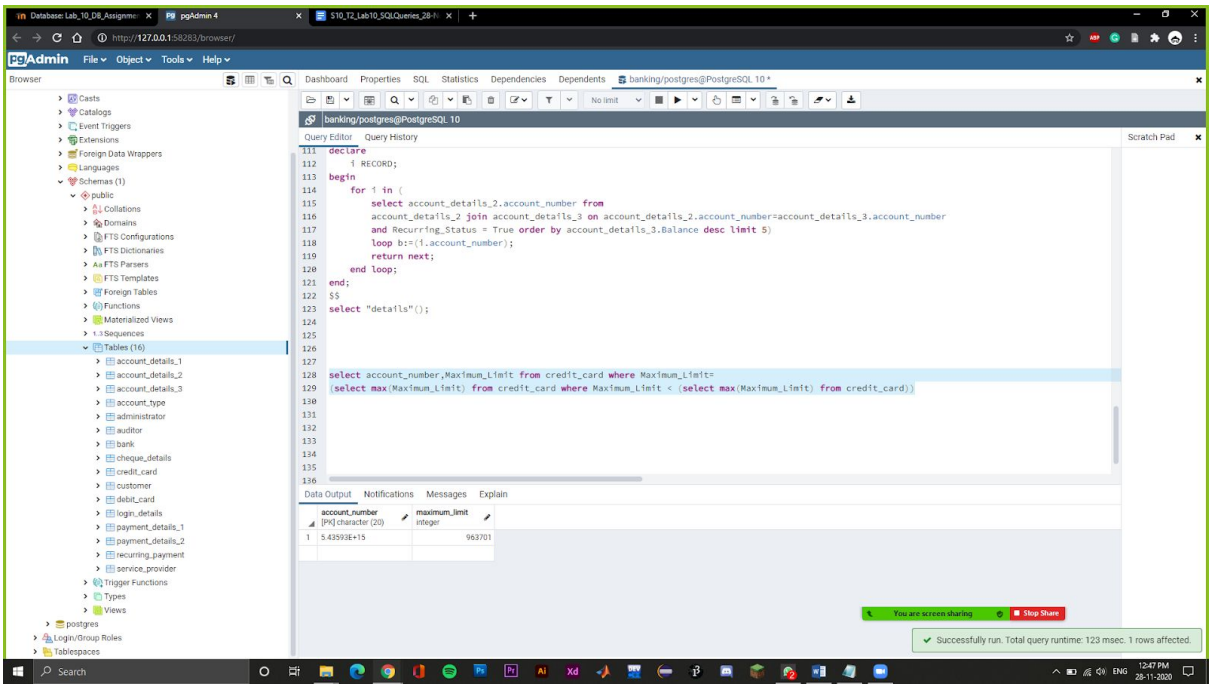
SELECT administrator.administrator_name,administrator.administrator_id FROM administrator RIGHT JOIN auditor ON administrator.auditor_id = auditor.auditor_id;



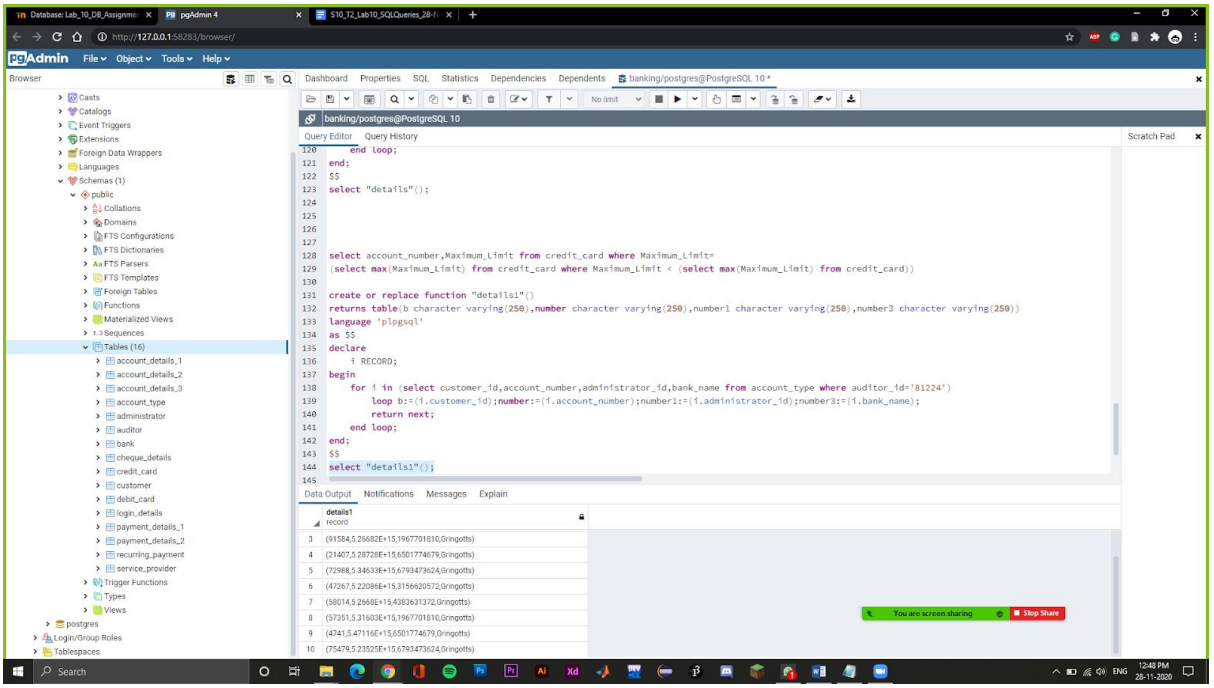
```
create or replace function "details"()
returns table(b character varying(250))
language 'plpgsql'
as $$
declare
	i RECORD;
begin
	for i in (
		select account_details_2.account_number from
		account_details_2 join account_details_3 on
account_details_2.account_number=account_details_3.account_number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



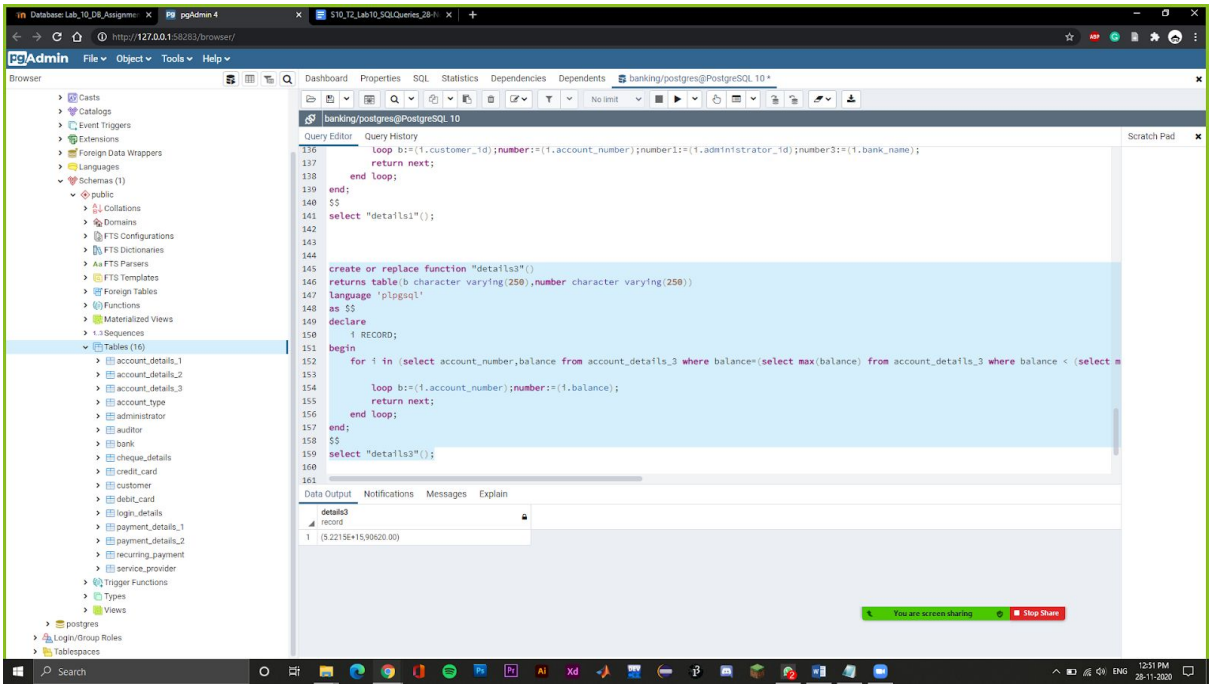
select account_number,Maximum_Limit from credit_card where Maximum_Limit=
(select max(Maximum_Limit) from credit_card where Maximum_Limit < (select
max(Maximum_Limit) from credit_card))



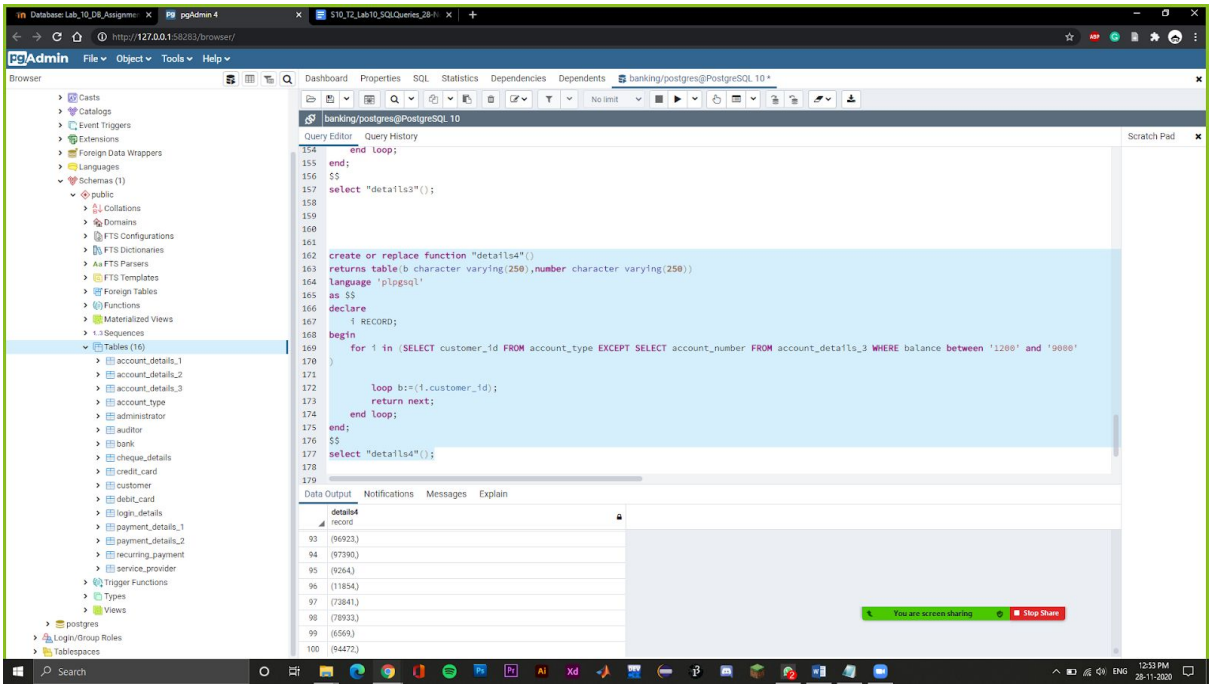
```
create or replace function "details1"()
returns table(b character varying(250),number character varying(250),number1 character
varying(250),number3 character varying(250))
language 'plpgsql'
as $$
declare
	i RECORD;
begin
	for i in (select customer_id,account_number,administrator_id,bank_name from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



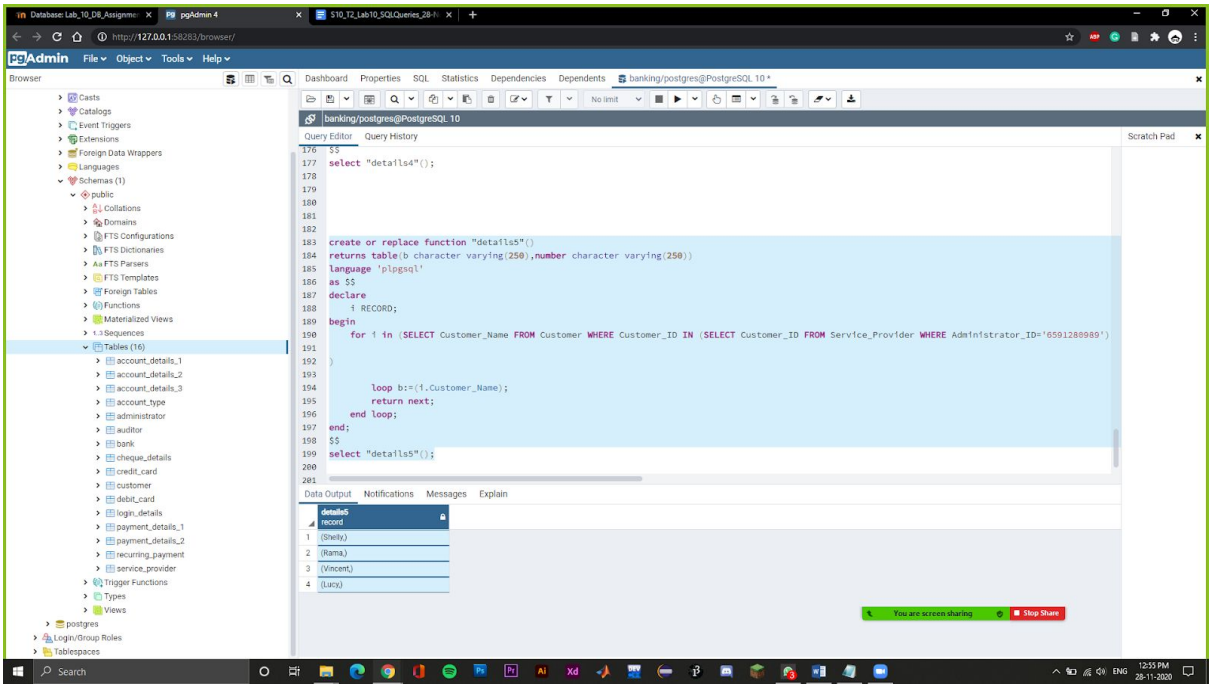
```
create or replace function "details3"()
returns table(b character varying(250),number character varying(250))
language 'plpgsql'
as $$
declare
	i RECORD;
begin
	for i in (select account_number,balance from account_details_3 where
balance=(select max(balance) from account_details_3 where balance < (select
max(balance) from account_details_3)))

		loop b:=(i.account_number);number:=(i.balance);
		return next;
	end loop;
end;
$$
select "details3"();
```



```
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 or replace function "details5"()
returns table(b character varying(250),number character varying(250))
language 'plpgsql'
as $$
declare
	i RECORD;
begin
	for i in (SELECT Customer_Name FROM Customer WHERE Customer_ID IN
(SELECT Customer_ID FROM Service_Provider WHERE Administrator_ID='6591280989')

)

		loop b:=(i.Customer_Name);
		return next;
	end loop;
end;
$$
select "details5"();
```



For the removal of redundancies, for most of the cases, i.e. many of the tables had a single primary key and so there were no redundancies in that. For two tables that are Encashment details, and Account details, we removed the redundancies while removing and modifying the DDL and schema accordingly. We analysed each and every table this way.

For the anomalies,i.e delete,insert,and update,

For any table if we change a tuple then the corresponding foreign keys associated with it also needs to be updated. Similarly we can extend it to the delete and insert anomalies as well. This consistency needs to be maintained and we have analyzed it for all the tables. It has been done in a generic sense to remove redundancies.

Credit Card: (BCNF)

- Primary Key : Account Number
- Foreign Key : Account Number
- Functional Dependency

  Account Number →Maximum Limit ,Credit Card Status, Expiry Date

- Since we have atomic attributes, hence it is in first normal form. Furthermore we have a single attribute primary key, hence it is in second normal form. Since there is one candidate key, hence it is in BCNF as well.

Debit Card (BCNF)

- Primary Key : Account Number
- Foreign Key : Account Number
- Functional Dependency

  Account Number →Credit Card Status, Expiry Date

- Since we have atomic attributes, hence it is in first normal form. Furthermore we have a single attribute primary key, hence it is in second normal form. Since there is one candidate key, hence it is in BCNF as well

Recurring Payment (BCNF)

- Primary Key : Account Number

- Foreign Key : Account Number
- Functional Dependency

  Account Number →Beneficiary account, Recurring amount

- Since we have atomic attributes, hence it is in first normal form. Furthermore we have a single attribute primary key, hence it is in second normal form. Since there is one candidate key, hence it is in BCNF as well.

Login Details (BCNF)

- Primary Key : Customer ID
- Foreign Key : Customer ID
- Functional Dependency

  Customer ID → Password

- Since we have atomic attributes, hence it is in first normal form. Furthermore we have a single attribute primary key, hence it is in second normal form. Since there is one candidate key, hence it is in BCNF as well

Payment Details (BCNF)

- Primary Key : Payment ID
- Foreign Key : Account Number
- Functional Dependency

  Payment ID → Account Number, Beneficiary Account Number, Amount

Account details (BCNF)

- Primary Key : Account Number
- Functional Dependency

  Account Number → Balance, Credit Card number, Debit Card number, Recurring Status

  Credit Card number→ Balance, Debit Card number, Recurring Status

  Debit Card number→ Balance, Recurring Status

(Account Number,Credit Card number) - 1

(Account Number, Debit Card number,) -2

(Account Number,Balance Recurring Status) - 3

- We have applied Heath's theorem for transforming a non-BCNF table to a BCNF table.

Let us take an example for explaining the heath's theorem where we have Initialized S = {R}

While S has a relation R' that is not in BCNF do:

Pick a FD: X->Y that holds in R' and violates BCNF

Add the relation XY to S

Update R' = R'-Y

Return S

So, now if s={ABCDE}

S = {ACDE, AB} // Pick FD: A->B which violates BCNF

S = {ACE, AB, CD} // Pick FD: C->D which violates BCNF

// Return S as all relations are in BCNF

Bank (BCNF)

- PRIMARY KEY:- Bank Name
- Functional Dependency

  Bank Name → Policy Version
- Since we have atomic attributes, hence it is in first normal form. Furthermore we have a single attribute primary key, hence it is in second normal form. Since there is one candidate key, hence it is in BCNF as well.

Auditor (BCNF)

- Primary Key : Auditor ID
- Foreign Key : Auditor ID
- Functional Dependency

Auditor ID →Auditor Name ,Audit Stage

- Since we have atomic attributes, hence it is in first normal form. Furthermore we have a single attribute primary key, hence it is in second normal form. Since there is one candidate key, hence it is in BCNF as well.

Encashment Details (BCNF)

- Primary Key : Payment ID
- Foreign Key : Account Number
- Functional Dependency

Payment ID→Cheque Number , Amount

Cheque Number → Amount

(Payment ID,Cheque Number) -1

(Payment ID,Amount)-2

- We have applied Heath's theorem for transforming a non-BCNF table to a BCNF table.

Let us take an example for explaining the heath's theorem where we have Initialized S = {R}

While S has a relation R' that is not in BCNF do:

Pick a FD: X->Y that holds in R' and violates BCNF

Add the relation XY to S

Update R' = R'-Y

Return S

So, now if s={ABCDE}

S = {ACDE, AB} // Pick FD: A->B which violates BCNF

S = {ACE, AB, CD} // Pick FD: C->D which violates BCNF

// Return S as all relations are in BCNF

Cheque details (BCNF)

- Primary Key : Account Number
- Foreign Key : Account Number
- Functional Dependency

Account Number → Cheques encashed, Cheque payments, Check books issued, Cheque Status

- Since we have atomic attributes, hence it is in first normal form. Furthermore we have a single attribute primary key, hence it is in second normal form. Since there is one candidate key, hence it is in BCNF as well.

Service Provider (BCNF)

- PRIMARY KEY:- Service Provider ID
- FOREIGN KEY:- Customer ID, Administrator ID, Auditor ID
- Functional Dependency

  Service Provider ID →Service Provider Name, Audit Details

- Since we have atomic attributes, hence it is in first normal form. Furthermore we have a single attribute primary key, hence it is in second normal form. Since there is one candidate key, hence it is in BCNF as well.

Administrator (BCNF)

- PRIMARY KEY:- Administrator ID
- FOREIGN KEY:- Bank Name, Auditor ID
- Functional Dependency

  Administrator ID →Administrator Name

- Since we have atomic attributes, hence it is in first normal form. Furthermore we have a single attribute primary key, hence it is in second normal form. Since there is one candidate key, hence it is in BCNF as well.

Account Type (BCNF)

- Primary Key : Auditor ID
- Foreign Key : Account Number, Administrator ID, Service Provider ID, Auditor ID, Bank Name
- Functional Dependency

  Auditor ID → Account Number, Administrator ID, Service Provider ID, Bank Name

Customer (BCNF)

- Primary Key : Customer ID

- Foreign Key : Auditor ID
- Functional Dependency

  Customer ID →Customer Name,Account number,Password,Auditor ID

- Since we have atomic attributes, hence it is in first normal form. Furthermore we have a single attribute primary key, hence it is in second normal form. Since there is one candidate key, hence it is in BCNF as well.