Q1

Banks

Primary key: {BankName, City}.
Both of them could be duplicated but combining these two attributes can make it unique.

```
        CREATE TABLE BANKS (
                BankName VARCHAR(20) NOT NULL,
                City VARCHAR(15) NOT NULL,
                NoAccounts INT DEFAULT 0 CONSTRAINT noAccCheck CHECK
(NoAccounts>=0),
                Security VARCHAR(10) CONSTRAINT securCheck CHECK (Security IN('weak',
'good', 'very good', 'excellent')),
                CONSTRAINT bpk PRIMARY KEY (BankName, City)
         );
```

Attribute constraints:

- NOT NULL for BankName and City because they are both primary keys.
- DEFAULT 0 to NoAccounts.
- CHECK (NoAccounts >= 0) a bank account can not less than 0.
- CHECK (Security IN(‘weak’, ‘good’, ‘very good’, ‘excellent’)) to make sure there are only these four categories in the Security attribute.

Robberies

Primary key: {BankName, City, Date}.
Because some banks may have been robbed more than once. Therefore it needs one more attribute Date to combine BankName and city to a group of primary keys to make it unique.

Foreign key: {BankName, City}.
BankName and City are connected to the Banks table to find a specific bank which has been robbed.

```
        CREATE TABLE ROBBERIES (
                BankName VARCHAR(20) NOT NULL,
                City VARCHAR(15) NOT NULL,
                Date DATE NOT NULL,
                Amount DECIMAL CHECK (Amount>0),
                CONSTRAINT rpk PRIMARY KEY (BankName, City, Date),
                CONSTRAINT rfk FOREIGN KEY(BankName, City) REFERENCES
BANKS(BankName, City) ON UPDATE CASCADE ON DELETE NO ACTION
         );
```

Actions:

- ON UPDATE CASCADE. If the Banks table updates new information it needs to update to the Robberies table as well.
- ON DELETE NO ACTION. If a bank is deleted from the Banks table then the record for robberies of this bank should still be kept in the Robberies table.

Attribute constraints:

- NOT NULL for BankName, City, Date because they are primary keys.
- CHECK (Amount > 0) money that was stolen by robber should not less than 1.

Plans

Primary key: {BankName, City, PlannedDate}.
Because that gang may plan to rob some banks more than once. Therefore it needs one more attribute PlanndDate to combine BankName and city to a group of primary keys to make it unique.

Foreign key: {BankName, City}.
BankName and City are connected to the Banks table to find a specific bank which was planned to rob by the gang.

```
        CREATE TABLE PLANS (
                BankName VARCHAR(20) NOT NULL,
                City VARCHAR(15) NOT NULL,
                PlannedDate DATE NOT NULL,
                NoRobbers INT CHECK (NoRobbers>0),
                CONSTRAINT ppk PRIMARY KEY (BankName, City, PlannedDate),
                CONSTRAINT pfk FOREIGN KEY(BankName, City) REFERENCES
BANKS(BankName, City) ON UPDATE CASCADE ON DELETE CASCADE
         );
```

Actions:

- ON UPDATE CASCADE. If the Banks table updates new information it needs to update to the Plans table as well.
- ON DELETE CASCADE. If a bank is deleted from the Banks table then the plan for robberies of this bank can be deleted as well. Because if the bank does not exist then it can not be robbed.

Attribute constraints:

- NOT NULL for BankName, City, PlannedDate because they are primary keys.
- CHECK (NoRobbers > 0) robbers have to not less than 1. Because 0 people can not rob a bank.

Robbers
Primary key: {RobberId}.
Because the Nickname may be duplicated. RobberId is the only unique identifier for a robber.

```
    CREATE TABLE ROBBERS (
        RobberId SERIAL NOT NULL,
        NickName VARCHAR(20),
        Age INT CONSTRAINT ageCheck CHECK (Age>0),
        NoYears INT CONSTRAINT noYearsCheck CHECK (NoYears <= Age AND
NoYears >= 0),
        CONSTRAINT robberspk PRIMARY KEY (RobberId)
     );
```

Attribute constraints:
- NOT NULL for RobberId because it is the primary key.
- CHECK (Age > 0) because age can not less than 0 if the person has been born.
- CHECK (NoYears <= Age AND NoYears > 0) because it is not possible to be in prison for more years than the robber has been alive and the year in prison can not be less than 0.

Skills
Primary key: {SkillId}.
SkillId is the unique identifier for skills.

```
    CREATE TABLE SKILLS (
        SkillId SERIAL NOT NULL,
        Description VARCHAR(30) UNIQUE,
        CONSTRAINT skillspk PRIMARY KEY (SkillId)
     );
```

HasSkills
Primary key: {RobberId, SkillId}.
Because one robber might have more than one skill. Therefore the combination of RobberId and SkillId can make it unique.

Foreign key: {RobberId, SkillId}.
RobberId is connected to the Robbers table to find a specific robber which has this skill. And SkillId is connected to the Skills table to find a specific skill.

```
    CREATE TABLE HASSKILLS (
        RobberId INT NOT NULL,
        SkillId INT NOT NULL,
        Preference INT CONSTRAINT preference CHECK (Preference BETWEEN 1
AND 3),
                Grade VARCHAR(2),
                CONSTRAINT hasspk PRIMARY KEY (RobberId, SkillId),
                CONSTRAINT hassrfk FOREIGN KEY(RobberId) REFERENCES
ROBBERS(RobberId) ON UPDATE CASCADE ON DELETE CASCADE,
        CONSTRAINT hassifk FOREIGN KEY(SkillId) REFERENCES SKILLS(SkillId) ON
UPDATE CASCADE ON DELETE CASCADE
         );
```

Actions:

- ON UPDATE CASCADE. If the information of a robber in the Robbers table is updated or the information of a skill in the Skills table is updated. Then it should update all these information in HasSkills table.
- ON DELETE CASCADE. If a robber is removed from the Robbers table or a skill is removed from the Skills table. Then the rows in HasSkills which contain these robberies or skills should also delete as well. Because this robber or skill does not exist anymore.

Attribute constraints:

- NOT NULL for RobberId and SkillId because they are the primary keys.
- CHECK (Preference BETWEEN 1 AND 3) as a robber has a preference between 1 and 3.

HasAccounts

Primary key: {RobberId, BankName, City}
Because one robber can have accounts in different banks and BankName and City are not unique. Therefore the combination of these three attributes can make them unique.

Foreign key: {RobberId, BankName, City}
RobberId is connected to the Robbers table to find a specific robber. BankName and City are connected to the Banks table to find a specific bank.

```
        CREATE TABLE HASACCOUNTS (
                RobberId INT NOT NULL,
                BankName VARCHAR(20) NOT NULL,
                City VARCHAR(15) NOT NULL,
                CONSTRAINT hasAccpk PRIMARY KEY (RobberId, BankName, City),
                CONSTRAINT hasAccRrfk FOREIGN KEY(RobberId) REFERENCES
ROBBERS(RobberId) ON UPDATE CASCADE ON DELETE CASCADE,
                CONSTRAINT hasAccBfk FOREIGN KEY(BankName, City) REFERENCES
        BANKS(BankName, City) ON UPDATE CASCADE ON DELETE CASCADE
         );
```

Actions:

- ON UPDATE CASCADE. If the RobberId of a robber is updated or the bank updates its name or moves to another city these information should be updated in HasAccounts table as well.
- ON DELETE CASCADE. If a robber is removed from the Robbers table then the bank account can not associate with a specific robber or if a bank is removed from the Banks table then the robber can not have an account in that bank anymore. Therefore these information needs to be deleted as well.

Attribute constraints:

- NOT NULL for RobberId, BankName and City because they are the primary keys.

Accomplices
Primary key: {RobberId, BankName, City, Date}
Because a robber might rob a bank multiple times and a robber might rob more than one bank. Therefore only the combination of RobberId, BankName, City and Date can make it unique.

Foreign key: {RobberId, BankName, City, Date}
RobberId is connected to the Robbers table to find a specific robber. BankName, City and Date are connected to the Robberies table to find the specific robbery.

```
        CREATE TABLE ACCOMPLICES (
                RobberId INT NOT NULL,
                BankName VARCHAR(20) NOT NULL,
                City VARCHAR(15) NOT NULL,
                Date Date NOT NULL,
                Share DECIMAL CONSTRAINT accShareCheck CHECK(Share >= 0),
                CONSTRAINT Accpk PRIMARY KEY (RobberId, BankName, City, Date),
                CONSTRAINT AccRfk FOREIGN KEY(RobberId) REFERENCES
ROBBERS(RobberId) ON UPDATE CASCADE ON DELETE NO ACTION,
                CONSTRAINT AccBfk FOREIGN KEY(BankName, City, Date) REFERENCES
        ROBBERIES(BankName, City, Date) ON UPDATE CASCADE ON DELETE NO ACTION
          );
```

Actions:

- ON UPDATE CASCADE. If a RobberId is changed in the Robbers table or a BankName or the City is changed in the Robberies table then they should update this information in the Accomplices table as well.
- ON DELETE NO ACTION. Because the Accomplices table is a record of which gang members participated in which robbery, and what share of the money they got. Therefore if there are some get changes in the robber or the bank will not affect this table.

Attribute constraints:

- NOT NULL for RobberId, BankName, City and Date because they are the primary keys.
- CHECK (Share >= 0) the money can not less than 0.

Q2

At first, I copy banks, robberies, plans and robbers to BANKS, ROBBERIES, PLANS and ROBBERS tables straightly.

1. \copy BANKS FROM C:\Users\wm088\Downloads\datafiles\banks_22.data

2. \copy ROBBERIES FROM C:\Users\wm088\Downloads\datafiles\robberies_22.data

3. \copy PLANS FROM C:\Users\wm088\Downloads\datafiles\plans_22.data

4. \copy ROBBERS(NickName,Age,NoYears) FROM C:\Users\wm088\Downloads\datafiles\robbers_22.data

Secondly, I create a TEMPSKILLS table to store hasskills data then insert the distinct Description from it to the SKILLS table.

5. SKILLS

```
        CREATE TABLE TEMPSKILLS (
                NickName VARCHAR(20),
                Description VARCHAR(20),
                Preference INT,
                Grade VARCHAR(2)
         );
\copy TEMPSKILLS FROM C:\Users\wm088\Downloads\datafiles\hasskills_22.data
INSERT INTO SKILLS (Description) SELECT  DISTINCT Description FROM
TEMPSKILLS;
```

Then I join the ROBBERS and the SKILLS tables by NickName and Description to find their RobberId and SkillId from TEMPSKILLS and insert into the HASSKILLS table.

6. HASSKILLS

```
INSERT INTO HASSKILLS (RobberId, SkillId, Preference, Grade)

SELECT ROBBERS.RobberId, SKILLS.SkillId, Preference, Grade

FROM TEMPSKILLS

INNER JOIN ROBBERS ON ROBBERS.NickName = TEMPSKILLS.NickName

INNER JOIN SKILLS ON SKILLS.Description = TEMPSKILLS.Description;
```

Next, I created a TEMPACCS table to store hasaccounts data. Then join the ROBBERS table by NickName from the TEMPACCS table to find the RobberId and insert them into the HASACCOUNTS table.

7. HASACCOUNTS

```
        CREATE TABLE TEMPACCS (

        NickName VARCHAR(20),
        BankName VARCHAR(20),
        City VARCHAR(20)
 );
        \copy TEMPACCS FROM C:\Users\wm088\Downloads\datafiles\hasaccounts_22.data

        INSERT INTO HASACCOUNTS (RobberId, BankName, City)

        SELECT ROBBERS.RobberId, BankName, City

        FROM TEMPACCS

        INNER JOIN ROBBERS ON ROBBERS.NickName = TEMPACCS.NickName;
```

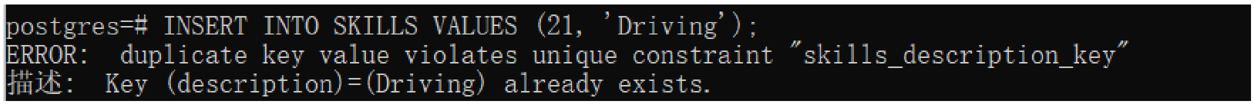
At last, I created a TEMPACOS table to store accomplices data. Then join the ROBBER table by NickName from the TEMPACOS table to find the RobberId and insert them into the ACCOMPLICES table.

8. ACCOMPLICES

```
        CREATE TABLE TEMPACOS (
                NickName VARCHAR(20),
                BankName VARCHAR(20),
                City VARCHAR(15),
                Date Date,
                Share DECIMAL
                  );
\copy TEMPACOS FROM C:\Users\wm088\Downloads\datafiles\accomplices_22.data

INSERT INTO ACCOMPLICES (RobberId, BankName, City, Date, Share)

SELECT ROBBERS.RobberId, BankName, City, Date, Share

FROM TEMPACOS

INNER JOIN ROBBERS ON ROBBERS.NickName = TEMPACOS.NickName;
```

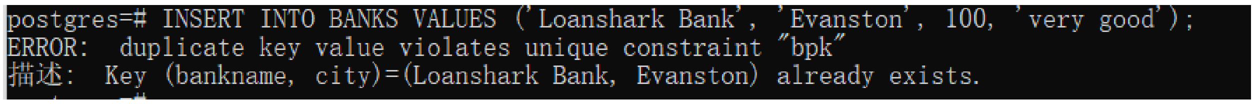
Q3

1. INSERT INTO SKILLS VALUES (21, 'Driving');



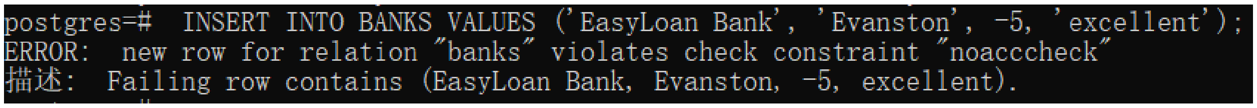
```
postgres=# INSERT INTO SKILLS VALUES (21, 'Driving');
ERROR:  duplicate key value violates unique constraint "skills_description_key"
描述:  Key (description)=(Driving) already exists.
```

2.
   a. INSERT INTO BANKS VALUES ('Loanshark Bank', 'Evanston', 100, 'very good');



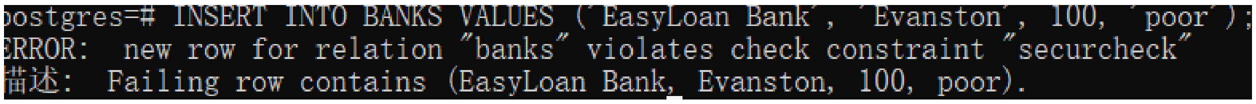
```
postgres=# INSERT INTO BANKS VALUES ('Loanshark Bank', 'Evanston', 100, 'very good');
ERROR:  duplicate key value violates unique constraint "bpk"
描述:  Key (bankname, city)=(Loanshark Bank, Evanston) already exists.
```

   b. INSERT INTO BANKS VALUES ('EasyLoan Bank', 'Evanston', -5, 'excellent');



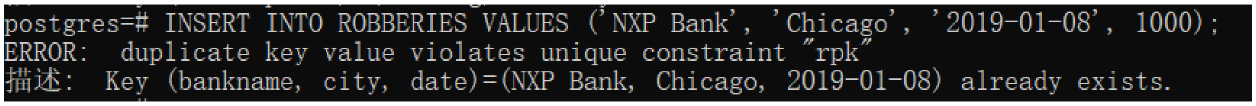
```
postgres=#  INSERT INTO BANKS VALUES ('EasyLoan Bank', 'Evanston', -5, 'excellent');
ERROR:  new row for relation "banks" violates check constraint "noacccheck"
描述:  Failing row contains (EasyLoan Bank, Evanston, -5, excellent).
```

   c. INSERT INTO BANKS VALUES ('EasyLoan Bank', 'Evanston', 100, 'poor');



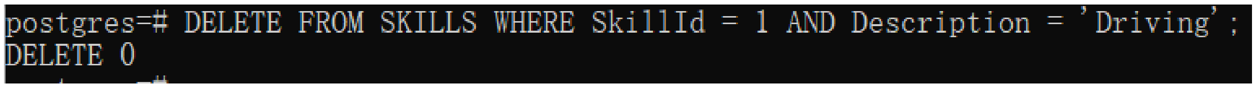
```
postgres=# INSERT INTO BANKS VALUES ('EasyLoan Bank', 'Evanston', 100, 'poor');
ERROR:  new row for relation "banks" violates check constraint "securcheck"
描述:  Failing row contains (EasyLoan Bank, Evanston, 100, poor).
```

3. INSERT INTO ROBBERIES VALUES ('NXP Bank', 'Chicago', '2019-01-08', 1000);



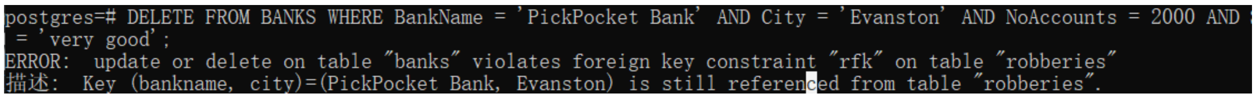
```
postgres=# INSERT INTO ROBBERIES VALUES ('NXP Bank', 'Chicago', '2019-01-08', 1000);
ERROR:  duplicate key value violates unique constraint "rpk"
描述:  Key (bankname, city, date)=(NXP Bank, Chicago, 2019-01-08) already exists.
```

4. DELETE FROM SKILLS WHERE SkillId = 1 AND Description = 'Driving';



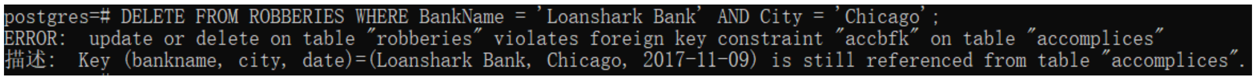
```
postgres=# DELETE FROM SKILLS WHERE SkillId = 1 AND Description = 'Driving';
DELETE 0
```

5. DELETE FROM BANKS WHERE BankName = 'PickPocket Bank' AND City = 'Evanston' AND NoAccounts = 2000 AND Security = 'very good';



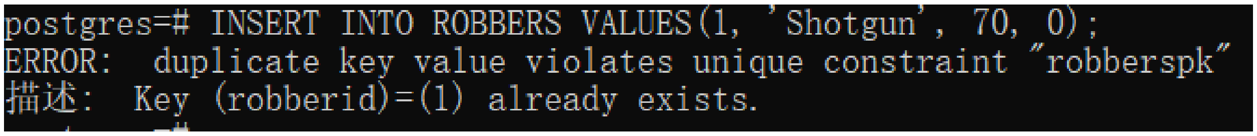
```
postgres=# DELETE FROM BANKS WHERE BankName = 'PickPocket Bank' AND City = 'Evanston' AND NoAccounts = 2000 AND
 = 'very good';
ERROR:  update or delete on table "banks" violates foreign key constraint "rfk" on table "robberies"
描述:  Key (bankname, city)=(PickPocket Bank, Evanston) is still referenced from table "robberies".
```

6. DELETE FROM ROBBERIES WHERE BankName = 'Loanshark Bank' AND City = 'Chicago';



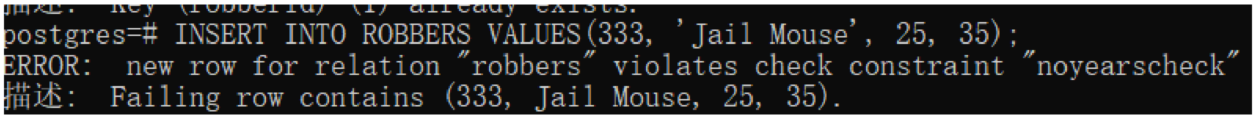
```
postgres=# DELETE FROM ROBBERIES WHERE BankName = 'Loanshark Bank' AND City = 'Chicago';
ERROR:  update or delete on table "robberies" violates foreign key constraint "accbfk" on table "accomplices"
描述:  Key (bankname, city, date)=(Loanshark Bank, Chicago, 2017-11-09) is still referenced from table "accomplices".
```

7.
   a. INSERT INTO ROBBERS VALUES(1, 'Shotgun', 70, 0);



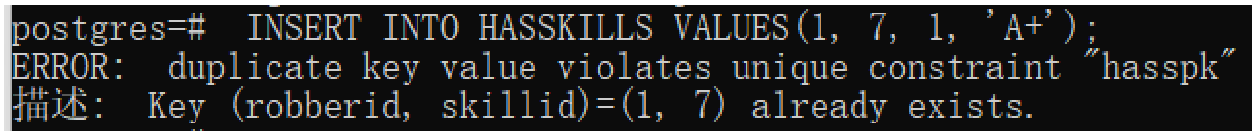
```
postgres=# INSERT INTO ROBBERS VALUES(1, 'Shotgun', 70, 0);
ERROR:  duplicate key value violates unique constraint "robberspk"
描述:  Key (robberid)=(1) already exists.
```

   b. INSERT INTO ROBBERS VALUES(333, 'Jail Mouse', 25, 35);



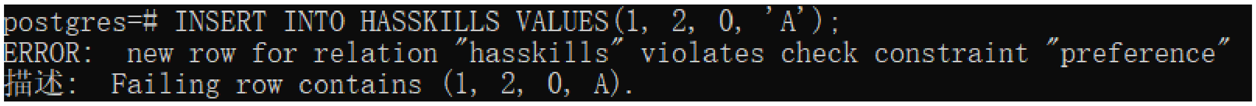
```
描述:  Key (robberid)=(1) already exists.
postgres=# INSERT INTO ROBBERS VALUES(333, 'Jail Mouse', 25, 35);
ERROR:  new row for relation "robbers" violates check constraint "noyearscheck"
描述:  Failing row contains (333, Jail Mouse, 25, 35).
```

8.
   a. INSERT INTO HASSKILLS VALUES(1, 7, 1, 'A+');



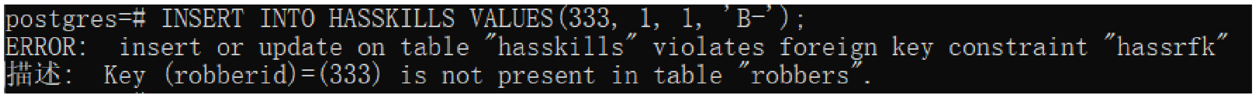
```
postgres=#  INSERT INTO HASSKILLS VALUES(1, 7, 1, 'A+');
ERROR:  duplicate key value violates unique constraint "hasspk"
描述:  Key (robberid, skillid)=(1, 7) already exists.
```

   b. INSERT INTO HASSKILLS VALUES(1, 2, 0, 'A');



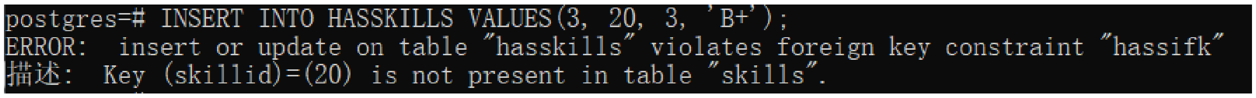
```
postgres=# INSERT INTO HASSKILLS VALUES(1, 2, 0, 'A');
ERROR:  new row for relation "hasskills" violates check constraint "preference"
描述:  Failing row contains (1, 2, 0, A).
```

   c. INSERT INTO HASSKILLS VALUES(333, 1, 1, 'B-');



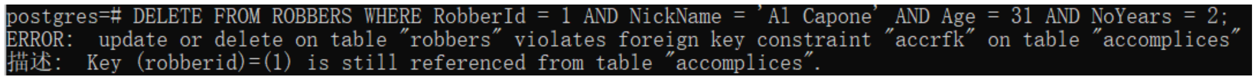
```
postgres=# INSERT INTO HASSKILLS VALUES(333, 1, 1, 'B-');
ERROR:  insert or update on table "hasskills" violates foreign key constraint "hassrfk"
描述:  Key (robberid)=(333) is not present in table "robbers".
```

   d. INSERT INTO HASSKILLS VALUES(3, 20, 3, 'B+');



```
postgres=# INSERT INTO HASSKILLS VALUES(3, 20, 3, 'B+');
ERROR:  insert or update on table "hasskills" violates foreign key constraint "hassifk"
描述:  Key (skillid)=(20) is not present in table "skills".
```

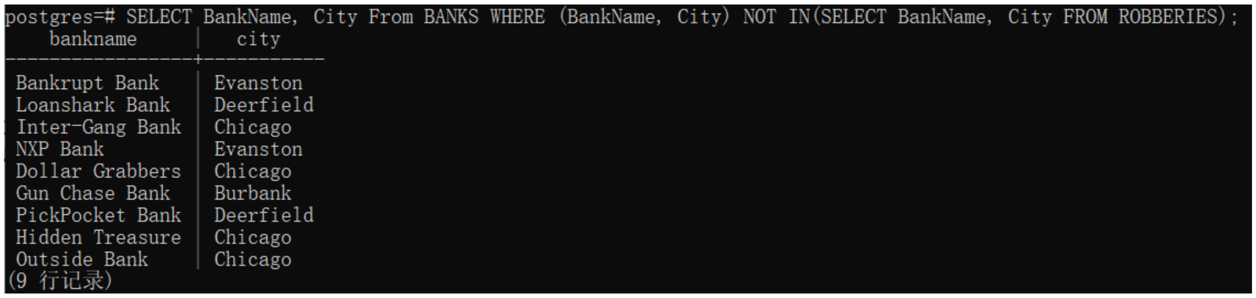
9. DELETE FROM ROBBERS WHERE RobberId = 1 AND NickName = 'Al Capone' AND Age = 31 AND NoYears = 2;



```
postgres=# DELETE FROM ROBBERS WHERE RobberId = 1 AND NickName = 'Al Capone' AND Age = 31 AND NoYears = 2;
ERROR:  update or delete on table "robbers" violates foreign key constraint "accrfk" on table "accomplices"
描述:  Key (robberid)=(1) is still referenced from table "accomplices".
```

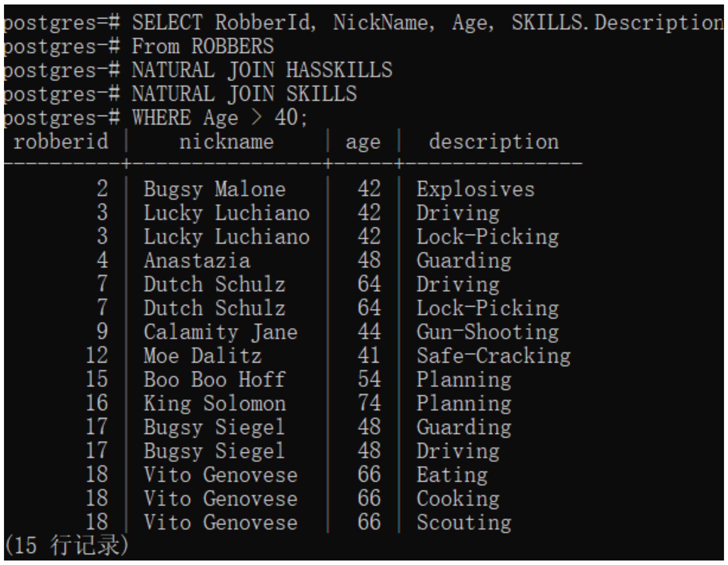
Q4

1. SELECT BankName, City From BANKS WHERE (BankName, City) NOT IN(SELECT BankName, City FROM ROBBERIES);



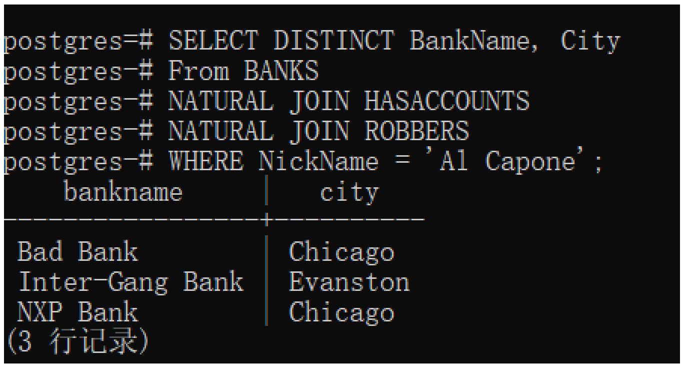
```
postgres=# SELECT BankName, City From BANKS WHERE (BankName, City) NOT IN(SELECT BankName, City FROM ROBBERIES);
    bankname     |   city
-----------------+-----------
 Bankrupt Bank   | Evanston
 Loanshark Bank  | Deerfield
 Inter-Gang Bank | Chicago
 NXP Bank        | Evanston
 Dollar Grabbers | Chicago
 Gun Chase Bank  | Burbank
 PickPocket Bank | Deerfield
 Hidden Treasure | Chicago
 Outside Bank    | Chicago
(9 行记录)
```

2. SELECT RobberId, NickName, Age, SKILLS.Description
   From ROBBERS
   NATURAL JOIN HASSKILLS
   NATURAL JOIN SKILLS
   WHERE Age > 40;



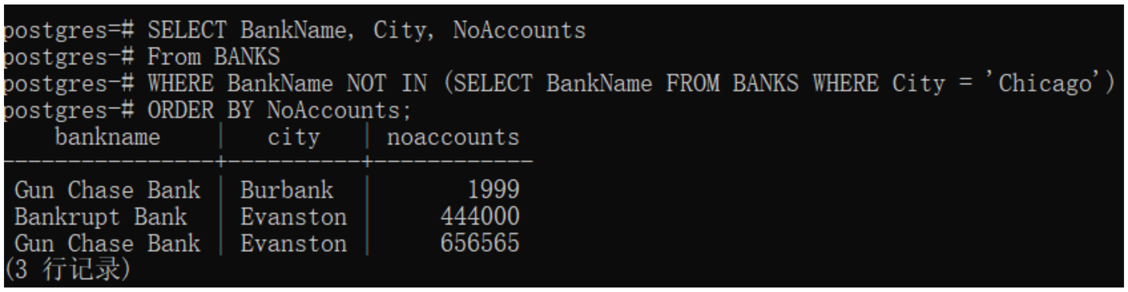
```
postgres=# SELECT RobberId, NickName, Age, SKILLS.Description
postgres-# From ROBBERS
postgres-# NATURAL JOIN HASSKILLS
postgres-# NATURAL JOIN SKILLS
postgres-# WHERE Age > 40;
 robberid |    nickname     | age |  description
----------+-----------------+-----+---------------
        2 | Bugsy Malone    |  42 | Explosives
        3 | Lucky Luchiano  |  42 | Driving
        3 | Lucky Luchiano  |  42 | Lock-Picking
        4 | Anastazia       |  48 | Guarding
        7 | Dutch Schulz    |  64 | Driving
        7 | Dutch Schulz    |  64 | Lock-Picking
        9 | Calamity Jane   |  44 | Gun-Shooting
       12 | Moe Dalitz      |  41 | Safe-Cracking
       15 | Boo Boo Hoff    |  54 | Planning
       16 | King Solomon    |  74 | Planning
       17 | Bugsy Siegel    |  48 | Guarding
       17 | Bugsy Siegel    |  48 | Driving
       18 | Vito Genovese   |  66 | Eating
       18 | Vito Genovese   |  66 | Cooking
       18 | Vito Genovese   |  66 | Scouting
(15 行记录)
```

3. SELECT DISTINCT BankName, City
   From BANKS
   NATURAL JOIN HASACCOUNTS
   NATURAL JOIN ROBBERS
   WHERE NickName = 'Al Capone';



```
postgres=# SELECT DISTINCT BankName, City
postgres-# From BANKS
postgres-# NATURAL JOIN HASACCOUNTS
postgres-# NATURAL JOIN ROBBERS
postgres-# WHERE NickName = 'Al Capone';
    bankname      |   city
------------------+----------
 Bad Bank         | Chicago
 Inter-Gang Bank  | Evanston
 NXP Bank         | Chicago
(3 行记录)
```

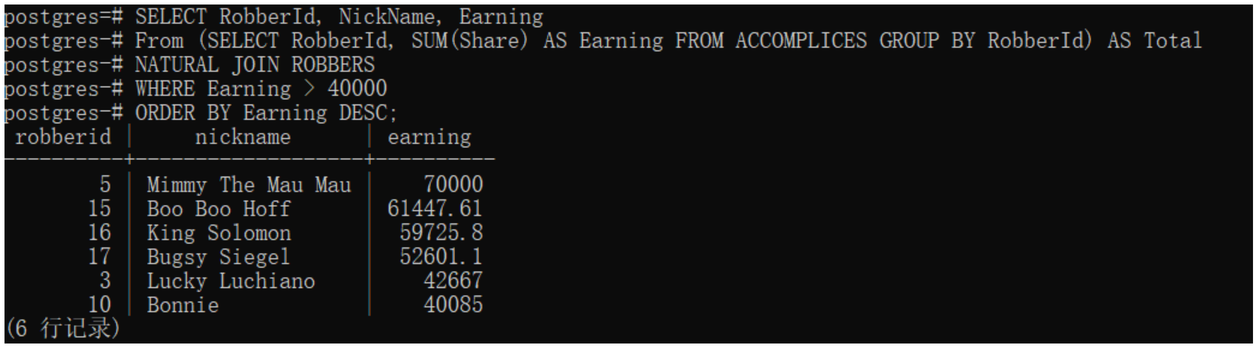
4. SELECT BankName, City, NoAccounts
   From BANKS
   WHERE BankName NOT IN (SELECT BankName FROM BANKS WHERE City = 'Chicago')
   ORDER BY NoAccounts;



```
postgres=# SELECT BankName, City, NoAccounts
postgres-# From BANKS
postgres-# WHERE BankName NOT IN (SELECT BankName FROM BANKS WHERE City = 'Chicago')
postgres-# ORDER BY NoAccounts;
    bankname     |   city   | noaccounts
-----------------+----------+------------
 Gun Chase Bank  | Burbank  |       1999
 Bankrupt Bank   | Evanston |     444000
 Gun Chase Bank  | Evanston |     656565
(3 行记录)
```

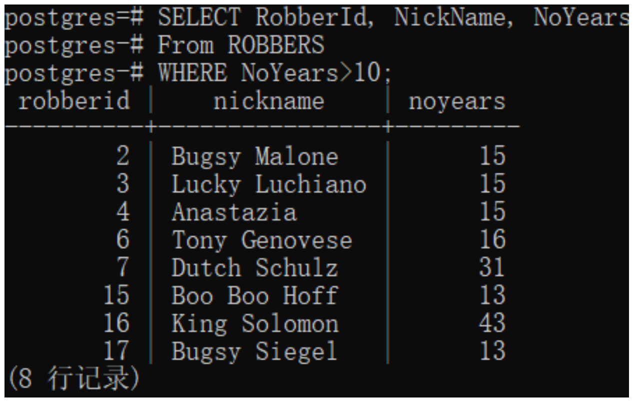
5. SELECT RobberId, NickName, Earning

From (SELECT RobberId, SUM(Share) AS Earning FROM ACCOMPLICES GROUP BY RobberId) AS Total
NATURAL JOIN ROBBERS
WHERE Earning > 40000
ORDER BY Earning DESC;



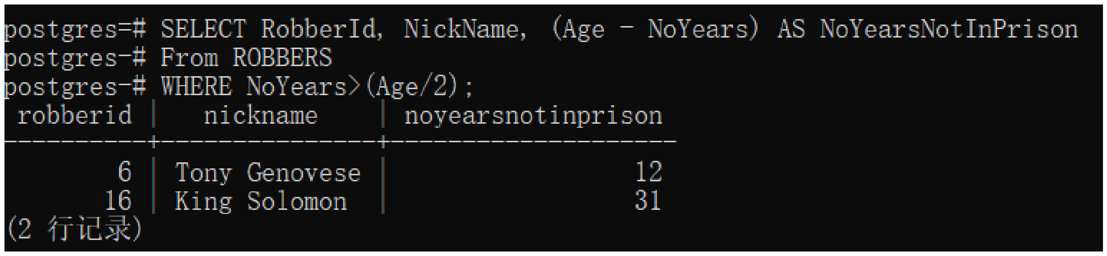
```
postgres=# SELECT RobberId, NickName, Earning
postgres-# From (SELECT RobberId, SUM(Share) AS Earning FROM ACCOMPLICES GROUP BY RobberId) AS Total
postgres-# NATURAL JOIN ROBBERS
postgres-# WHERE Earning > 40000
postgres-# ORDER BY Earning DESC;
 robberid |     nickname      | earning
----------+-------------------+----------
        5 | Mimmy The Mau Mau |    70000
       15 | Boo Boo Hoff      | 61447.61
       16 | King Solomon      |  59725.8
       17 | Bugsy Siegel      |  52601.1
        3 | Lucky Luchiano    |    42667
       10 | Bonnie            |    40085
(6 行记录)
```

6. SELECT RobberId, NickName, NoYears
   From ROBBERS
   WHERE NoYears>10;



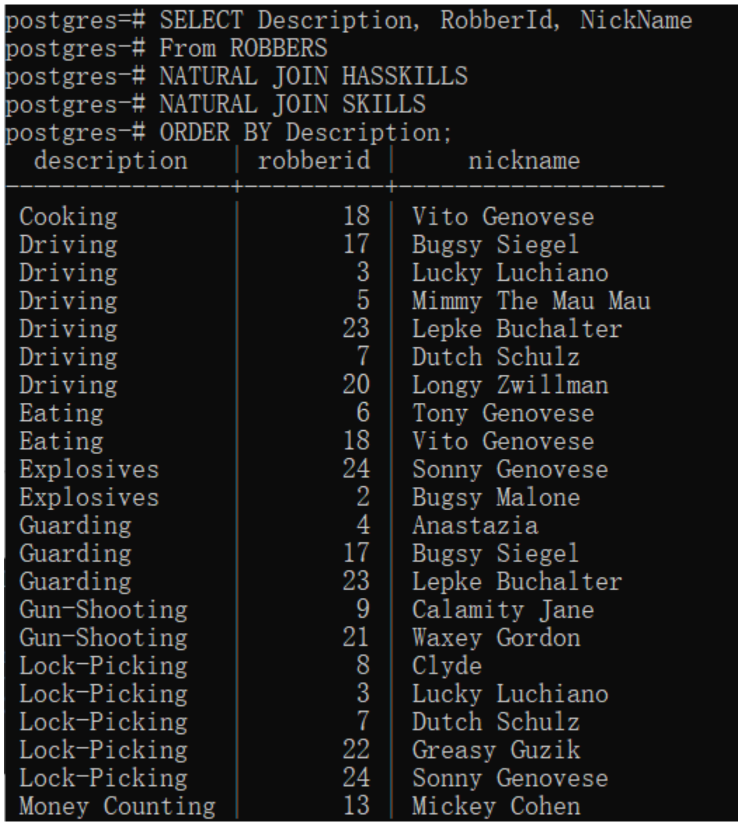
```
postgres=# SELECT RobberId, NickName, NoYears
postgres-# From ROBBERS
postgres-# WHERE NoYears>10;
 robberid |    nickname    | noyears
----------+----------------+---------
        2 | Bugsy Malone   |      15
        3 | Lucky Luchiano |      15
        4 | Anastazia      |      15
        6 | Tony Genovese  |      16
        7 | Dutch Schulz   |      31
       15 | Boo Boo Hoff   |      13
       16 | King Solomon   |      43
       17 | Bugsy Siegel   |      13
(8 行记录)
```

7. SELECT RobberId, NickName, (Age - NoYears) AS NoYearsNotInPrison
   From ROBBERS
   WHERE NoYears>(Age/2);



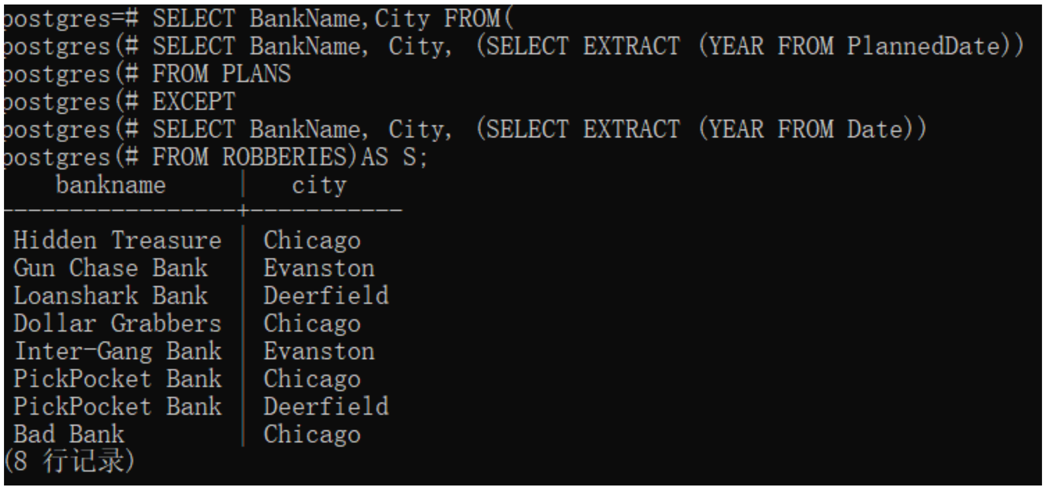
```
postgres=# SELECT RobberId, NickName, (Age - NoYears) AS NoYearsNotInPrison
postgres-# From ROBBERS
postgres-# WHERE NoYears>(Age/2);
 robberid |   nickname    | noyearsnotinprison
----------+---------------+--------------------
        6 | Tony Genovese |                 12
       16 | King Solomon  |                 31
(2 行记录)
```

8. SELECT Description, RobberId, NickName
   From ROBBERS
   NATURAL JOIN HASSKILLS
   NATURAL JOIN SKILLS
   ORDER BY Description;



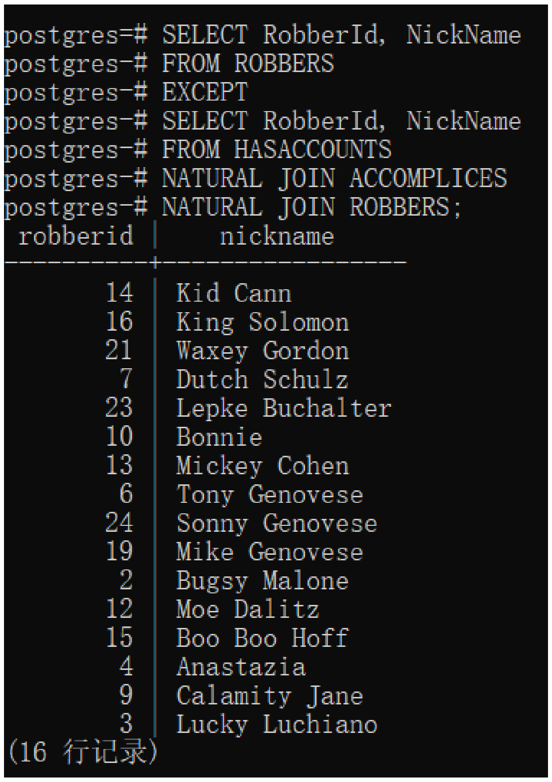
```
postgres=# SELECT Description, RobberId, NickName
postgres-# From ROBBERS
postgres-# NATURAL JOIN HASSKILLS
postgres-# NATURAL JOIN SKILLS
postgres-# ORDER BY Description;
  description    | robberid |      nickname
-----------------+----------+--------------------
 Cooking         |       18 | Vito Genovese
 Driving         |       17 | Bugsy Siegel
 Driving         |        3 | Lucky Luchiano
 Driving         |        5 | Mimmy The Mau Mau
 Driving         |       23 | Lepke Buchalter
 Driving         |        7 | Dutch Schulz
 Driving         |       20 | Longy Zwillman
 Eating          |        6 | Tony Genovese
 Eating          |       18 | Vito Genovese
 Explosives      |       24 | Sonny Genovese
 Explosives      |        2 | Bugsy Malone
 Guarding        |        4 | Anastazia
 Guarding        |       17 | Bugsy Siegel
 Guarding        |       23 | Lepke Buchalter
 Gun-Shooting    |        9 | Calamity Jane
 Gun-Shooting    |       21 | Waxey Gordon
 Lock-Picking    |        8 | Clyde
 Lock-Picking    |        3 | Lucky Luchiano
 Lock-Picking    |        7 | Dutch Schulz
 Lock-Picking    |       22 | Greasy Guzik
 Lock-Picking    |       24 | Sonny Genovese
 Money Counting  |       13 | Mickey Cohen
```

1. SELECT BankName,City FROM(
   SELECT BankName, City, (SELECT EXTRACT (YEAR FROM PlannedDate))
   FROM PLANS
   EXCEPT
   SELECT BankName, City, (SELECT EXTRACT (YEAR FROM Date))
   FROM ROBBERIES)AS EXC;



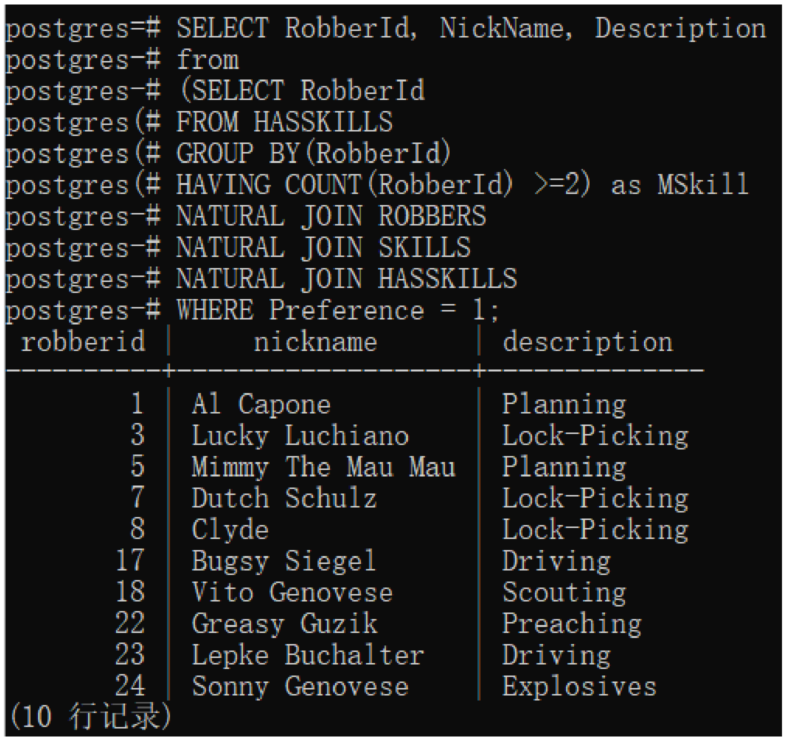
```
postgres=# SELECT BankName,City FROM(
postgres(# SELECT BankName, City, (SELECT EXTRACT (YEAR FROM PlannedDate))
postgres(# FROM PLANS
postgres(# EXCEPT
postgres(# SELECT BankName, City, (SELECT EXTRACT (YEAR FROM Date))
postgres(# FROM ROBBERIES)AS S;
    bankname      |    city
------------------+-----------
 Hidden Treasure  | Chicago
 Gun Chase Bank   | Evanston
 Loanshark Bank   | Deerfield
 Dollar Grabbers  | Chicago
 Inter-Gang Bank  | Evanston
 PickPocket Bank  | Chicago
 PickPocket Bank  | Deerfield
 Bad Bank         | Chicago
(8 行记录)
```

2. SELECT RobberId, NickName
   FROM ROBBERS
   EXCEPT
   SELECT RobberId, NickName
   FROM HASACCOUNTS
   NATURAL JOIN ACCOMPLICES
   NATURAL JOIN ROBBERS;



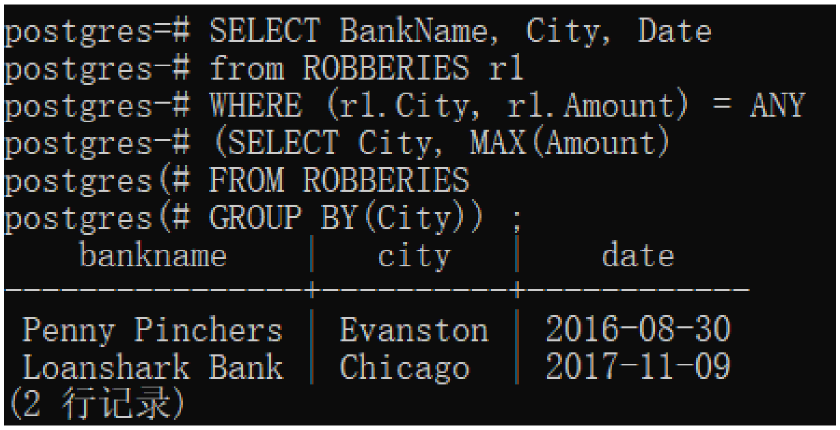
```
postgres=# SELECT RobberId, NickName
postgres-# FROM ROBBERS
postgres-# EXCEPT
postgres-# SELECT RobberId, NickName
postgres-# FROM HASACCOUNTS
postgres-# NATURAL JOIN ACCOMPLICES
postgres-# NATURAL JOIN ROBBERS;
 robberid |    nickname
----------+-----------------
       14 | Kid Cann
       16 | King Solomon
       21 | Waxey Gordon
        7 | Dutch Schulz
       23 | Lepke Buchalter
       10 | Bonnie
       13 | Mickey Cohen
        6 | Tony Genovese
       24 | Sonny Genovese
       19 | Mike Genovese
        2 | Bugsy Malone
       12 | Moe Dalitz
       15 | Boo Boo Hoff
        4 | Anastazia
        9 | Calamity Jane
        3 | Lucky Luchiano
(16 行记录)
```

3. SELECT RobberId, NickName, Description
   FROM
   (SELECT RobberId
   FROM HASSKILLS
   GROUP BY(RobberId)
   HAVING COUNT(RobberId) >=2) as MSkill
   NATURAL JOIN ROBBERS
   NATURAL JOIN SKILLS
   NATURAL JOIN HASSKILLS
   WHERE Preference = 1;



```
postgres=# SELECT RobberId, NickName, Description
postgres-# from
postgres-# (SELECT RobberId
postgres(# FROM HASSKILLS
postgres(# GROUP BY(RobberId)
postgres(# HAVING COUNT(RobberId) >=2) as MSkill
postgres-# NATURAL JOIN ROBBERS
postgres-# NATURAL JOIN SKILLS
postgres-# NATURAL JOIN HASSKILLS
postgres-# WHERE Preference = 1;
 robberid |     nickname       | description
----------+--------------------+--------------
        1 | Al Capone          | Planning
        3 | Lucky Luchiano     | Lock-Picking
        5 | Mimmy The Mau Mau  | Planning
        7 | Dutch Schulz       | Lock-Picking
        8 | Clyde              | Lock-Picking
       17 | Bugsy Siegel       | Driving
       18 | Vito Genovese      | Scouting
       22 | Greasy Guzik       | Preaching
       23 | Lepke Buchalter    | Driving
       24 | Sonny Genovese     | Explosives
(10 行记录)
```

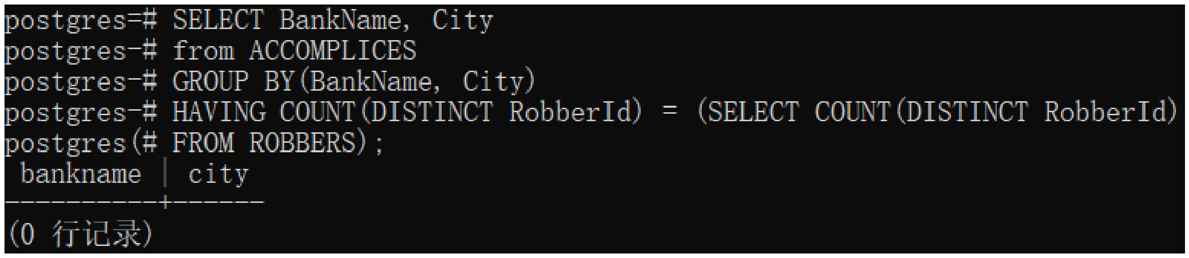
4. SELECT BankName, City, Date
   FROM ROBBERIES r1
   WHERE (r1.City, r1.Amount) = ANY
   (SELECT City, MAX(Amount)
   FROM ROBBERIES
   GROUP BY(City)) ;



```
postgres=# SELECT BankName, City, Date
postgres-# from ROBBERIES r1
postgres-# WHERE (r1.City, r1.Amount) = ANY
postgres-# (SELECT City, MAX(Amount)
postgres(# FROM ROBBERIES
postgres(# GROUP BY(City)) ;
    bankname     |   city    |    date
-----------------+-----------+------------
 Penny Pinchers  | Evanston  | 2016-08-30
 Loanshark Bank  | Chicago   | 2017-11-09
(2 行记录)
```

5. SELECT BankName, City
   FROM ACCOMPLICES
   GROUP BY(BankName, City)

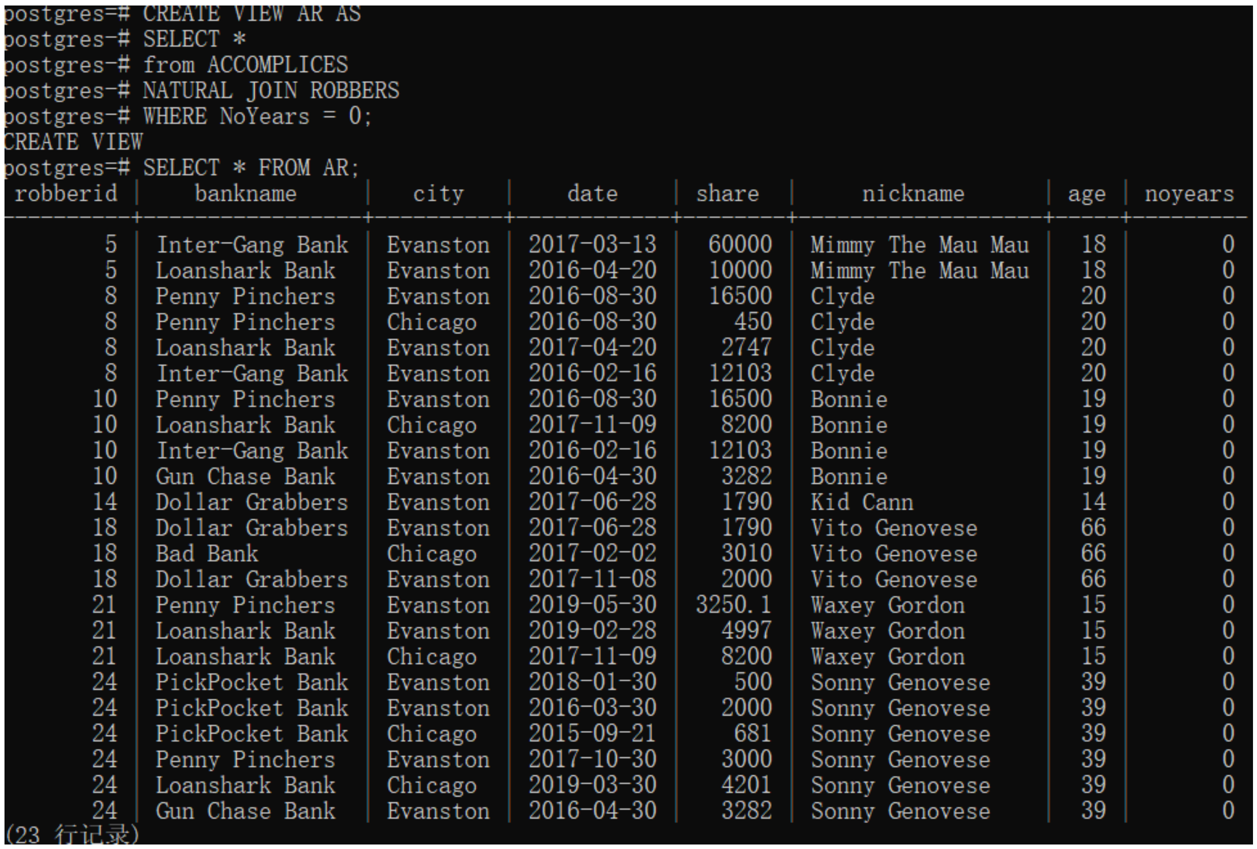
HAVING COUNT(DISTINCT RobberId) = (SELECT COUNT(DISTINCT RobberId)
FROM ROBBERS);



```
postgres=# SELECT BankName, City
postgres-# from ACCOMPLICES
postgres-# GROUP BY(BankName, City)
postgres-# HAVING COUNT(DISTINCT RobberId) = (SELECT COUNT(DISTINCT RobberId)
postgres(# FROM ROBBERS);
 bankname | city
----------+------
(0 行记录)
```

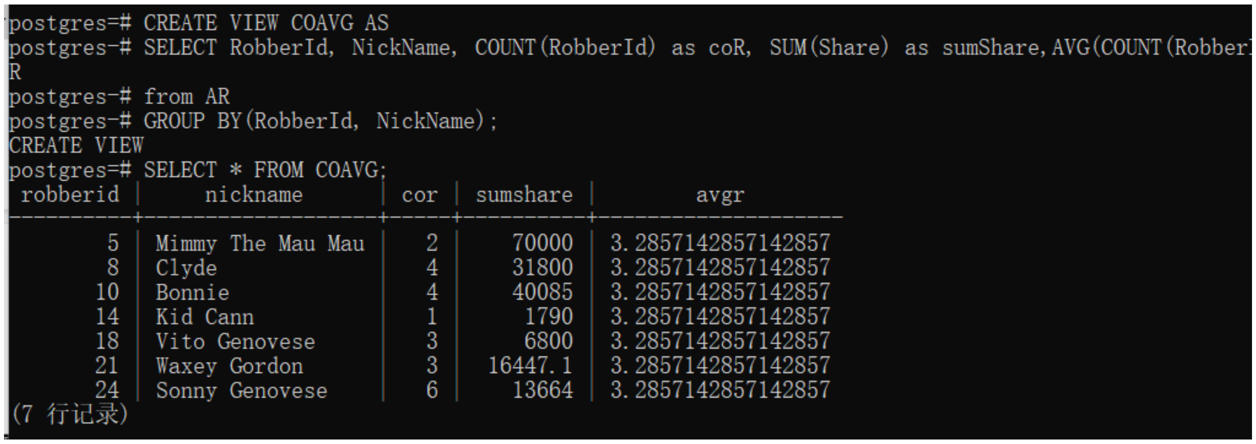
Q6

1. **Stepwise**
   CREATE VIEW AR AS
   SELECT *
   from ACCOMPLICES
   NATURAL JOIN ROBBERS
   WHERE NoYears = 0;



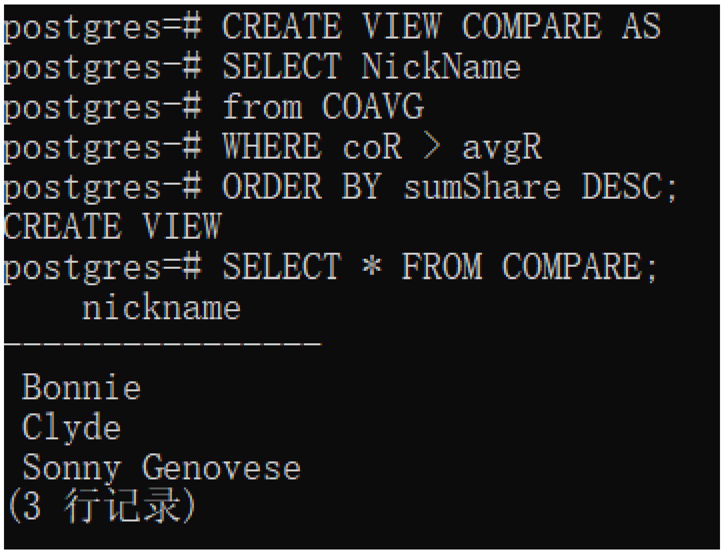
```
postgres=# CREATE VIEW AR AS
postgres-# SELECT *
postgres-# from ACCOMPLICES
postgres-# NATURAL JOIN ROBBERS
postgres-# WHERE NoYears = 0;
CREATE VIEW
postgres=# SELECT * FROM AR;
 robberid |    bankname     |   city   |    date    | share  |     nickname      | age | noyears
----------+-----------------+----------+------------+--------+-------------------+-----+---------
        5 | Inter-Gang Bank | Evanston | 2017-03-13 |  60000 | Mimmy The Mau Mau |  18 |       0
        5 | Loanshark Bank  | Evanston | 2016-04-20 |  10000 | Mimmy The Mau Mau |  18 |       0
        8 | Penny Pinchers  | Evanston | 2016-08-30 |  16500 | Clyde             |  20 |       0
        8 | Penny Pinchers  | Chicago  | 2016-08-30 |    450 | Clyde             |  20 |       0
        8 | Loanshark Bank  | Evanston | 2017-04-20 |   2747 | Clyde             |  20 |       0
        8 | Inter-Gang Bank | Evanston | 2016-02-16 |  12103 | Clyde             |  20 |       0
       10 | Penny Pinchers  | Evanston | 2016-08-30 |  16500 | Bonnie            |  19 |       0
       10 | Loanshark Bank  | Chicago  | 2017-11-09 |   8200 | Bonnie            |  19 |       0
       10 | Inter-Gang Bank | Evanston | 2016-02-16 |  12103 | Bonnie            |  19 |       0
       10 | Gun Chase Bank  | Evanston | 2016-04-30 |   3282 | Bonnie            |  19 |       0
       14 | Dollar Grabbers | Evanston | 2017-06-28 |   1790 | Kid Cann          |  14 |       0
       18 | Dollar Grabbers | Evanston | 2017-06-28 |   1790 | Vito Genovese     |  66 |       0
       18 | Bad Bank        | Chicago  | 2017-02-02 |   3010 | Vito Genovese     |  66 |       0
       18 | Dollar Grabbers | Evanston | 2017-11-08 |   2000 | Vito Genovese     |  66 |       0
       21 | Penny Pinchers  | Evanston | 2019-05-30 | 3250.1 | Waxey Gordon      |  15 |       0
       21 | Loanshark Bank  | Evanston | 2019-02-28 |   4997 | Waxey Gordon      |  15 |       0
       21 | Loanshark Bank  | Chicago  | 2017-11-09 |   8200 | Waxey Gordon      |  15 |       0
       24 | PickPocket Bank | Evanston | 2018-01-30 |    500 | Sonny Genovese    |  39 |       0
       24 | PickPocket Bank | Evanston | 2016-03-30 |   2000 | Sonny Genovese    |  39 |       0
       24 | PickPocket Bank | Chicago  | 2015-09-21 |    681 | Sonny Genovese    |  39 |       0
       24 | Penny Pinchers  | Evanston | 2017-10-30 |   3000 | Sonny Genovese    |  39 |       0
       24 | Loanshark Bank  | Chicago  | 2019-03-30 |   4201 | Sonny Genovese    |  39 |       0
       24 | Gun Chase Bank  | Evanston | 2016-04-30 |   3282 | Sonny Genovese    |  39 |       0
(23 行记录)
```

CREATE VIEW COAVG AS
SELECT RobberId, NickName, COUNT(RobberId) as coR, SUM(Share) as sumShare,AVG(COUNT(RobberId)) over () AS avgR
from AR
GROUP BY(RobberId, NickName);



```
postgres=# CREATE VIEW COAVG AS
postgres-# SELECT RobberId, NickName, COUNT(RobberId) as coR, SUM(Share) as sumShare,AVG(COUNT(Robber
R
postgres-# from AR
postgres-# GROUP BY(RobberId, NickName);
CREATE VIEW
postgres=# SELECT * FROM COAVG;
 robberid |      nickname       | cor | sumshare |        avgr
----------+---------------------+-----+----------+--------------------
        5 | Mimmy The Mau Mau   |   2 |    70000 | 3.2857142857142857
        8 | Clyde               |   4 |    31800 | 3.2857142857142857
       10 | Bonnie              |   4 |    40085 | 3.2857142857142857
       14 | Kid Cann            |   1 |     1790 | 3.2857142857142857
       18 | Vito Genovese       |   3 |     6800 | 3.2857142857142857
       21 | Waxey Gordon        |   3 |  16447.1 | 3.2857142857142857
       24 | Sonny Genovese      |   6 |    13664 | 3.2857142857142857
(7 行记录)
```

CREATE VIEW COMPARE AS
SELECT NickName
from COAVG
WHERE coR > avgR
ORDER BY sumShare DESC;

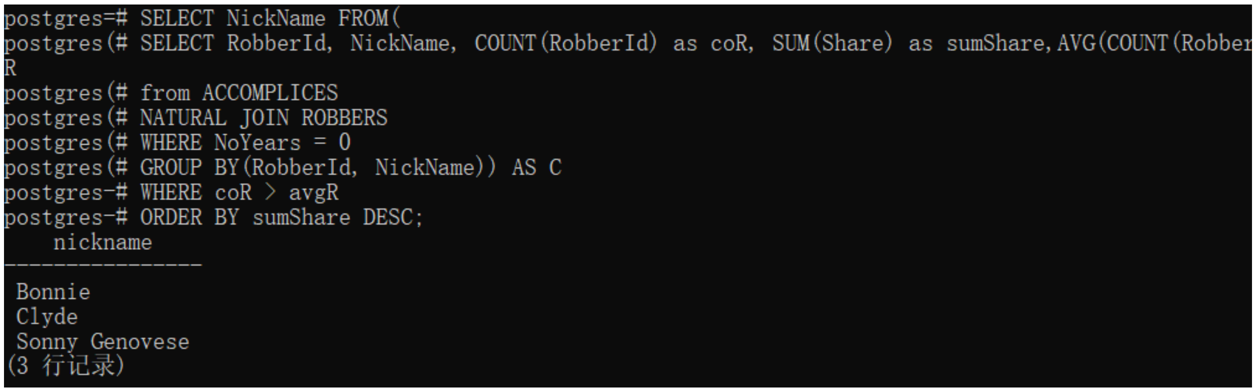


```
postgres=# CREATE VIEW COMPARE AS
postgres-# SELECT NickName
postgres-# from COAVG
postgres-# WHERE coR > avgR
postgres-# ORDER BY sumShare DESC;
CREATE VIEW
postgres=# SELECT * FROM COMPARE;
    nickname
----------------
 Bonnie
 Clyde
 Sonny Genovese
(3 行记录)
```

Single nested query

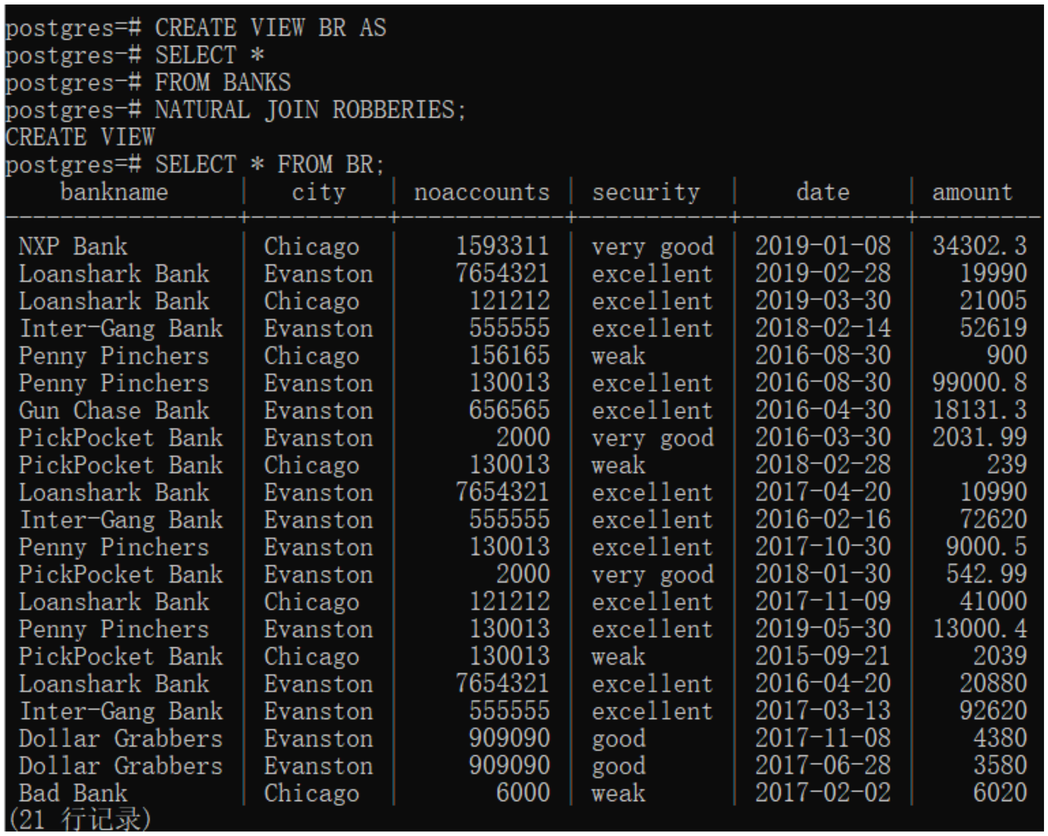
SELECT NickName FROM(
SELECT RobberId, NickName, COUNT(RobberId) as coR, SUM(Share) as sumShare,AVG(COUNT(RobberId)) over () AS avgR
from ACCOMPLICES

NATURAL JOIN ROBBERS
WHERE NoYears = 0
GROUP BY(RobberId, NickName)) AS C
WHERE coR > avgR
ORDER BY sumShare DESC;



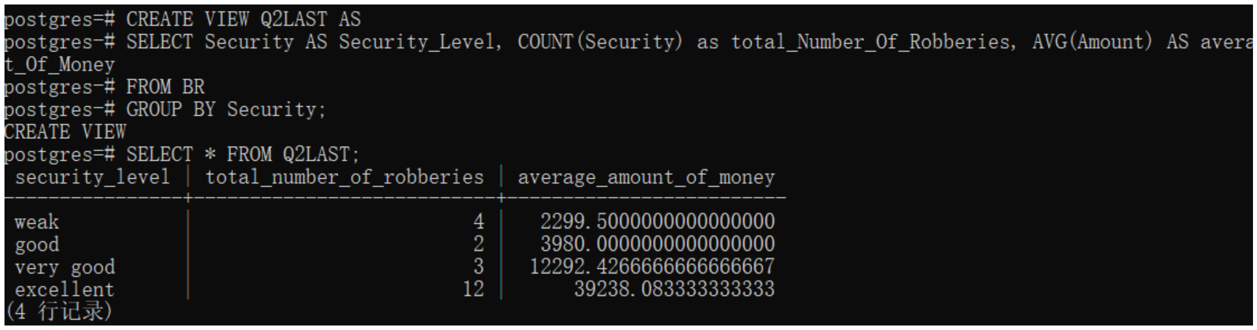
```
postgres=# SELECT NickName FROM(
postgres(# SELECT RobberId, NickName, COUNT(RobberId) as coR, SUM(Share) as sumShare,AVG(COUNT(Robber
R
postgres(# from ACCOMPLICES
postgres(# NATURAL JOIN ROBBERS
postgres(# WHERE NoYears = 0
postgres(# GROUP BY(RobberId, NickName)) AS C
postgres-# WHERE coR > avgR
postgres-# ORDER BY sumShare DESC;
    nickname
----------------
 Bonnie
 Clyde
 Sonny Genovese
(3 行记录)
```

2. **Stepwise**
   CREATE VIEW BR AS
   SELECT *
   FROM BANKS
   NATURAL JOIN ROBBERIES;



```
postgres=# CREATE VIEW BR AS
postgres-# SELECT *
postgres-# FROM BANKS
postgres-# NATURAL JOIN ROBBERIES;
CREATE VIEW
postgres=# SELECT * FROM BR;
    bankname     |   city   | noaccounts | security  |    date    | amount
-----------------+----------+------------+-----------+------------+---------
 NXP Bank        | Chicago  |    1593311 | very good | 2019-01-08 | 34302.3
 Loanshark Bank  | Evanston |    7654321 | excellent | 2019-02-28 |   19990
 Loanshark Bank  | Chicago  |     121212 | excellent | 2019-03-30 |   21005
 Inter-Gang Bank | Evanston |     555555 | excellent | 2018-02-14 |   52619
 Penny Pinchers  | Chicago  |     156165 | weak      | 2016-08-30 |     900
 Penny Pinchers  | Evanston |     130013 | excellent | 2016-08-30 | 99000.8
 Gun Chase Bank  | Evanston |     656565 | excellent | 2016-04-30 | 18131.3
 PickPocket Bank | Evanston |       2000 | very good | 2016-03-30 | 2031.99
 PickPocket Bank | Chicago  |     130013 | weak      | 2018-02-28 |     239
 Loanshark Bank  | Evanston |    7654321 | excellent | 2017-04-20 |   10990
 Inter-Gang Bank | Evanston |     555555 | excellent | 2016-02-16 |   72620
 Penny Pinchers  | Evanston |     130013 | excellent | 2017-10-30 |  9000.5
 PickPocket Bank | Evanston |       2000 | very good | 2018-01-30 |  542.99
 Loanshark Bank  | Chicago  |     121212 | excellent | 2017-11-09 |   41000
 Penny Pinchers  | Evanston |     130013 | excellent | 2019-05-30 | 13000.4
 PickPocket Bank | Chicago  |     130013 | weak      | 2015-09-21 |    2039
 Loanshark Bank  | Evanston |    7654321 | excellent | 2016-04-20 |   20880
 Inter-Gang Bank | Evanston |     555555 | excellent | 2017-03-13 |   92620
 Dollar Grabbers | Evanston |     909090 | good      | 2017-11-08 |    4380
 Dollar Grabbers | Evanston |     909090 | good      | 2017-06-28 |    3580
 Bad Bank        | Chicago  |       6000 | weak      | 2017-02-02 |    6020
(21 行记录)
```

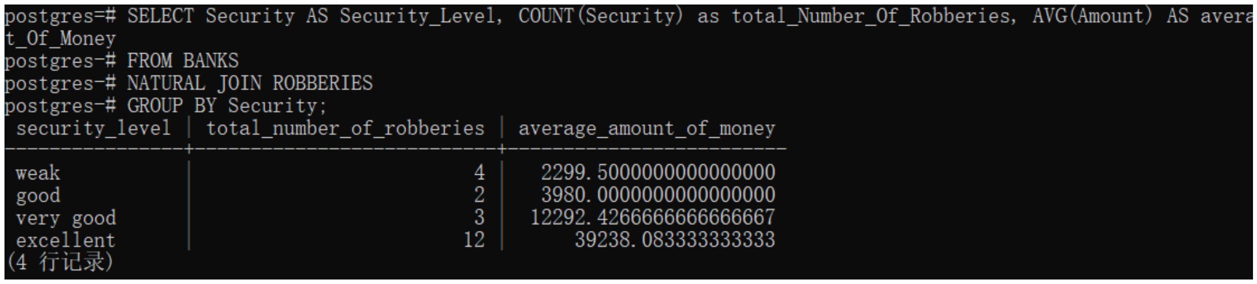
CREATE VIEW Q2LAST AS
SELECT Security AS Security_Level, COUNT(Security) as total_Number_Of_Robberies,
AVG(Amount) AS average_Amount_Of_Money
FROM BR
GROUP BY Security;



```
postgres=# CREATE VIEW Q2LAST AS
postgres-# SELECT Security AS Security_Level, COUNT(Security) as total_Number_Of_Robberies, AVG(Amount) AS avera
t_Of_Money
postgres-# FROM BR
postgres-# GROUP BY Security;
CREATE VIEW
postgres=# SELECT * FROM Q2LAST;
 security_level | total_number_of_robberies | average_amount_of_money
----------------+---------------------------+-------------------------
 weak           |                         4 |   2299.5000000000000000
 good           |                         2 |   3980.0000000000000000
 very good      |                         3 |  12292.4266666666666667
 excellent      |                        12 |      39238.083333333333
(4 行记录)
```

Single nested query

SELECT Security AS Security_Level, COUNT(Security) as total_Number_Of_Robberies,
AVG(Amount) AS average_Amount_Of_Money
FROM BANKS
NATURAL JOIN ROBBERIES
GROUP BY Security;



```
postgres=# SELECT Security AS Security_Level, COUNT(Security) as total_Number_Of_Robberies, AVG(Amount) AS avera
t_Of_Money
postgres-# FROM BANKS
postgres-# NATURAL JOIN ROBBERIES
postgres-# GROUP BY Security;
 security_level | total_number_of_robberies | average_amount_of_money
----------------+---------------------------+-------------------------
 weak           |                         4 |   2299.5000000000000000
 good           |                         2 |   3980.0000000000000000
 very good      |                         3 |  12292.4266666666666667
 excellent      |                        12 |      39238.083333333333
(4 行记录)
```