SWEN304 Project 1
Rohan Gajadhar 300562245

Question 1)

Banks
Primary key: BankName, City
Constraints: NoAccounts > 0

BankName and City is a suitable primary key as it ensures uniqueness. The primary key could have just been BankName, but it is possible that there could be two or more Banks with the same name. Combining BankName and City will help ensure uniqueness as the local banking authority has ensured that a combination of name and city is unique.

The NoAccounts attribute must be greater than 0. As a bank with no accounts cannot be robbed and is not a functioning bank.

Robberies
Primary key: BankName, City, Date
Foreign key: BankName, City → Banks
Constraints: Amount > 0

I decided that BankName, City and Date is a suitable primary key. I found it necessary to combine the three keys as a primary key. The primary key could have just been BankName and Date. However, following on from the logic of my decision on the Banks primary key, it is possible that two or more banks with the same name could have been robbed on the same day. A combination of these three keys to form the primary key, ensures uniqueness.

The foreign keys are necessary to ensure the referenced bank is also in Banks. This is so that we cannot insert a robbery of a bank that is not included in our database (eg: it is in a different area not covered by Chicago)

The Amount attribute must be greater than 0. A robbery is only considered a robbery if money was stolen. It would be illogical to insert a value that is less than zero as that would mean the robbers gave money to the bank.

Plans
Primary key: BankName, City, PlannedDate
Foreign key: BankName, City → Banks
Constraints: NoRobbers > 0

Again, I found it necessary to specify BankName and City, as well as the PlannedDate to form the primary key. BankName and City must also be foreign keys to ensure that when they are

referenced in this table that it is a valid bank in Banks. This is so we cannot insert a plan to rob a bank that is not included in our database (eg: it is in a different area not covered by Chicago)

The NoRobbers attribute must be greater than 0. You cannot have negative (or 0) robbers.

Robbers
Primary key: RobberId
Constraints: Age > 0, NoYears < Age

As RobberId is unique to each robber, this makes it the ideal choice for the primary key.

A constraint is necessary to ensure that the number of years a robber is in prison does not exceed their age. Age must be greater than 0 and NoYears must be less than Age.

Skills
Primary key: SkillId
Constraints: Description is UNIQUE
NOT NULL: Description

As SkillId is unique to each skill, this makes it the ideal choice for the primary key.

Description cannot be null as it would render this table useless. SkillId is only an integer and does not contain any information about the details of the skill. Description must also be unique so we do not assign multiple SkillId's to the same Description. We also do not want duplicate skills.

HasSkills
Primary key: RobberId, SkillId
Foreign key: RobberId → Robbers, SkillId → Skills
Constraints: Preference > 0

A robber may have multiple skills, or a skill may be possessed by multiple robbers. Combining RobberId and SkillId ensures specificity and uniqueness.
RobberId and SkillId are foreign keys to ensure that when a RobberId is referenced in this table then it must be in Robbers. If SkillId is referenced this also must be in the Skills table. This is so we cannot insert data about a robber who is not in our database. This is also so we cannot apply a skill that is not in our database, to a robber.

Preference must be greater than 0 as a robber must have a preference.

HasAccounts
Primary key: RobberId, BankName, City
Foreign key: RobberId → Robbers, BankName, City → Banks

It is possible for a robber to have multiple bank accounts in different banks, therefore it is necessary for BankName and City to be combined as mentioned previously. We must also combine RobberId.
Foreign key ensures that RobberId is a robber in Robbers, and that BankName and City is a bank in Banks. We want to ensure that we cannot insert account data about a robber that is not in our database. We also want to ensure that we can only insert account information for a bank that is in our database.

Accomplices
Primary key: RobberId, BankName, City, Date
Foreign key: RobberId → Robbers, BankName, City, Date → Robberies

It is necessary for this combination of keys to form the primary key. BankName and City are needed to uniquely identify the bank, RobberId to identify the robber, and Date to specify the date on which the robbery took place. We could not have used Share as it is probable that the bank robbers all received an equal share. Including this in the primary key would not uniquely identify anything.
Foreign key ensures that RobberId is a robber in Robbers, BankName and City is a bank in Robberies, and that Date is a date in Robberies. This is so we cannot insert an accomplice who is not in the Robbers table. We also need BankName, City, and Date to match with a robbery in Robberies to ensure that we are not inserting a robbery for our accomplice that is outside of Chicago.

CREATE TABLE Statements

```
CREATE TABLE Banks (
    BankName VARCHAR(50) NOT NULL,
    City VARCHAR(50) NOT NULL,
    NoAccounts INT CHECK (NoAccounts > 0),
    Security VARCHAR(50) CHECK (Security IN ('excellent', 'very good',
'good', 'weak')),
    PRIMARY KEY (BankName, City)
);
```

```sql
CREATE TABLE Robberies (
    BankName VARCHAR(50) NOT NULL,
    City VARCHAR(50) NOT NULL,
    Date DATE NOT NULL,
    Amount DECIMAL (10, 2) CHECK (Amount > 0),
    PRIMARY KEY (BankName, City, Date),
    FOREIGN KEY (BankName, City) REFERENCES Banks (BankName, City)
);

CREATE TABLE Plans (
    BankName VARCHAR(50) NOT NULL,
    City VARCHAR(50) NOT NULL,
    NoRobbers INT CHECK (NoRobbers > 0),
    PlannedDate DATE NOT NULL,
    PRIMARY KEY (BankName, City, PlannedDate),
    FOREIGN KEY (BankName, City) REFERENCES Banks (BankName, City)
);

CREATE TABLE Robbers (
    RobberId SERIAL,
    Nickname VARCHAR(50),
    Age INT CHECK (Age > 0),
    NoYears INT CHECK (NoYears < Age),
    PRIMARY KEY (RobberId)
);

CREATE TABLE Skills (
    SkillId SERIAL,
    Description VARCHAR(50) NOT NULL
    PRIMARY KEY (SkillId)
    CONSTRAINT unique_Description UNIQUE (Description)
);
```

```sql
CREATE TABLE HasSkills (
    RobberId INT NOT NULL,
    SkillId INT NOT NULL,
    Preference INT CHECK (Preference > 0),
    GRADE VARCHAR(50),
    PRIMARY KEY (RobberId, SkillId),
    FOREIGN KEY (RobberId) REFERENCES Robbers (RobberId),
    FOREIGN KEY (SkillId) REFERENCES Skills (SkillId)
);

CREATE TABLE HasAccounts (
    RobberId INT NOT NULL,
    BankName VARCHAR(50) NOT NULL,
    City VARCHAR(50) NOT NULL,
    PRIMARY KEY (RobberId, BankName, City),
    FOREIGN KEY (RobberId) REFERENCES Robbers (RobberId),
    FOREIGN KEY (BankName, City) REFERENCES Banks (BankName, City)
);

CREATE TABLE Accomplices (
    RobberId INT NOT NULL,
    BankName VARCHAR(50) NOT NULL,
    City VARCHAR(50) NOT NULL,
    Date DATE NOT NULL,
    Share DECIMAL (10, 2),
    PRIMARY KEY (RobberId, BankName, City, Date),
    FOREIGN KEY (RobberId) REFERENCES Robbers (RobberId),
    FOREIGN KEY (BankName, City, Date) REFERENCES Robberies (BankName,
City, Date)
);
```

Question 2)

1) I

```
\copy Banks FROM
/am/kings/home1/gajadhroha/Documents/SWEN304/banks_23.data
\copy Robberies FROM
/am/kings/home1/gajadhroha/Documents/SWEN304/robberies_23.data
\copy Plans(BankName, City, PlannedDate, NoRobbers) FROM
/am/kings/home1/gajadhroha/Documents/SWEN304/plans_23.data
\copy Robbers(Nickname, Age, NoYears) FROM
/am/kings/home1/gajadhroha/Documents/SWEN304/robbers_23.data

-- Creating Skills table
CREATE TABLE defineSkills (
    defineNickname VARCHAR(50),
    defineDescription VARCHAR(50) NOT NULL,
    definePreference INT CHECK (definePreference > 0),
    defineGrade VARCHAR(50)
);

\copy defineSkills FROM
/am/kings/home1/gajadhroha/Documents/SWEN304/hasskills_23.data

INSERT INTO Skills (Description)
    SELECT DISTINCT defineDescription
    FROM defineSkills;

-- Creating HasSkills table
INSERT INTO HasSkills (RobberId, SkillId, Preference, Grade)
    SELECT Robbers.RobberId, Skills.SkillId, definePreference, defineGrade
    FROM defineSkills
    INNER JOIN Robbers ON Robbers.Nickname = defineNickname
    INNER JOIN Skills ON Skills.Description = defineDescription;

DROP TABLE defineSkills;
```

```
-- Creating HasAccounts and Accomplices tables
CREATE TABLE defineAccounts (
    defineNickname VARCHAR(50),
    defineBankName VARCHAR(50),
    defineCity VARCHAR(50)
);

\copy defineAccounts FROM
/am/kings/home1/gajadhroha/Documents/SWEN304/hasaccounts_23.data

INSERT INTO HasAccounts (RobberId, BankName, City)
    SELECT Robbers.RobberId, defineBankName, defineCity
    FROM defineAccounts
    INNER JOIN Robbers ON Robbers.Nickname = defineNickname;

DROP TABLE defineAccounts;

CREATE TABLE defineAccomplices (
    defineNickname VARCHAR(50),
    defineBankName VARCHAR(50),
    defineCity VARCHAR(50),
    defineDate DATE,
    defineShare DECIMAL (10, 2)
);

\copy defineAccomplices FROM
/am/kings/home1/gajadhroha/Documents/SWEN304/accomplices_23.data

INSERT INTO Accomplices (RobberId, BankName, City, Date, Share)
    SELECT Robbers.RobberId, defineBankName, defineCity, defineDate,
defineShare
    FROM defineAccomplices
    INNER JOIN Robbers ON Robbers.Nickname = defineNickname;

DROP TABLE defineAccomplices;
```

2) I implemented the Banks, Robberies, Plans and Robbers tables first. These were simple to implement as no conversions were necessary. Plans and Robbers required the order of attributes to be specified to match the data.

I then implemented the Skills table followed by the HasSkills table. HasSkills depends on SkillId and RobberId which is why I implemented those tables previously. As there was no Skills data file, I had to create a temporary table for Skills to collect the definition from the HasSkills data file. I then selected all the distinct definitions and inserted them into the Skills table. To populate the HasSkills table I performed an inner join on defineSkills, Robbers and Skills tables based on the matching values for Nicknames and Descriptions. This allows for HasSkills to be populated with data from all three tables where Nickname and Description match, and are consistent through each table.

Finally I implemented the HasAccounts and Accomplices tables. Almost all of their attributes depend on tables defined previously (Banks, Robbers, Robberies), so I decided to implement them last. To populate HasAccounts I created a temporary table to hold the raw data. I then performed a similar inner join as before, to ensure that RobberId and the Nicknames matched. This allowed for HasAccounts to be populated with RobberId instead of Nickname. Finally, an almost identical approach had to be followed for the Accomplices table. With the temporary table created to hold the raw data, I then performed an inner join on Robbers. This allowed for the accomplices table to be populated with the RobberId instead of Nickname.

Question 3)

1)

a) INSERT INTO Skills (SkillId, Description) VALUES (21, ‘Driving’);

ERROR: duplicate key value violates unique constraint "unique_description"
DETAIL: Key (description)=(Driving) already exists.

The insertion violates the UNIQUE constraint for the Description attribute. This means there is already another Description attribute with the value of ‘Driving’.

2)

a) INSERT INTO Banks (BankName, City, NoAccounts, Security) VALUES ('Loanshark Bank', 'Evanston', 100, 'very good');

ERROR: duplicate key value violates unique constraint "banks_pkey"
DETAIL: Key (bankname, city)=(Loanshark Bank, Evanston) already exists.

The insertion violates the unique constraint for BankName and City. As these attributes are primary keys they must be unique. BankName and City already contain values for Loanshark Bank and Evanston.

b) INSERT INTO Banks (BankName, City, NoAccounts, Security) VALUES (‘EasyLoan Bank’, ‘Evanston’, -5, ‘excellent’);

ERROR: new row for relation "banks" violates check constraint "banks_noaccounts_check"
DETAIL: Failing row contains (EasyLoan Bank, Evanston, -5, excellent).

The insertion violates the CHECK constraint that NoAccounts is greater than 0. We are trying to insert a value for NoAccounts that is negative.

c) INSERT INTO Banks (BankName, City, NoAccounts, Security) VALUES (‘EasyLoan Bank’, ‘Evanston’, 100, ‘poor’);

ERROR: new row for relation "banks" violates check constraint "banks_security_check"
DETAIL: Failing row contains (EasyLoan Bank, Evanston, 100, poor)

This insertion violates the CHECK constraint that the security level of the bank is valid. ‘Poor’ is not a recognised security level so it is rejected.

3)

a) INSERT INTO Robberies (BankName, City, Date, Amount) VALUES ('NXP Bank', 'Chicago', '2019-01-08', 1000);

ERROR: duplicate key value violates unique constraint "robberies_pkey"
DETAIL: Key (bankname, city, date)=(NXP Bank, Chicago, 2019-01-08) already exists.

The insertion violates the unique constraint for BankName, City, and Date. As these attributes are primary keys they must be unique. BankName, City, and Date already contain values for NXP Bank, Chicago, 2019-01-08. The amount value is different but I am assuming that no banks were robbed twice on the same day.

4)

a) DELETE FROM Skills WHERE SkillId = 1;

ERROR: update or delete on table "skills" violates foreign key constraint "hasskills_skillid_fkey" on table "hasskills"
DETAIL: Key (skillid)=(1) is still referenced from table "hasskills".

This deletion violates the foreign key constraint for SkillId in HasSkills, as it is still being referenced it cannot be deleted.

5)

a) 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 "robberies_bankname_city_fkey" on table "robberies"
DETAIL: Key (bankname, city)=(PickPocket Bank, Evanston) is still referenced from table "robberies".

The deletion violates the foreign key constraint for the Robberies table as it is still being referenced, where BankName, City are PickPocket Bank and Evanston.

6)

DELETE FROM Robberies WHERE BankName = 'Loanshark Bank' AND City = 'Chicago';

ERROR: update or delete on table "robberies" violates foreign key constraint "accomplices_bankname_city_date_fkey" on table "accomplices"
DETAIL: Key (bankname, city, date)=(Loanshark Bank, Chicago, 2017-11-09) is still referenced from table "accomplices".

This deletion violates the foreign key constraint as Loanshark Bank Chicago is still being referenced in the accomplices table.

7)

a) INSERT INTO Robbers (RobberId, Nickname, Age, NoYears) VALUES (1, 'Shotgun', 70, 0);

ERROR: duplicate key value violates unique constraint "robbers_pkey"
DETAIL: Key (robberid)=(1) already exists.

The insertion violates the unique constraint for RobberId. As the RobberId attribute is a primary key it must be unique. RobberId already contains a value of 1.

b) INSERT INTO Robbers (RobberId, Nickname, Age, NoYears) VALUES (999, 'Jail Mouse', 25, 35);

ERROR: new row for relation "robbers" violates check constraint "robbers_check"
DETAIL: Failing row contains (999, Jail Mouse, 25, 35).

This insertion violates the constraint that the number of years a robber has been in prison (NoYears) is less than the age of the robber.

8)

a) INSERT INTO HasSkills (RobberId, SkillId, Preference, Grade) VALUES (1, 7, 1, 'A+');

ERROR: duplicate key value violates unique constraint "hasskills_pkey"
DETAIL: Key (robberid, skillid)=(1, 7) already exists.

The insertion violates the unique constraint for RobberId, SkillId. As the RobberId and SkillId attributes are primary keys they must be unique. RobberId and SkillId already contain values of 1 and 7.

b) INSERT INTO HasSkills (RobberId, SkillId, Preference, Grade) VALUES (1, 2, 0, 'A');

ERROR: new row for relation "hasskills" violates check constraint "hasskills_preference_check"
DETAIL: Failing row contains (1, 2, 0, A).

This insertion violates the constraint that all robbers must have a preference. A robber's preference must be greater than 0.

c) INSERT INTO HasSkills (RobberId, SkillId, Preference, Grade) VALUES (999, 1, 1, 'B-');

ERROR: insert or update on table "hasskills" violates foreign key constraint "hasskills_robberid_fkey"
DETAIL: Key (robberid)=(999) is not present in table "robbers".

This insertion violates the foreign key constraint that RobberId must be in the Robbers table. There is no RobberId of 999 in the Robbers table.

d) INSERT INTO HasSkills (RobberId, SkillId, Preference, Grade) VALUES (3, 20, 3, 'B+');

ERROR: insert or update on table "hasskills" violates foreign key constraint "hasskills_skillid_fkey"
DETAIL: Key (skillid)=(20) is not present in table "skills".

This insertion violates the foreign key constraint that SkillId must be in the Skills table. There is no SkillId of 20 in the Skills table.

9)

a) 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 "hasaccounts_robberid_fkey" on table "hasaccounts"
DETAIL: Key (robberid)=(1) is still referenced from table "hasaccounts".

The deletion violates the foreign key constraint for the HasAccounts table as it is still being referenced, where RobberId is 1.

Question 4)

1)

```
gajadhroha=> SELECT BankName, City FROM Banks
gajadhroha-> WHERE (BankName, City) NOT IN (
gajadhroha(>     SELECT BankName, City FROM Robberies
gajadhroha(> );
    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 rows)
```

2)

```
gajadhroha=> SELECT r.RobberId, r.Nickname, r.Age, s.Description
gajadhroha-> FROM Robbers r
gajadhroha-> JOIN HasSkills hs ON r.RobberId = hs.RobberId
gajadhroha-> JOIN Skills s ON hs.SkillId = s.SkillId
gajadhroha-> WHERE r.Age > 40;
 robberid |    nickname    | age |  description
----------+----------------+-----+---------------
       18 | Vito Genovese  |  66 | Cooking
       17 | Bugsy Siegel   |  48 | Driving
        3 | Lucky Luchiano |  42 | Driving
        7 | Dutch Schulz   |  64 | Driving
       18 | Vito Genovese  |  66 | Eating
        2 | Bugsy Malone   |  42 | Explosives
        4 | Anastazia      |  48 | Guarding
       17 | Bugsy Siegel   |  48 | Guarding
        9 | Calamity Jane  |  44 | Gun-Shooting
        3 | Lucky Luchiano |  42 | Lock-Picking
        7 | Dutch Schulz   |  64 | Lock-Picking
       15 | Boo Boo Hoff   |  54 | Planning
       16 | King Solomon   |  74 | Planning
       12 | Moe Dalitz     |  41 | Safe-Cracking
       18 | Vito Genovese  |  66 | Scouting
(15 rows)
```

3)

```
gajadhroha=> SELECT b.BankName, b.City
gajadhroha-> FROM Banks b
gajadhroha-> JOIN HasAccounts ha ON b.BankName = ha.BankName AND b.City = ha.City
gajadhroha-> JOIN Robbers r ON ha.RobberId = r.RobberId
gajadhroha-> WHERE r.Nickname = 'Al Capone';
    bankname     |   city
-----------------+----------
 Bad Bank        | Chicago
 Inter-Gang Bank | Evanston
 NXP Bank        | Chicago
(3 rows)
```

4)

```
gajadhroha=> SELECT BankName, City, NoAccounts
gajadhroha-> FROM Banks
gajadhroha-> WHERE City != 'Chicago'
gajadhroha-> ORDER BY NoAccounts ASC;
    bankname     |   city    | noaccounts
-----------------+-----------+------------
 Gun Chase Bank  | Burbank   |       1999
 PickPocket Bank | Evanston  |       2000
 PickPocket Bank | Deerfield |       6565
 Penny Pinchers  | Evanston  |     130013
 Bankrupt Bank   | Evanston  |     444000
 Inter-Gang Bank | Evanston  |     555555
 Gun Chase Bank  | Evanston  |     656565
 NXP Bank        | Evanston  |     656565
 Dollar Grabbers | Evanston  |     909090
 Loanshark Bank  | Deerfield |    3456789
 Loanshark Bank  | Evanston  |    7654321
(11 rows)
```

5)

```
gajadhroha=> SELECT r.RobberId, r.Nickname, SUM(a.Share) AS TotalEarnings
gajadhroha-> FROM Robbers r
gajadhroha-> JOIN Accomplices a ON r.RobberId = a.RobberId
gajadhroha-> GROUP BY r.RobberId
gajadhroha-> HAVING SUM(a.Share) > 40000
gajadhroha-> ORDER BY TotalEarnings DESC;
 robberid |     nickname      | totalearnings
----------+-------------------+---------------
        5 | Mimmy The Mau Mau |      70000.00
       15 | Boo Boo Hoff      |      61447.61
       16 | King Solomon      |      59725.80
       17 | Bugsy Siegel      |      52601.10
        3 | Lucky Luchiano    |      42667.00
       10 | Bonnie            |      40085.00
(6 rows)
```

6)

```
gajadhroha=> SELECT RobberId, Nickname, NoYears AS NumberofYears
gajadhroha-> FROM Robbers
gajadhroha-> WHERE NoYears > 10;
 robberid |    nickname     | numberofyears
----------+-----------------+---------------
        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 rows)
```

7)

```
gajadhroha=> SELECT RobberId, Nickname, (Age - NoYears) AS NumberofYears
gajadhroha-> FROM Robbers
gajadhroha-> WHERE Age/2 < NoYears;
 robberid |   nickname     | numberofyears
----------+----------------+---------------
        6 | Tony Genovese  |            12
       16 | King Solomon   |            31
(2 rows)
```

8)

```
gajadhroha=> SELECT s.Description, r.RobberId, r.Nickname
gajadhroha-> FROM Skills s
gajadhroha-> JOIN HasSkills hs ON s.SkillId = hs.SkillId
gajadhroha-> JOIN Robbers r ON r.RobberId = hs.RobberId
gajadhroha-> ORDER BY s.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
 Money Counting |       14 | Kid Cann
 Money Counting |       19 | Mike Genovese
 Planning       |       15 | Boo Boo Hoff
 Planning       |        8 | Clyde
 Planning       |        5 | Mimmy The Mau Mau
 Planning       |        1 | Al Capone
 Planning       |       16 | King Solomon
 Preaching      |       22 | Greasy Guzik
 Preaching      |       10 | Bonnie
 Preaching      |        1 | Al Capone
 Safe-Cracking  |        1 | Al Capone
 Safe-Cracking  |       24 | Sonny Genovese
 Safe-Cracking  |       12 | Moe Dalitz
 Safe-Cracking  |       11 | Meyer Lansky
 Scouting       |        8 | Clyde
 Scouting       |       18 | Vito Genovese
(38 rows)
```

Question 5

1) This includes NXP bank Chicago because the plans were made after the bank was robbed.

```
gajadhroha=> SELECT DISTINCT p.BankName, p.City
gajadhroha-> FROM Plans p
gajadhroha-> LEFT JOIN Robberies r ON p.BankName = r.BankName AND p.City = r.Cit
y AND p.PlannedDate < r.Date;
    bankname     |   city
-----------------+-----------
 Bad Bank        | Chicago
 Dollar Grabbers | Chicago
 Gun Chase Bank  | Evanston
 Hidden Treasure | Chicago
 Inter-Gang Bank | Evanston
 Loanshark Bank  | Deerfield
 NXP Bank        | Chicago
 PickPocket Bank | Chicago
 PickPocket Bank | Deerfield
(9 rows)
```

2)

```
gajadhroha=> SELECT DISTINCT r.RobberId, r.Nickname
gajadhroha-> From Robbers r
gajadhroha-> INNER JOIN HasAccounts ha ON ha.RobberId = r.RobberId
gajadhroha-> LEFT JOIN Robberies rob ON rob.BankName != ha.BankName AND rob.City != ha.City;

 robberid |     nickname
----------+------------------
       14 | Kid Cann
       22 | Greasy Guzik
       21 | Waxey Gordon
        5 | Mimmy The Mau Mau
       23 | Lepke Buchalter
       19 | Mike Genovese
       20 | Longy Zwillman
        1 | Al Capone
        7 | Dutch Schulz
       13 | Mickey Cohen
       18 | Vito Genovese
       24 | Sonny Genovese
       11 | Meyer Lansky
        2 | Bugsy Malone
       12 | Moe Dalitz
       17 | Bugsy Siegel
        8 | Clyde
       15 | Boo Boo Hoff
        4 | Anastazia
        9 | Calamity Jane
        3 | Lucky Luchiano
(21 rows)
```

3)

```
gajadhroha=> SELECT r.RobberId, r.Nickname, s.Description
gajadhroha-> FROM Robbers r
JOIN HasSkills hs ON r.RobberId = hs.RobberId
gajadhroha-> JOIN HasSkills hs ON r.RobberId = hs.RobberId
gajadhroha-> JOIN Skills s ON hs.SkillId = s.SkillId
gajadhroha-> WHERE r.RobberId IN (
gajadhroha(>     SELECT RobberId
gajadhroha(>     FROM HasSkills
gajadhroha(>     GROUP BY RobberId
gajadhroha(>     HAVING COUNT(*) >= 2
gajadhroha(> )
gajadhroha-> AND hs.Preference = 1;
 robberid |     nickname      | description
----------+-------------------+-------------
       22 | Greasy Guzik      | Preaching
        3 | Lucky Luchiano    | Lock-Picking
       17 | Bugsy Siegel      | Driving
        5 | Mimmy The Mau Mau | Planning
        7 | Dutch Schulz      | Lock-Picking
       24 | Sonny Genovese    | Explosives
        1 | Al Capone         | Planning
       18 | Vito Genovese     | Scouting
       23 | Lepke Buchalter   | Driving
        8 | Clyde             | Lock-Picking
(10 rows)
```