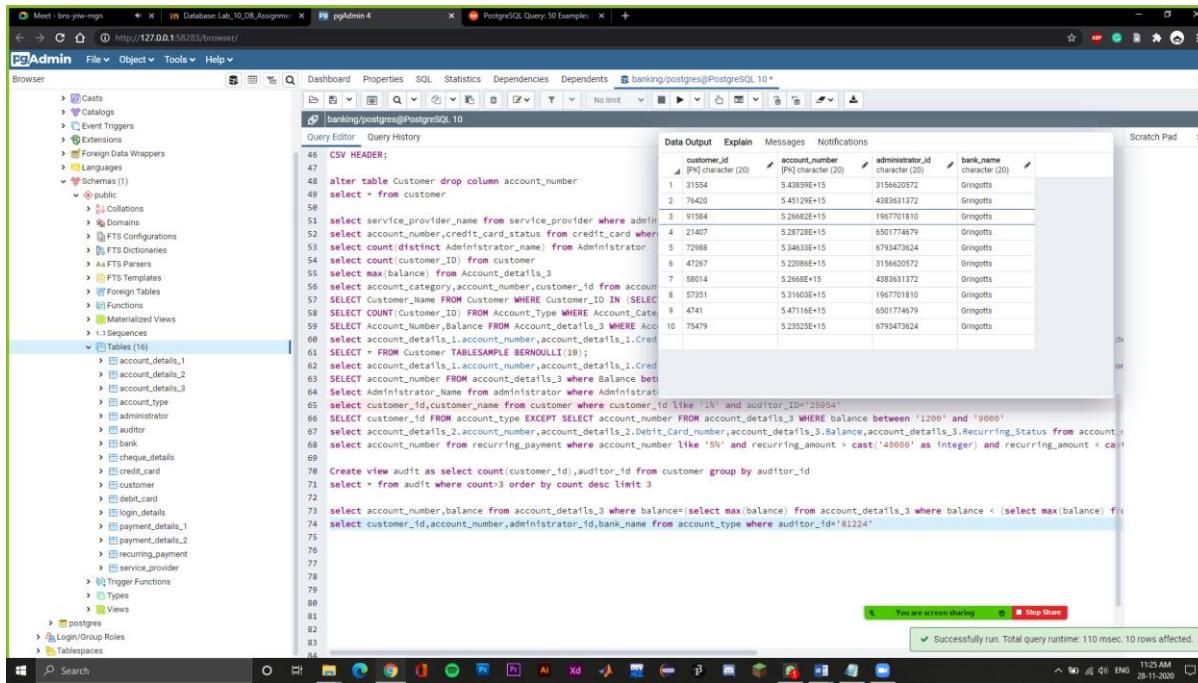


select customer_id,account_number,administrator_id,bank_name from account_type where auditor_id='81224'
 ->Showing customer id, account number, administrator id, bank name when auditor id is 81224.



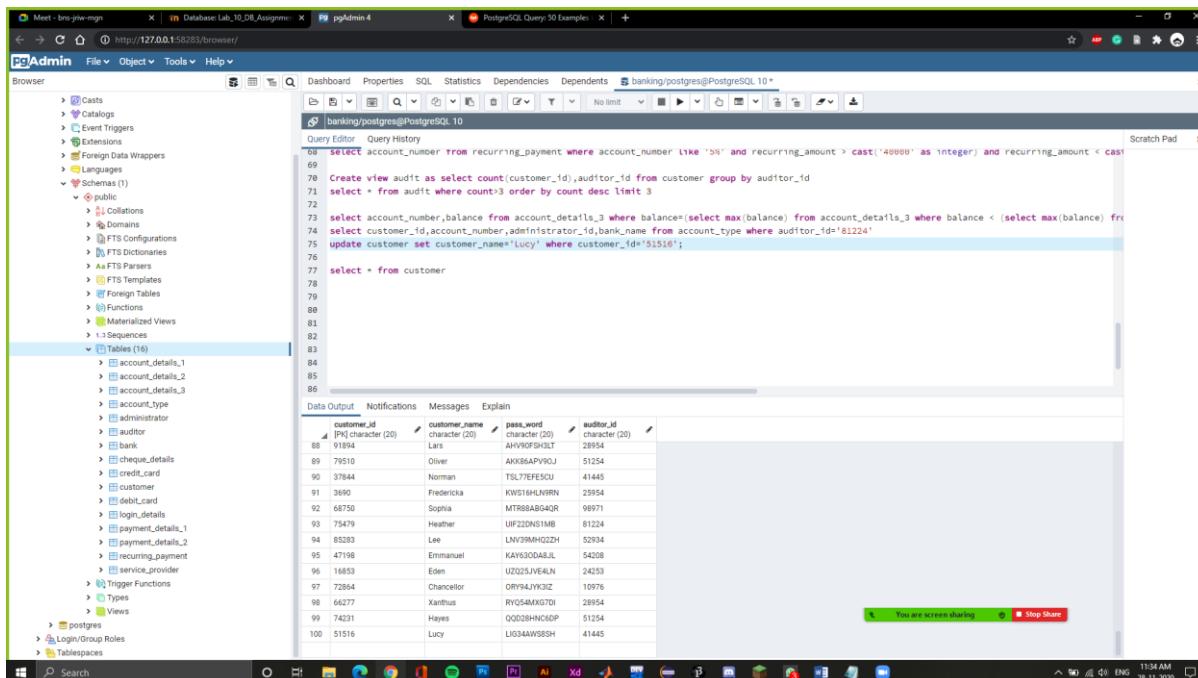
The screenshot shows the pgAdmin 4 interface with the following details:

- Browser:** Meet - bns-grw-mgn - Database Lab_10_DB_Assignment - pgAdmin 4 - PostgreSQL Query: 50 Examples
- Query Editor:** banking/postgres@PostgreSQL 10*


```

46 CSV HEADER;
47
48 alter table customer drop column account_number
49 select * from customer
50
51 select service_provider_name from service_provider where adminstrator_id=81224
52 select account_number,credit_card,status from credit_card where
53 select count(distinct administrator_name) from Administrator
54 select count(customer_ID) from customer
55 select max(balance) from account_details_3
56 select account_category,account_number,customer_id from account
57 SELECT Customer_Name FROM Customer WHERE Customer_ID IN (SELECT
58 SELECT COUNT(Customer_ID) FROM Account_Type WHERE Account_Cat
59 SELECT Account_Number,Balance FROM Account_Details_3 WHERE Acco
60 select account_details_1.account_number,account_details_1.Credit
61 SELECT * FROM Customer TABLESAMPLE BERNOULLI(10);
62 select account_details_1.account_number,account_details_1.Credit
63 select account_number,account_details_1.account_number,account_det
64 Select Administrator_Name from administrator where administrator_id=81224
65 select customer_id,customer_name from customer where customer_id like '8%' and auditor_id='81224'
66 SELECT customer_id FROM account_type EXCEPT SELECT account_number,account_details_3 WHERE balance between '1200' and '9000'
67 select account_details_2.account_number,account_details_2.Debit_Card_number,account_details_3.Balance,account_details_3.Recurring_Status from account
68 select account_number from recurring_payment where account_number like '5%' and recurring_amount > cast('40000' as integer) and recurring_amount < cast('60000' as integer)
69
70 Create view audit as select count(customer_id),auditor_id from customer group by auditor_id
71 select * from audit where count>3 order by count desc limit 3
72
73 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))
74 select customer_id,account_number,administrator_id,bank_name from account_type where auditor_id='81224'
75 update customer set customer_name='Lucy' where customer_id='51516';
76
77 select * from customer
78
79
80
81
82
83
84
85
86
      
```
- Data Output:** Shows the results of the query, listing 10 rows of customer data.
- Notifications:** Shows a message: "Successfully run. Total query runtime: 110 msec. 10 rows affected."

update customer set customer_name='Lucy' where customer_id='51516';
 ->Update customer name as lucy whose customer id is 51516



The screenshot shows the pgAdmin 4 interface with the following details:

- Browser:** Meet - bns-grw-mgn - Database Lab_10_DB_Assignment - pgAdmin 4 - PostgreSQL Query: 50 Examples
- Query Editor:** banking/postgres@PostgreSQL 10*


```

69 select account_number from recurring_payment where account_number like '5%' and recurring_amount > cast('40000' as integer) and recurring_amount < cast('60000' as integer)
70 Create view audit as select count(customer_id),auditor_id from customer group by auditor_id
71 select * from audit where count>3 order by count desc limit 3
72
73 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))
74 select customer_id,account_number,administrator_id,bank_name from account_type where auditor_id='81224'
75 update customer set customer_name='Lucy' where customer_id='51516';
76
77 select * from customer
78
79
80
81
82
83
84
85
86
      
```

- Data Output:** Shows the results of the update query, listing 100 rows of customer data.
- Notifications:** Shows a message: "Successfully run. Total query runtime: 113 msec. 100 rows affected."

```
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
```

The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:

```
68 select account_number from recurring_payment where account_number like '5%' and recurring_amount > cast('40000' as integer) and recurring_amount < cast('50000' as integer)
69
70 Create view audit as select count(customer_id),auditor_id from customer group by auditor_id
71 select * from audit where count>3 order by count desc limit 3
72
73 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 where balance > (select min(balance) from account_details_3 where account_number like '5%')) and account_number like '5%' )
74 select customer_id,account_number,administrator_id,bank_name from account_type where auditor_id='81224'
75 update customer set customer_name='Lucy' where customer_id='51516';
76
77 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
78
79
80
81
82
83
84
85
86
```

The results table shows:

account_category	max
character (20)	numeric
1. Savings	96620.00
2. Current	96347.00

Message bar: You are screen sharing. Stop Share. Successfully run. Total query runtime: 11s.

```
set statement_timeout=1000;
select pg_sleep(2000);
->Set max query time as 1000 ms
```

The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:

```
68 select account_number from recurring_payment where account_number like '5%' and recurring_amount > cast('40000' as integer) and recurring_amount < cast('50000' as integer)
69
70 Create view audit as select count(customer_id),auditor_id from customer group by auditor_id
71 select * from audit where count>3 order by count desc limit 3
72
73 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 where balance > (select min(balance) from account_details_3 where account_number like '5%')) and account_number like '5%' )
74 select customer_id,account_number,administrator_id,bank_name from account_type where auditor_id='81224'
75 update customer set customer_name='Lucy' where customer_id='51516';
76
77 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
78
79 set statement_timeout=1000;
80 select pg_sleep(2000);
81
82
83
84
85
86
```

The message bar shows: ERROR: canceling statement due to statement timeout. SQL state: 57014.

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.

The screenshot shows the pgAdmin 4 interface with the 'Query Editor' tab active. The query window contains the SQL code for creating the 'admin' view:

```
69 Create view audit as select count(customer_id),administrator_id from customer group by auditor_id
70 select * from audit where count>3 order by count desc limit 3
71
72
73 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))
74 select customer_id,account_number,administrator_id,bank_name from account_type where auditor_id='81224'
75 update customer set customer_name='Lucy' where customer_id='51516';
76 select account_type.account_category, max(account_details_3.balance) from account_details_3 join account_type on account_type.account_number=account_details_3.account_number
77
78 set statement_timeout=1000;
79 select pg_sleep(2000);
80
81 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
82 select * from admin where count>2
83
84
85
86
87
```

Below the query window, the 'Data Output' tab is selected, showing the results of the 'admin' view query:

count	administrator_id
1	2500608078
2	6602972422
3	2927065725
4	924093888
5	515053322
6	3745992793
7	2999339791
8	6591260969
9	745671857
10	438361372
11	8146816319
12	3840737835
13	1175693089
14	4754379056

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.

The screenshot shows the pgAdmin 4 interface with the 'Query Editor' tab active. The query window contains the SQL code for creating the 'audit' view:

```
69 Create view audit as select count(customer_id),auditor_id from customer group by auditor_id
70 select * from audit where count>3 order by count desc limit 3
71
72
73 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))
74 select customer_id,account_number,administrator_id,bank_name from account_type where auditor_id='81224'
75 update customer set customer_name='Lucy' where customer_id='51516';
76 select account_type.account_category, max(account_details_3.balance) from account_details_3 join account_type on account_type.account_number=account_details_3.account_number
77
78 set statement_timeout=1000;
79 select pg_sleep(2000);
80
81 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
82 select * from admin where count>2
83
84
85
86
87
```

Below the query window, the 'Data Output' tab is selected, showing the results of the 'audit' view query:

auditor_id	auditor_name	count
1	Tiger	10
2	Maggie	10
3	Fredenika	10
4	Zane	10
5	Xander	10
6	Lana	10
7	Leslie	10
8	Plato	10
9	Jordan	10
10	Micah	10

```
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.

```
17 update customer set customer_name='Lucy' where customer_id=3191;
18 select account_type.account_category, max(account_details_3.balance) from account_details_3 join account_type on account_details_3.account_number=account.account_number;
19
20 set statement_timeout=10000;
21 select pg_sleep(2000);
22
23 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;
24 select * from admin where count>2;
25
26 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;
27
28 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);
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
```

service_provider_name	service_provider_id	administrator_id
Timothy	5.54259E+15	8600365895
Cody	5.58024E+15	3156620572
Cooper	5.56938E+15	3765992793
Zane	5.52954E+15	3076160406
Brock	5.57911E+15	3687598091
Robert	5.59892E+15	3156620572
Cameron	5.57823E+15	0772427985
Marko	5.45547E+15	7456711857
Alan	5.57871E+15	4754379096
Nathan	5.53877E+15	6591280989

```
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)
```

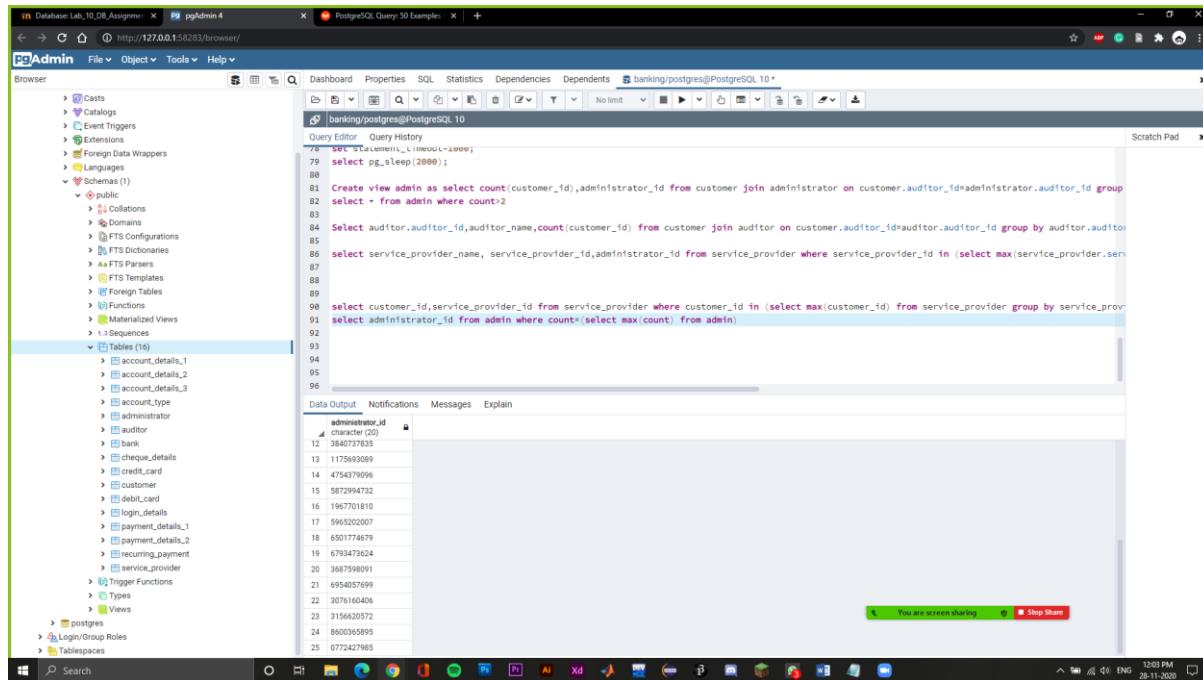
->Select Maximum Customer ID from the pool of customers who are being serviced by a certain service provider and show the corresponding service provider ID as well

```
70 update customer set customer_name='Lucy' where customer_id=3191;
71 select account_type.account_category, max(account_details_3.balance) from account_details_3 join account_type on account_details_3.account_number=account.account_number;
72
73 set statement_timeout=10000;
74 select pg_sleep(2000);
75
76 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;
77 select * from admin where count>2;
78
79 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;
80
81 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 service_provider_id);
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

customer_id	service_provider_id
7750	5.38134E+15
89	5.35489E+15
90	5.50859E+15
91	5.43871E+15
92	5.31827E+15
93	5.39408E+15
94	5.5114E+15
95	5.26641E+15
96	5.2951E+15
97	5.165E+15
98	5.22928E+15
99	5.32086E+15
100	5.1087E+15

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.



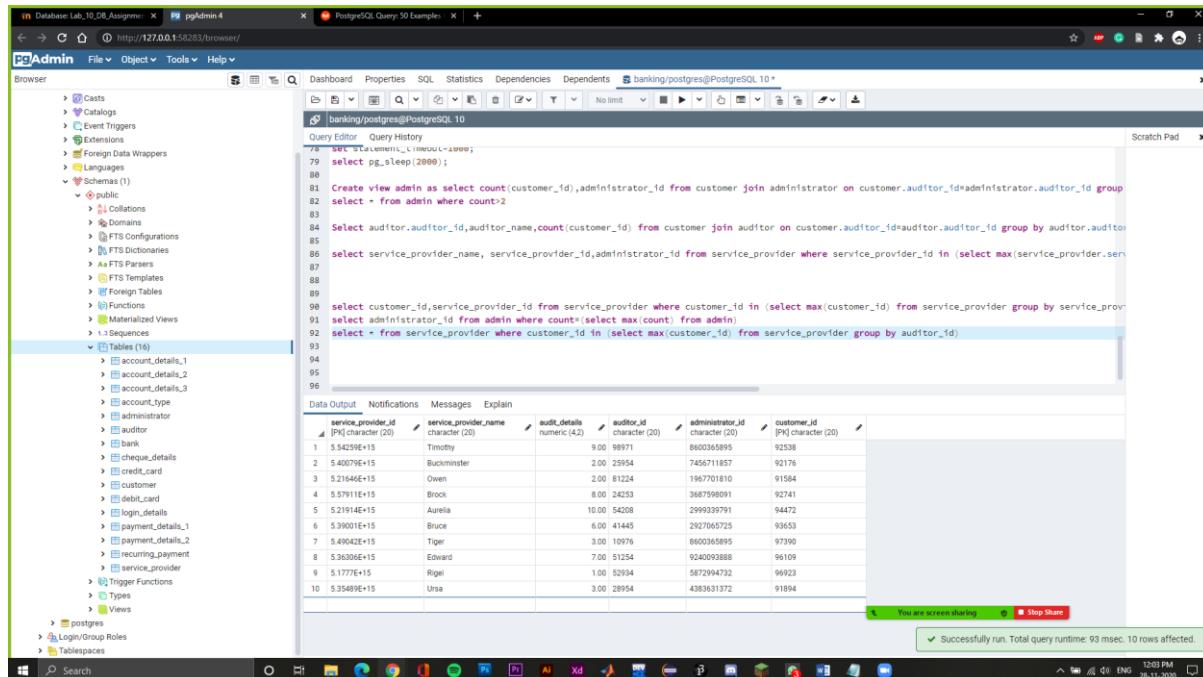
```

78 set statement_timeout=10000;
79 select pg_sleep(2000);
80
81 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
82 select * from admin where count=2
83
84 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
85
86 select service_provider_name, service_provider_id,administrator_id from service_provider where service_provider_id in (select max(service_provider_id) from service_provider group by service_provider_name)
87
88
89 select customer_id,service_provider_id from service_provider where customer_id in (select max(customer_id) from service_provider group by service_provider_name)
90
91 select administrator_id from admin where count=(select max(count) from admin)
92
93
94
95
96

```

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



```

78 set statement_timeout=10000;
79 select pg_sleep(2000);
80
81 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
82 select * from admin where count=2
83
84 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
85
86 select service_provider_name, service_provider_id,administrator_id from service_provider where service_provider_id in (select max(service_provider_id) from service_provider group by service_provider_name)
87
88
89 select customer_id,service_provider_id from service_provider where customer_id in (select max(customer_id) from service_provider group by auditor_id)
90
91 select administrator_id from admin where count=(select max(count) from admin)
92 select * from service_provider where customer_id in (select max(customer_id) from service_provider group by auditor_id)
93
94
95
96

```

service_provider_id	service_provider_name	audit_details	auditor_id	administrator_id	customer_id
1 5.54259E+15	Timothy	9.00	98971	8600365995	92538
2 5.40076E+15	Buckminster	2.00	25954	7450711857	92176
3 5.21646E+15	Owen	2.00	81224	1967701810	91584
4 5.57911E+15	Brock	8.00	24253	3687996091	92741
5 5.21914E+15	Aurelia	10.00	54208	299939791	94472
6 5.39001E+15	Bruce	6.00	41445	2927065725	93653
7 5.49042E+15	Tiger	3.00	10976	8600365995	97390
8 5.36306E+15	Edward	7.00	51254	9240938888	96109
9 5.17771E+15	Rigel	1.00	52934	587294732	96923
10 5.35489E+15	Ursa	3.00	28954	4383631372	91894

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.

```
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;
```

The screenshot shows the pgAdmin 4 interface with the 'Query Editor' tab active. The query window contains the SQL code to create a view named 'detail'. Below the code, the 'Data Output' pane displays the results of the query, showing 14 rows of customer data. The columns are labeled: customer_name, customer_id, account_number, account_category, and bank_name. The data includes entries for Lucy, Caleb, Nigel, Abel, Wyatt, Robin, Rafael, Riley, Charlotte, Charissa, Hu, Shaina, Madonna, and Kadem.

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

```
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
JOIN account_details_1 ON customer.customer_id = account_details_1.customer_id
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;
```

The screenshot shows the pgAdmin 4 interface with the 'Query Editor' tab active. The query window contains the SQL code to create a view named 'detail' with a complex join condition. Below the code, the 'Data Output' pane displays the results of the query, showing 100 rows of joined customer and account details. The columns are labeled: account_number, credit_card_number, debit_card_number, balance, and recurring_status. The data includes various account numbers, card numbers, balances, and recurring status values.

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.

```

Database Lab_10_DB_Assignment | pgAdmin 4 | http://127.0.0.1:58283/browser/
PostgreSQL Query: 50 Examples | S10_T2_Lab10_SQLQueries_28-h | +
PgAdmin File Object Tools Help *
Browser
  > Casts
  > Catalogs
  > Event Triggers
  > Extensions
  > Foreign Data Wrappers
  > Languages
  > Schemas (1)
    > public
      > Collations
      > Domains
      > FTS Configurations
      > FTS Dictionaries
      > Aa FTS Parsers
      > FTS Templates
      > Foreign Tables
      > Functions
      > Materialized Views
      > Sequences
    > Tables (16)
      > account_details_1
      > account_details_2
      > account_details_3
      > account_type
      > administrator
      > bank
      > cheque_details
      > credit_card
      > customer
      > debit_card
      > login_details
      > payment_details_1
      > payment_details_2
      > recurring_payment
      > service_provider
      > Trigger Functions
      > Types
      > Views
Query Editor Query History
  select service_provider_name, service_provider_id, administrator_id from service_provider where service_provider_id in (select max(service_provider_id) from service_provider group by service_provider)
  select customer_id, service_provider_id from service_provider where customer_id in (select max(customer_id) from service_provider group by service_provider)
  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
  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.Expiry_Date
  SELECT customer.customer_name, customer.customer_id FROM customer LEFT JOIN auditor ON customer.auditor_id = auditor.auditor_id;
Data Output Notifications Messages Explain
  customer_name   customer_id
  character(20)   [PK] character(20)
  1 Uver          1
  90 Norma        37844
  91 Frederick    3690
  92 Sophia       68750
  93 Heather      75479
  94 Lee           85283
  95 Emmanuel     47198
  96 Eden          16853
  97 Chancellor   72864
  98 Xanthus      66277
  99 Hayes         74231
  100 Lucy         51516
  You are screen sharing. Stop Share
Show all

```

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.

```

Database Lab_10_DB_Assignment | pgAdmin 4 | http://127.0.0.1:58283/browser/
PostgreSQL Query: 50 Examples | S10_T2_Lab10_SQLQueries_28-h | +
PgAdmin File Object Tools Help *
Browser
  > Casts
  > Catalogs
  > Event Triggers
  > Extensions
  > Foreign Data Wrappers
  > Languages
  > Schemas (1)
    > public
      > Collations
      > Domains
      > FTS Configurations
      > FTS Dictionaries
      > Aa FTS Parsers
      > FTS Templates
      > Foreign Tables
      > Functions
      > Materialized Views
      > Sequences
    > Tables (16)
      > account_details_1
      > account_details_2
      > account_details_3
      > account_type
      > administrator
      > bank
      > cheque_details
      > credit_card
      > customer
      > debit_card
      > login_details
      > payment_details_1
      > payment_details_2
      > recurring_payment
      > service_provider
      > Trigger Functions
      > Types
      > Views
Query Editor Query History
  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 "banking"."details"(s varying character(250))
  returns table(b character varying(250))
  language 'plpgsql'
  as $$
  declare
  i RECORDS;
  begin
  for i in (
  select account_details_2.account_number
  from account_details_2 join account_details_3 on account_details_2.account_number =
  and Recurring_Status = True order by account_details_3.Balance desc limit 5)
  loop b:=i.account_number;
  return next;
  end loop;
  end;
  $$;
  select "banking"."details"()
Data Output Notifications Messages Explain
  administrator_name   administrator_id
  character(20)   [PK] character(20)
  19 Evelyn        6793473624
  20 Dean           3076160406
  21 Nicole         2927065725
  22 Celeste        7456711857
  23 TaShya         6602972422
  24 Cherokee       1967701810
  25 Lois            4754379096
  You are screen sharing. Stop Share
Show all

```

```

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

The screenshot shows the pgAdmin 4 interface with the following details:

- Left Panel (Browser):** Shows the database schema structure under the 'banking' schema, including Casts, Catalogs, Event Triggers, Extensions, Languages, Schemas, and Tables (16).
- Right Panel (Query Editor):** Displays the SQL code for the 'details' function.
- Data Output:** Shows the results of the 'details' function, which are the top 5 account numbers with the highest balances.

```

107 select account_number,Maximum_Limit from credit_card where Maximum_Limit=
108 (select max(Maximum_Limit)
109   from credit_card
110   where balance <
111     (select max((select Maximum_Limit) from credit_card where Maximum_Limit < (select max(Maximum_Limit) from credit_card)) from credit_card)
112
113 create or replace function "details"()
114 returns table(b character varying(250))
115 language 'plpgsql'
116 as $$
117 declare
118     i RECORD;
119 begin
120     for i in (
121         select account_details_2.account_number from
122         account_details_2 join account_details_3 on account_details_2.account_number=account_details_3.account_number
123         and Recurring_Status = True order by account_details_3.Balance desc limit 5)
124     loop b:=(i.account_number);
125     return next;
126    end loop;
127 end;
128 $$
129 select "details"();
130
131

```

details	character varying
1	5.47116E+15
2	5.30618E+15
3	5.57274E+15
4	5.35999E+15
5	5.59783E+15

```

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

The screenshot shows the PgAdmin 4 interface with a query editor window. The query is:

```

111 declare
112     f RECORD;
113 begin
114     for f in (
115         select account_details_2.account_number from
116             account_details_2 join account_details_3 on account_details_2.account_number=account_details_3.account_number
117             and Recurring_Status = true order by account_details_3.Balance desc limit 5)
118     loop b:=f.account_number;
119     return next;
120     end loop;
121 end;
122 $$;
123 select "details"();
124
125
126
127
128 select account_number,Maximum_Limit from credit_card where Maximum_Limit=
129 (select max(Maximum_Limit) from credit_card where Maximum_Limit < (select max(Maximum_Limit) from credit_card))
130
131
132
133
134
135
136

```

The results pane shows the following table:

account_number	maximum_limit
543992e+15	963701

Message bar: You are screen sharing. Stop Share.

System status: Successfully run. Total query runtime: 123 msec. 1 rows affected.

```

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'

```

Database: Lab_10_DB_Assignment | pgAdmin 4 | http://127.0.0.1:58283/browser/ | S10_72_Lab10_SQL/Queries_28-1 | +
PgAdmin File Object Tools Help
Browser
> Casts
> Catalogs
> Event Triggers
> Extensions
> Foreign Data Wrappers
> Languages
> Schemas (1)
  > public
    > Collations
    > Domains
  > FTS Configurations
  > FTS Dictionaries
    > FTS Parsers
  > FTS Templates
  > Foreign Tables
  > Functions
  > Materialized Views
  > Sequences
Tables (16)
  > account_details_1
  > account_details_2
  > account_details_3
  > account_type
  > administrator
  > auditor
  > bank
  > cheque_details
  > credit_card
  > customer
  > debit_card
  > login_details
  > payment_details_1
  > payment_details_2
  > recurring_payment
  > service_provider
  > Trigger Functions
  > Types
  > Views
  > postgres
  > Login/Group Roles
  > Tablespaces
Query Editor
banking/postgres@PostgreSQL 10
128     end loop;
129   end;
130   $$
131   select "details1"();
132
133   select account_number,Maximum_Limit from credit_card where Maximum_Limit<
134   (select max(Maximum_Limit) from credit_card where Maximum_Limit < (select max(Maximum_Limit) from credit_card))
135
136   create or replace function "details1"()
137   returns table(b character varying(250),number character varying(250),number1 character varying(250),number3 character varying(250))
138   language 'plpgsql'
139   as $$
140   declare
141       i RECORD;
142   begin
143       for i in (select customer_id,account_number,administrator_id,bank_name from account_type where auditor_id='81224')
144           loop
145               b:=(i.customer_id);number:=(i.account_number);number1:=(i.administrator_id);number3:=(i.bank_name);
146               return next;
147           end loop;
148   end;
149   $$
150   select "details1"();
Data Output Notifications Messages Explain
details1
record
1 (91584.526682E+15,1967701810.0,ingotts)
2 (21407.528728E+15,5501774679.0,ingotts)
3 (72988.34832E+15,6793473624.0,ingotts)
4 (47267.522086E+15,3156602052.0,ingotts)
5 (58014.525668E+15,4383631372.0,ingotts)
6 (57351.531602E+15,1967701810.0,ingotts)
7 (47415.47116E+15,5501774679.0,ingotts)
8 (75479.523325E+15,6793473624.0,ingotts)
9 (47415.47116E+15,5501774679.0,ingotts)
10 (75479.523325E+15,6793473624.0,ingotts)

```

```

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 "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"();

```

record
93 (96923)
94 (97390)
95 (9264)
96 (11854)
97 (73841)
98 (78933)
99 (6569)
100 (94472)

```

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

The screenshot shows the pgAdmin 4 interface with two tabs open:

- Query Editor:** Contains the SQL code for creating a function named `func_1()` that returns a trigger. It checks if `new.bank_name` is NULL. If so, it updates the `administrator` table to set `bank_name` to 'Gringotts' where `administrator_id` matches `new.administrator_id`. It then raises a notice stating 'Updated to default bank!'. The function ends with a return statement. It also creates a trigger named `"def_bank"` that triggers after an insert on the `administrator` table, executing the `func_1()` procedure for each row.
- Data Output:** Shows the result of the `INSERT` statement. It displays a single row inserted into the `administrator` table with the values: `(Audit_Details, Auditor_ID, Administrator_ID, Administrator_Name)` as `(18, 52934, 9240093889, 'Arka')`.
- Notifications:** Shows a single message: `NOTICE: Updated to default bank!`
- Messages:** Shows the message: `INSERT 0 1`
- Explain:** Shows the execution plan for the `INSERT` statement.
- Administrator Table Data:** Below the Query Editor, a table shows the data for the `administrator` table. The columns are `audit_details`, `auditor_id`, `administrator_id`, `administrator_name`, and `bank_name`. The data includes rows for Dean, Nicole, Celeste, TaShya, Cherokee, and Los, all with `bank_name` set to 'Gringotts'.
- Second Tab:** Shows a `Select * from administrator` query and a `DELETE FROM administrator WHERE administrator_id = '9240093889'` command. The resulting table shows the same data as the first tab, but with one row removed where `administrator_id` is 9240093889.

```

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

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with tables like 'account_details_1', 'account_details_2', and 'account_details_3'. The 'account_details_3' table is currently selected. The main area is the Query Editor, which contains the following SQL code:

```

1 Create or replace function func_3()
2 returns trigger
3 language 'plpgsql'
4 as $$
5 begin
6 if(new.balance < 100)
7 then raise notice 'Balance must be in 3 digits';
8 return old;
9 end if;
10 return new;
11 end
12 $$;

13
14 create trigger "bal"
15 before insert on account_details_3
16 for each row execute procedure func_3();

17
18 INSERT INTO account_details_3(Account_Number,Balance,Recurring_Status)
19 VALUES ('5.43999E+15',74,false);

20
21 delete from account_details_3 where balance = 74

22
23 select * from account_details_3

```

The message bar at the top right says: "You are currently running PostgreSQL 10. You can click here to see what's new in this version, however the current version is not supported by the documentation." A tooltip for the PostgreSQL 10 logo says: "Please click here for more information".