<u>**SWEN304 – PROJECT ONE**</u>
<u>**Lavanya Sajwan – 300381661**</u>

QUESTION ONE: Defining the Database

CREATE TABLE BANKS(
BankName VARCHAR(25) NOT NULL,
City VARCHAR(15) NOT NULL,
NoAccounts INT CONSTRAINT noAccountsChk CHECK (NoAccounts >= 0),
Security VARCHAR(10) DEFAULT 'weak' CONSTRAINT securityChk CHECK (Security IN('weak', 'good', 'very good', 'excellent')),
CONSTRAINT bpk PRIMARY KEY (BankName, City)
);

Primary Key: BankName, City
Both of these make up the primary key as there are multiple instances of both, but together they make a specific bank.

Attribute constraints:
- NOT NULL for BankName and City as they are primary keys
- CHECK (NoAccounts >=0) because a bank cannot have less than zero accounts.
- DEFAULT 'weak' for the Security with the assumption that each bank is categorised a 'weak' security and then is later specifically categorised.
- CHECK (Security IN('weak', 'good', 'very good', 'excellent') is there as the specific categories of security can only be set to those four.

CREATE TABLE ROBBERIES(
BankName VARCHAR(25) NOT NULL,
City VARCHAR(15) NOT NULL,
Date DATE NOT NULL,
Amount DECIMAL CONSTRAINT amountChk 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
);

Primary Key: BankName, City, Date
Together, these make up the primary key as there can be double ups on city, banks and dates that robberies have occurred, this is why all of them combined make it a unique identifier.

Foreign Key: BankName, City
A specific Bank in a City, needs to be specified to connect to the Bank table and if not it 'doesn't exist'.

Attribute constraints:
- NOT NULL BankName, City and Date as thy are primary keys

- CHECK (Amount >=0) as there has to be money stolen or an attempt of theft in order to be classified as a robbery.

Actions:

- ON UPDATE CASCADE = If Bank data is updated, then the robberies should also update in order to hold accurate information.
- ON DELETE NO ACTION = If Bank data is deleted for any instance (e.g. a bank being shut-down, so all data attributed to it is ignored) then the robberies table should remain the same as the robbery has till occurred in the past and still needs to be kept.

```
CREATE TABLE PLANS(
BankName VARCHAR(25) NOT NULL,
City VARCHAR(15) NOT NULL,
PlannedDate DATE NOT NULL,
NoRobbers INT CONSTRAINT NoRobbersChk CHECK (NoRobbers >=1),
CONSTRAINT ppk PRIMARY KEY ( BankName, City, PlannedDate),
CONSTRAINT pfk FOREIGN KEY (BankName, City) REFERENCES BANKS
(BankName, City) ON UPDATE CASCADE ON DELETE CASCADE
);
```

Primary Key: BankName, City, PlannedDate
Together, these make up the primary key as there can be double ups on city, banks and planned dates that robberies might occur at, this is why all of them combined make it a unique identifier.

Foreign Key: BankName, City
A specific Bank in a City, needs to be specified to connect to the Bank table and if not it 'doesn't exist'.

Attribute constraints:

- NOT NULL BankName, City and PlannedDate as thy are primary keys
- CHECK (NoRobbers >=1) as 0 people can't rob a bank.

Actions:

- ON UPDATE CASCADE = If Bank data is updated, then the robberies should also update in order to hold accurate information.
- ON DELETE CASCADE = If Bank data is deleted for any instance (e.g. a bank being shut-down, so all data attributed to it is ignored) then the possible plans to rob the place can be deleted as you cannot rob a bank that does not exist.

```
CREATE TABLE ROBBERS(
RobberId SERIAL UNIQUE,
Nickname VARCHAR(20),
Age INT CONSTRAINT robbersAge CHECK (Age > 0),
NoYears INT CONSTRAINT totalYears CHECK (Age >= NoYears),
CONSTRAINT rbpk PRIMARY KEY (RobberID)
);
CREATE SEQUENCE robbers_robberID;
ALTER SEQUENCE robbers_robberID restart with 1;
```

Primary Key: RobberID
RobberID the primary key as this is the unique identifier to differentiate between robbers as multiple robbers can share a nickname. It is set as unique as not one can be the same.

Attribute constraints:
- CHECK (Age>0) as you have to be born to rob a bank.
- CHECK (NoYears=<Age), can't be in jail longer than your age.

```
CREATE TABLE SKILLS(
SkillID SERIAL PRIMARY KEY,
Description VARCHAR(50) UNIQUE
);
CREATE SEQUENCE skills_skillsID;
ALTER SEQUENCE skills_skillsID restart with 1;
```

Primary Key: SkillID
SkillID is the primary key as this is the unique identifier of the skills displayed. It also has a data-type SERIAL in order to increment the ID's.

```
CREATE TABLE HASSKILLS(
RobberID INT NOT NULL,
SkillID INT NOT NULL,
Preference INT CONSTRAINT hasSkillsPreference CHECK (Preference>=1),
Grade VARCHAR(2) CONSTRAINT hasSkillsGrade CHECK (Grade IN('A+', 'A+ ',
'A', 'A ', 'B+', 'B+ ', 'B', 'B ','C+', 'C+ ', 'C', 'C ')),
CONSTRAINT hsrifk FOREIGN KEY (RobberID) REFERENCES ROBBERS
(RobberId) ON UPDATE CASCADE ON DELETE CASCADE,
 CONSTRAINT hssifk FOREIGN KEY (SkillID) REFERENCES SKILLS (SkillID) ON
UPDATE CASCADE ON DELETE CASCADE
);
```

Foreign Key: RobberID, SkillID
RobberID connects to the robbers table and SkillID connects to the Skills table. Otherwise both "don't exist".

Attribute constraints:

- NOT NULL RobberID and SkillID as they are needed to uniquely identify,
- CHECK (Preference >=1) as a robber cannot have a preference less than one.
- CHECK (Grade IN('A+', 'A+ ', 'A', 'A ', 'B+', 'B+ ', 'B', 'B ','C+', 'C+ ', 'C', 'C ')) as the grades are set to those and cannot deviate from them.

Actions:

- ON UPDATE CASCADE = If RobberID is updated, then the hasskills table will be updated to match this. If SkillsID is changed in anyway, than hasskills table will be updated to match this.
- ON DELETE CASCADE = If any robbers or skills are deleted, than the hasskills table containing the RobberID and SkillID will be deleted to math this as you can't have a skill without the specific idenfier of the said skill and the robber attributed to it.

CREATE TABLE HASACCOUNTS(
RobberID INT NOT NULL,
BankName VARCHAR(25) NOT NULL,
City VARCHAR(15) NOT NULL,
CONSTRAINT hafk FOREIGN KEY (RobberID) REFERENCES ROBBERS(RobberID) ON UPDATE CASCADE ON DELETE CASCADE,
CONSTRAINT habfk FOREIGN KEY (BankName, City) REFERENCES BANKS (BankName, City) ON UPDATE CASCADE ON DELETE CASCADE
);

Foreign Key: (RobberID), (BankName, City)
Both of these connect to associated tables such as Robbers and Banks.

Attribute Constraints:

- NOT NULL RobberID, BankName, City as they are needed to uniquely identify in the connected foreign tables.

Actions:

- ON UPDATE CASCADE = If any RobberID's are modified in any way in the Robber table, the has accounts table is updated to mirror this change. If any changes occur to the bankname and city, the has accounts table also changes them.
- ON DELETE CASCADE = If the RobbersID are deleted, then they are not in the robbers table and will consequently have no accounts associated with them and so should be deleted from this table. If the BankName and City are deleted that specific bank no longer exists and therefore can have no accounts.

CREATE TABLE ACCOMPLICES(
RobberID INT NOT NULL,
BankName VARCHAR(25) NOT NULL,
City VARCHAR(15) NOT NULL,
RobberyDate DATE NOT NULL,
Share REAL CONSTRAINT accomplicesShare CHECK (Share >= 0),
CONSTRAINT afk FOREIGN KEY (RobberID) REFERENCES ROBBERS (RobberID) ON UPDATE CASCADE ON DELETE NO ACTION,
CONSTRAINT abfk FOREIGN KEY (BankName, City, RobberyDate) REFERENCES ROBBERIES (BankName, City, Date) ON UPDATE CASCADE ON DELETE NO ACTION
);

Foreign Key: (RobberID), (BankName, City, RobberyDate)
Both of these connect to associated tables such as Robbers and Robberies.

Attribute Constraints:

- NOT NULL RobberID, BankName, City, RobberyDate as they are needed to uniquely identify in the connected foreign tables.
- CHECK (Share >=0) can't have a negative amount of money.

Actions:

- ON UPDATE CASCADE = If any RobberID's are modified in any way in the Robber table, the has accounts table is updated to mirror this change. If any changes occur to the bankname and city, the has accounts table also changes them.
- ON DELETE NO ACTION = If the RobbersID are deleted, then they are not in the robbers table, but they did previously rob so therefore, that is why there is no action at this delete. Also if all references to robberies is deleted as in any part of the foreign key, they still did occur and were accomplices hence the use of on delete no action.

QUESTION TWO: Populating your Database with Data

BANKS

\copy BANKS FROM ~/Desktop/pro1/banks_19.data

ROBBERIES

\copy ROBBERIES FROM ~/Desktop/pro1/robberies_19.data

PLANS

\copy PLANS FROM ~/Desktop/pro1/plans_19.data

ROBBERS

\copy ROBBERS(NickName, Age, NoYears) FROM ~/Desktop/pro1/robbers_19.data

SKILLS

CREATE TABLE TEMPSKILLS(NickName VARCHAR(20), Description VARCHAR(50), Preference INT, Grade VARCHAR(2));
\copy TEMPSKILLS FROM ~/Desktop/pro1/hasskills_19.data
insert INTO SKILLS(Description) (select distinct Description from TEMPSKILLS);

HASSKILLS

insert into HasSkills(select RobberId, SkillId, Preference, Grade FROM (Skills natural join TEMPSKILLS) natural join Robbers);

HASACCOUNTS

CREATE TABLE TEMPACCOUNTS(NickName VARCHAR(20), BankName VARCHAR(25), City VARCHAR(15));
\copy TEMPACCOUNTS FROM ~/Desktop/pro1/hasaccounts_19.data
insert INTO HASACCOUNTS(SELECT RobberId, BankName, City FROM (ROBBERS inner join TEMPACCOUNTS on TEMPACCOUNTS.NickName = ROBBERS.NickName));

ACCOMPLICES

```
CREATE TABLE TEMPACCOMPLICES(NickName VARCHAR(20), BankName VARCHAR(25), City VARCHAR(15), RobberyDate DATE, Share REAL);
\copy TEMPACCOMPLICES FROM ~/Desktop/pro1/accomplices_19.data
insert INTO ACCOMPLICES (SELECT RobberId, BankName, City, RobberyDate, Share FROM (ROBBERS inner join TEMPACCOMPLICES on TEMPACCOMPLICES.NickName = ROBBERS.NickName));
```

I essentially followed the steps outlined in question one in order to get where I did for this question. This involved some iteration going back and changing a few steps in order to get certain things working, such as swapping rows and changing amount datatype.

QUESTION THREE: Checking your Database

1a)

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

1b)

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

1c)

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

2a)

```
INSERT INTO SKILLS VALUES ('20', 'Guarding');
ERROR:  duplicate key value violates unique constraint "skills_description_key"
DETAIL:  Key (description)=(Guarding) already exists.
```

3a)

```
INSERT INTO ROBBERIES VALUES ('NXP BANK', 'Chicago', '2019-01-08', 1000);
ERROR:  insert or update on table "robberies" violates foreign key constraint "rfk"
DETAIL:  Key (bankname, city)=(NXP BANK, Chicago) is not present in table "banks".
```

4a)

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

4b)
DELETE FROM BANKS WHERE BankName = 'Outside Bank' AND City = 'Chicago' AND NoAccounts = 5000 AND Security = 'good';
DELETE 1
This is deleted as no other referenced table relies on ‘Outside Bank’

5a)
INSERT INTO ROBBERS VALUES (1, 'Shotgun', 70, 0);
ERROR: duplicate key value violates unique constraint "rbpk"
DETAIL: Key (robberid)=(1) already exists.

5b)
INSERT INTO ROBBERS VALUES (333, 'Jail Mouse', 25, 35);
ERROR: new row for relation "robbers" violates check constraint "totalyears"
DETAIL: Failing row contains (333, Jail Mouse, 25, 35).

6a)
INSERT INTO HASSKILLS VALUES (333, 1, 1, 'B-');
ERROR: new row for relation "hasskills" violates check constraint "hasskillsgrade"
DETAIL: Failing row contains (333, 1, 1, B-).

6b)
INSERT INTO HASSKILLS VALUES (3, 20, 3, 'B+');
ERROR: insert or update on table "hasskills" violates foreign key constraint "hssifk"
DETAIL: Key (skillid)=(20) is not present in table "skills".

6c)
INSERT INTO HASSKILLS VALUES (1, 7, 1, 'A+');
```
INSERT 0 1
```

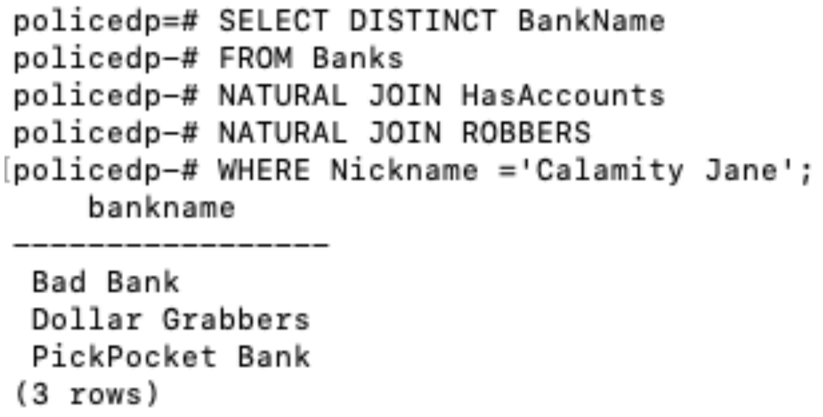
6d)
INSERT INTO HASSKILLS VALUES (1, 2, 0, 'A')
Worked?

7a)
DELETE FROM SKILLS WHERE SkillID = 1 AND Description = 'Driving';
DELETE 0
This is not deleted as the combination is no within the skills table. Driving is associated with another number.

8a)
DELETE FROM robbers WHERE RobberId = 1 AND NickName = 'Al Capone' AND Age = 21 AND NoYears = 2;
DELETE 0

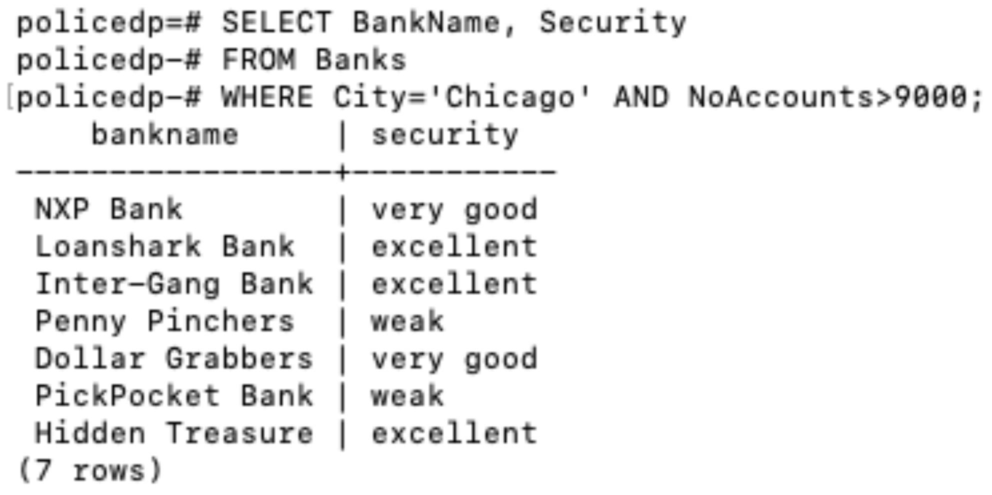
QUESTION FOUR: Simple Database Queries

1)



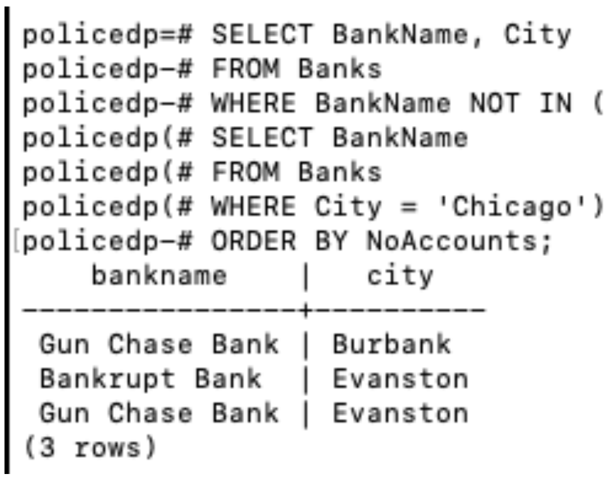
```
policedp=# SELECT DISTINCT BankName
policedp-# FROM Banks
policedp-# NATURAL JOIN HasAccounts
policedp-# NATURAL JOIN ROBBERS
[policedp-# WHERE Nickname ='Calamity Jane';
    bankname
-----------------
 Bad Bank
 Dollar Grabbers
 PickPocket Bank
(3 rows)
```

2)



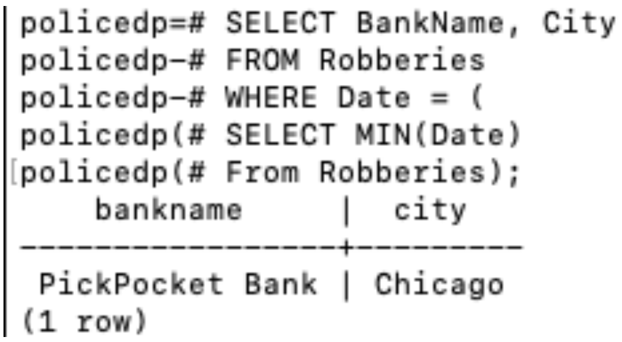
```
policedp=# SELECT BankName, Security
policedp-# FROM Banks
[policedp-# WHERE City='Chicago' AND NoAccounts>9000;
    bankname     | security
-----------------+-----------
 NXP Bank        | very good
 Loanshark Bank  | excellent
 Inter-Gang Bank | excellent
 Penny Pinchers  | weak
 Dollar Grabbers | very good
 PickPocket Bank | weak
 Hidden Treasure | excellent
(7 rows)
```

3)



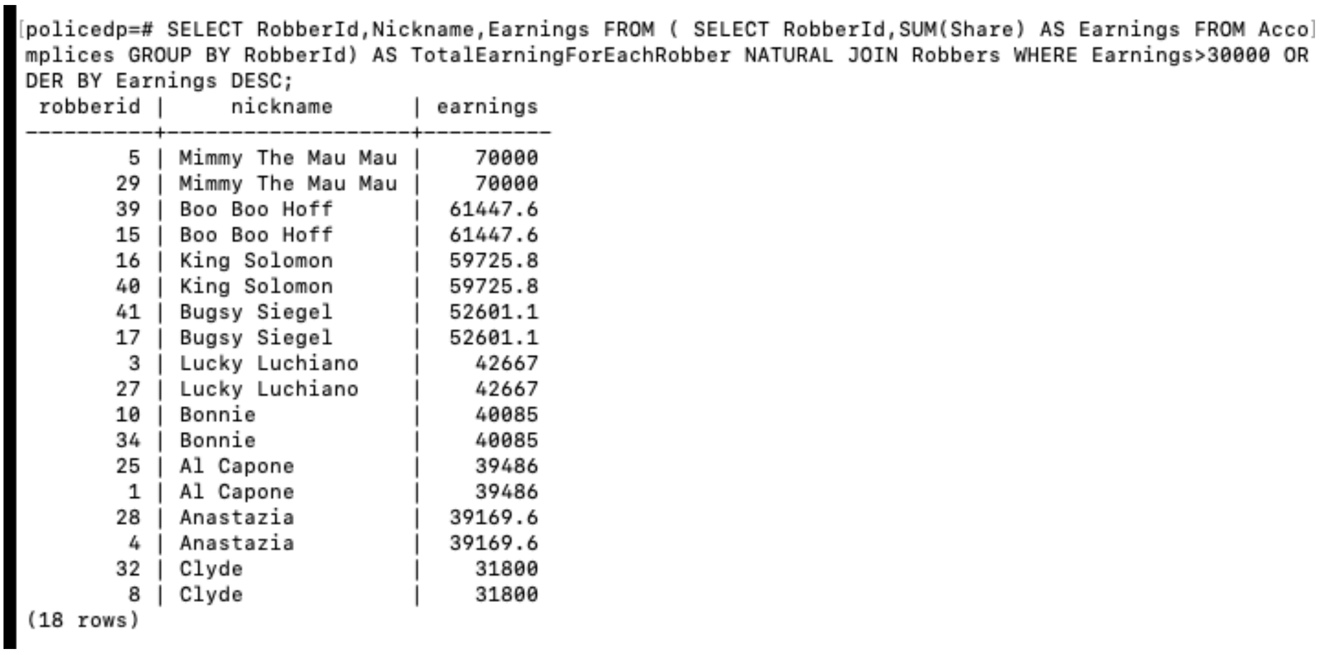
```
policedp=# SELECT BankName, City
policedp-# FROM Banks
policedp-# WHERE BankName NOT IN (
policedp(# SELECT BankName
policedp(# FROM Banks
policedp(# WHERE City = 'Chicago')
[policedp-# ORDER BY NoAccounts;
    bankname    |   city
----------------+----------
 Gun Chase Bank | Burbank
 Bankrupt Bank  | Evanston
 Gun Chase Bank | Evanston
(3 rows)
```

4)



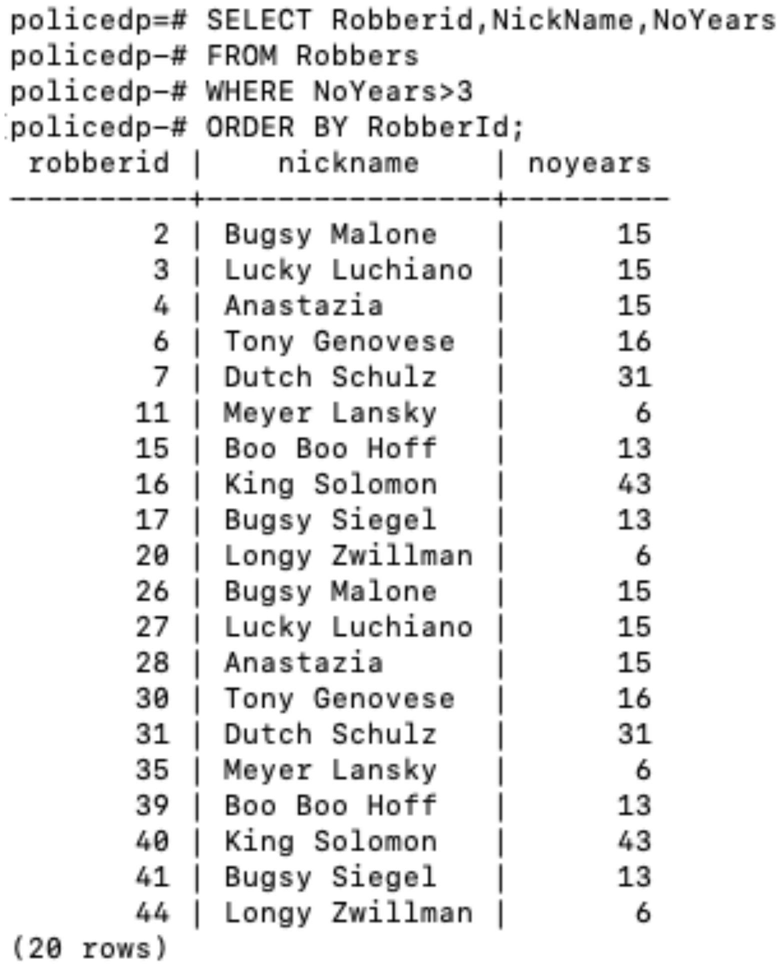
```
policedp=# SELECT BankName, City
policedp-# FROM Robberies
policedp-# WHERE Date = (
policedp(# SELECT MIN(Date)
[policedp(# From Robberies);
    bankname     |  city
-----------------+---------
 PickPocket Bank | Chicago
(1 row)
```

5)



```
[policedp=# SELECT RobberId,Nickname,Earnings FROM ( SELECT RobberId,SUM(Share) AS Earnings FROM Acco]
mplices GROUP BY RobberId) AS TotalEarningForEachRobber NATURAL JOIN Robbers WHERE Earnings>30000 OR
DER BY Earnings DESC;
 robberid |     nickname      | earnings
----------+-------------------+----------
        5 | Mimmy The Mau Mau |    70000
       29 | Mimmy The Mau Mau |    70000
       39 | Boo Boo Hoff      |  61447.6
       15 | Boo Boo Hoff      |  61447.6
       16 | King Solomon      |  59725.8
       40 | King Solomon      |  59725.8
       41 | Bugsy Siegel      |  52601.1
       17 | Bugsy Siegel      |  52601.1
        3 | Lucky Luchiano    |    42667
       27 | Lucky Luchiano    |    42667
       10 | Bonnie            |    40085
       34 | Bonnie            |    40085
       25 | Al Capone         |    39486
        1 | Al Capone         |    39486
       28 | Anastazia         |  39169.6
        4 | Anastazia         |  39169.6
       32 | Clyde             |    31800
        8 | Clyde             |    31800
(18 rows)
```

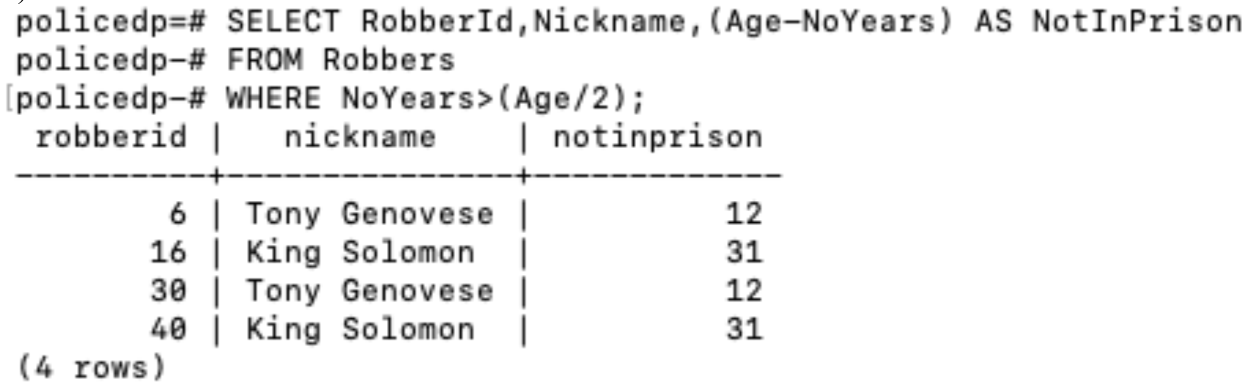
6)



```
policedp=# SELECT Robberid,NickName,NoYears
policedp-# FROM Robbers
policedp-# WHERE NoYears>3
policedp-# ORDER BY RobberId;
 robberid |    nickname    | noyears
----------+----------------+---------
        2 | Bugsy Malone   |      15
        3 | Lucky Luchiano |      15
        4 | Anastazia      |      15
        6 | Tony Genovese  |      16
        7 | Dutch Schulz   |      31
       11 | Meyer Lansky   |       6
       15 | Boo Boo Hoff   |      13
       16 | King Solomon   |      43
       17 | Bugsy Siegel   |      13
       20 | Longy Zwillman |       6
       26 | Bugsy Malone   |      15
       27 | Lucky Luchiano |      15
       28 | Anastazia      |      15
       30 | Tony Genovese  |      16
       31 | Dutch Schulz   |      31
       35 | Meyer Lansky   |       6
       39 | Boo Boo Hoff   |      13
       40 | King Solomon   |      43
       41 | Bugsy Siegel   |      13
       44 | Longy Zwillman |       6
(20 rows)
```

7)

8)

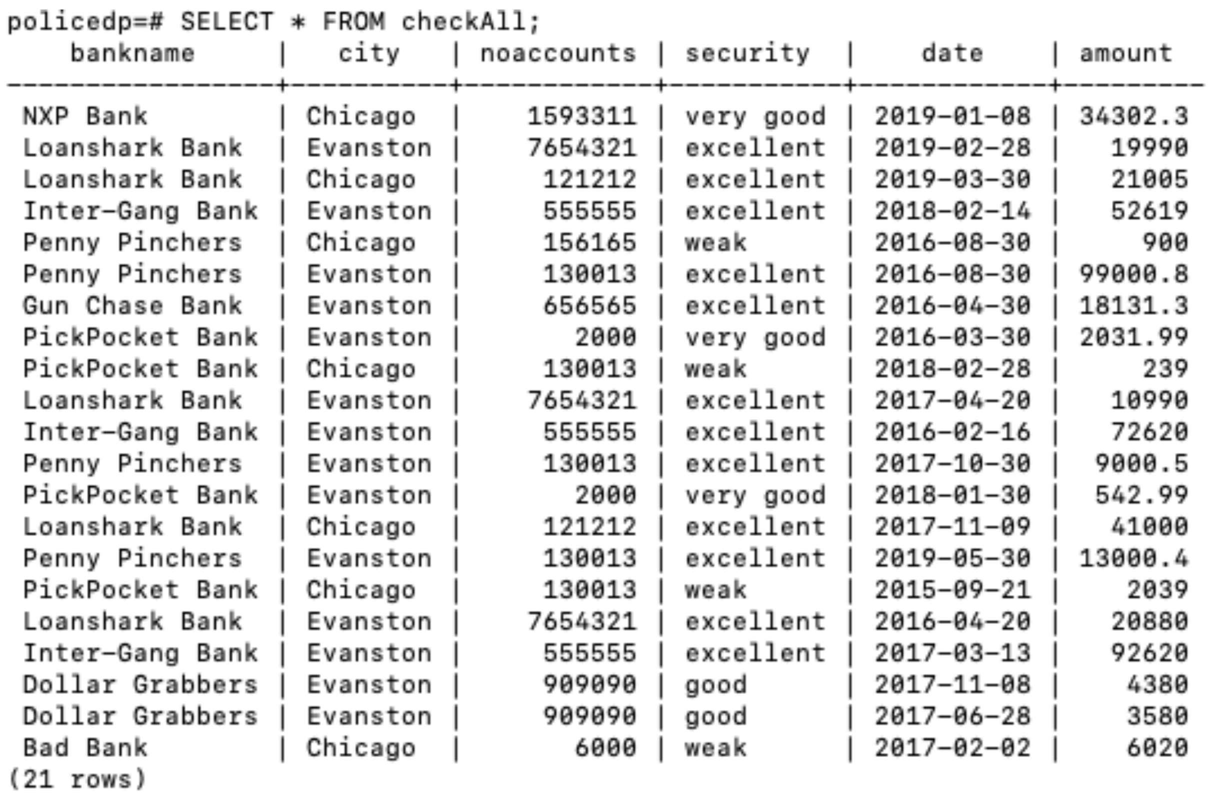


```
policedp=# SELECT RobberId,Nickname,(Age-NoYears) AS NotInPrison
policedp-# FROM Robbers
[policedp-# WHERE NoYears>(Age/2);
 robberid |    nickname    | notinprison
----------+----------------+-------------
        6 | Tony Genovese  |          12
       16 | King Solomon   |          31
       30 | Tony Genovese  |          12
       40 | King Solomon   |          31
(4 rows)
```

QUESTION FIVE: Complex Database Queries

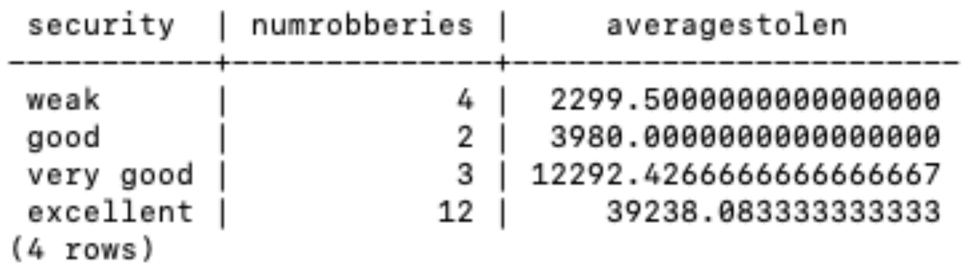
1. The police department wants to know whether bank branches with lower security levels are more attractive for robbers than those with higher security levels. Construct a view containing the Security level, the total Number of robberies that occurred in bank branches of that security level, and the average Amount of money that was stolen during these robberies.

Stepwise:
CREATE VIEW checkAll AS SELECT * FROM BANKS NATURAL JOIN ROBBERIES;
SELECT * FROM checkAll;



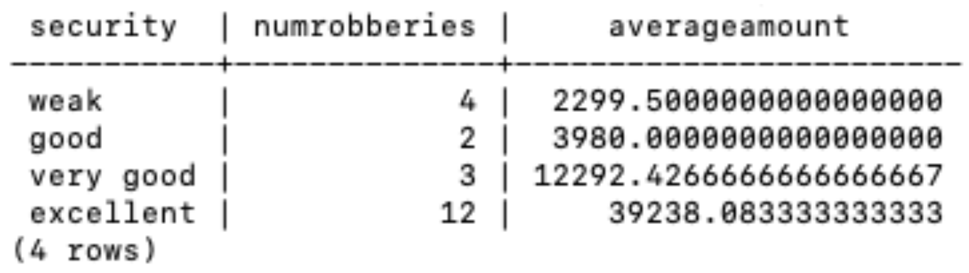
```
policedp=# SELECT * FROM checkAll;
    bankname     |   city   | noaccounts | security  |    date    | amount
-----------------+----------+------------+-----------+------------+---------
 NXP Bank        | Chicago  |    1593311 | very good | 2019-01-08 | 34302.3
 Loanshark Bank  | Evanston |    7654321 | 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 rows)
```

CREATE VIEW securityLevelTargeted AS SELECT Security, Count(Security) AS numRobberies, AVG(Amount) AS averageStolen FROM checkAll GROUP BY Security;
SELECT * FROM securityLevelTargeted;



```
 security  | numrobberies |      averagestolen
-----------+--------------+------------------------
 weak      |            4 |  2299.5000000000000000
 good      |            2 |  3980.0000000000000000
 very good |            3 | 12292.4266666666666667
 excellent |           12 |     39238.083333333333
(4 rows)
```

Single Nested Query:
SELECT Security, Count(Security) AS numRobberies, AVG(Amount) AS averageAmount FROM ROBBERIES NATURAL JOIN BANKS GROUP BY Security;

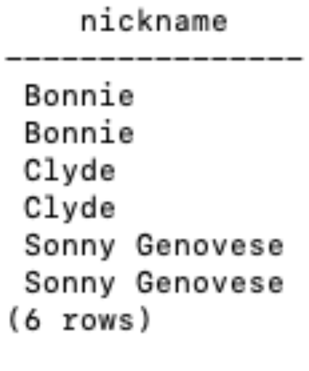


```
 security   | numrobberies |     averageamount
------------+--------------+------------------------
 weak       |            4 |  2299.5000000000000000
 good       |            2 |  3980.0000000000000000
 very good  |            3 | 12292.4266666666666667
 excellent  |           12 |     39238.083333333333
(4 rows)
```

2. The police department wants to know which robbers are most active, but were never penalised. Construct a view that contains the Nicknames of all robbers who participated in more robberies than the average, but spent no time in prison. The answer should be sorted in decreasing order of the individual total "earnings" of the robbers.

Stepwise:

Single Nested Query:
SELECT NickName FROM ROBBERS NATURAL JOIN ACCOMPLICES WHERE NoYears = 0 GROUP BY RobberId, NickName HAVING count(*) > (SELECT AVG(numRobberies) FROM (SELECT COUNT(*) AS numRobberies FROM ACCOMPLICES GROUP BY RobberId) AS R) ORDER BY SUM(Share) DESC;



```
     nickname
----------------
 Bonnie
 Bonnie
 Clyde
 Clyde
 Sonny Genovese
 Sonny Genovese
(6 rows)
```

3. The police department wants to increase security at those bank branches that are most likely to be victims in the near future. Construct a view containing the BankName, the City, and Security level of all bank branches that have not been robbed in 2018, but where plans for a robbery in 2020 are known. The answer should be sorted in decreasing order of the number of robbers who have accounts in that bank branch.

4. The police department wants to know which robbers are most likely to attack a particular bank branch. Robbing bank branches with a certain security level might require certain skills. For example, maybe every robbery of a branch with "excellent" security requires a robber with "Explosives" skill. Construct a view containing the Security level, the Skills (if any) that appear in every single bank robbery with that security level, and the Nicknames of all the robbers in the database who possess each of those skills.

5. The police department has a theory that bank robberies in Chicago are more profitable than in any other city. Construct a view that shows the average share of all robberies in Chicago, and the average share of all robberies in the city that observes the highest average share (other than Chicago). The average share of a robbery is computed based on the number of participants in that particular robbery.