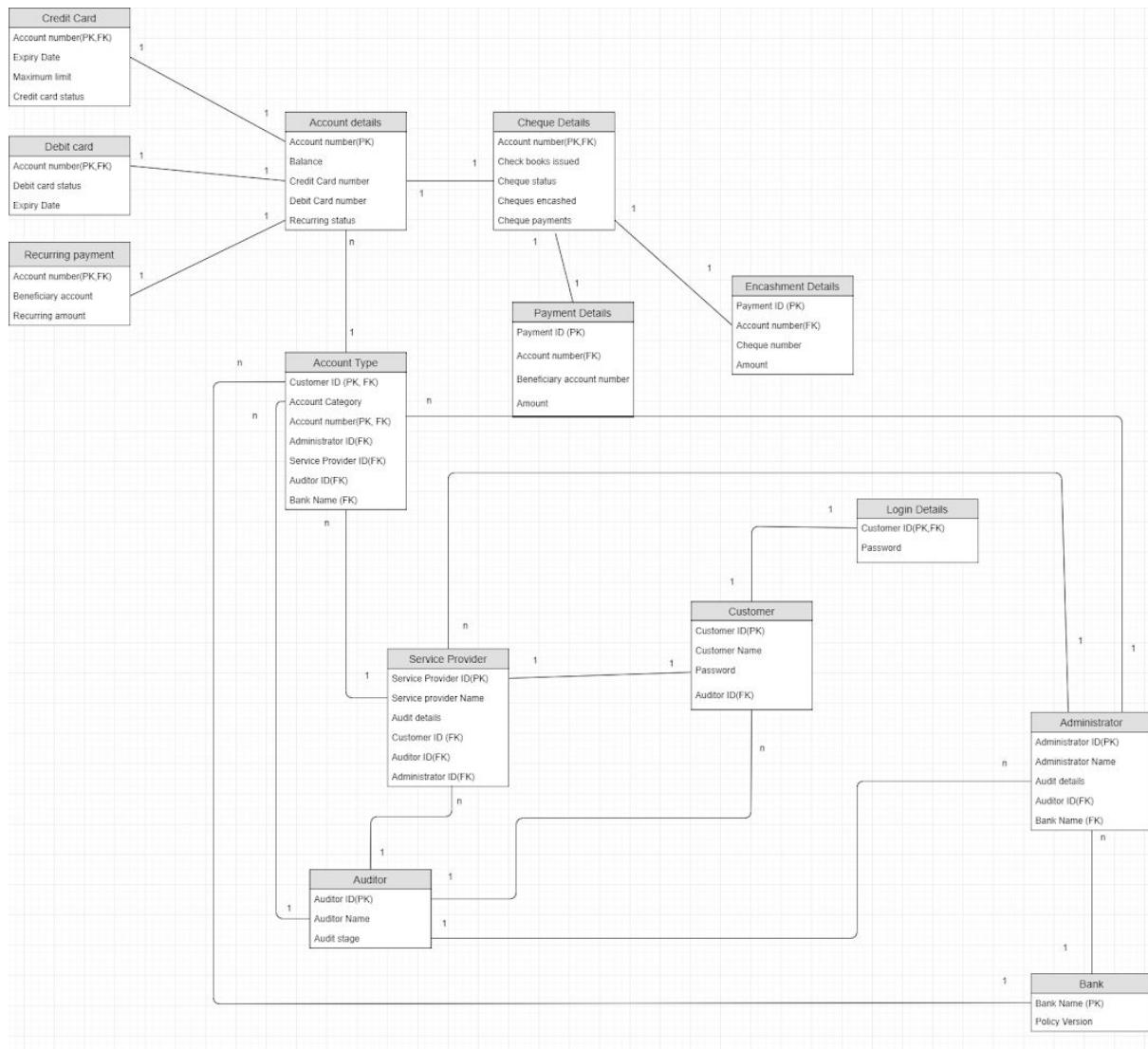


Updated ER diagram without normalization



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)

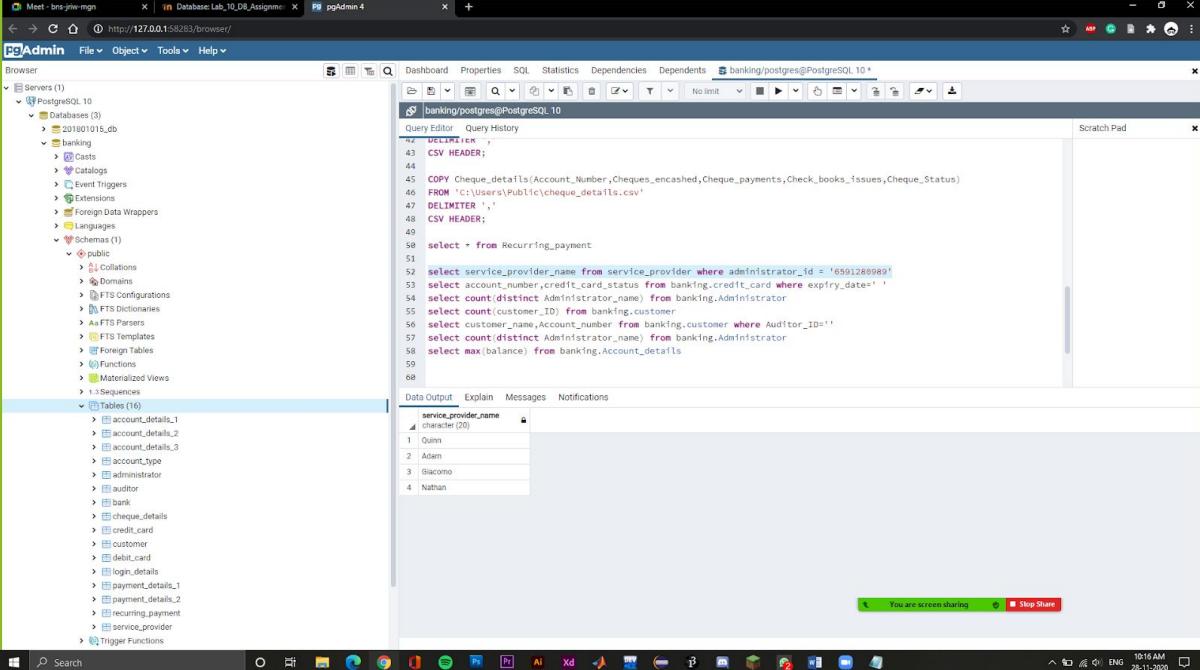
CREATE TABLE Payment_Details_1(
Payment_ID CHAR(20) NOT NULL,
Account_Number CHAR(20) NOT NULL,
Cheque_Number CHAR(20),
PRIMARY KEY (Payment_ID),
FOREIGN KEY (Account_Number) REFERENCES Cheque_Details
ON DELETE CASCADE
)

```

```
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:

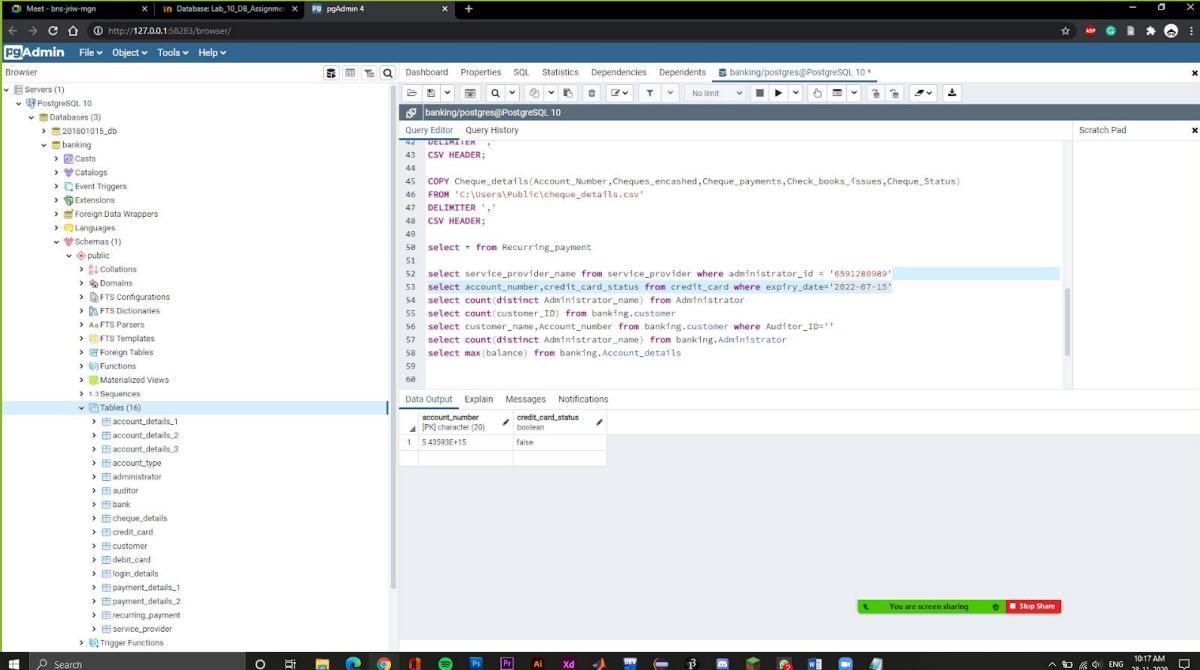
select service_provider_name from service_provider where administrator_id = '6591280989'
-> Here, we are selecting the name of the service provider for the administrator id 6591280989.



The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:`SELECT * FROM service_provider WHERE administrator_id = '6591280989'`

The results show a single row with the value 'Quinn'.

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.



The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:`SELECT * FROM credit_card WHERE expiry_date = '2022-07-15'`

The results show a single row with the account number '543993E15' and credit card status 'false'.

select count(distinct Administrator_name) from Administrator

-> Here, we are giving the total number of distinct administrators.

The screenshot shows the pgAdmin 4 interface. On the left, the 'Servers' tree view shows a connection to 'PostgreSQL 10' and the 'banking' database. The 'Tables' node under 'banking' has 16 entries. In the center, the 'Query Editor' window displays a SQL query to count distinct administrator names. The 'Data Output' tab shows the result: count = 25. A status bar at the bottom right indicates 'Successfully run. Total query runtime: 101 msec. 1 rows affected.'

```
DELIMITER ','
CSV HEADER;
COPY Cheque_details(Account_Number,Cheques_Encashed,Cheque_Payments,Check_Books_Issues,Cheque_Status)
FROM 'C:\Users\Public\cheque_details.csv'
DELIMITER ','
CSV HEADER;
select * from Recurring_payment
select service_provider_name from service_provider where administrator_Id = '6591288989'
select account_number,credit_card_status from credit_card where expiry_date='2022-07-15'
select count(distinct Administrator_name) from Administrator
select count(customer_ID) from banking.customer
select customer_name,Account_number from banking.customer where Auditor_ID=''
select count(distinct Administrator_name) from banking.Administrator
select max(balance) from banking.Account_Details
count
1 25
```

select count(customer_ID) from customer

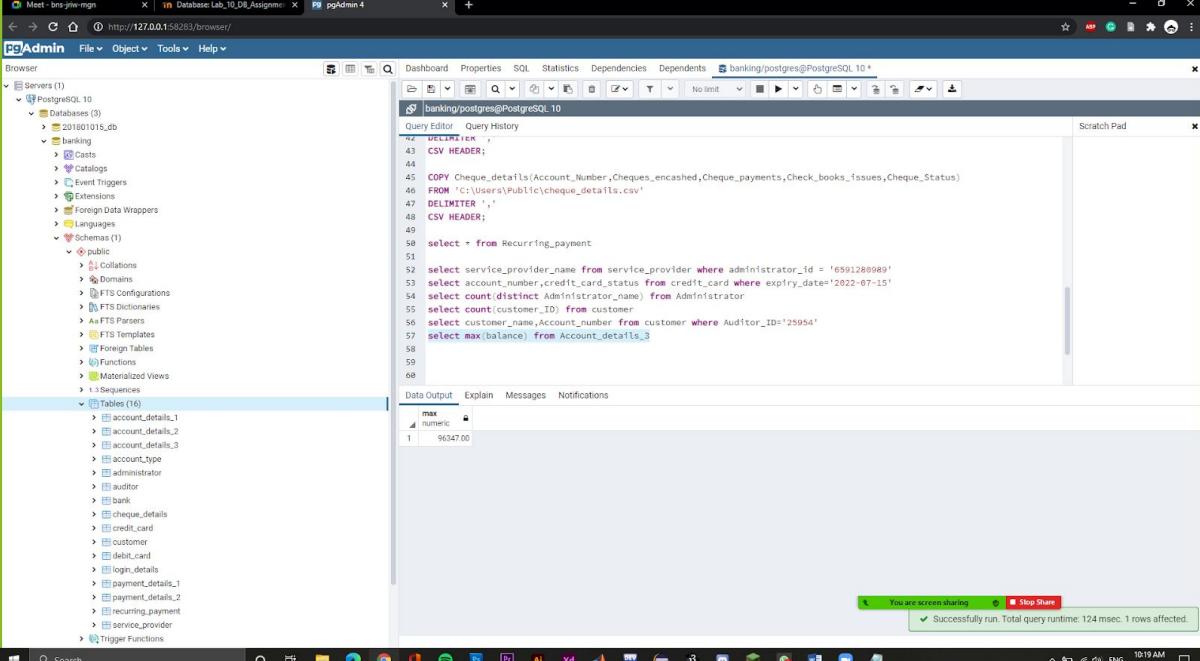
-> Here, we are calculating the total number of customers.

The screenshot shows the pgAdmin 4 interface. On the left, the 'Servers' tree view shows a connection to 'PostgreSQL 10' and the 'banking' database. The 'Tables' node under 'banking' has 16 entries. In the center, the 'Query Editor' window displays a SQL query to count customer IDs. The 'Data Output' tab shows the result: count = 100. A status bar at the bottom right indicates 'Successfully run. Total query runtime: 105 msec. 1 rows affected.'

```
DELIMITER ','
CSV HEADER;
COPY Cheque_Details(Account_Number,Cheques_Encashed,Cheque_Payments,Check_Books_Issues,Cheque_Status)
FROM 'C:\Users\Public\cheque_details.csv'
DELIMITER ','
CSV HEADER;
select * from Recurring_payment
select service_provider_name from service_provider where administrator_Id = '6591288989'
select account_number,credit_card_status from credit_card where expiry_date='2022-07-15'
select count(distinct Administrator_name) from Administrator
select count(customer_ID) from customer
select customer_name,Account_number from banking.customer where Auditor_ID=''
select count(distinct Administrator_name) from banking.Administrator
select max(balance) from banking.Account_Details
count
1 100
```

select max(balance) from Account_details_3

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



The screenshot shows the pgAdmin 4 interface. On the left, the 'Servers' tree view shows 'PostgreSQL 10' and '201601015_db'. Under 'Tables (16)', there is a list of tables including 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, and trigger_functions. The 'Query Editor' tab is active, displaying the following SQL code:

```
42 DELIMITER ;
43 CSV HEADER;
44
45 COPY Cheque_details(Account_Number,Cheques_Encashed,Cheque_Payments,Check_Books_Issues,Cheque_Status)
46 FROM 'C:\Users\Public\cheque_details.csv';
47 DELIMITER ',';
48 CSV HEADER;
49
50 select * from Recurring_payment
51
52 select service_provider_name from service_provider where administrator_Id = '6591288999'
53 select account_number,credit_card_status from credit_card where expiry_date='2022-07-15'
54 select count(distinct Administrator_name) from Administrator
55 select count(customer_ID) from customer
56 select customer_name,Account_number from customer where Auditor_ID='25954'
57 select max(balance) from Account_details_3
58
59
60
61
62
63
64
65
66
```

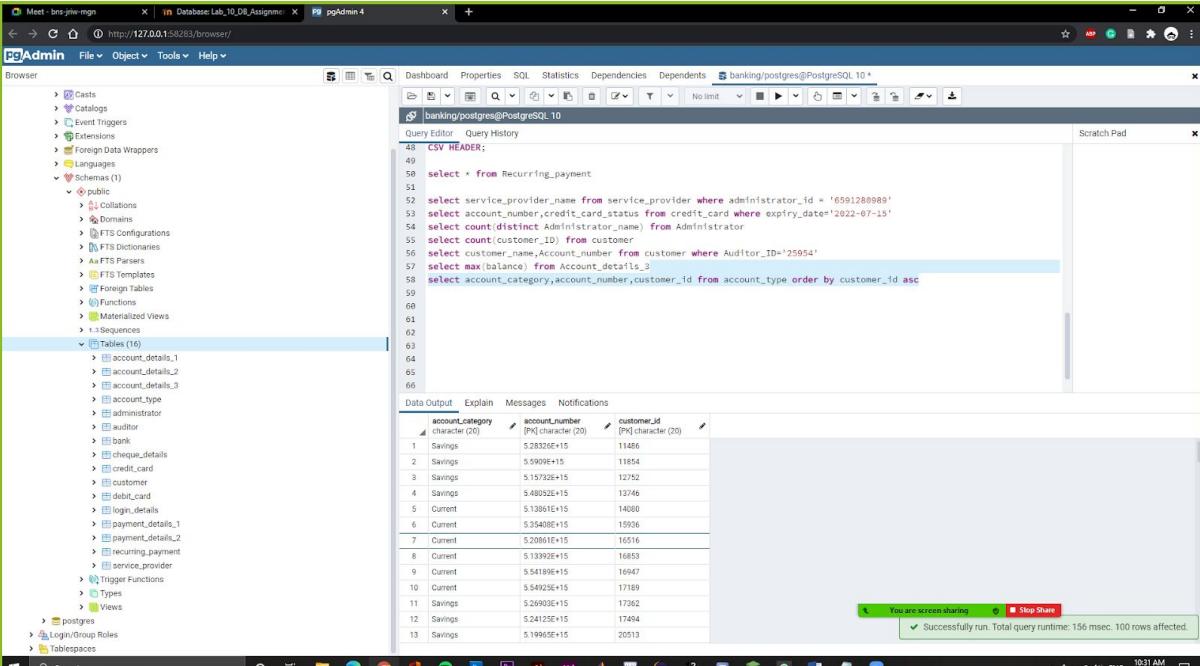
The 'Data Output' tab shows the result of the query:

	max
1	96347.00

A status bar at the bottom right indicates: You are screen sharing, Stop Share, Successfully run. Total query runtime: 124 msec. 1 rows affected. The system tray shows the date and time as 10:19 AM 28-11-2020.

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.



The screenshot shows the pgAdmin 4 interface. On the left, the 'Servers' tree view shows 'PostgreSQL 10' and '201601015_db'. Under 'Tables (16)', there is a list of tables including 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, and trigger_functions. The 'Query Editor' tab is active, displaying the following SQL code:

```
48 CSV HEADER;
49
50 select * from Recurring_payment
51
52 select service_provider_name from service_provider where administrator_Id = '6591288999'
53 select account_number,credit_card_status from credit_card where expiry_date='2022-07-15'
54 select count(distinct Administrator_name) from Administrator
55 select count(customer_ID) from customer
56 select customer_name,Account_number from customer where Auditor_ID='25954'
57 select max(balance) from Account_details_3
58 select account_category,account_number,customer_id from account_type order by customer_id asc
59
60
61
62
63
64
65
66
```

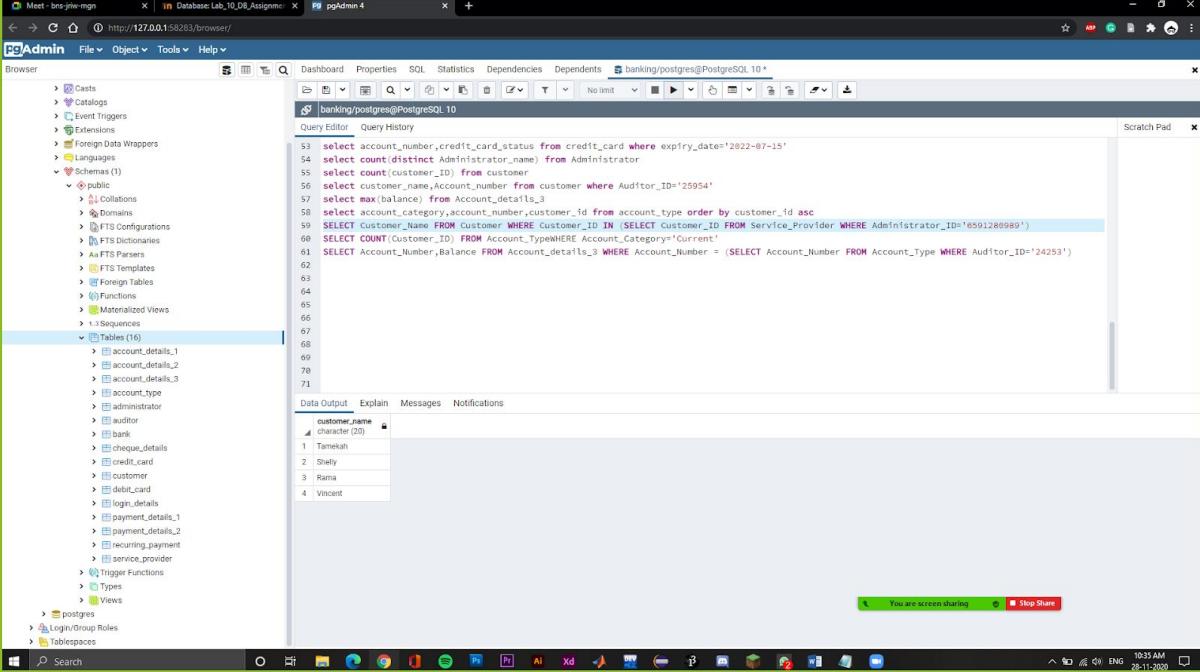
The 'Data Output' tab shows the result of the query:

account_category	account_number	customer_id
Savings	5.28326E+15	11486
Savings	5.5909E+15	11854
Savings	5.15732E+15	12752
Savings	5.48052E+15	13746
Current	5.13661E+15	14080
Current	5.35408E+15	15936
Current	5.20861E+15	16516
Current	5.13392E+15	16853
Current	5.54189E+15	16947
Current	5.54925E+15	17189
Savings	5.26903E+15	17362
Savings	5.24125E+15	17494
Savings	5.19965E+15	20513

A status bar at the bottom right indicates: You are screen sharing, Stop Share, Successfully run. Total query runtime: 156 msec. 100 rows affected. The system tray shows the date and time as 10:31 AM 28-11-2020.

`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



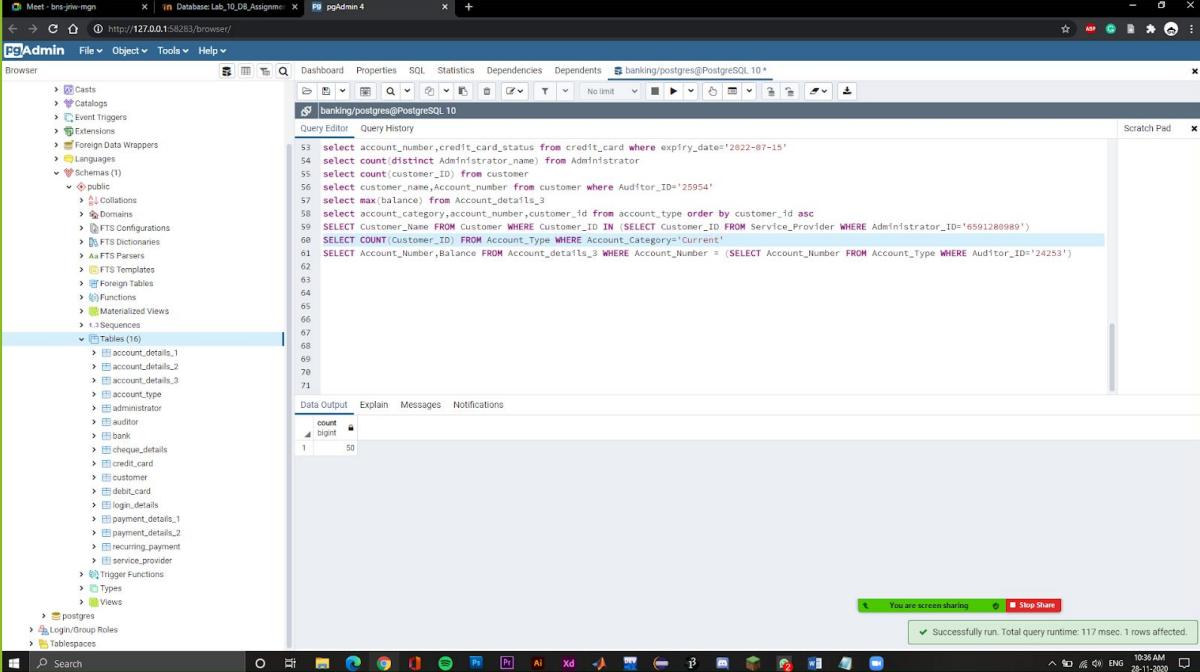
```

Meet - bns-jnw-mgn  Database Lab_10_DB_Assignment pgAdmin 4
http://127.0.0.1:58283/browser/
PgAdmin File Object Tools Help
Browser
Query Editor Query History
banking/postgres@PostgreSQL 10*
S3 select account_number,credit_card_status from credit_card where expiry_date='2022-07-15'
S4 select count(distinct Administrator_name) from Administrator
S5 select count(customer_ID) from customer
S6 select customer_name,Account_number from customer where Auditor_ID='25954'
S7 select max(balance) from Account_details_3
S8 select account_category,account_number,customer_id from account_type order by customer_id asc
S9 SELECT Customer_Name FROM Customer WHERE Customer_ID IN (SELECT Customer_ID FROM Service_Provider WHERE Administrator_ID='6591280989')
S10 SELECT COUNT(Customer_ID) FROM Account_Type WHERE Account_Category='Current'
S11 SELECT Account_Number,Balance FROM Account_details_3 WHERE Account_Number = (SELECT Account_Number FROM Account_Type WHERE Auditor_ID='24253')
S12
S13
S14
S15
S16
S17
S18
S19
S20
S21
S22
S23
S24
S25
S26
S27
S28
S29
S30
S31
S32
S33
S34
S35
S36
S37
S38
S39
S40
S41
S42
S43
S44
S45
S46
S47
S48
S49
S50
S51
S52
S53
S54
S55
S56
S57
S58
S59
S60
S61
S62
S63
S64
S65
S66
S67
S68
S69
S70
S71
Data Output Explain Messages Notifications
customer_name
character (20)
1 Tameekah
2 Shelly
3 Rama
4 Vincent

```

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

Show number of customers who have account category as current.



```

Meet - bns-jnw-mgn  Database Lab_10_DB_Assignment pgAdmin 4
http://127.0.0.1:58283/browser/
PgAdmin File Object Tools Help
Browser
Query Editor Query History
banking/postgres@PostgreSQL 10*
S3 select account_number,credit_card_status from credit_card where expiry_date='2022-07-15'
S4 select count(distinct Administrator_name) from Administrator
S5 select count(customer_ID) from customer
S6 select customer_name,Account_number from customer where Auditor_ID='25954'
S7 select max(balance) from Account_details_3
S8 select account_category,account_number,customer_id from account_type order by customer_id asc
S9 SELECT Customer_Name FROM Customer WHERE Customer_ID IN (SELECT Customer_ID FROM Service_Provider WHERE Administrator_ID='6591280989')
S10 SELECT COUNT(Customer_ID) FROM Account_Type WHERE Account_Category='Current'
S11 SELECT Account_Number,Balance FROM Account_details_3 WHERE Account_Number = (SELECT Account_Number FROM Account_Type WHERE Auditor_ID='24253')
S12
S13
S14
S15
S16
S17
S18
S19
S20
S21
S22
S23
S24
S25
S26
S27
S28
S29
S30
S31
S32
S33
S34
S35
S36
S37
S38
S39
S40
S41
S42
S43
S44
S45
S46
S47
S48
S49
S50
S51
S52
S53
S54
S55
S56
S57
S58
S59
S60
S61
S62
S63
S64
S65
S66
S67
S68
S69
S70
S71
Data Output Explain Messages Notifications
count
bigint
1 50

```

```
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_number,credit_card_status FROM credit_card WHERE expiry_date='2022-07-15'
SELECT count(DISTINCT Administrator_name) FROM Administrator
SELECT count(customer_id) FROM customer
SELECT customer_name,Account_number FROM customer WHERE Auditor_ID='25954'
SELECT max(balance) FROM Account_Details_3
SELECT account_category,account_number,customer_id FROM account_type ORDER BY customer_id ASC
SELECT Customer_Name FROM customer WHERE Customer_ID IN (SELECT Customer_ID FROM Service_Provider WHERE Administrator_ID='6591280989')
SELECT COUNT(Customer_ID) FROM Account_Type WHERE Account_Categories='Current'
SELECT Account_Number,Balance FROM Account_Details_3 WHERE Account_Number IN (SELECT Account_Number FROM Account_Type WHERE Auditor_ID='24253')

```

account_number	balance
5.23951E+15	8871.00
5.44359E+15	44200.00
5.50073E+15	63260.00
5.42506E+15	35404.00
5.53198E+15	33502.00
5.55406E+15	21337.00
5.33645E+15	24579.00
5.13361E+15	96347.00
5.2983E+15	15968.00
5.13392E+15	37761.00

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 account_number,credit_card_status FROM credit_card WHERE expiry_date='2022-07-15'
SELECT count(DISTINCT Administrator_name) FROM Administrator
SELECT count(customer_id) FROM customer
SELECT customer_name,Account_number FROM customer WHERE Auditor_ID='25954'
SELECT max(balance) FROM Account_Details_3
SELECT account_category,account_number,customer_id FROM account_type ORDER BY customer_id ASC
SELECT Customer_Name FROM customer WHERE Customer_ID IN (SELECT Customer_ID FROM Service_Provider WHERE Administrator_ID='6591280989')
SELECT COUNT(Customer_ID) FROM Account_Type WHERE Account_Categories='Current'
SELECT Account_Number,Balance FROM Account_Details_3 WHERE Account_Number IN (SELECT Account_Number FROM Account_Type WHERE Auditor_ID='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

```

account_number	credit_card_number	debit_card_number
5.42593E+15	842045	519260
5.57274E+15	910396	931455
5.12088E+15	547558	341100
5.43839E+15	866231	634495
5.53794E+15	382283	22635
5.19309E+15	475026	731921
5.29511E+15	765554	41190
5.29656E+15	953075	649026
5.39921E+15	834921	15321
5.59094E+15	236319	642075
5.58388E+15	488428	374068
5.51593E+15	682428	992780
5.40256E+15	911941	839945

SELECT * FROM Customer TABLESAMPLE BERNOULLI(10);

```

49
50   select * from Recurring_payment
51
52   select service_provider_name from service_provider where administrator_id = '6591288989'
53   select account_number,credit_card_status from credit_card where expiry_date='2022-07-15'
54   select count(distinct Administrator_name) from Administrator
55   select count(customer_ID) from customer
56   select count(service_provider_name,account_number) from customer where Auditor_ID='25954'
57   select count(account_number,account_detail_1.Balance) from account_detail_1 join account_detail_3
58   select account_category,account_number,customer_id from account_type order by customer_id asc
59   SELECT Customer_Name FROM Customer WHERE Customer_ID IN (SELECT Customer_ID FROM Service_Provider WHERE Administrator_ID='6591288989')
60   SELECT COUNT(Customer_ID) FROM Account_Type WHERE Account_Category='Current'
61   SELECT Account_Number,Balance FROM Account_detail_3 WHERE Account_Number IN (SELECT Account_Number FROM Account_Type WHERE Auditor_ID='24253')
62   select account_detail_1.account_number,account_detail_1.Credit_Card_number,account_detail_2.Debit_Card_number from account_detail_1 join account_detail_2
63   SELECT * FROM Customer TABLESAMPLE BERNOULLI(10);
64
65
66
67

```

customer_id	customer_name	account_number	pass_word	auditor_id
1 94914	Wyatt	519405E+15	KVY00PR83N	52934
2 6569	Riley	559505E+15	LWMBAXK4ED	10976
3 8886	Adara	547807E+15	GGV12TZQ9HS	28954
4 76225	Zahir	55477E+15	QF17P963TD	41445
5 86002	Shane	54045E+15	PQC22LYJUM	98971
6 74830	Brooke	51206E+15	UYZ6V6WAVO	41445
7 60024	Jada	537859E+15	A0H55KOPAF	28954
8 6843	Darryl	54807E+15	SIB4UJLTT	51254
9 3612	Emma	542734E+15	PTR9ZSBK0B	25954
10 47267	Forrest	555668E+15	I2TIAUDG1GD	81224
11 40594	Willow	549754E+15	YXCS2UJC2PE	24253
12 17189	Mira	55559E+15	GZ4ANVABGS	98971
13 9264	Aristote	529791E+15	FPG65R0B3MA	54208

```

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

```

32   select service_provider_name from service_provider where administrator_id = '6591288989'
33   select account_number,credit_card_status from credit_card where expiry_date='2022-07-15'
34   select count(distinct Administrator_name) from Administrator
35   select count(customer_ID) from customer
36   select count(service_provider_name,account_number) from customer where Auditor_ID='25954'
37   select count(account_number,account_detail_1.Balance) from account_detail_1 join account_detail_3
38   select account_category,account_number,customer_id from account_type order by customer_id asc
39   SELECT Customer_Name FROM Customer WHERE Customer_ID IN (SELECT Customer_ID FROM Service_Provider WHERE Administrator_ID='6591288989')
40   SELECT COUNT(Customer_ID) FROM Account_Type WHERE Account_Category='Current'
41   SELECT Account_Number,Balance FROM Account_detail_3 WHERE Account_Number IN (SELECT Account_Number FROM Account_Type WHERE Auditor_ID='24253')
42   select account_detail_1.account_number,account_detail_1.Credit_Card_number,account_detail_3.Balance from account_detail_1 join account_detail_3 or
43   SELECT * FROM Customer TABLESAMPLE BERNOULLI(10);
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67

```

account_number	credit_card_number	balance
1 54593E+15	842045	74469.00
2 55724E+15	910396	83778.00
3 512088E+15	547558	1676.00
4 54389E+15	866231	46821.00
5 52794E+15	382283	82911.00
6 519309E+15	475026	35977.00
7 52951E+15	765554	8871.00
8 529650E+15	953075	66356.00
9 539921E+15	834921	36205.00
10 55909E+15	256319	10200.00
11 558388E+15	488428	1361.00
12 551505E+15	662428	6068.00
13 540256E+15	911941	43958.00

`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.

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

- Browser:** Meet - bns-yhw-mgn
- Database:** Database Lab_10_DB_Assigned
- Port:** 127.0.0.1:58283/browser/
- Tool:** pgAdmin 4
- Query Editor:** banking/postgres@PostgreSQL 10*
- Query:** A complex multi-table query involving account_details_1, account_details_2, account_details_3, and account_type tables to filter accounts with a balance between 74469 and 818183.
- Data Output:** Shows 13 rows of account numbers ranging from 5.41599E+15 to 5.10223E+15.
- Messages:** Successfully run. Total query runtime: 149 msec. 24 rows affected.
- Notifications:** None
- System Bar:** Shows the date and time as 10:48 AM 28-11-2020.

`Select Administrator_Name from administrator where Administrator_Name like 'A%'`

-> Showing the administrator name starting from A.

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

- Browser:** Meet - bns-yhw-mgn
- Database:** Database Lab_10_DB_Assigned
- Port:** 127.0.0.1:58283/browser/
- Tool:** pgAdmin 4
- Query Editor:** banking/postgres@PostgreSQL 10*
- Query:** A complex multi-table query involving account_details_1, account_details_2, account_details_3, and account_type tables to filter administrators with names starting with 'A'.
- Data Output:** Shows 3 rows of administrator names: Addison, Adam, and Aphrodite.
- Messages:** Successfully run. Total query runtime: 149 msec. 24 rows affected.
- Notifications:** None
- System Bar:** Shows the date and time as 10:50 AM 28-11-2020.

```
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.

Met - bns-jrw-mgn | Database: Lab_10_DB_Assignment | pgAdmin 4 | postgresql - quick random row | +

http://127.0.0.1:58283/browser/

File Object Tools Help

Browser

Casts Catalogs Event Triggers Extensions Functions Types Triggers 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

+ postges

Login Group Roles Tablespace

Dashboard Properties SQL Statistics Dependencies Dependents banking/postgres@PostgreSQL 10*

Query Editor Query History

customer_id character (20) customer_name character (20)
1 11486 Cameran

Data Output Explain Messages Notifications

```
37  
38 COPY Recurring_payment(Account_Number,Beneficiary_account,Recurrence_Type,Recurring_StartDate,Recurring_EndDate,Recurring_Payment_Amount,Recurring_Payment_Frequency)  
39 FROM 'C:\Users\Public\recurring_payment.csv'  
40 DELIMITER ','  
41 CSV HEADER;  
42  
43 COPY Cheque_details(Account_Number,Cheques_enveloped,Cheque_purpose)  
44 FROM 'C:\Users\Public\cheque_details.csv'  
45 DELIMITER ','  
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 admiral  
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 customer_name,Account_number from customer where Auditor_ID='25954'  
56 select max(balance) from Account_details_3  
57 select count(account_category,account_number,customer_id) from account_type order by customer_id asc  
58 SELECT Customer_ID FROM Customer WHERE Customer_ID IN (SELECT Customer_ID FROM Service_Provider WHERE Administrator_ID='6591289899')  
59 SELECT COUNT(Customer_ID) FROM Account_Details_3 WHERE Account_Type='Current'  
60 SELECT COUNT(Account_Number) FROM Account_Details_2 WHERE Account_Number IN (SELECT Account_Number FROM Account_Type WHERE Auditor_ID='24253')  
61 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  
62 SELECT * FROM Customer TABLESAMPLE BERNOULLI(10);  
63 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  
64 SELECT account_number FROM account_details_3 WHERE balance between '74469' AND '818183'  
65 Select Administrator_Name from administrator where Administrator_Name like '%AS'  
66 select customer_id,customer_name from customer where customer_id like '1%' and auditor_ID='25954'  
67  
68  
69  
70  
71  
72  
73  
74  
75
```

You are screen sharing Stop Share

```
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

The screenshot shows the pgAdmin 4 interface connected to a PostgreSQL 10 database named 'banking'. The left sidebar lists various database objects such as Schemas, Tables, and Views. The main window contains a Query Editor with a history of commands, a Data Output table showing 10 rows of customer data, and an Explain plan. A status bar at the bottom indicates that the user is screen sharing and that a query was successfully run.

```
36 COPY RECURRING_PAYMENT(Account_Number,Beneficiary_Account,Recurrence_Period,Amount,Currency,Last_Paid,Next_Paid,Status) FROM 'C:\Users\Public\recurring_payment.csv' DELIMITER ',' CSV HEADER; 37 38 COPY Cheque_Details(Account_Number,Cheques_Encahsed,Cheque_Payee,Balance) FROM 'C:\Users\Public\cheque_details.csv' DELIMITER ',' CSV HEADER; 39 40 alter table Customer drop column account_number 41 select * from customer 42 43 select service_provider_name from service_provider where admin_id=1 44 select account_number,credit_card_status from credit_card where 45 select count(distinct administrator_name) from Administrator 46 select count(customer_id) from customer 47 select max(balance) from account_details_3 48 select account_category,account_number,customer_id from account_type order by customer_id asc 49 50 SELECT Customer_Name FROM Customer WHERE Customer_ID IN (SELECT Customer_ID FROM Service_Provider WHERE Administor_ID='6591289989') 51 52 SELECT COUNT(Customer_ID) FROM Account_Type WHERE Account_Catogory='current' 53 54 SELECT COUNT(*) FROM (SELECT * FROM account_type WHERE account_number IN (SELECT Account_Number FROM Account_Type WHERE Auditor_ID='242531') 55 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 56 select * FROM CUSTOMER TABLESAMPLE BERNoulli(10); 57 select account_details_1.account_number,account_details_1.credit_card_number,account_details_2.account_number,account_details_2.debit_card_number from account_details_1 join account_details_2 58 59 select account_number FROM account_details_3 WHERE balance between '74469' AND '818183' 60 61 Select Administrator_Name from administrator where Administor_Name='AS' 62 63 select customer_id,customer_name from customer where customer_id like '1%' and auditor_ID='25954' 64 65 select customer_id FROM account_type EXCEPT SELECT account_number FROM account_details_3 WHERE balance between '1288' and '9088' 66 67 68 69 70 71 72 73 74
```

You are screen sharing
Successfully run. Total query runtime: 181 msec. 100 rows affected.

```

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

account_number	debit_card_number	balance	recurring_status
5.57724E+15	931455	83776.00	true
5.12688E+15	341100	1676.00	true
5.39921E+15	15321	36205.00	true
5.58886E+15	374068	13361.00	true
5.51656E+15	992780	6886.00	true
5.40256E+15	89945	43950.00	true
5.32634E+15	669457	33620.00	true
5.27297E+15	73829	33381.00	true
5.19242E+15	818183	31951.00	true
5.54189E+15	49269	24558.00	true
5.59783E+15	809031	83717.00	true
5.12286E+15	750334	14122.00	true
5.42259E+15	73875	21630.00	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.

account_number	debit_card_number	balance	recurring_status
5.57724E+15	931455	83776.00	true
5.12688E+15	341100	1676.00	true
5.43859E+15	15321	36205.00	true
5.29556E+15	374068	13361.00	true
5.39921E+15	992780	6886.00	true
5.51656E+15	89945	43950.00	true
5.45129E+15	669457	33620.00	true
5.35408E+15	73829	33381.00	true
5.33129E+15	818183	31951.00	true
5.44035E+15	49269	24558.00	true
5.45987E+15	809031	83717.00	true
5.17107E+15	750334	14122.00	true
5.24666E+15	73875	21630.00	true

```
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

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

- Left Sidebar (Browser):** Shows the database structure:
 - Schemas: public, bank, banking
 - Tables: 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.
 - PostgreSQL objects: Login Group Roles, Tablespaces.
- Central Area:**
 - Query Editor:** Contains the following SQL script:

```
44 FROM 'C:/Users/Public/cheque_details.csv'
45 DELIMITER ','
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 admin
52 select account_number,credit_card_status from credit_card when
53 select count(distinct Administrator_name) from Administrator
54 select 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),FRONT_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.Cred
61 SELECT * FROM Customer TABLESAMPLE BERNoulli(10);
62 select account_details_1.account_number,account_details_1.Cred
63 SELECT account_number FROM account_Details_3 where Balance between '74469' AND '818183'
64 Select Administrator_Name from administrator where Administrator_Name like '%AS'
65 select customer_id,customer_name from customer where customer_id like '1%' and auditor_ID='25954'
66 SELECT customer_id FROM account_type EXCEPT SELECT account_number FROM account_Details_3 WHERE balance between '1209' and '90808'
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('100000' 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
74
75
76
77
78
79
80
81
82
```
 - Data Output:** Shows the results of the last three queries:

count	auditor_id	character (20)
1	10	25954
2	10	52934
3	10	81224
 - Explain:** Shows the execution plan for the last query.
 - Messages:** Shows no messages.
 - Notifications:** Shows no notifications.
- Right Sidebar (Scratch Pad):** Shows a message: "You are screen sharing".

```
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.

customer_id	account_number	administrator_id	bank_name
1 31554	543859E+15	3156620572	Griegotts
2 76420	545129E+15	4383613172	Griegotts
3 91584	526682E+15	1957701810	Griegotts
4 21407	528728E+15	6501774679	Griegotts
5 72988	534633E+15	6793473624	Griegotts
6 47267	522066E+15	3156620572	Griegotts
7 58014	526684E+15	4383613172	Griegotts
8 57351	531603E+15	1957701810	Griegotts
9 4741	547116E+15	6501774679	Griegotts
10 75479	523525E+15	6793473624	Griegotts

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

Update customer name as lucy whose customer id is 51516

customer_id	customer_name	pass_word	administrator_id
88 91894	Lhrs	AHV90fSHQJ	28954
89 79310	Oliver	A4K06APV0J	51254
90 37844	Norman	TSJ77FE50U	41445
91 3690	Fredericka	KW516H1NRRN	25954
92 68750	Sophia	MTR88AB64QR	98971
93 75479	Heather	UfZ2DZNS1M	81224
94 85283	Lee	LNV99MnQZZH	52934
95 47198	Emmanuel	KAY530DAJL	54208
96 16853	Eden	UZ025lV64LN	24253
97 72864	Chancellor	CRY94JYK3Z	10976
98 66277	Xanthus	RYQ24MKG70I	28954
99 74231	Hayes	QQD2B1HNO5P	51254
100 51516	Lucy	LIG3AWG8SH	41445

```

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
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
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=acco
78
79
80
81
82
83
84
85
86

```

The results table shows:

account_category	max
character (29)	numeric
Savings	90620.00
Current	96347.00

Message bar: You are screen sharing. Successfully run. Total query runtime: 111ms.

```

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 identical to the one above:

```

68 select account_number from recurring_payment where account_number like '5%' and recurring_amount > cast('40000' as integer) and recurring_amount < cast
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
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=acco
78 set statement_timeout=1000;
79 select pg_sleep(2000);

```

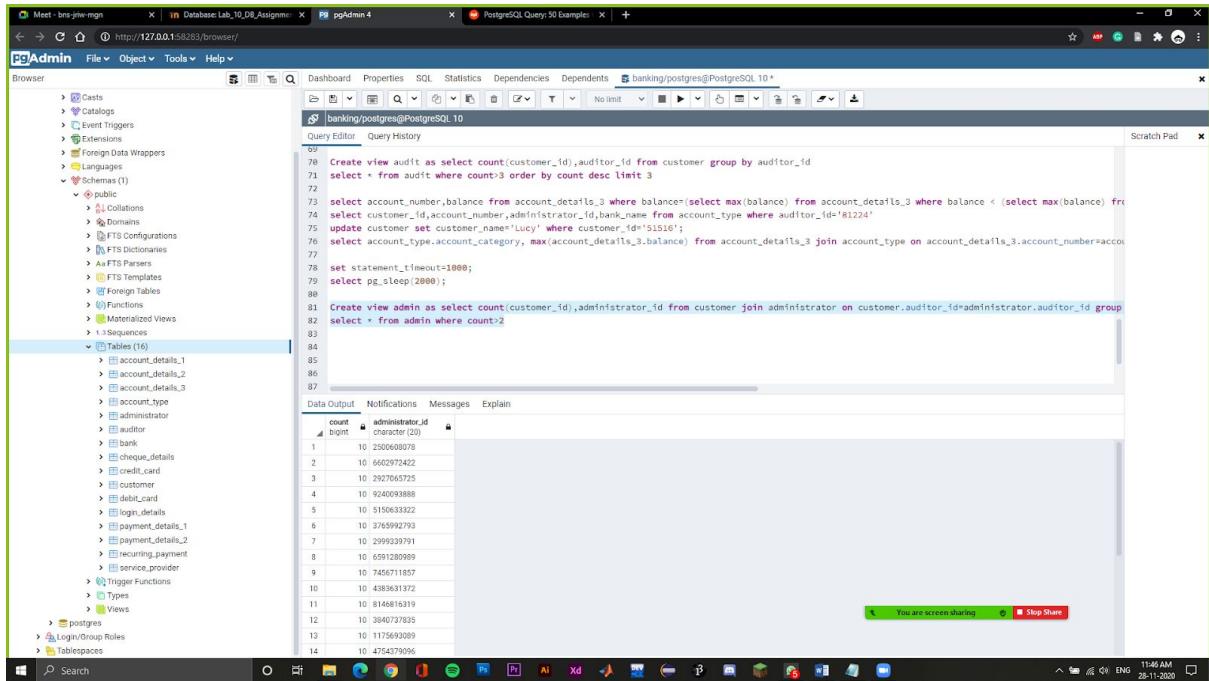
The results table shows:

account_category	max
character (29)	numeric
Savings	90620.00
Current	96347.00

Message bar: You are screen sharing. 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 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.



```

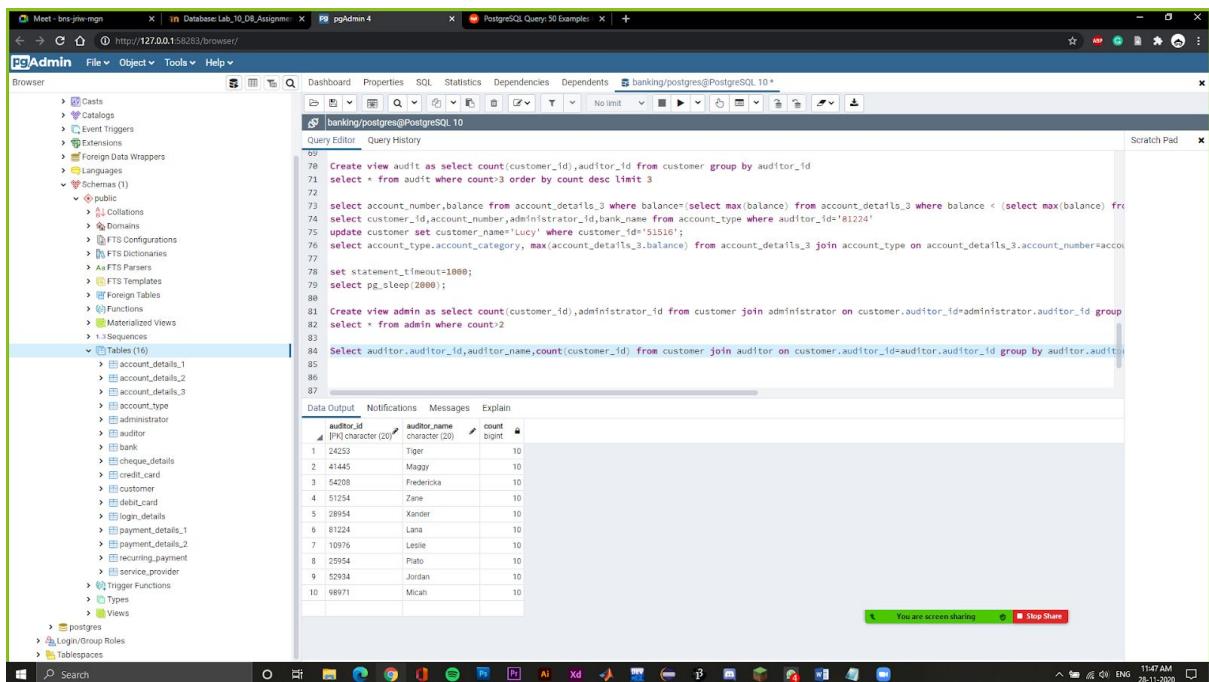
Meet - bns-jhv-mgn          Database Lab_10_DB_Assignment          pgAdmin 4          PostgreSQL Query 50 Examples: | +
http://127.0.0.1:58283/browser/          banking/postgres@PostgreSQL 10          Scratch Pad: x
PgAdmin File Object Tools Help
Browser
  Casts
  Catalogs
  Event Triggers
  Extensions
  Foreign Data Wrappers
  Languages
  Schemas (1)
    public
      Collations
      Domains
      FTS Configurations
      FTS Dictionaries
      FTS Parsers
      FTS Templates
      Foreign Tables
      Functions
      Materialized Views
    Sequences (1)
    Tables (16)
      account_details_1
      account_details_2
      account_details_3
      account_type
      administrator
      auditor
      bank
      cheque_details
      credit_card
      customer
      debt_card
      login_details
      payment_details_1
      payment_details_2
      recurring_payment
      service_provider
      Trigger Functions
      Types
      Views
  postgres
  Login/Group Roles
  Tablespaces
Search
Query Editor: Query History
  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
  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_details_3.account_number=acco
  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
  82 select * from admin where count>2
  83
  84
  85
  86
  87
  
```

The screenshot shows the pgAdmin 4 interface with the 'Query Editor' tab active. The query window contains the SQL code for creating a view named 'admin'. The results pane below shows 14 rows of data from the 'admin' view, which lists administrator IDs and their counts. The operating system taskbar at the bottom indicates it's 11:46 AM on 28-11-2020.

count	administrator_id
1	10
2	10
3	10
4	10
5	10
6	10
7	10
8	10
9	10
10	10
11	10
12	10
13	10
14	10

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.



```

Meet - bns-jhv-mgn          Database Lab_10_DB_Assignment          pgAdmin 4          PostgreSQL Query 50 Examples: | +
http://127.0.0.1:58283/browser/          banking/postgres@PostgreSQL 10          Scratch Pad: x
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 (1)
    Tables (16)
      account_details_1
      account_details_2
      account_details_3
      account_type
      administrator
      auditor
      bank
      cheque_details
      credit_card
      customer
      debt_card
      login_details
      payment_details_1
      payment_details_2
      recurring_payment
      service_provider
      Trigger Functions
      Types
      Views
  postgres
  Login/Group Roles
  Tablespaces
Search
Query Editor: Query History
  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
  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_details_3.account_number=acco
  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
  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.audit
  85
  86
  87
  
```

The screenshot shows the pgAdmin 4 interface with the 'Query Editor' tab active. The query window contains the SQL code for creating a view named 'admin'. The results pane below shows 10 rows of data from the 'admin' view, which lists auditor IDs, names, and counts. The operating system taskbar at the bottom indicates it's 11:47 AM on 28-11-2020.

auditor_id	auditor_name	count
1	Tiger	10
2	Maggie	10
3	Frederika	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)
```

The screenshot shows the pgAdmin 4 interface with a query editor containing the following SQL code:

```

75  update customer set customer_name='LUCY' where customer_id=10;
76  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;
77  set statement_timeout=1000;
78  select pg_sleep(2000);
79
80
81 Create view admin as select count(customer_id),administrator_id from customer join administrator on customer.administrator_id=administrator.administrator_id group by administrator.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.ser
87
88
89
90
91
92
93

```

The Data Output tab displays the results of the last SELECT statement:

service_provider_name	service_provider_id	administrator_id
character (20)	character (20)	character (20)
1 Timothy	5.54250E+15	8650365893
2 Cody	5.58024E+15	3156620572
3 Cooper	5.59393E+15	3765992793
4 Zane	5.59594E+15	3076160496
5 Brock	5.57911E+15	3687598691
6 Robert	5.59920E+15	3156620572
7 Cameron	5.57823E+15	0772427985
8 Mariko	5.45547E+15	7456711857
9 Alan	5.57871E+15	4754379969
10 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)
```

The screenshot shows the pgAdmin 4 interface with a query editor containing the following SQL code:

```

76  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;
77  set statement_timeout=1000;
78  select pg_sleep(2000);
79
80
81 Create view admin as select count(customer_id),administrator_id from customer join administrator on customer.administrator_id=administrator.administrator_id group by administrator.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.ser
87
88
89
90
91
92
93
94

```

The Data Output tab displays the results of the last SELECT statement:

customer_id	service_provider_id
[PK] character (20)	[PK] character (20)
88 17362	5.38134E+15
89 91894	5.53489E+15
90 79510	5.50859E+15
91 37844	5.43871E+15
92 3690	5.31827E+15
93 60750	5.39408E+15
94 75479	5.51140E+15
95 85283	5.26641E+15
96 47198	5.29521E+15
97 16853	5.1656E+15
98 72884	5.22928E+15
99 66277	5.32008E+15
100 74231	5.1087E+15

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

```

Database: Lab_10_DB_Assignment | pgAdmin 4
http://127.0.0.1:58283/browser/
PostgreSQL Query: 50 Examples | + | X
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 | 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 | Query History | banking/postgres@PostgreSQL 10 |
Scratch Pad | Data Output | Notifications | Messages | Explain |


```

76 set statement_timeout=10000;
77 select pg_sleep(2000);
78
79
80
81 Create view admin as select count(customer_id),administrator_id from customer join administrator on customer.administrator_id=administrator.administrator_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 select administrator_id from admin where count=(select max(count) from admin);
91
92
93
94
95
96

```


```

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

```

Database: Lab_10_DB_Assignment | pgAdmin 4
http://127.0.0.1:58283/browser/
PostgreSQL Query: 50 Examples | + | X
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 | 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 | Query History | banking/postgres@PostgreSQL 10 |
Scratch Pad | Data Output | Notifications | Messages | Explain |


```

76 set statement_timeout=10000;
77 select pg_sleep(2000);
78
79
80
81 Create view admin as select count(customer_id),administrator_id from customer join administrator on customer.administrator_id=administrator.administrator_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 select administrator_id from admin where count=(select max(count) from admin);
91
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 | administrator_id | customer_id |
|---------------------|-----------------------|---------------|------------------|-------------|
| 1 5.54259E+15       | Timothy               | 9.00          | 98071            | 92538       |
| 2 5.40079E+15       | Buckminster           | 2.00          | 25954            | 7459711857  |
| 3 5.21646E+15       | Owen                  | 2.00          | 81224            | 1967701810  |
| 4 5.57911E+15       | Brock                 | 8.00          | 24253            | 3687598091  |
| 5 5.21914E+15       | Aurelia               | 10.00         | 54208            | 94472       |
| 6 5.39001E+15       | Bruce                 | 6.00          | 41445            | 292705725   |
| 7 5.49026E+15       | Tiger                 | 3.00          | 10975            | 93653       |
| 8 5.36306E+15       | Edward                | 7.00          | 51254            | 9240053888  |
| 9 5.17771E+15       | Rigel                 | 1.00          | 52934            | 96109       |
| 10 5.35489E+15      | Ursula                | 3.00          | 28954            | 64923       |


```

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

The screenshot shows the pgAdmin 4 interface with the 'Query Editor' tab active. The query window contains the SQL code for creating a view 'detail'. The results pane below shows a table with 14 rows of data, mapping customers to their accounts and bank details.

```
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)
87
88 select customer_id, service_provider_id from service_provider where customer_id in (select max(customer_id) from service_provider group by service_provider)
89
90 select administrator_id from admin where count=(select max(count) from admin)
91
92 select * from service_provider where customer_id in (select max(customer_id) from service_provider group by auditor_id)
93
94 create view detail as select customer.customer_name, customer.customer_id, account_type.account_number, account_type.account_category, account_type.bank_name
95
96 select * from detail
```

customer_name	customer_id	account_number	account_category	bank_name
Lucy	51516	\$43993E+15	Current	Gringotts
Caleb	85820	\$57274E+15	Savings	Gringotts
Nigel	92538	\$12088E+15	Current	Gringotts
Abel	31554	\$43859E+15	Savings	Gringotts
Wyatt	94914	\$37984E+15	Current	Gringotts
Robin	86418	\$19396E+15	Savings	Gringotts
Rafael	9140	\$29511E+15	Current	Gringotts
Riley	6569	\$29650E+15	Savings	Gringotts
Charlotte	4118	\$39921E+15	Current	Gringotts
Charissa	11854	\$59094E+15	Savings	Gringotts
Hu	85671	\$58388E+15	Current	Gringotts
Shane	78933	\$51585E+15	Savings	Gringotts
Madonna	81129	\$40256E+15	Current	Gringotts
Kadeem	76420	\$45129E+15	Savings	Gringotts

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

The screenshot shows the pgAdmin 4 interface with the 'Query Editor' tab active. The query window contains a complex multi-table join query. The results pane below shows a table with 100 rows of data, combining information from four different account types (Credit Card, Debit Card, Balance, and Recurring Status).

```
87
88 select customer_id, service_provider_id from service_provider where customer_id in (select max(customer_id) from service_provider group by service_provider)
89
90 select administrator_id from admin where count=(select max(count) from admin)
91
92 select * from service_provider where customer_id in (select max(customer_id) from service_provider group by auditor_id)
93
94 create view detail as select customer.customer_name, customer.customer_id, account_type.account_number, account_type.account_category, account_type.bank_name
95
96 select * from detail
```

```
100
101 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
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
```

account_number	credit_card_number	debit_card_number	balance	recurring_status
5.2983E+15	122792	496767	1596.00	false
5.26903E+15	58321	648560	8504.00	false
5.10219E+15	474568	407794	1900.00	false
5.358E+15	704374	764484	2576.00	true
5.35899E+15	558743	977210	83741.00	true
5.2213E+15	698592	27630	90620.00	false
5.18903E+15	871731	114097	8263.00	false
5.23523E+15	520515	544134	36435.00	false
5.19227E+15	696446	111938	40958.00	true
5.43205E+15	606978	793337	50487.00	true
5.13392E+15	432251	521540	37751.00	false
5.40563E+15	689587	163870	61248.00	false
5.21621E+15	815133	7386	56323.00	true
5.57094E+15	134014	218096	65353.00	true

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

The screenshot shows the pgAdmin 4 interface connected to a PostgreSQL 10 database. The left sidebar displays the schema tree, including public, banking, and information schemas. The banking schema contains tables like account_details, customer, auditor, and bank. The Query Editor window shows a multi-table JOIN query. The Data Output grid below shows the results of the query, listing various customers with their names, IDs, and other account-related details.

customer_name	customer_id
Uwe	79710
Norman	37644
Fredericka	3690
Sophia	68750
Heather	75479
Lee	85283
Emmanuel	47198
Eden	16653
Chancellor	72664
Xanthus	66277
Hayes	74231
Lucy	51516

You are screen sharing Stop Share

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

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

- Database:** Lab_10_DB_Assignment
- Port:** 5432
- Host:** 127.0.0.1
- Browser:** banking/postgres@PostgreSQL-10
- Query Editor:** Contains the following PostgreSQL code:

```
101
102
103 SELECT customer.customer_name,customer.customer_id FROM customer LEFT JOIN auditor ON customer.auditor_id = auditor.auditor_id;
104
105 SELECT administrator.administrator_name,administrator.administrator_id FROM administrator RIGHT JOIN auditor ON administrator.auditor_id = auditor.auditor_id;
106
107 create or replace function "banking"."details"(s varying character(250))
108 returns table(b character varying(250))
109 language 'plpgsql'
110 as $$
111 declare
112     i RECORD;
113 begin
114     for i in (
115         select account_details_2.account_number from
116             account_details_2 join account_details_3 on account_details_2.account_number=s
117             and Recurring_Status = True order by account_details_3.Balance desc limit 5)
118     loop b:=(i.account_number);
119     return next;
120 end loop;
121 end;
122 $$;
123 select "banking"."details"()
124
125
```
- Data Output:** Shows the results of the function call, listing administrators and their IDs:

administrator.name	administrator_id
Evelyn	6792473524
Dean	3076160406
Nicole	2927657525
Celeste	7456711957
TaShya	6602972422
Cherokee	1967701810
Leia	4754799096
- Notifications:** You are screen sharing.
- Messages:** Stop Share.
- Explain:** Not visible in the screenshot.

```

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"();

```

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

- Left Panel (Browser):** Shows the database structure with Schemas (1) and Tables (16) expanded.
- Center Panel (Query Editor):** Displays the SQL code for the function creation.
- Bottom Panel (Data Output):** Shows the result of the function execution, which is a table with one column 'details' containing five rows of account numbers.

```

167 select account_number,Maximum_Limit from credit_card where Maximum_Limit<
168 (select max(Maximum_Limit)
169   from credit_card
170   where balance <
171     (select max((select Maximum_Limit) from credit_card where Maximum_Limit < (select max(Maximum_Limit) from credit_card))) from credit_card)
172
173 create or replace function "details"()
174 returns table(b character varying(250))
175 language 'plpgsql'
176 as $$
177 declare
178     i RECORD;
179 begin
180     for i in (
181         select account_details_2.account_number from
182         account_details_2 join account_details_3 on account_details_2.account_number=account_details_3.account_number
183         and Recurring_Status = True order by account_details_3.Balance desc limit 5)
184     loop b:=(i.account_number);
185     return next;
186     end loop;
187 end;
188 $$
189 select "details"();
190
191

```

details
character varying
1 5.47116E+15
2 5.30910E+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))
```

The screenshot shows the PgAdmin 4 interface with a database browser on the left and a query editor on the right. The query editor contains a PL/pgSQL block that performs a self-join on the 'credit_card' table to find accounts with maximum limits less than the second-highest maximum limit. The results are displayed in a table titled 'Data Output'.

account_number	maximum_limit
543993+15	963701

```

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"();

```

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

- Left Panel (Browser):** Shows the database schema structure, including Schemas (1) and Tables (10).
- Top Bar:** Includes tabs for Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, Schemas (1), public, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Testdates, Foreign Tables, Functions, Materialized Views, and Sequences.
- Query Editor:** Contains the SQL code for creating the 'details1' function.
- Data Output Tab:** Displays the results of the 'select "details1"();' command, showing 10 rows of data.
- Bottom Status Bar:** Shows the date and time (28-11-2020, 12:48 PM) and a message: 'You are access sharing'.

```

120     end loop;
121   end;
122   $$
123   select "details"();
124
125
126
127
128   select account_number,Maximum_Limit from credit_card where Maximum_Limit=
129   (select max(Maximum_Limit) from credit_card where Maximum_Limit < (select max(Maximum_Limit) from credit_card));
130
131   create or replace function "details1"()
132   returns table(b character varying(250),number character varying(250),number1 character varying(250),
133   number3 character varying(250))
134   language 'plpgsql'
135   as $$
136   declare
137       i RECORD;
138   begin
139       for i in (select customer_id,account_number,administrator_id,bank_name from account_type where auditor_id='81224')
140           loop
141               b:=(i.customer_id);number:=(i.account_number);number1:=(i.administrator_id);number3:=(i.b
ank_name);
142           return next;
143       end loop;
144   end;
145   $$

details1
record
3 (91584.526682E+15,1967701810,0,ingots)
4 (21407.529728E+15,6501774675,0,ingots)
5 (72988.534932E+15,6793473625,0,ingots)
6 (47287.522086E+15,3166620572,0,ingots)
7 (50914.526682E+15,4303637372,0,ingots)
8 (57351.531602E+15,1967701810,0,ingots)
9 (47415.47116E+15,6501774675,0,ingots)
10 (75479.523525E+15,6793473624,0,ingots)

```

```

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"();

```

The screenshot shows the PgAdmin 4 interface. The left sidebar displays the database schema with various objects like Schemas, Tables, and Functions. The main area is the Query Editor, which contains the SQL code for the 'details3' function. Below the editor is a Data Output pane showing the results of the function execution. The status bar at the bottom right indicates the session is being shared.

```

126     loop b:=(i.customer_id);number:=(i.account_number);number1:=(i.administrator_id);number3:=(i.bank_name);
127         return next;
128     end loop;
129 end;
130 $$
131 select "details1"();
132
133
134 create or replace function "details3"()
135 returns table(b character varying(250),number character varying(250))
136 language 'plpgsql'
137 as $$
138 declare
139     i RECORD;
140 begin
141     for i in (select account_number,balance from account_details_3 where
142 balance=(select max(balance) from account_details_3 where balance < (select
143 max(balance) from account_details_3)))
144         loop b:=(i.account_number);number:=(i.balance);
145         return next;
146     end loop;
147 end;
148 $$
149 select "details3"();
150
151
152
153
154
155
156
157
158
159
160
161

```

details3	record
1	(5.2215e+15,90520.00)

```

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"();

```

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

- Database:** Lab_10_DB_Assignment
- Connection:** pgAdmin 4 (banking/postgres@PostgreSQL 10)
- Query Editor:** Contains the SQL code for creating the function.
- Browser:** Shows the database schema tree on the left, including Schemas, Tables, and Views.
- Data Output:** Shows the results of the function execution, which is empty (0 rows).
- System Bar:** Includes icons for search, file operations, and system status (You are screen sharing, Stop Share).

```

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"();

```

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

- Query Editor:** Contains the SQL code for creating the function "details5". The code includes a cursor loop to select customer names from the Customer table based on IDs found in the Service_Provider table.
- Browser:** Shows the database schema structure, including Schemas, Tables (16), and various system objects like casts, catalogs, and functions.
- Results:** A table titled "details5" with a single column "Record" containing four rows of data: 1 (Shelly), 2 (Rama), 3 (Vincent), and 4 (Lucy).

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,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.