实　　验　　报　　告

学　　号	2 3 1 2 0 0 1 4 0 2 9	姓　　名	张心顺	专业班级	计算机 23 级
课程名称	数据库系统			学期	2025 年秋季学期
任课教师	刘艳艳	完成日期	251224	上机课时间	251224
实 验 名 称	openGauss 金融场景化实验				

一、实验要求（10%）

根据实验指导书,以金融行业为场景,设计数据库模型,并使用华为 openGauss 构建金融场景下的数据库。通过对数据库中的对象(表、约束、视图、索引等)创建,掌握基础 SQL 语法,并通过对表中数据的增删改查,模拟金融场景下的业务实现。

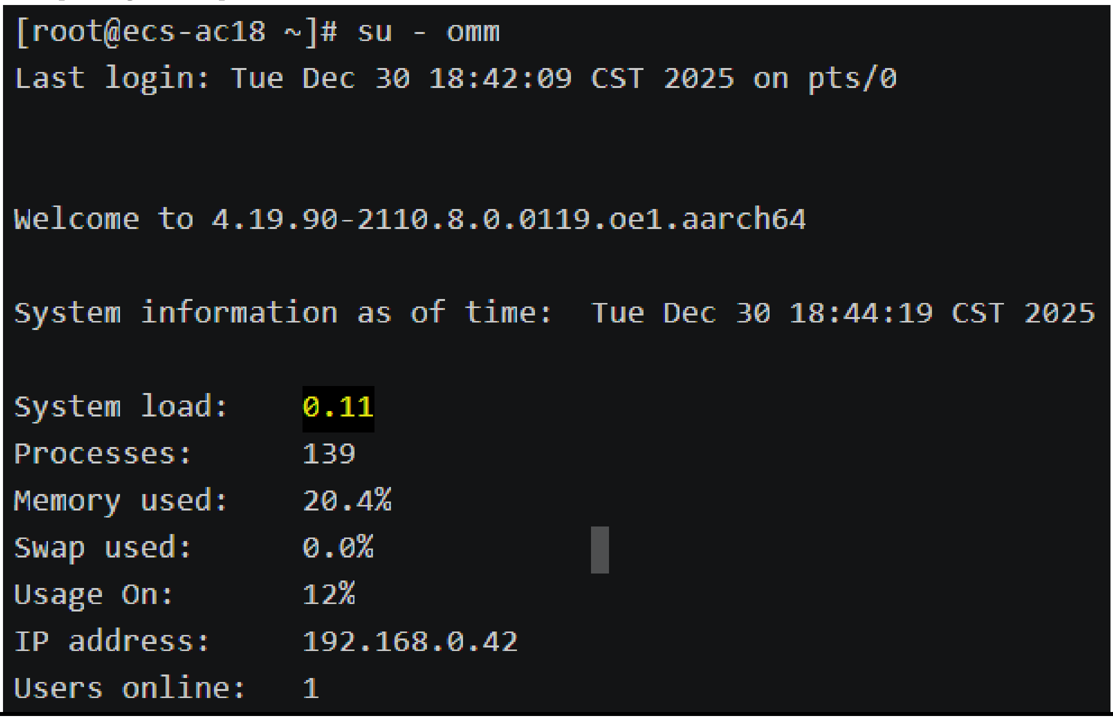
二、实验内容及步骤（80%）

以下是根据《openGauss 金融场景化综合应用实验》文档内容整理出的所有需要执行的代码及项目描述，按实验流程顺序排列，标注序号，便于复制执行：

一、实验环境准备与数据库创建

1. 切换到 omm 用户并登录 openGauss 数据库

```
su - omm
gs_om -t start
gsql -d postgres -p 26000 -r
```

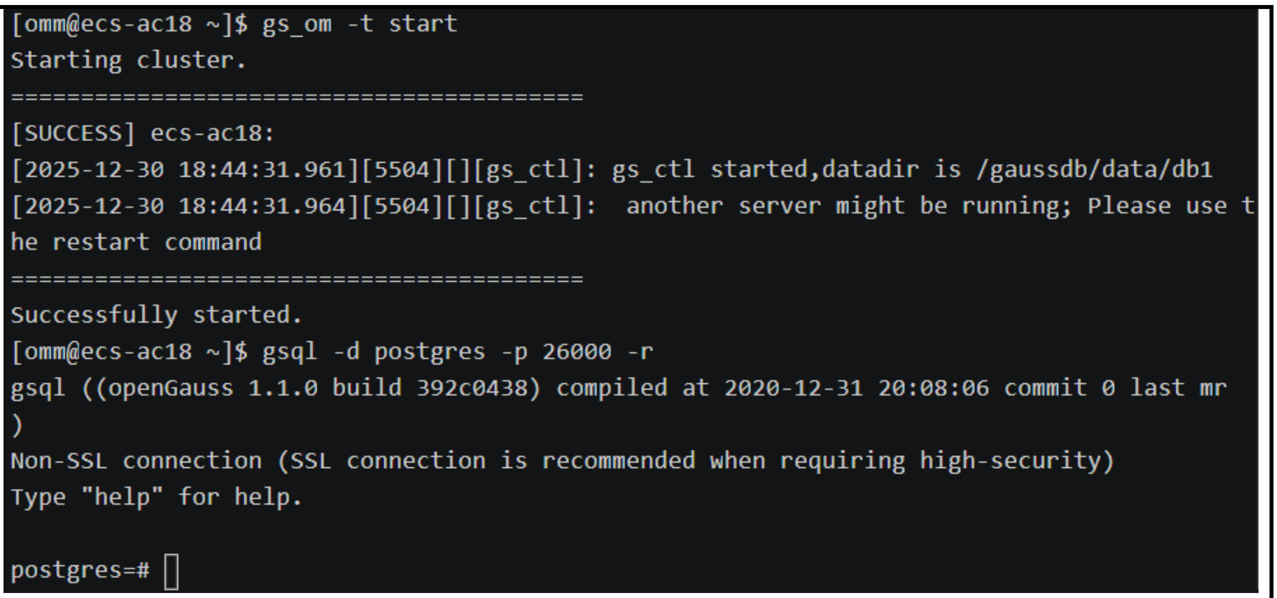


```
[root@ecs-ac18 ~]# su - omm
Last login: Tue Dec 30 18:42:09 CST 2025 on pts/0


Welcome to 4.19.90-2110.8.0.0119.oe1.aarch64

System information as of time:  Tue Dec 30 18:44:19 CST 2025

System load:     0.11
Processes:       139
Memory used:     20.4%
Swap used:       0.0%
Usage On:        12%
IP address:      192.168.0.42
Users online:    1
```

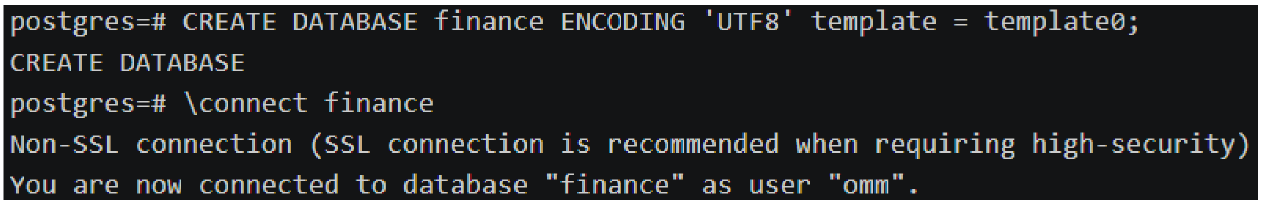


```
[omm@ecs-ac18 ~]$ gs_om -t start
Starting cluster.
=========================================
[SUCCESS] ecs-ac18:
[2025-12-30 18:44:31.961][5504][][gs_ctl]: gs_ctl started,datadir is /gaussdb/data/db1
[2025-12-30 18:44:31.964][5504][][gs_ctl]:  another server might be running; Please use t
he restart command
=========================================
Successfully started.
[omm@ecs-ac18 ~]$ gsql -d postgres -p 26000 -r
gsql ((openGauss 1.1.0 build 392c0438) compiled at 2020-12-31 20:08:06 commit 0 last mr
)
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

postgres=# 
```

2. 创建金融数据库 finance 并连接

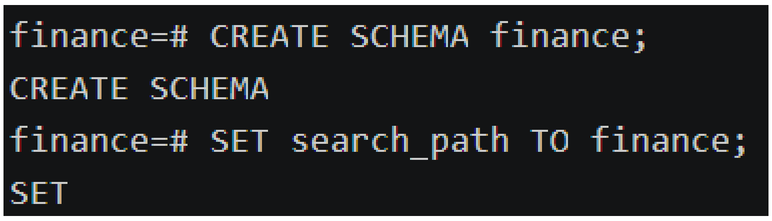
CREATE DATABASE finance ENCODING ’UTF8’ template = template0;
\connect finance



```
postgres=# CREATE DATABASE finance ENCODING 'UTF8' template = template0;
CREATE DATABASE
postgres=# \connect finance
Non-SSL connection (SSL connection is recommended when requiring high-security)
You are now connected to database "finance" as user "omm".
```

3. 创建并设置 finance schema

CREATE SCHEMA finance;
SET search_path TO finance;

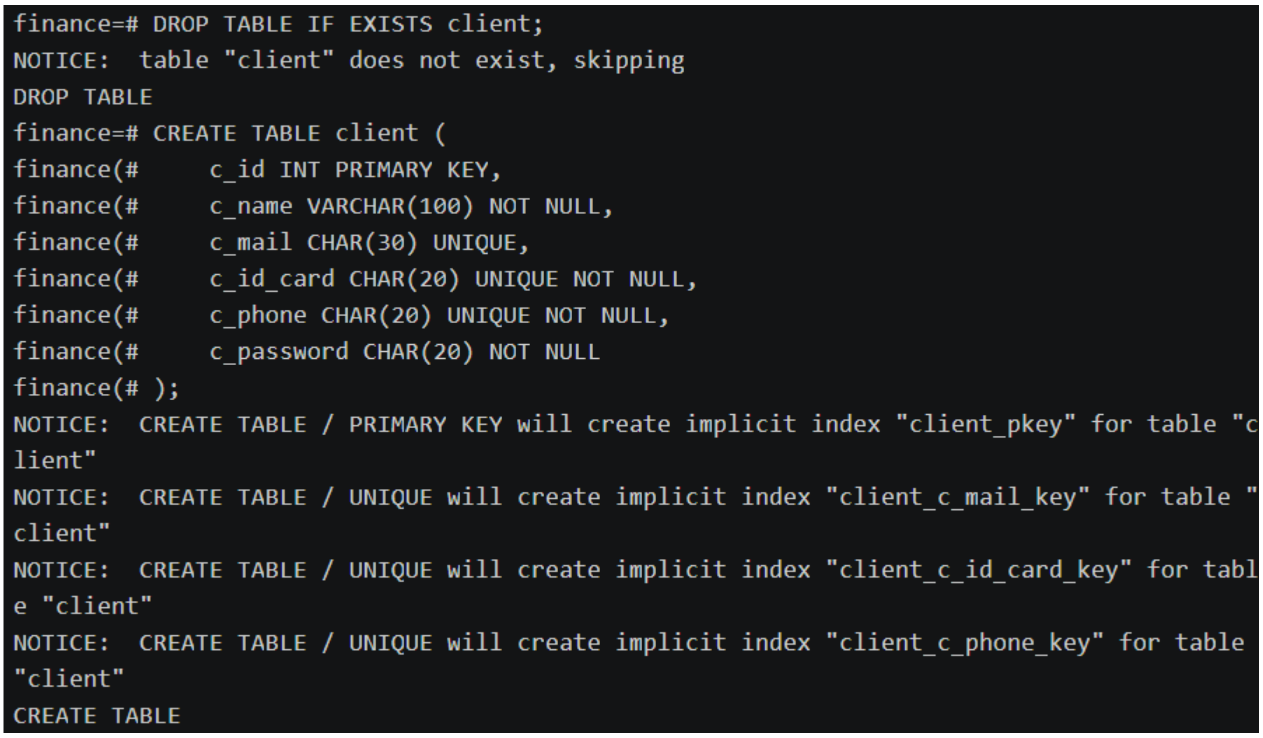


```
finance=# CREATE SCHEMA finance;
CREATE SCHEMA
finance=# SET search_path TO finance;
SET
```

二、创建数据表（金融数据模型）

4. 创建客户信息表 client

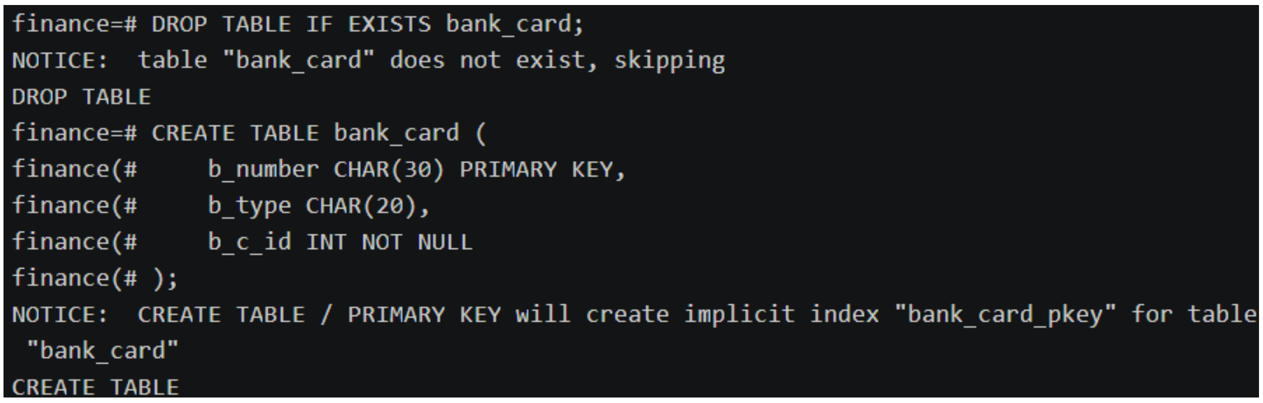
```
DROP TABLE IF EXISTS client;
CREATE TABLE client (
    c_id INT PRIMARY KEY,
    c_name VARCHAR(100) NOT NULL,
    c_mail CHAR(30) UNIQUE,
    c_id_card CHAR(20) UNIQUE NOT NULL,
    c_phone CHAR(20) UNIQUE NOT NULL,
    c_password CHAR(20) NOT NULL
);
```



```
finance=# DROP TABLE IF EXISTS client;
NOTICE:  table "client" does not exist, skipping
DROP TABLE
finance=# CREATE TABLE client (
finance(#     c_id INT PRIMARY KEY,
finance(#     c_name VARCHAR(100) NOT NULL,
finance(#     c_mail CHAR(30) UNIQUE,
finance(#     c_id_card CHAR(20) UNIQUE NOT NULL,
finance(#     c_phone CHAR(20) UNIQUE NOT NULL,
finance(#     c_password CHAR(20) NOT NULL
finance(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "client_pkey" for table "c
lient"
NOTICE:  CREATE TABLE / UNIQUE will create implicit index "client_c_mail_key" for table "
client"
NOTICE:  CREATE TABLE / UNIQUE will create implicit index "client_c_id_card_key" for tabl
e "client"
NOTICE:  CREATE TABLE / UNIQUE will create implicit index "client_c_phone_key" for table
"client"
CREATE TABLE
```

5. 创建银行卡信息表 bank_card

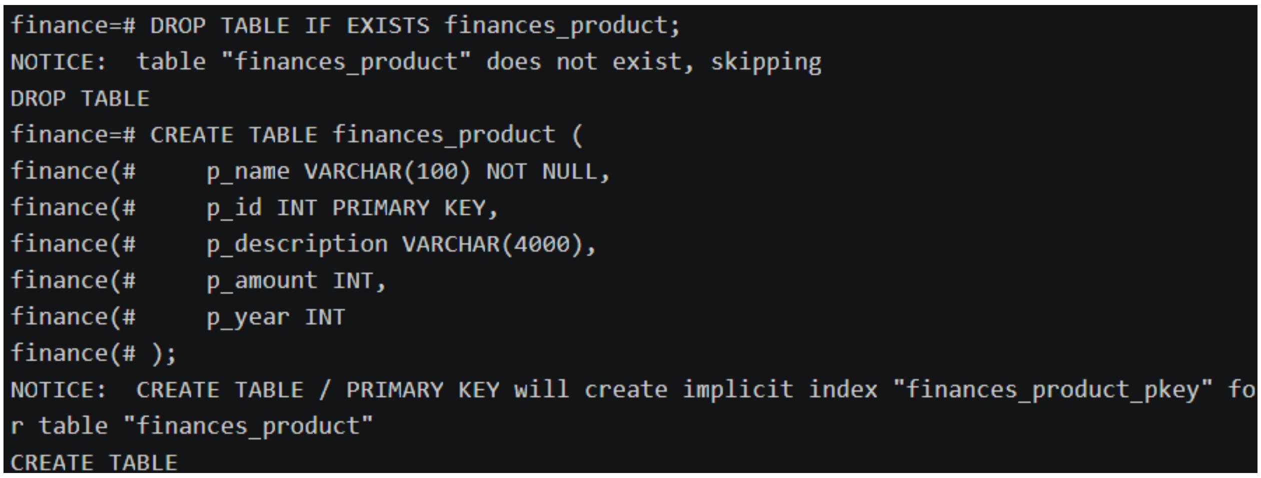
```
DROP TABLE IF EXISTS bank_card;
CREATE TABLE bank_card (
    b_number CHAR(30) PRIMARY KEY,
    b_type CHAR(20),
    b_c_id INT NOT NULL
);
```



```
finance=# DROP TABLE IF EXISTS bank_card;
NOTICE:  table "bank_card" does not exist, skipping
DROP TABLE
finance=# CREATE TABLE bank_card (
finance(#     b_number CHAR(30) PRIMARY KEY,
finance(#     b_type CHAR(20),
finance(#     b_c_id INT NOT NULL
finance(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "bank_card_pkey" for table
 "bank_card"
CREATE TABLE
```

6. 创建理财产品信息表 finances_product

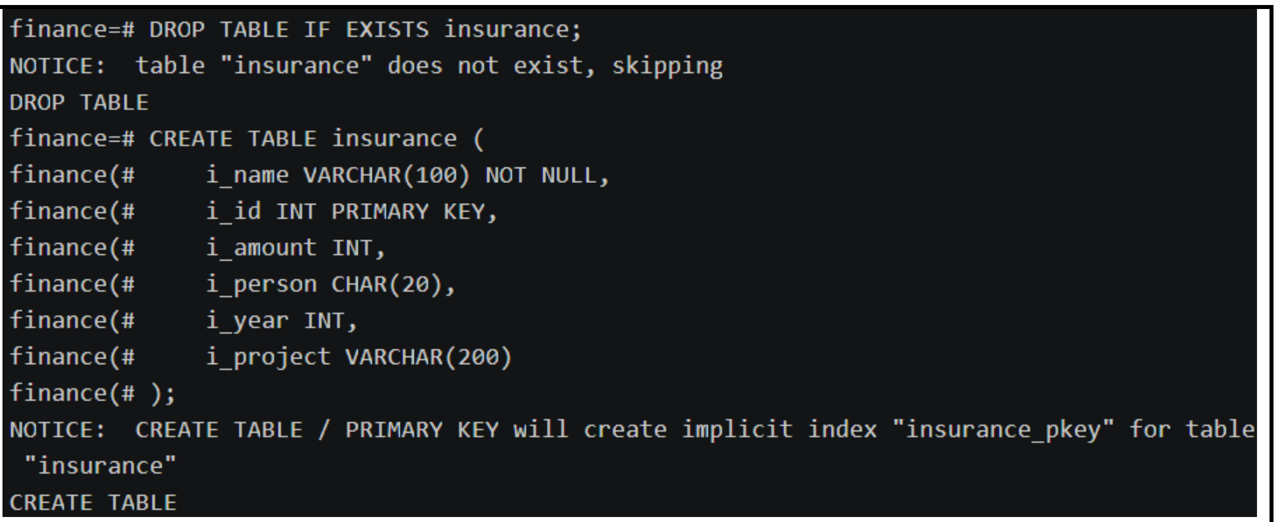
```
DROP TABLE IF EXISTS finances_product;
CREATE TABLE finances_product (
    p_name VARCHAR(100) NOT NULL,
    p_id INT PRIMARY KEY,
    p_description VARCHAR(4000),
    p_amount INT,
    p_year INT
);
```



```
finance=# DROP TABLE IF EXISTS finances_product;
NOTICE:  table "finances_product" does not exist, skipping
DROP TABLE
finance=# CREATE TABLE finances_product (
finance(#     p_name VARCHAR(100) NOT NULL,
finance(#     p_id INT PRIMARY KEY,
finance(#     p_description VARCHAR(4000),
finance(#     p_amount INT,
finance(#     p_year INT
finance(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "finances_product_pkey" fo
r table "finances_product"
CREATE TABLE
```

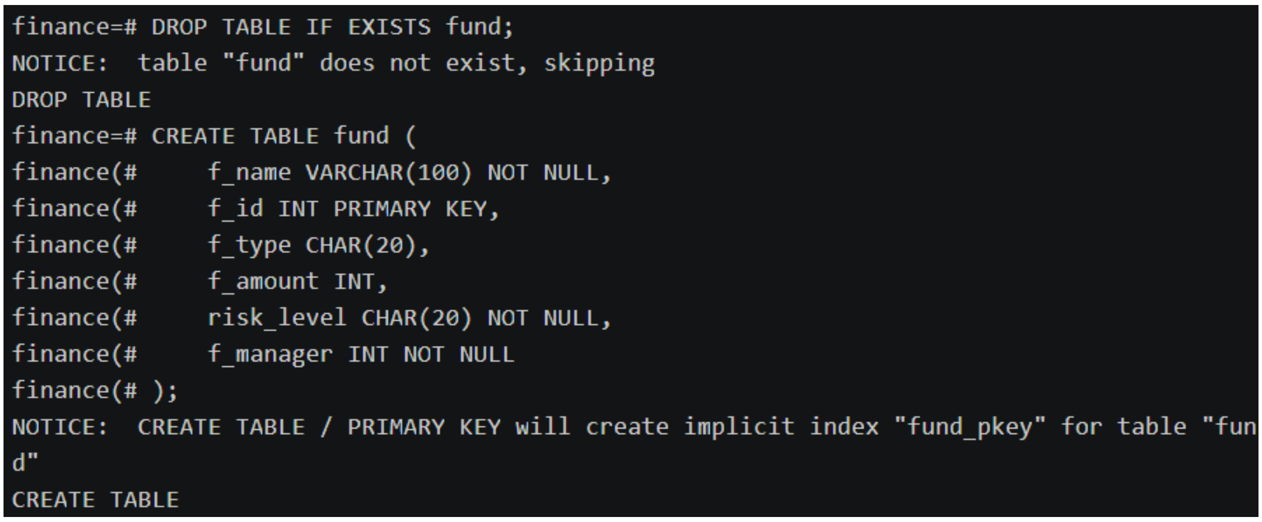
7. 创建保险信息表 insurance

```
DROP TABLE IF EXISTS insurance;
CREATE TABLE insurance (
    i_name VARCHAR(100) NOT NULL,
    i_id INT PRIMARY KEY,
    i_amount INT,
    i_person CHAR(20),
    i_year INT,
    i_project VARCHAR(200)
);
```


```
finance=# DROP TABLE IF EXISTS insurance;
NOTICE:  table "insurance" does not exist, skipping
DROP TABLE
finance=# CREATE TABLE insurance (
finance(#     i_name VARCHAR(100) NOT NULL,
finance(#     i_id INT PRIMARY KEY,
finance(#     i_amount INT,
finance(#     i_person CHAR(20),
finance(#     i_year INT,
finance(#     i_project VARCHAR(200)
finance(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "insurance_pkey" for table
 "insurance"
CREATE TABLE
```

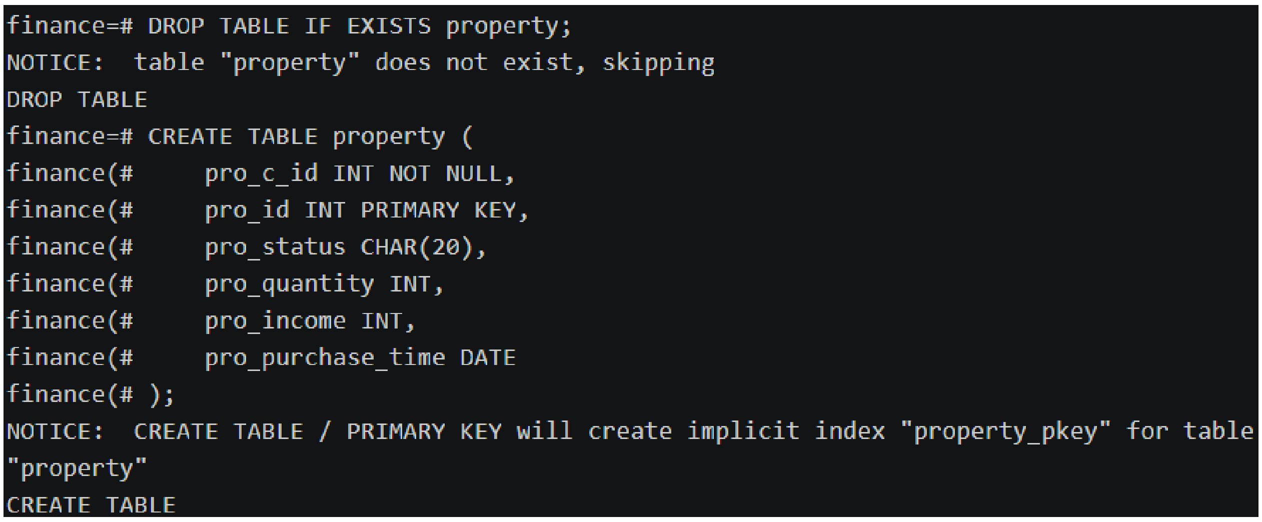
8. 创建基金信息表 fund

```
DROP TABLE IF EXISTS fund;
CREATE TABLE fund (
    f_name VARCHAR(100) NOT NULL,
    f_id INT PRIMARY KEY,
    f_type CHAR(20),
    f_amount INT,
    risk_level CHAR(20) NOT NULL,
    f_manager INT NOT NULL
);
```


```
finance=# DROP TABLE IF EXISTS fund;
NOTICE:  table "fund" does not exist, skipping
DROP TABLE
finance=# CREATE TABLE fund (
finance(#     f_name VARCHAR(100) NOT NULL,
finance(#     f_id INT PRIMARY KEY,
finance(#     f_type CHAR(20),
finance(#     f_amount INT,
finance(#     risk_level CHAR(20) NOT NULL,
finance(#     f_manager INT NOT NULL
finance(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "fund_pkey" for table "fun
d"
CREATE TABLE
```

9. 创建资产信息表 property

```
DROP TABLE IF EXISTS property;
CREATE TABLE property (
    pro_c_id INT NOT NULL,
    pro_id INT PRIMARY KEY,
    pro_status CHAR(20),
    pro_quantity INT,
    pro_income INT,
    pro_purchase_time DATE
);
```

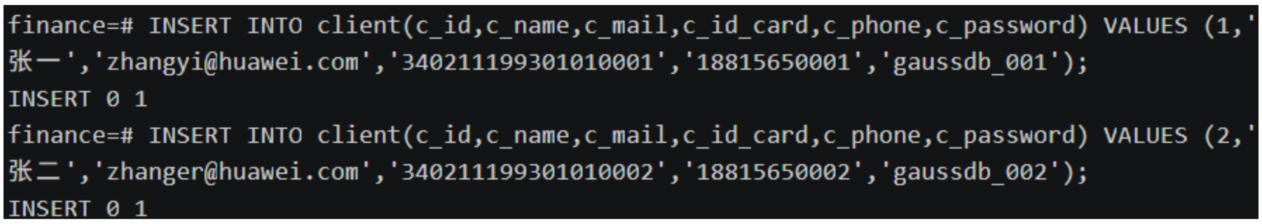


```
finance=# DROP TABLE IF EXISTS property;
NOTICE:  table "property" does not exist, skipping
DROP TABLE
finance=# CREATE TABLE property (
finance(#     pro_c_id INT NOT NULL,
finance(#     pro_id INT PRIMARY KEY,
finance(#     pro_status CHAR(20),
finance(#     pro_quantity INT,
finance(#     pro_income INT,
finance(#     pro_purchase_time DATE
finance(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "property_pkey" for table
"property"
CREATE TABLE
```

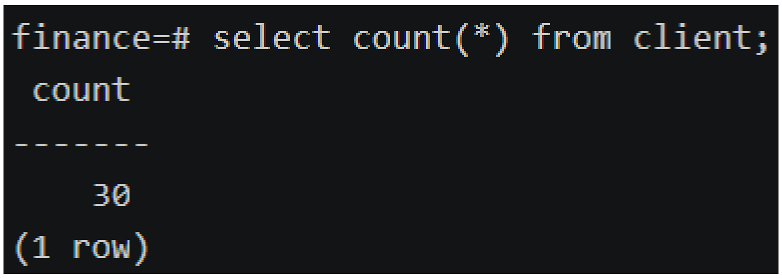
三、插入测试数据

10. 向 client 表插入 30 条客户数据

```
INSERT INTO client(c_id,c_name,c_mail,c_id_card,c_phone,c_password) VALUES (1,'张一','zhangyi@huawei.com','340211199301010001','18815650001','gaussdb_001');
...
```
（此处省略第 2 至 30 条 INSERT 语句）



```
finance=# INSERT INTO client(c_id,c_name,c_mail,c_id_card,c_phone,c_password) VALUES (1,'
张一','zhangyi@huawei.com','340211199301010001','18815650001','gaussdb_001');
INSERT 0 1
finance=# INSERT INTO client(c_id,c_name,c_mail,c_id_card,c_phone,c_password) VALUES (2,'
张二','zhanger@huawei.com','340211199301010002','18815650002','gaussdb_002');
INSERT 0 1
```

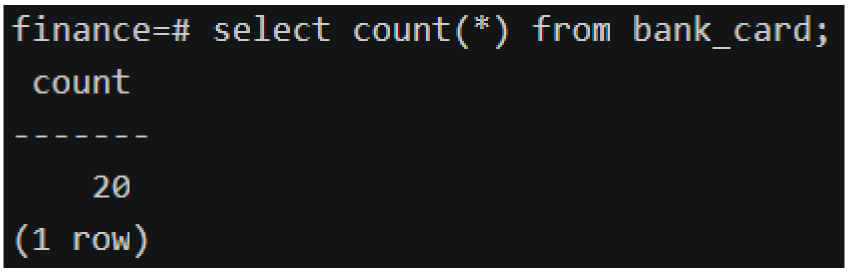


```
finance=# select count(*) from client;
 count
-------
    30
(1 row)
```

11. 向 bank_card 表插入 20 条银行卡数据

```
INSERT INTO bank_card(b_number,b_type,b_c_id) VALUES ('6222021302020000001','
```

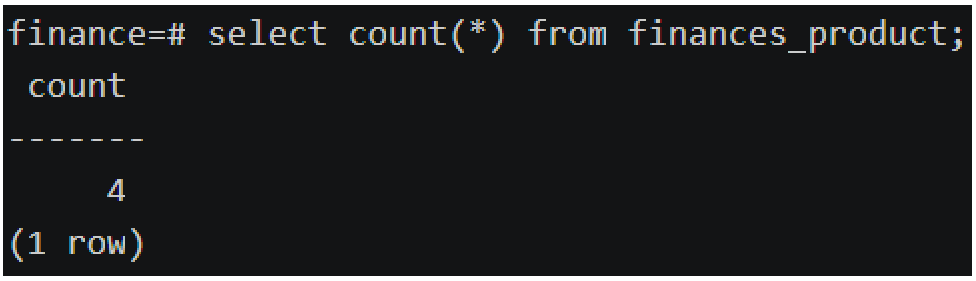
信用卡',1);
...
（此处省略第 2 至 20 条 INSERT 语句）



```
finance=# select count(*) from bank_card;
 count
-------
    20
(1 row)
```

12. 向 finances_product 表插入 4 条理财产品数据

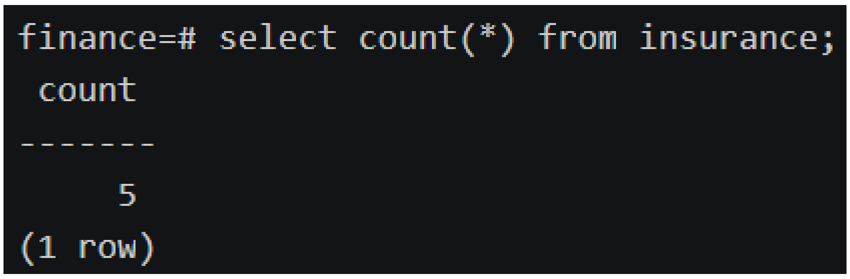
INSERT INTO finances_product(p_name,p_id,p_description,p_amount,p_year) VALUES ('债券',1,'...',50000,6);
...
（此处省略第 2 至 4 条 INSERT 语句）



```
finance=# select count(*) from finances_product;
 count
-------
     4
(1 row)
```

13. 向 insurance 表插入 5 条保险数据

INSERT INTO insurance(i_name,i_id,i_amount,i_person,i_year,i_project) VALUES ('健康保险',1,2000,'老人',30,'平安保险');
...
（此处省略第 2 至 5 条 INSERT 语句）

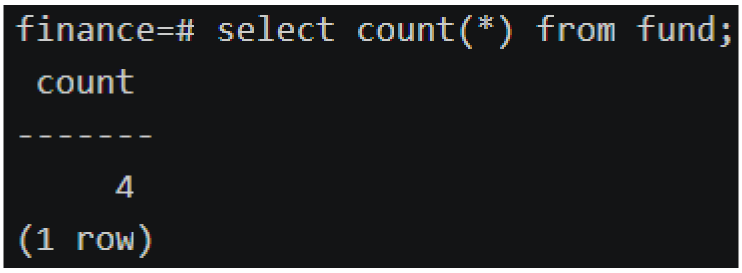


```
finance=# select count(*) from insurance;
 count
-------
     5
(1 row)
```

14. 向 fund 表插入 4 条基金数据

INSERT INTO fund(f_name,f_id,f_type,f_amount,risk_level,f_manager) VALUES ('股票',1,'股票型',10000,'高',1);

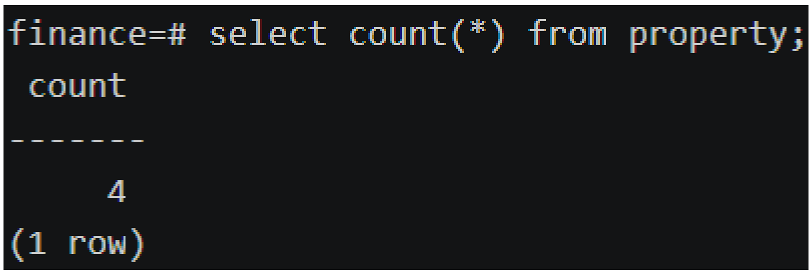
...
（此处省略第 2 至 4 条 INSERT 语句）



```
finance=# select count(*) from fund;
 count
-------
     4
(1 row)
```

15. 向 property 表插入 4 条资产数据

INSERT INTO property(pro_c_id,pro_id,pro_status,pro_quantity,pro_income,pro_purchase_time) VALUES (5,1,'可用',4,8000,'2018-07-01');
...
（此处省略第 2 至 4 条 INSERT 语句）

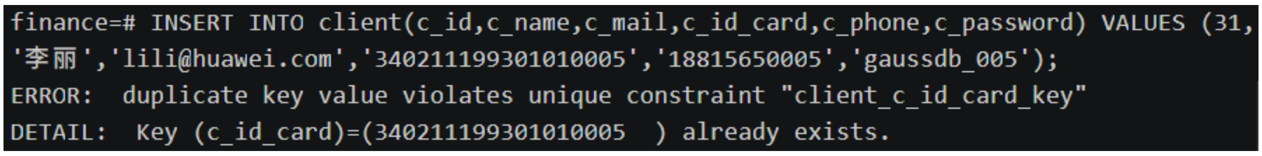


```
finance=# select count(*) from property;
 count
-------
     4
(1 row)
```

四、手工插入数据与约束管理

16. 尝试插入重复数据（应失败）

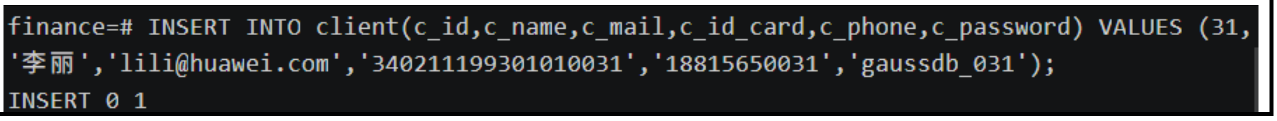
INSERT INTO client(c_id,c_name,c_mail,c_id_card,c_phone,c_password) VALUES (31,'李丽','lili@huawei.com','340211199301010005','18815650005','gaussdb_005');



```
finance=# INSERT INTO client(c_id,c_name,c_mail,c_id_card,c_phone,c_password) VALUES (31,
'李丽','lili@huawei.com','340211199301010005','18815650005','gaussdb_005');
ERROR:  duplicate key value violates unique constraint "client_c_id_card_key"
DETAIL:  Key (c_id_card)=(340211199301010005  ) already exists.
```

17. 成功插入新客户数据

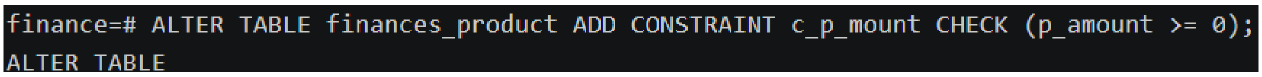
INSERT INTO client(c_id,c_name,c_mail,c_id_card,c_phone,c_password) VALUES (31,'李丽','lili@huawei.com','340211199301010031','18815650031','gaussdb_031');



```
finance=# INSERT INTO client(c_id,c_name,c_mail,c_id_card,c_phone,c_password) VALUES (31,
'李丽','lili@huawei.com','340211199301010031','18815650031','gaussdb_031');
INSERT 0 1
```

18. 为 finances_product 表添加金额非负约束

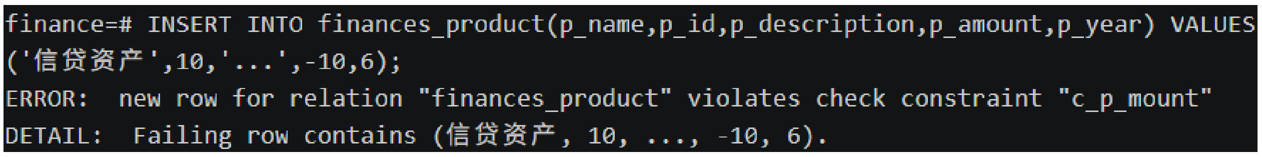
ALTER TABLE finances_product ADD CONSTRAINT c_p_mount CHECK (p_amount >= 0);



```
finance=# ALTER TABLE finances_product ADD CONSTRAINT c_p_mount CHECK (p_amount >= 0);
ALTER TABLE
```

19. 尝试插入负金额数据（应失败）

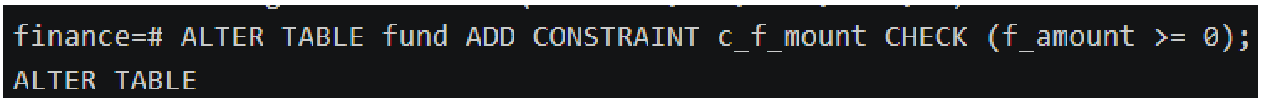
INSERT INTO finances_product(p_name,p_id,p_description,p_amount,p_year) VALUES ('信贷资产',10,'...',-10,6);



```
finance=# INSERT INTO finances_product(p_name,p_id,p_description,p_amount,p_year) VALUES
('信贷资产',10,'...',-10,6);
ERROR:  new row for relation "finances_product" violates check constraint "c_p_mount"
DETAIL:  Failing row contains (信贷资产, 10, ..., -10, 6).
```

20. 为 fund 表添加金额非负约束

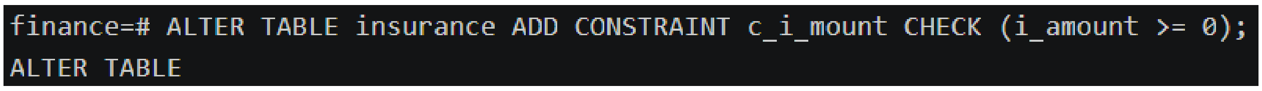
ALTER TABLE fund ADD CONSTRAINT c_f_mount CHECK (f_amount >= 0);



```
finance=# ALTER TABLE fund ADD CONSTRAINT c_f_mount CHECK (f_amount >= 0);
ALTER TABLE
```

21. 为 insurance 表添加金额非负约束

ALTER TABLE insurance ADD CONSTRAINT c_i_mount CHECK (i_amount >= 0);

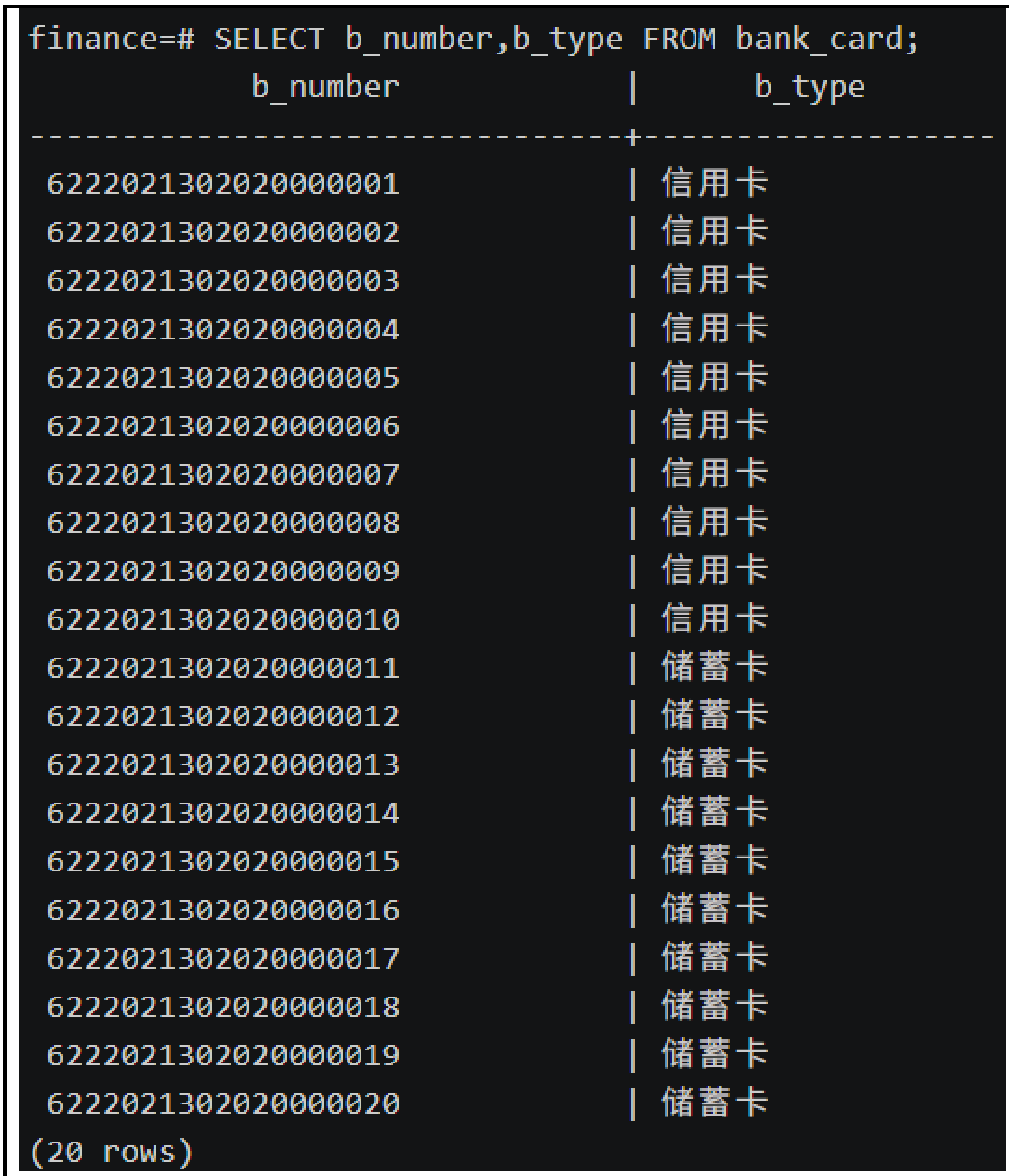


```
finance=# ALTER TABLE insurance ADD CONSTRAINT c_i_mount CHECK (i_amount >= 0);
ALTER TABLE
```

五、数据查询操作

22. 单表查询：查询银行卡信息

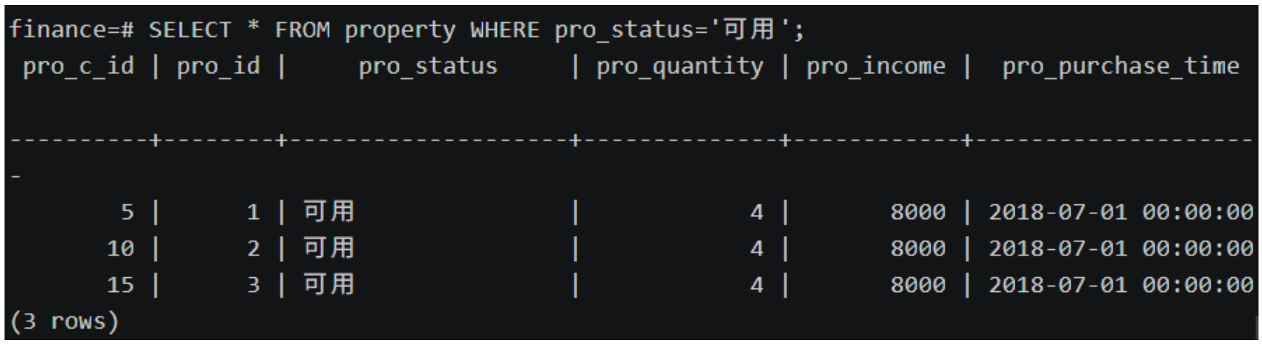
SELECT b_number,b_type FROM bank_card;



```
finance=# SELECT b_number,b_type FROM bank_card;
           b_number               |      b_type
----------------------------------+--------------------
 6222021302020000001              | 信用卡
 6222021302020000002              | 信用卡
 6222021302020000003              | 信用卡
 6222021302020000004              | 信用卡
 6222021302020000005              | 信用卡
 6222021302020000006              | 信用卡
 6222021302020000007              | 信用卡
 6222021302020000008              | 信用卡
 6222021302020000009              | 信用卡
 6222021302020000010              | 信用卡
 6222021302020000011              | 储蓄卡
 6222021302020000012              | 储蓄卡
 6222021302020000013              | 储蓄卡
 6222021302020000014              | 储蓄卡
 6222021302020000015              | 储蓄卡
 6222021302020000016              | 储蓄卡
 6222021302020000017              | 储蓄卡
 6222021302020000018              | 储蓄卡
 6222021302020000019              | 储蓄卡
 6222021302020000020              | 储蓄卡
(20 rows)
```

23. 条件查询：查询可用资产

SELECT * FROM property WHERE pro_status='可用';

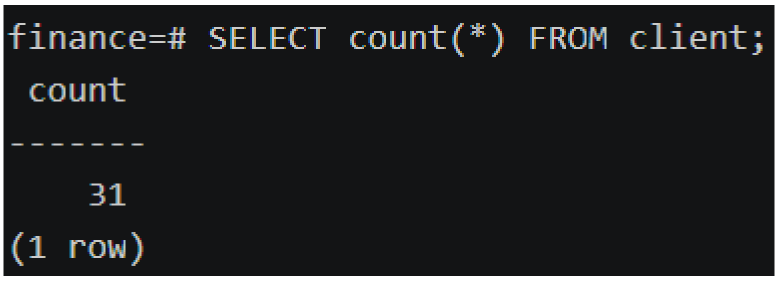


```
finance=# SELECT * FROM property WHERE pro_status='可用';
 pro_c_id | pro_id |      pro_status      | pro_quantity | pro_income |  pro_purchase_time


----------+--------+--------------------+--------------+------------+--------------------
-
        5 |      1 | 可用                |            4 |       8000 | 2018-07-01 00:00:00
       10 |      2 | 可用                |            4 |       8000 | 2018-07-01 00:00:00
       15 |      3 | 可用                |            4 |       8000 | 2018-07-01 00:00:00
(3 rows)
```

24. 聚合查询：统计客户数量

SELECT count(*) FROM client;



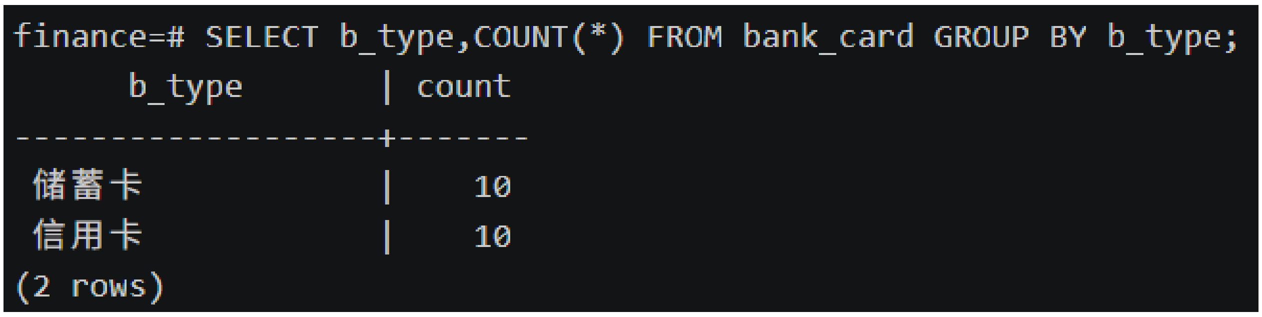
```
finance=# SELECT count(*) FROM client;
 count
-------
    31
(1 row)
```

25. 聚合查询：按类型统计银行卡数量

SELECT b_type,COUNT(*) FROM bank_card GROUP BY b_type;

26. 聚合查询：计算平均保险金额

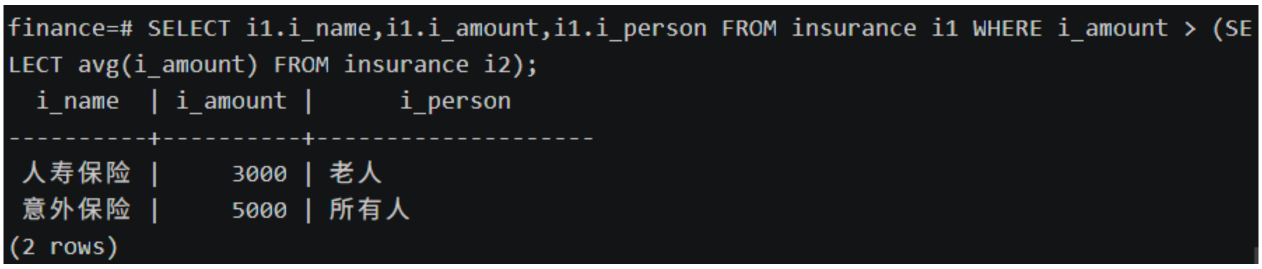
SELECT AVG(i_amount) FROM insurance;



```
finance=# SELECT b_type,COUNT(*) FROM bank_card GROUP BY b_type;
       b_type        | count
---------------------+-------
 储蓄卡               |    10
 信用卡               |    10
(2 rows)
```

27. 子查询：查询保险金额大于平均值的保险

SELECT i1.i_name,i1.i_amount,i1.i_person FROM insurance i1 WHERE i_amount > (SELECT avg(i_amount) FROM insurance i2);

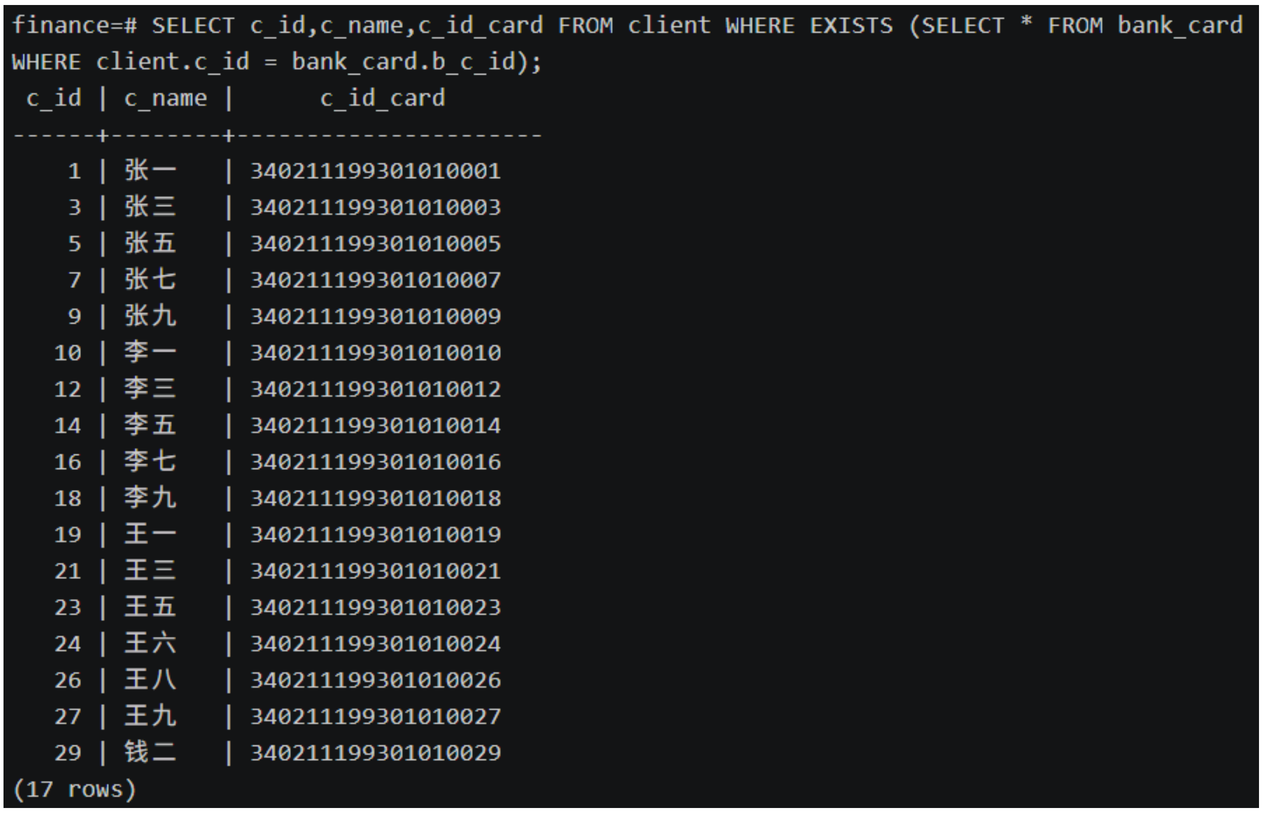


```
finance=# SELECT i1.i_name,i1.i_amount,i1.i_person FROM insurance i1 WHERE i_amount > (SE
LECT avg(i_amount) FROM insurance i2);
  i_name  | i_amount |      i_person
----------+----------+--------------------
 人寿保险 |     3000 | 老人
 意外保险 |     5000 | 所有人
(2 rows)
```

28. 半连接查询：查询有银行卡的客户

SELECT c_id,c_name,c_id_card FROM client WHERE EXISTS (SELECT * FROM bank_card

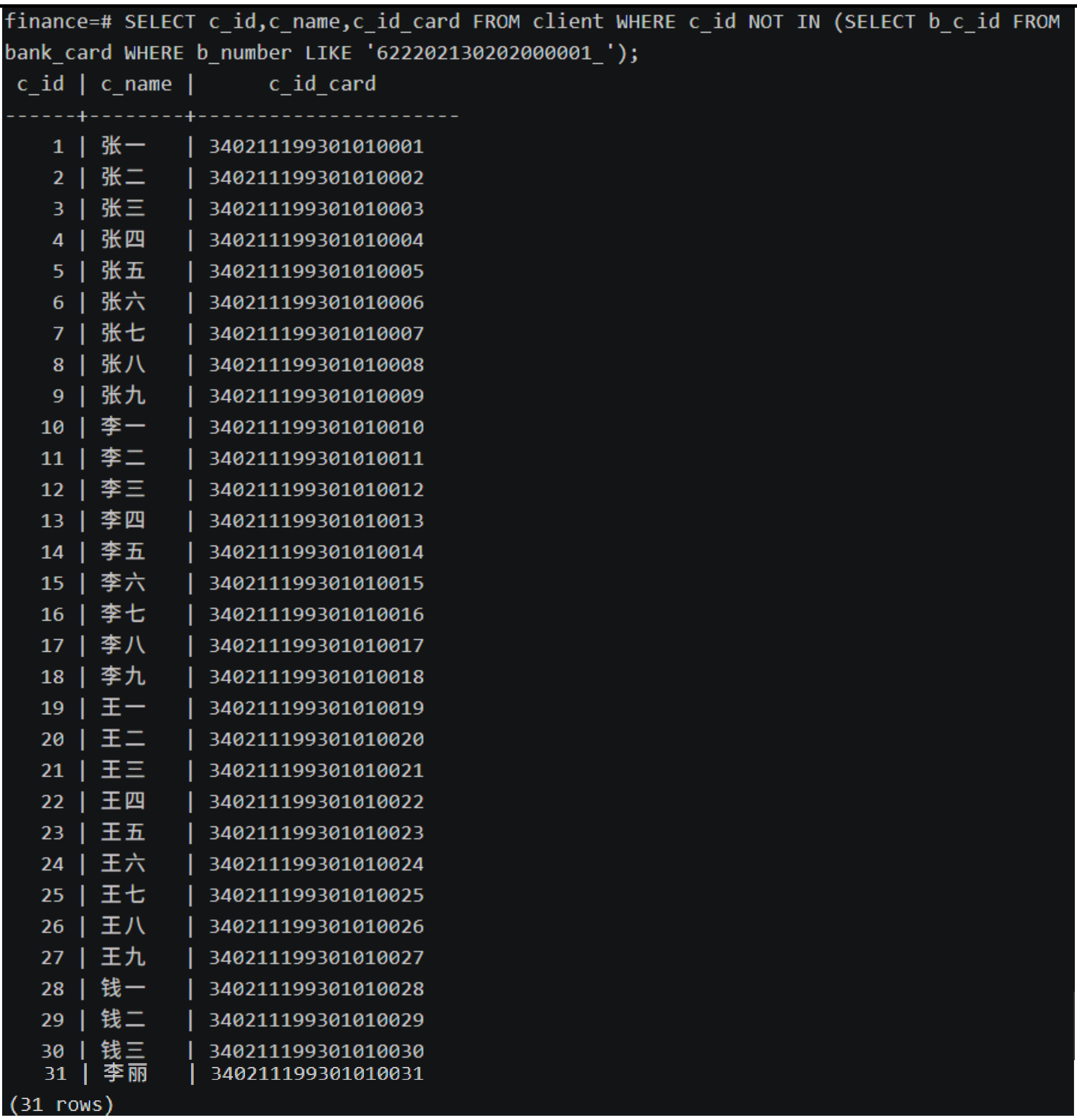
WHERE client.c_id = bank_card.b_c_id);



```
finance=# SELECT c_id,c_name,c_id_card FROM client WHERE EXISTS (SELECT * FROM bank_card
WHERE client.c_id = bank_card.b_c_id);
 c_id | c_name |      c_id_card
------+--------+---------------------
    1 | 张一    | 340211199301010001
    3 | 张三    | 340211199301010003
    5 | 张五    | 340211199301010005
    7 | 张七    | 340211199301010007
    9 | 张九    | 340211199301010009
   10 | 李一    | 340211199301010010
   12 | 李三    | 340211199301010012
   14 | 李五    | 340211199301010014
   16 | 李七    | 340211199301010016
   18 | 李九    | 340211199301010018
   19 | 王一    | 340211199301010019
   21 | 王三    | 340211199301010021
   23 | 王五    | 340211199301010023
   24 | 王六    | 340211199301010024
   26 | 王八    | 340211199301010026
   27 | 王九    | 340211199301010027
   29 | 钱二    | 340211199301010029
(17 rows)
```

29. 反连接查询：查询没有特定银行卡前缀的客户

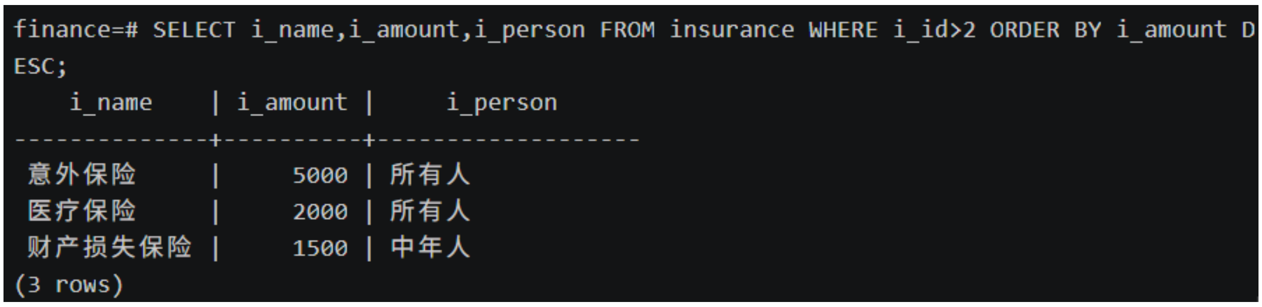
SELECT c_id,c_name,c_id_card FROM client WHERE c_id NOT IN (SELECT b_c_id FROM bank_card WHERE b_number LIKE '622202130202000001_');



```
finance=# SELECT c_id,c_name,c_id_card FROM client WHERE c_id NOT IN (SELECT b_c_id FROM
bank_card WHERE b_number LIKE '622202130202000001_');
 c_id | c_name |      c_id_card
------+--------+----------------------
    1 | 张一   | 340211199301010001
    2 | 张二   | 340211199301010002
    3 | 张三   | 340211199301010003
    4 | 张四   | 340211199301010004
    5 | 张五   | 340211199301010005
    6 | 张六   | 340211199301010006
    7 | 张七   | 340211199301010007
    8 | 张八   | 340211199301010008
    9 | 张九   | 340211199301010009
   10 | 李一   | 340211199301010010
   11 | 李二   | 340211199301010011
   12 | 李三   | 340211199301010012
   13 | 李四   | 340211199301010013
   14 | 李五   | 340211199301010014
   15 | 李六   | 340211199301010015
   16 | 李七   | 340211199301010016
   17 | 李八   | 340211199301010017
   18 | 李九   | 340211199301010018
   19 | 王一   | 340211199301010019
   20 | 王二   | 340211199301010020
   21 | 王三   | 340211199301010021
   22 | 王四   | 340211199301010022
   23 | 王五   | 340211199301010023
   24 | 王六   | 340211199301010024
   25 | 王七   | 340211199301010025
   26 | 王八   | 340211199301010026
   27 | 王九   | 340211199301010027
   28 | 钱一   | 340211199301010028
   29 | 钱二   | 340211199301010029
   30 | 钱三   | 340211199301010030
   31 | 李丽   | 340211199301010031
(31 rows)
```

30. 排序查询：按保额降序查询保险

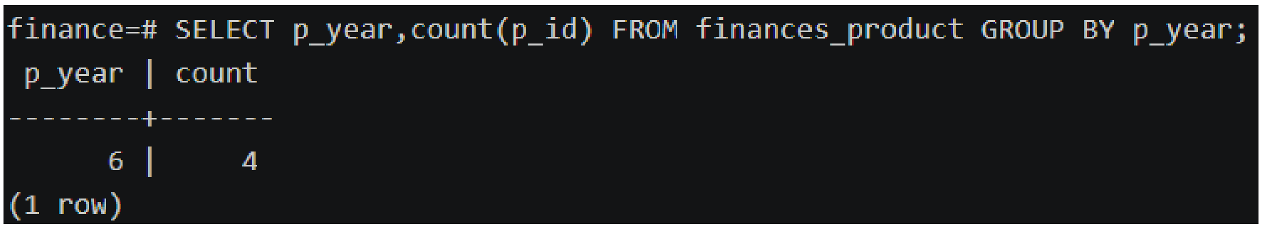
SELECT i_name,i_amount,i_person FROM insurance WHERE i_id>2 ORDER BY i_amount DESC;



```
finance=# SELECT i_name,i_amount,i_person FROM insurance WHERE i_id>2 ORDER BY i_amount D
ESC;
    i_name     | i_amount |     i_person
---------------+----------+-------------------
 意外保险      |     5000 | 所有人
 医疗保险      |     2000 | 所有人
 财产损失保险  |     1500 | 中年人
(3 rows)
```

31. 分组查询：按年份统计理财产品数量

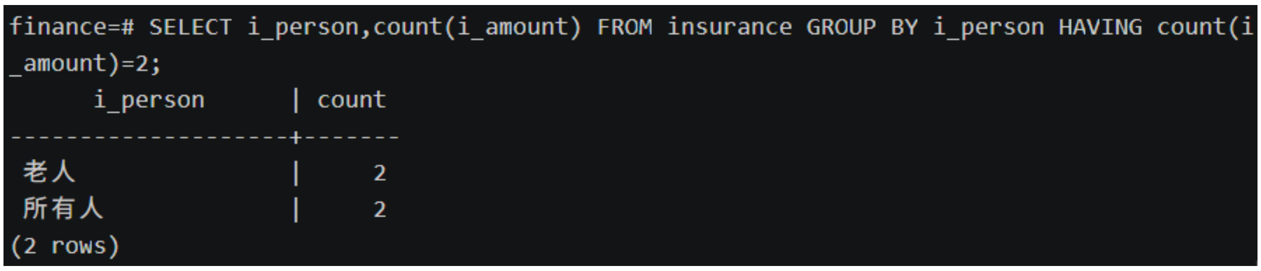
SELECT p_year,count(p_id) FROM finances_product GROUP BY p_year;



```
finance=# SELECT p_year,count(p_id) FROM finances_product GROUP BY p_year;
 p_year | count
--------+-------
      6 |     4
(1 row)
```

32. HAVING 查询：统计保险金额数量为 2 的人群

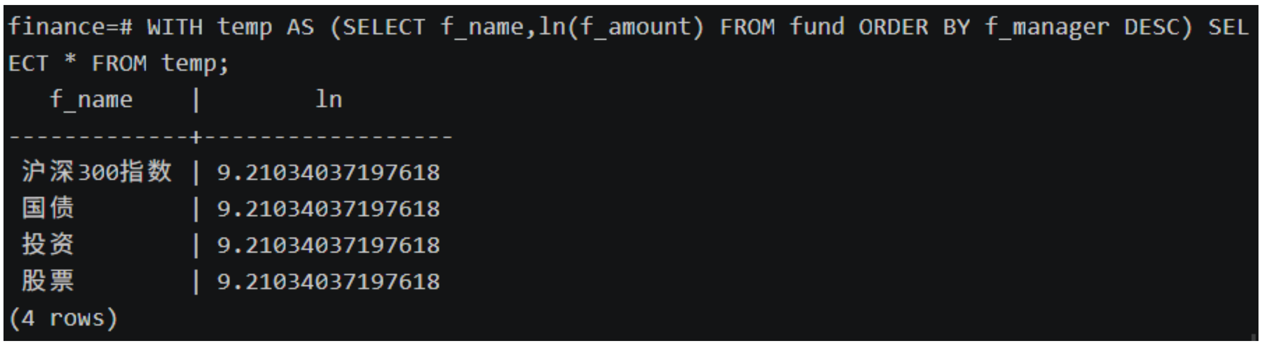
SELECT i_person,count(i_amount) FROM insurance GROUP BY i_person HAVING count(i_amount)=2;



```
finance=# SELECT i_person,count(i_amount) FROM insurance GROUP BY i_person HAVING count(i
_amount)=2;
      i_person       | count
--------------------+-------
 老人                |     2
 所有人              |     2
(2 rows)
```

33. WITH AS 查询：使用临时 SQL 片段查询基金

WITH temp AS (SELECT f_name,ln(f_amount) FROM fund ORDER BY f_manager DESC) SELECT * FROM temp;

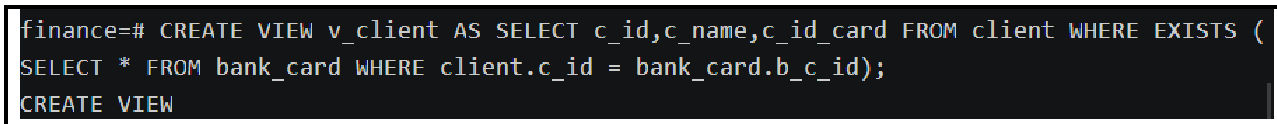


```
finance=# WITH temp AS (SELECT f_name,ln(f_amount) FROM fund ORDER BY f_manager DESC) SEL
ECT * FROM temp;
   f_name    |        ln
-------------+------------------
 沪深300指数 | 9.21034037197618
 国债        | 9.21034037197618
 投资        | 9.21034037197618
 股票        | 9.21034037197618
(4 rows)
```

六、视图操作

34. 创建视图：基于半连接查询

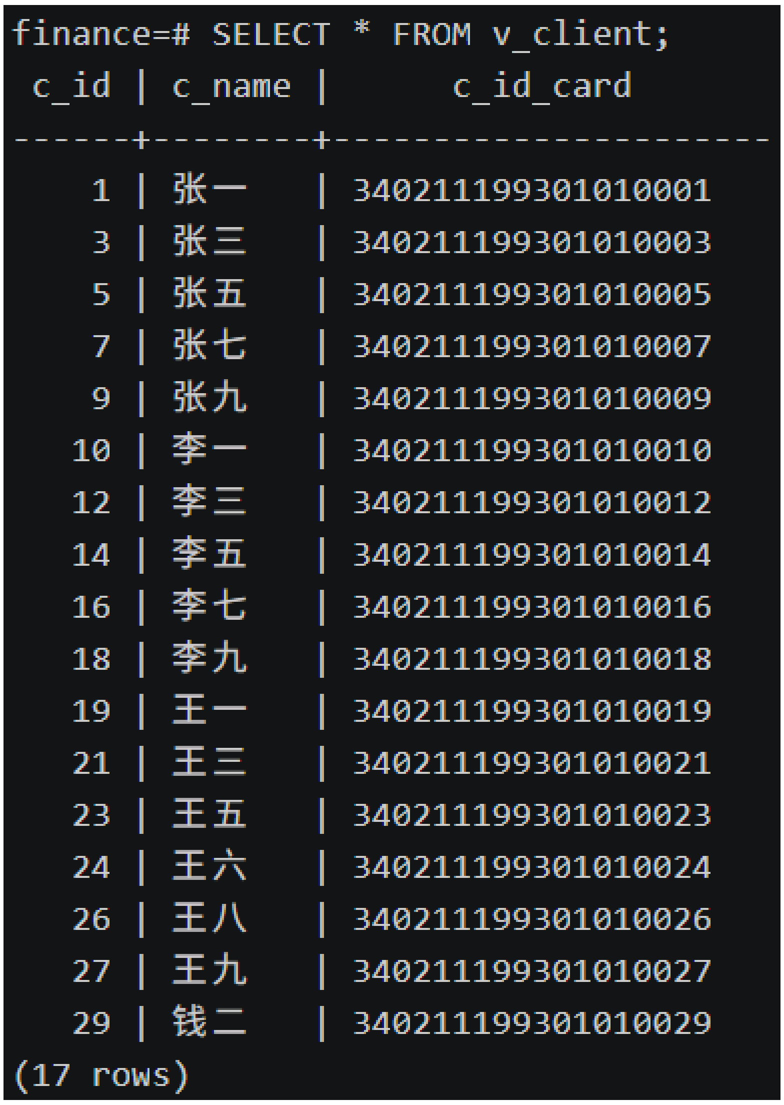
CREATE VIEW v_client AS SELECT c_id,c_name,c_id_card FROM client WHERE EXISTS (SELECT * FROM bank_card WHERE client.c_id = bank_card.b_c_id);



```
finance=# CREATE VIEW v_client AS SELECT c_id,c_name,c_id_card FROM client WHERE EXISTS (
SELECT * FROM bank_card WHERE client.c_id = bank_card.b_c_id);
CREATE VIEW
```

35. 使用视图查询

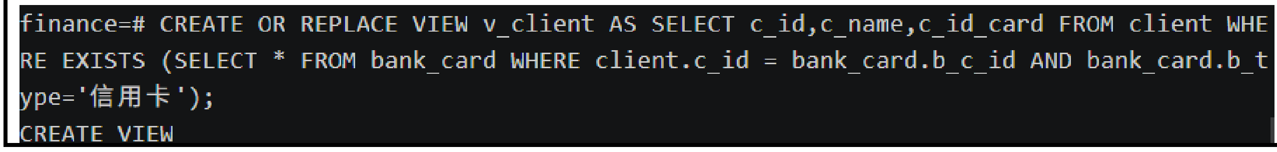
SELECT * FROM v_client;



```
finance=# SELECT * FROM v_client;
 c_id | c_name |      c_id_card
------+--------+----------------------
    1 | 张一   | 340211199301010001
    3 | 张三   | 340211199301010003
    5 | 张五   | 340211199301010005
    7 | 张七   | 340211199301010007
    9 | 张九   | 340211199301010009
   10 | 李一   | 340211199301010010
   12 | 李三   | 340211199301010012
   14 | 李五   | 340211199301010014
   16 | 李七   | 340211199301010016
   18 | 李九   | 340211199301010018
   19 | 王一   | 340211199301010019
   21 | 王三   | 340211199301010021
   23 | 王五   | 340211199301010023
   24 | 王六   | 340211199301010024
   26 | 王八   | 340211199301010026
   27 | 王九   | 340211199301010027
   29 | 钱二   | 340211199301010029
(17 rows)
```

36. 修改视图：仅保留信用卡客户

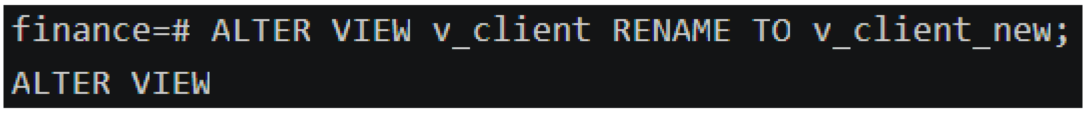
CREATE OR REPLACE VIEW v_client AS SELECT c_id, c_name, c_id_card FROM client WHERE EXISTS (SELECT * FROM bank_card WHERE client.c_id = bank_card.b_c_id AND bank_card.b_type='信用卡');



```
finance=# CREATE OR REPLACE VIEW v_client AS SELECT c_id,c_name,c_id_card FROM client WHE
RE EXISTS (SELECT * FROM bank_card WHERE client.c_id = bank_card.b_c_id AND bank_card.b_t
ype='信用卡');
CREATE VIEW
```

37. 重命名视图

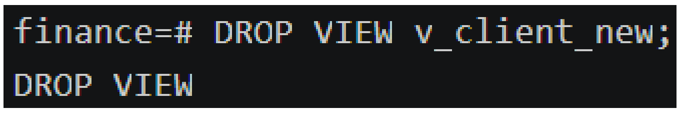
ALTER VIEW v_client RENAME TO v_client_new;



```
finance=# ALTER VIEW v_client RENAME TO v_client_new;
ALTER VIEW
```

38. 删除视图

DROP VIEW v_client_new;

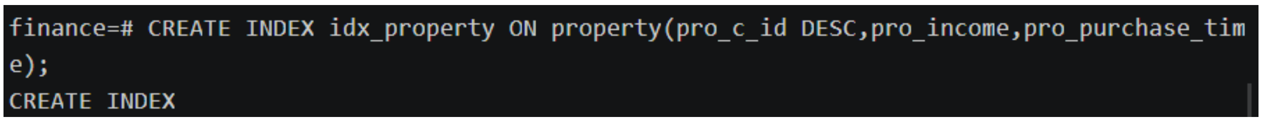


```
finance=# DROP VIEW v_client_new;
DROP VIEW
```

七、索引操作

39. 创建复合索引

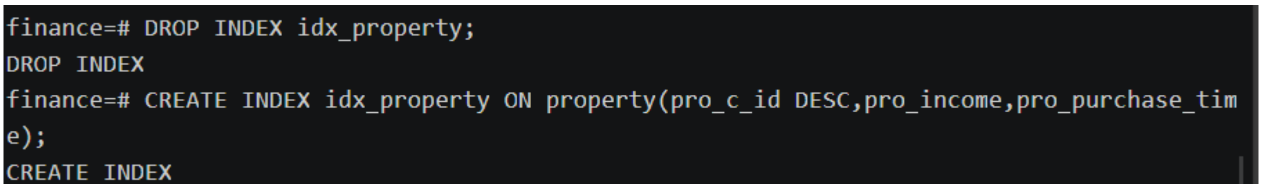
CREATE INDEX idx_property ON property(pro_c_id DESC,pro_income,pro_purchase_time);



```
finance=# CREATE INDEX idx_property ON property(pro_c_id DESC,pro_income,pro_purchase_tim
e);
CREATE INDEX
```

40. 删除并重建索引

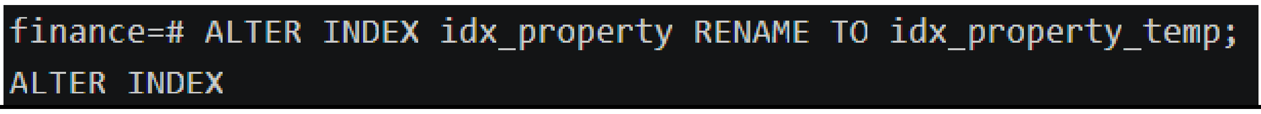
DROP INDEX idx_property;
CREATE INDEX idx_property ON property(pro_c_id DESC,pro_income,pro_purchase_time);



```
finance=# DROP INDEX idx_property;
DROP INDEX
finance=# CREATE INDEX idx_property ON property(pro_c_id DESC,pro_income,pro_purchase_tim
e);
CREATE INDEX
```

41. 重命名索引

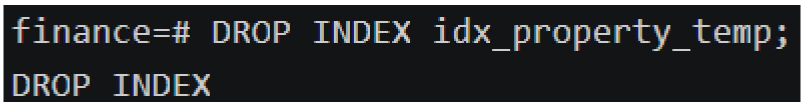
ALTER INDEX idx_property RENAME TO idx_property_temp;



```
finance=# ALTER INDEX idx_property RENAME TO idx_property_temp;
ALTER INDEX
```

42. 删除索引

DROP INDEX idx_property_temp;

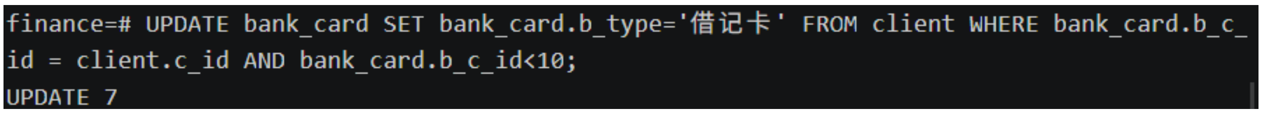


```
finance=# DROP INDEX idx_property_temp;
DROP INDEX
```

八、数据修改与删除

43. 更新数据：修改银行卡类型

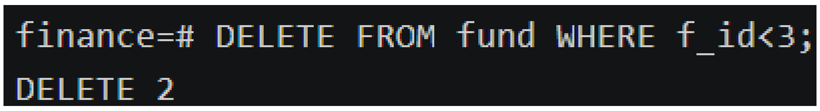
UPDATE bank_card SET bank_card.b_type='借记卡' FROM client WHERE bank_card.b_c_id = client.c_id AND bank_card.b_c_id<10;



```
finance=# UPDATE bank_card SET bank_card.b_type='借记卡' FROM client WHERE bank_card.b_c_
id = client.c_id AND bank_card.b_c_id<10;
UPDATE 7
```

44. 删除数据：删除基金表中编号小于 3 的记录

DELETE FROM fund WHERE f_id<3;

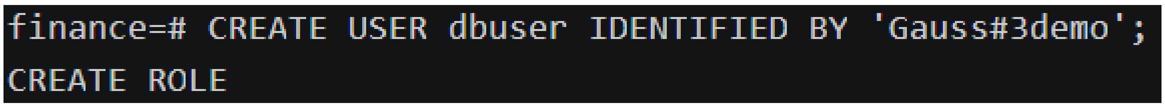


```
finance=# DELETE FROM fund WHERE f_id<3;
DELETE 2
```

九、用户与权限管理

45. 创建新用户

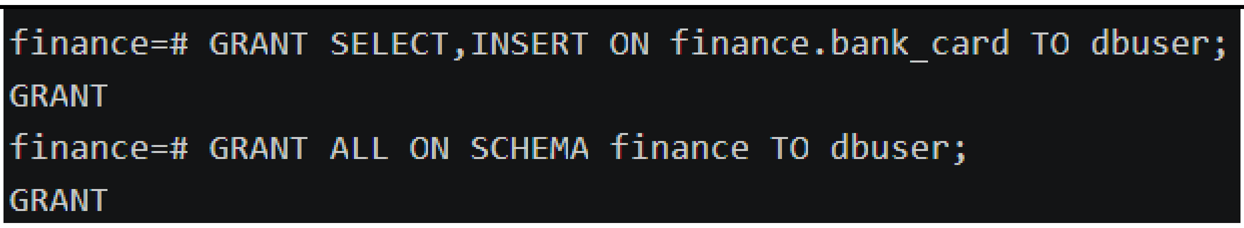
CREATE USER dbuser IDENTIFIED BY 'Gauss#3demo';



```
finance=# CREATE USER dbuser IDENTIFIED BY 'Gauss#3demo';
CREATE ROLE
```

46. 授予用户表权限

GRANT SELECT,INSERT ON finance.bank_card TO dbuser;
GRANT ALL ON SCHEMA finance TO dbuser;



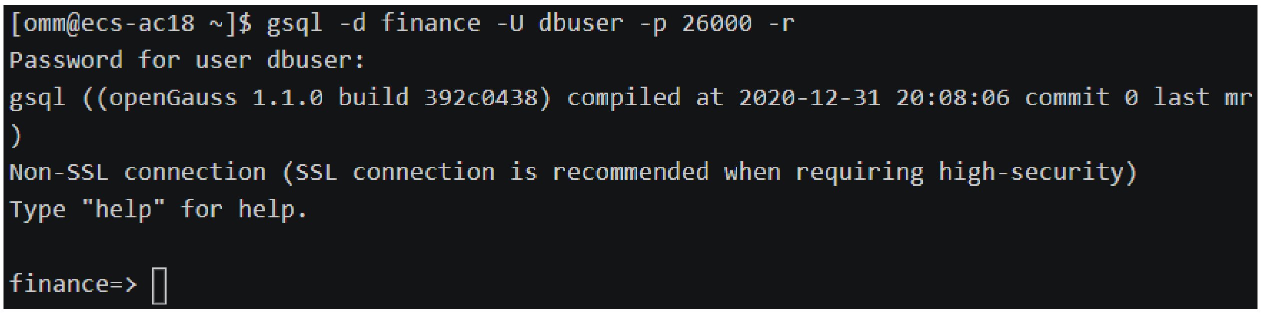
```
finance=# GRANT SELECT,INSERT ON finance.bank_card TO dbuser;
GRANT
finance=# GRANT ALL ON SCHEMA finance TO dbuser;
GRANT
```

47. 退出数据库

\q

48. 使用新用户连接数据库

gsql -d finance -U dbuser -p 26000 -r

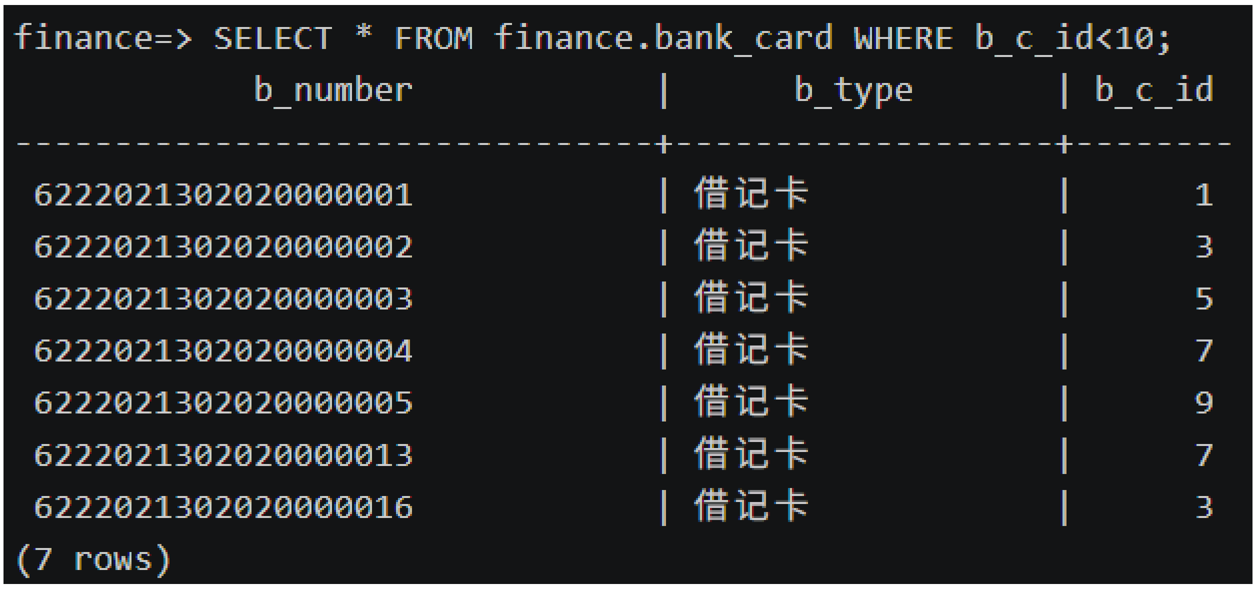


```
[omm@ecs-ac18 ~]$ gsql -d finance -U dbuser -p 26000 -r
Password for user dbuser:
gsql ((openGauss 1.1.0 build 392c0438) compiled at 2020-12-31 20:08:06 commit 0 last mr
)
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

finance=> 
```

49. 新用户查询表数据

SELECT * FROM finance.bank_card WHERE b_c_id<10;

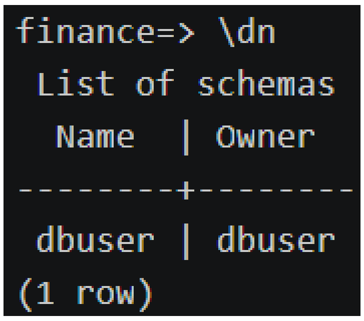


```
finance=> SELECT * FROM finance.bank_card WHERE b_c_id<10;
           b_number            |       b_type        | b_c_id
-------------------------------+---------------------+--------
 6222021302020000001           | 借记卡              |      1
 6222021302020000002           | 借记卡              |      3
 6222021302020000003           | 借记卡              |      5
 6222021302020000004           | 借记卡              |      7
 6222021302020000005           | 借记卡              |      9
 6222021302020000013           | 借记卡              |      7
 6222021302020000016           | 借记卡              |      3
(7 rows)
```

十、Schema 删除操作

50. 查看 schema 列表

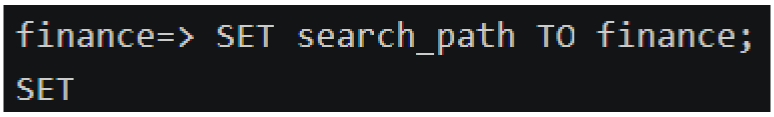
\dn



```
finance=> \dn
 List of schemas
  Name   | Owner
--------+--------
 dbuser | dbuser
(1 row)
```

51. 设置搜索路径

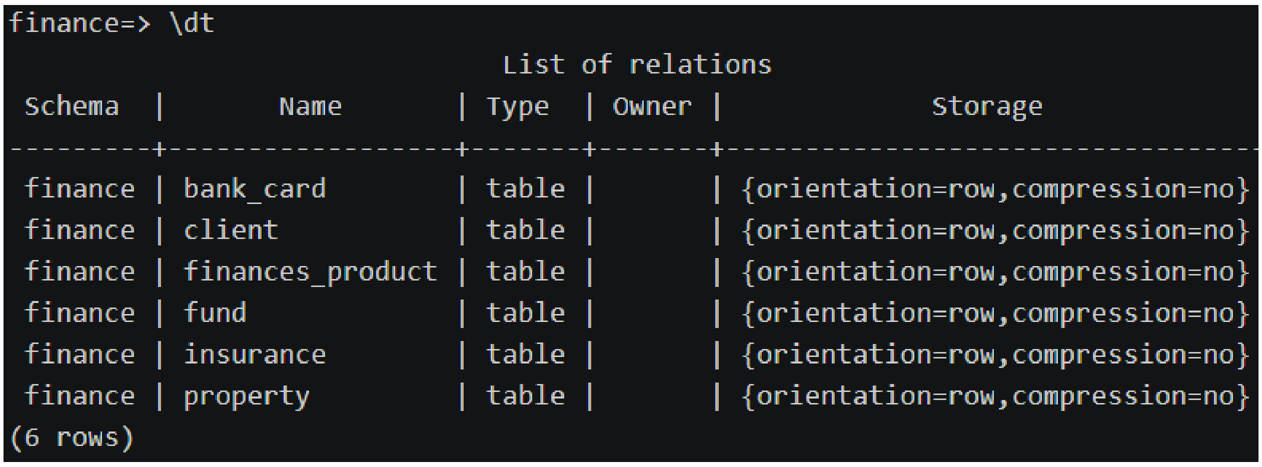
SET search_path TO finance;



```
finance=> SET search_path TO finance;
SET
```

52. 查看当前 schema 下的表

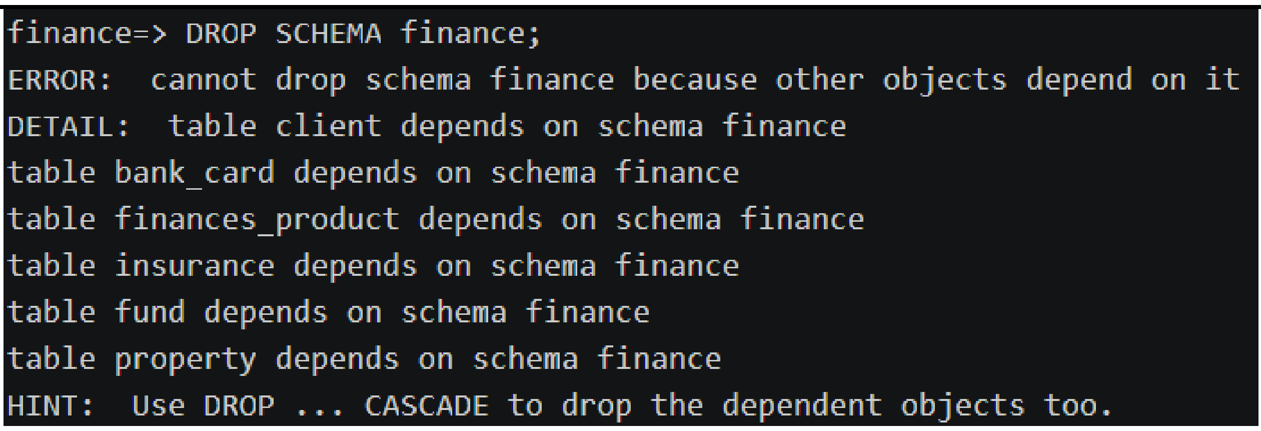
\dt



```
finance=> \dt
                                  List of relations
 Schema  |       Name        | Type  | Owner |             Storage
---------+-------------------+-------+-------+----------------------------------
 finance | bank_card         | table |       | {orientation=row,compression=no}
 finance | client            | table |       | {orientation=row,compression=no}
 finance | finances_product  | table |       | {orientation=row,compression=no}
 finance | fund              | table |       | {orientation=row,compression=no}
 finance | insurance         | table |       | {orientation=row,compression=no}
 finance | property          | table |       | {orientation=row,compression=no}
(6 rows)
```

53. 尝试删除 schema（应失败）

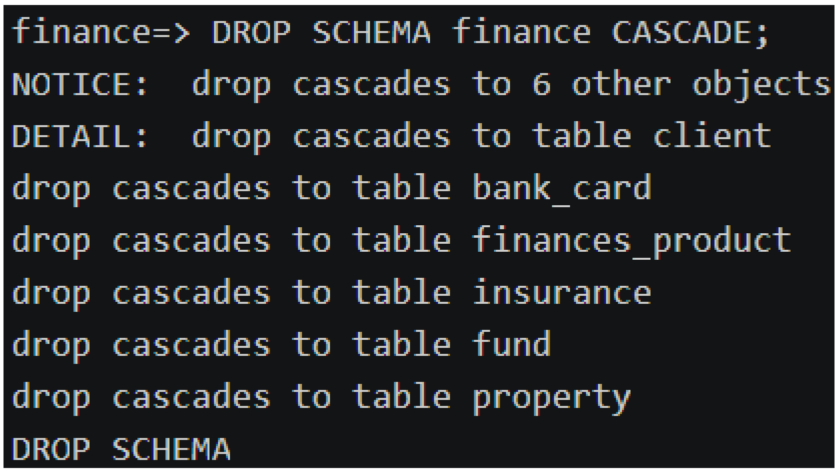
DROP SCHEMA finance;



```
finance=> DROP SCHEMA finance;
ERROR:  cannot drop schema finance because other objects depend on it
DETAIL:  table client depends on schema finance
table bank_card depends on schema finance
table finances_product depends on schema finance
table insurance depends on schema finance
table fund depends on schema finance
table property depends on schema finance
HINT:  Use DROP ... CASCADE to drop the dependent objects too.
```

54. 级联删除 schema 及其所有对象

DROP SCHEMA finance CASCADE;



```
finance=> DROP SCHEMA finance CASCADE;
NOTICE:  drop cascades to 6 other objects
DETAIL:  drop cascades to table client
drop cascades to table bank_card
drop cascades to table finances_product
drop cascades to table insurance
drop cascades to table fund
drop cascades to table property
DROP SCHEMA
```

十一、JDBC 连接准备（数据库端配置）

55. 修改 pg_hba.conf 以允许远程连接

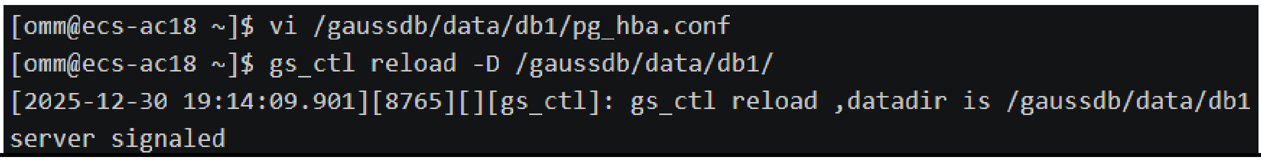
vi /gaussdb/data/db1/pg_hba.conf

添加行：

host all all 0.0.0.0/0 sha256

重载配置：

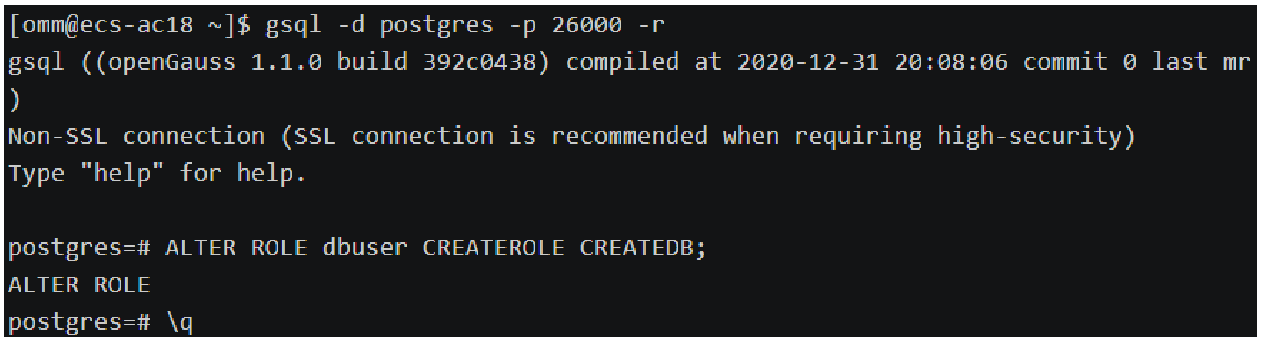
gs_ctl reload -D /gaussdb/data/db1/



```
[omm@ecs-ac18 ~]$ vi /gaussdb/data/db1/pg_hba.conf
[omm@ecs-ac18 ~]$ gs_ctl reload -D /gaussdb/data/db1/
[2025-12-30 19:14:09.901][8765][][gs_ctl]: gs_ctl reload ,datadir is /gaussdb/data/db1
server signaled
```

56. 授予用户更多权限

```
gsql -d postgres -p 26000 -r
ALTER ROLE dbuser CREATEROLE CREATEDB;
\q
```



```
[omm@ecs-ac18 ~]$ gsql -d postgres -p 26000 -r
gsql ((openGauss 1.1.0 build 392c0438) compiled at 2020-12-31 20:08:06 commit 0 last mr
)
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

postgres=# ALTER ROLE dbuser CREATEROLE CREATEDB;
ALTER ROLE
postgres=# \q
```

57. 修改监听地址

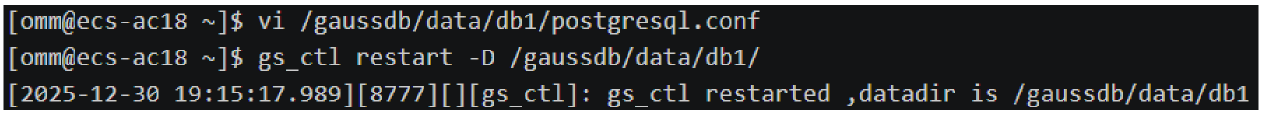
```
vi /gaussdb/data/db1/postgresql.conf
```

修改为：

```
listen_addresses = ’*’
```

重启数据库：

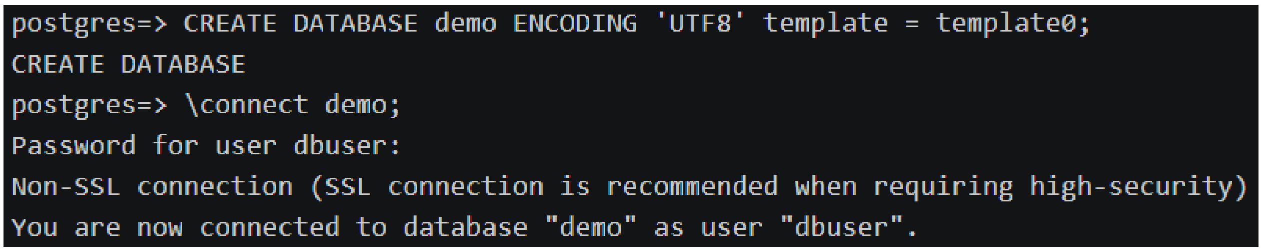
```
gs_ctl restart -D /gaussdb/data/db1/
```



```
[omm@ecs-ac18 ~]$ vi /gaussdb/data/db1/postgresql.conf
[omm@ecs-ac18 ~]$ gs_ctl restart -D /gaussdb/data/db1/
[2025-12-30 19:15:17.989][8777][][gs_ctl]: gs_ctl restarted ,datadir is /gaussdb/data/db1
```

58. 创建测试数据库和表

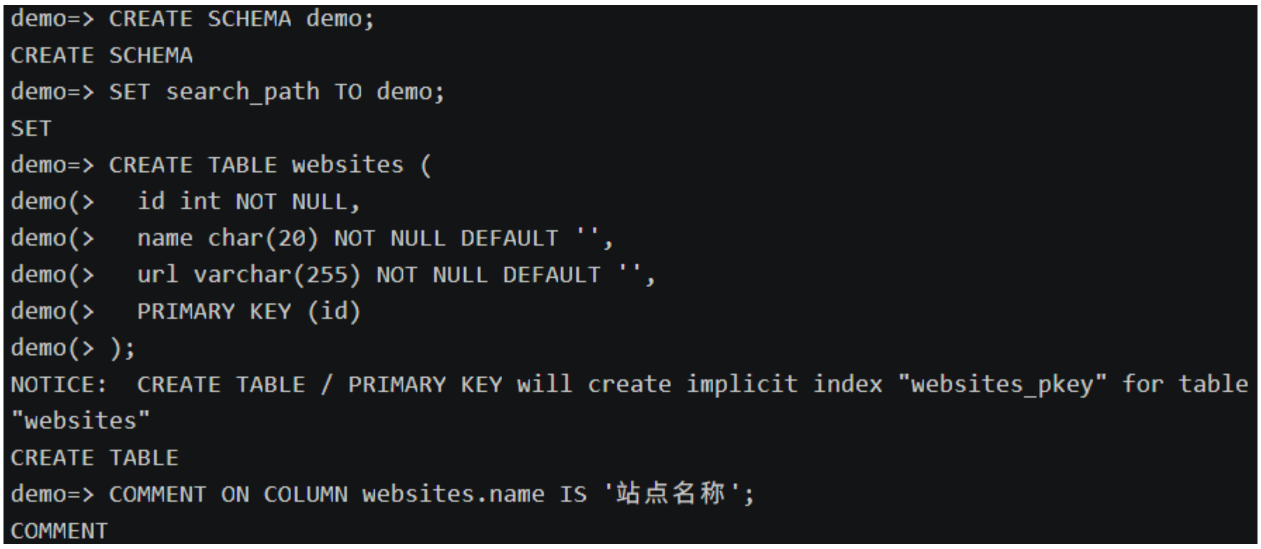
```
gsql -d postgres -p 26000 -U dbuser -r
CREATE DATABASE demo ENCODING ’UTF8’ template = template0;
\connect demo;
```



```
postgres=> CREATE DATABASE demo ENCODING 'UTF8' template = template0;
CREATE DATABASE
postgres=> \connect demo;
Password for user dbuser:
Non-SSL connection (SSL connection is recommended when requiring high-security)
You are now connected to database "demo" as user "dbuser".
```

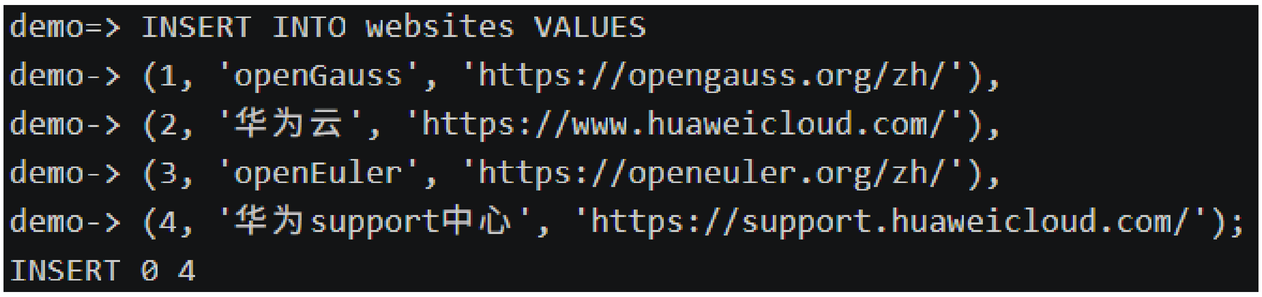
```
CREATE SCHEMA demo;
SET search_path TO demo;
```

```
CREATE TABLE websites (
    id INT NOT NULL,
    name CHAR(20) NOT NULL DEFAULT '',
    url VARCHAR(255) NOT NULL DEFAULT '',
    PRIMARY KEY (id)
);
COMMENT ON COLUMN websites.name IS '站点名称';
```



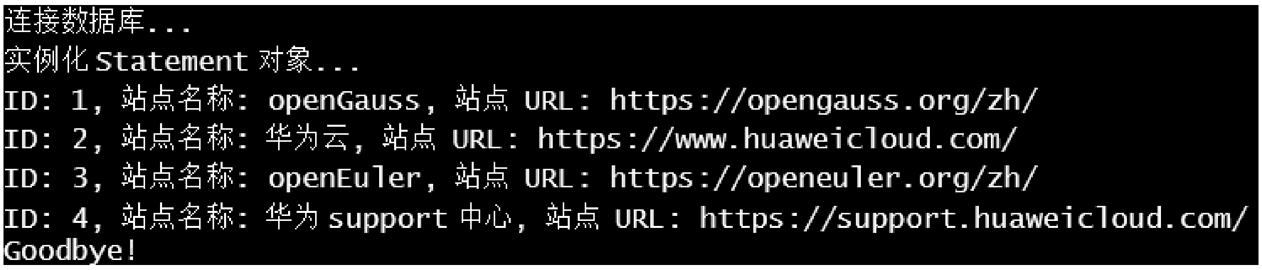
```
demo=> CREATE SCHEMA demo;
CREATE SCHEMA
demo=> SET search_path TO demo;
SET
demo=> CREATE TABLE websites (
demo(>   id int NOT NULL,
demo(>   name char(20) NOT NULL DEFAULT '',
demo(>   url varchar(255) NOT NULL DEFAULT '',
demo(>   PRIMARY KEY (id)
demo(> );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "websites_pkey" for table
"websites"
CREATE TABLE
demo=> COMMENT ON COLUMN websites.name IS '站点名称';
COMMENT
```

```
INSERT INTO websites VALUES
(1, 'openGauss', 'https://opengauss.org/zh/'),
(2, '华为云', 'https://www.huaweicloud.com/'),
(3, 'openEuler', 'https://openeuler.org/zh/'),
(4, '华为 support 中心', 'https://support.huaweicloud.com/');
```



```
demo=> INSERT INTO websites VALUES
demo-> (1, 'openGauss', 'https://opengauss.org/zh/'),
demo-> (2, '华为云', 'https://www.huaweicloud.com/'),
demo-> (3, 'openEuler', 'https://openeuler.org/zh/'),
demo-> (4, '华为support中心', 'https://support.huaweicloud.com/');
INSERT 0 4
```

十二、Java 连接



```
连接数据库...
实例化Statement 对象...
ID: 1, 站点名称: openGauss, 站点 URL: https://opengauss.org/zh/
ID: 2, 站点名称: 华为云, 站点 URL: https://www.huaweicloud.com/
ID: 3, 站点名称: openEuler, 站点 URL: https://openeuler.org/zh/
ID: 4, 站点名称: 华为 support 中心, 站点 URL: https://support.huaweicloud.com/
Goodbye!
```

三、心得总结（写出自己在完成实验过程中遇到的问题、解决方法，以及体会、收获等）（10%）

本次 openGauss 金融场景化综合实验全面模拟了银行核心业务的数据建模与操作流程，成功构建了涵盖客户、银行卡、理财产品、保险、基金及资产六大实体的金融数据库系统。通过完整的实验流程，系统掌握了 openGauss 数据库的创建与管理、表结构设计与约束定义、数据的增删改查、复杂查询语句编写、视图与索引优化、用户权限控制等核心技能，并最终通过 JDBC 实现了 Java 应用程序与数据库的远程连接与交互。实验不仅加深了对数据库理论知识的理解，更提升了在实际业务场景中运用 openGauss 解决复杂数据管理问题的实战能力，为今后从事数据库开发与运维工作奠定了扎实的基础。

一开始又忘记启动数据库服务了，导致停顿了一段时间，后来没遇到什么问题。