Q1

1.

primary keys:

Banks: BankName, City

"the local banking authority ensure that the combination of name and city is unique"

Robberies: BankName, City, Date

Plans: BankName, City, PlannedDate

Robbers: RobberId

Skills: SkillId

HasSkills: RobberId, SkillId

HasAccounts: BankName, City, RobberId

Accomplices: BankName, City, Date, RobberId

foreign keys:

Banks: -

Robberies: Banks(BankName, City)

Plans: Banks(BankName, City)

Robbers: -

Skills: -

HasSkills: Robbers(RobberId), Skills(SkillId)

HasAccounts: Banks(BankName, City), Robbers(RobberId)

Accomplices: Robberies(BankName, City, Date), Robbers(RobberId)

2.

```
CREATE DOMAIN secRating AS CHAR(10)
   DEFAULT 'weak'
   CONSTRAINT allowed_values
       CHECK ( VALUE = 'excellent' OR
               VALUE = 'very good' OR
               VALUE = 'good' OR
               VALUE = 'weak');

CREATE DOMAIN skillGrade AS CHAR(3)
   DEFAULT 'C-'
   CONSTRAINT allowed_values
       CHECK ( VALUE = 'A+' OR
               VALUE = 'A' OR
               VALUE = 'A-' OR
               VALUE = 'B+' OR
               VALUE = 'B' OR
               VALUE = 'B-' OR
               VALUE = 'C+' OR
               VALUE = 'C' OR
               VALUE = 'C-' OR
               VALUE = 'D+' OR
               VALUE = 'D' OR
               VALUE = 'D-');

CREATE TABLE Banks (
  BankName TEXT,
```

```
  City TEXT,
  NoAccounts INT CHECK (NoAccounts >= 0),
  Security secRating,
  PRIMARY KEY(BankName, City)
);
--response: CREATE TABLE

CREATE TABLE Robberies (
  BankName TEXT,
  City TEXT,
  Date DATE,
  Amount NUMERIC,
  PRIMARY KEY(BankName, City, Date),
  FOREIGN KEY(BankName, City)
      REFERENCES Banks(BankName, City)
      ON DELETE RESTRICT

);
--response: CREATE TABLE

CREATE TABLE Plans (
  BankName TEXT,
  City TEXT,
  PlannedDate DATE,
  NoRobbers INT CHECK (NoRobbers > 0),
  PRIMARY KEY(BankName, City, PlannedDate),
  FOREIGN KEY(BankName, City)
      REFERENCES Banks(BankName, City)
      ON DELETE CASCADE
);
--response: CREATE TABLE

CREATE TABLE Robbers (
  RobberId SERIAL PRIMARY KEY,
  Nickname TEXT,
  Age INT CHECK (Age > 0),
  NoYears INT CHECK (NoYears < Age)
);
--response: CREATE TABLE

CREATE TABLE Skills (
  SkillId SERIAL PRIMARY KEY ,
  Description TEXT UNIQUE NOT NULL
);
--response: CREATE TABLE

CREATE TABLE HasSkills (
  RobberId INT,
  SkillId INT,
  Preference INT CHECK (Preference > 0),
  Grade CHAR(3),
  UNIQUE (RobberId, Preference),
  PRIMARY KEY(RobberId, SkillId),
```

```
  FOREIGN KEY(RobberId)
      REFERENCES Robbers(RobberId)
      ON DELETE CASCADE,
  FOREIGN KEY(SkillId)
      REFERENCES Skills(SkillId)
      ON DELETE CASCADE
);
--response: CREATE TABLE

CREATE TABLE HasAccounts (
  RobberId INT,
  BankName TEXT,
  City TEXT,
  PRIMARY KEY(RobberId, BankName, City),
  FOREIGN KEY(BankName, City)
      REFERENCES Banks(BankName, City)
      ON DELETE CASCADE,
  FOREIGN KEY(RobberId)
      REFERENCES Robbers(RobberId)
      ON DELETE CASCADE
);
--response: CREATE TABLE

CREATE TABLE Accomplices (
  RobberId INT,
  BankName TEXT,
  City TEXT,
  Date DATE,
  Share NUMERIC,
  PRIMARY KEY(BankName, City, Date, RobberId),
  FOREIGN KEY(BankName, City, Date)
      REFERENCES Robberies(BankName, City, Date)
      ON DELETE RESTRICT,
  FOREIGN KEY(RobberId)
      REFERENCES Robbers(RobberId)
      ON DELETE RESTRICT
);
--response: CREATE TABLE
```

3.

The Robberies forigen key uses ON DELETE RESTRICT because this prevents actions that would delete a location without first dealing with robberies recorded for that location.
The Plans forigen key uses ON DELETE CASCADE because if a location is deleted there is no reason to remember unused plans there.
The HasSkills forigen keys both use ON DELETE CASCADE because if either the robber or skill is deleted then the record of their preference and grade in that skill is unnecessary.
The HasAccounts forigen keys both use ON DELETE CASCADE for similar reasons to HasSkills.
The Accomplices forigen keys both use ON DELETE RESTRICT to protect the information on robbery cooperation if robbers or locations are deleted.

4.

In Banks I prevent negative values for noAccounts as it is not possible to have less than 0 accounts.

Plans require greater than 0 robbers as a plan with no robbers seems equivalent to no plan

For Robbers I added a constraint preventing robbers 0 or younger, I didn't want to set the limit higher as children could be used by the robbers and may need to be entered as Accomplices. I check that time in prison is less than the age of the robber.

In HasSkils I included a UNIQUE constraint for (RobberId, Preference) this ensures that there is only one skill at each preference level for each robber. I also check that preference values are greater than 0 because 1 is supposed to be the highest preference.

Both share in Accomplices and amount in Robberies are unrestricted because money could be lost during a failed robbery attempt resulting in a negative value.

Q2

1.

```
\copy Banks FROM
/am/phoenix/home1/straigchri/swen304/project1/data/banks_21.data
--response: COPY 20

\copy Robbers(nickname, age, noyears) FROM
/am/phoenix/home1/straigchri/swen304/project1/data/robbers_21.data
--response: COPY 24

\copy Robberies FROM
/am/phoenix/home1/straigchri/swen304/project1/data/robberies_21.data
--response: COPY 21

\copy Plans FROM
/am/phoenix/home1/straigchri/swen304/project1/data/plans_21.data
--response: COPY 11

CREATE TABLE SkillsData (
  Nickname TEXT,
  Description TEXT,
  Preference INT,
  Grade CHAR(3)
);
--response: CREATE TABLE

\copy SkillsData FROM
/am/phoenix/home1/straigchri/swen304/project1/data/hasskills_21.data
--response: COPY 38

INSERT INTO Skills(description) SELECT DISTINCT Description FROM SkillsData;
--response: INSERT 0 12

INSERT INTO HasSkills
SELECT RobberId, SkillId, Preference, Grade FROM
   (SELECT RobberId, Description, Preference, Grade FROM Robbers
       INNER JOIN SkillsData
           ON Robbers.Nickname = SkillsData.Nickname) as RSD
```

```
   INNER JOIN Skills
       ON Skills.Description = RSD.Description;
--response: INSERT 0 38

DROP TABLE SkillsData;
--response: DROP TABLE

CREATE TABLE AccountsData (
  Nickname TEXT,
  BankName TEXT,
  City TEXT
);
--response: CREATE TABLE

\copy AccountsData FROM
/am/phoenix/home1/straigchri/swen304/project1/data/hasaccounts_21.data
--response: COPY 31

INSERT INTO HasAccounts
SELECT RobberId, BankName, City FROM AccountsData
   INNER JOIN Robbers
       ON Robbers.Nickname = AccountsData.Nickname;
--response: INSERT 0 31

DROP TABLE AccountsData;
--response: DROP TABLE

CREATE TABLE AccomplicesData (
  Nickname TEXT,
  BankName TEXT,
  City TEXT,
  Date DATE,
  Share NUMERIC
);
--response: CREATE TABLE

\copy AccomplicesData FROM
/am/phoenix/home1/straigchri/swen304/project1/data/accomplices_21.data
--response: COPY 76

INSERT INTO Accomplices
SELECT RobberId, BankName, City, Date, Share FROM AccomplicesData
   INNER JOIN Robbers
       ON Robbers.Nickname = AccomplicesData.Nickname;
--response: INSERT 0 76

DROP TABLE AccomplicesData;
--response: DROP TABLE
```

2.

I first filled the tables that could be directly imported.
Then I loaded the hasSkills_21.data file into a temporary table, that allowed me to fill the skills table. With both the skillIds and robberIds generated I used joins with Robbers, Skills and the temporary table to fill HasSkills.
hasaccounts_21.data was loaded into a temporary table which was joined with the robberIds in robbers to fill HasAccounts. The same procedure was used with accomplices_21.data to fill Accomplices

Q3

1.

```
INSERT INTO banks
VALUES ('Loanshark Bank', 'Evanston', 100, 'very good');
--response: ERROR: duplicate key value violates unique constraint
"banks_pkey"
-- Detail: Key (bankname, city)=(Loanshark Bank, Evanston) already exists.

-- the key ('Loanshark Bank', 'Evanston') is already in banks

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

-- it is not possible to have a negative number of accounts

INSERT INTO banks
VALUES ('EasyLoan Bank', 'Evanston', 100, 'poor');
--response: ERROR: value for domain secrating violates check constraint
"allowed_values"

-- poor is not a valid security rating
```

2.

```
INSERT INTO skills VALUES (21, 'Driving');
--response: ERROR: duplicate key value violates unique constraint
"skills_description_key"
-- Detail: Key (description)=(Driving) already exists.

-- the Driving skill already exists
```

3.

```
INSERT INTO robberies VALUES ('NXP Bank', 'Chicago', '2019-01-08', 1000);
--response: ERROR: duplicate key value violates unique constraint
"robberies_pkey"
-- Detail: Key (bankname, city, date)=(NXP Bank, Chicago, 2019-01-08)
already exists.

-- there is already a robbery at that location on that date
```

4.

```
DELETE FROM banks
WHERE bankname = 'PickPocket Bank'
 AND city = 'Evanston'
 AND noaccounts = 2000
 AND security = 'very good';
--response: 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".

-- deleting this tuple world violate the foreign key of rows in the
robberies table
```

5.

```
DELETE FROM banks
WHERE bankname = 'Loanshark Bank'
 AND city = 'Chicago'
 AND noaccounts = ''
 AND security = '';
--response: invalid input syntax for integer: ""
-- LINE 4:   AND noaccounts = ''

-- noaccounts is an int so it can't be an empty string
```

6.

```
INSERT INTO Robbers VALUES (1, 'Shotgun', 70, 0);
--response: ERROR: duplicate key value violates unique constraint
"robbers_pkey"
-- Detail: Key (robberid)=(1) already exists.

--robber 1 already exists

INSERT INTO Robbers VALUES (666, 'Jail Mouse', 25, 35);
--response: ERROR: new row for relation "robbers" violates check constraint
"robbers_check"
-- Detail: Failing row contains (666, Jail Mouse, 25, 35).

-- cannot be in jail longer than alive i.e. age < no years
```

7.

```
INSERT INTO HasSkills VALUES (1, 7, 1, 'A+');
--response: ERROR: duplicate key value violates unique constraint
"hasskills_pkey"
-- Detail: Key (robberid, skillid)=(1, 7) already exists.

-- there is already a robber with that skill recorded

INSERT INTO HasSkills VALUES (1, 2, 0, 'A');
--response: ERROR: new row for relation "hasskills" violates check
constraint "hasskills_preference_check"
-- Detail: Failing row contains (1, 2, 0, A ).

-- 1 is the minimum preference level
```

```
INSERT INTO HasSkills VALUES (666, 1, 1, 'B-');
--response: ERROR: insert or update on table "hasskills" violates foreign
key constraint "hasskills_robberid_fkey"
-- Detail: Key (robberid)=(666) is not present in table "robbers".

-- there is no robber 666

INSERT INTO HasSkills VALUES (3, 30, 3, 'B+');
--response: ERROR: insert or update on table "hasskills" violates foreign
key constraint "hasskills_skillid_fkey"
-- Detail: Key (skillid)=(30) is not present in table "skills".

-- there is no skill 30
```

8.

```
DELETE FROM skills
WHERE skillId = 7 AND description = 'Planning';
--response: DELETE 0

-- skill 7 is 'Safe-Cracking' not 'Planning' so nothing is deleted
```

9.

```
DELETE FROM Robbers
WHERE robberid = 1
 AND nickname = 'Al Capone'
 AND age = 31
 AND noyears = 2;
--response: ERROR: update or delete on table "robbers" violates foreign key
constraint "accomplices_robberid_fkey" on table "accomplices"
-- Detail: Key (robberid)=(1) is still referenced from table "accomplices".

-- deleting this tuple world violate the foreign key of rows in the
accomplices table
```

Q4

1.

```
SELECT Robbers.RobberId, Nickname, Age, Description FROM
   (SELECT RobberId, Description FROM
       Skills INNER JOIN HasSkills
           ON Skills.SkillId = HasSkills.SkillId) as skills
   INNER JOIN Robbers
       ON Robbers.RobberId = skills.RobberId
WHERE Age >= 20 AND Age <= 40;
```

```
 robberid |      nickname    | age |  description
----------+------------------+-----+----------------
        1 | Al Capone        |  31 | Planning
        1 | Al Capone        |  31 | Preaching
        1 | Al Capone        |  31 | Safe-Cracking
        8 | Clyde            |  20 | Lock-Picking
```

```
      8 | Clyde         |  20 | Planning
      8 | Clyde         |  20 | Scouting
      22 | Greasy Guzik     |  25 | Preaching
      22 | Greasy Guzik     |  25 | Lock-Picking
      23 | Lepke Buchalter |  25 | Driving
      23 | Lepke Buchalter |  25 | Guarding
      20 | Longy Zwillman  |  35 | Driving
      11 | Meyer Lansky     |  34 | Safe-Cracking
      13 | Mickey Cohen     |  24 | Money Counting
      19 | Mike Genovese   |  35 | Money Counting
      24 | Sonny Genovese  |  39 | Explosives
      24 | Sonny Genovese  |  39 | Safe-Cracking
      24 | Sonny Genovese  |  39 | Lock-Picking
      6 | Tony Genovese    |  28 | Eating
(18 rows)
```

2.

```
SELECT Banks.BankName, Banks.City
FROM Banks
   LEFT JOIN Robberies
       ON (Banks.BankName, Banks.City) = (Robberies.BankName,
Robberies.City)
WHERE (Robberies.BankName, Robberies.City) IS NULL;
```

```
     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)
```

3.

```
SELECT BankName, City FROM HasAccounts, Robbers
WHERE Robbers.Nickname = 'Al Capone'
 AND HasAccounts.RobberId = Robbers.RobberId;
```

```
     bankname    |   city
-----------------+----------
 Bad Bank        | Chicago
 Inter-Gang Bank | Evanston
 NXP Bank        | Chicago
(3 rows)
```

4.

```
SELECT BankName, City, NoAccounts
```

```
FROM Banks
WHERE BankName NOT IN (SELECT BankName FROM Banks WHERE City = 'Deerfield')
ORDER BY NoAccounts;
```

```
      bankname    |    city    | noaccounts
------------------+------------+------------
 Gun Chase Bank   | Burbank    |    1999
 Outside Bank     | Chicago    |    5000
 Bad Bank         | Chicago    |    6000
 Dollar Grabbers  | Chicago    |    56005
 Inter-Gang Bank  | Chicago    |    100000
 Penny Pinchers   | Evanston   |    130013
 Penny Pinchers   | Chicago    |    156165
 Bankrupt Bank    | Evanston   |    444000
 Inter-Gang Bank  | Evanston   |    555555
 Gun Chase Bank   | Evanston   |    656565
 NXP Bank         | Evanston   |    656565
 Dollar Grabbers  | Evanston   |    909090
 Hidden Treasure  | Chicago    |    999999
 NXP Bank         | Chicago    |    1593311
(14 rows)
```

5.

```
SELECT Robbers.RobberId, Nickname, earnings
FROM Robbers, (SELECT RobberId, SUM(Share) as earnings FROM Accomplices
GROUP BY RobberId) as etable
WHERE Robbers.RobberId = etable.RobberId
 AND earnings > 30000
ORDER BY earnings DESC;
```

```
 robberid |      nickname     | earnings
----------+-------------------+----------
       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
       1  | Al Capone         |    39486
       4  | Anastazia         | 39169.62
       8  | Clyde             |    31800
(9 rows)
```

6.

```
SELECT RobberId, Nickname, NoYears
FROM Robbers
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 rows)
```

7.

```
SELECT RobberId, Nickname, (Age - NoYears) as NoYearsNot
FROM Robbers
WHERE NoYears > Age / 2;
```

```
 robberid |   nickname      | noyearsnot
----------+---------------+------------
      6 | Tony Genovese |         12
      16 | King Solomon  |         31
(2 rows)
```

8.

```
SELECT Description, RobberId, Nickname
FROM Skills, Robbers
WHERE (RobberId, SkillId) IN (SELECT RobberId, SkillId FROM HasSkills)
ORDER BY Description;
```

```
  description    | robberid |     nickname
-----------------+----------+-------------------
 Cooking         |     18 | Vito Genovese
 Driving         |     23 | Lepke Buchalter
 Driving         |     7 | Dutch Schulz
 Driving         |     17 | Bugsy Siegel
 Driving         |     5 | Mimmy The Mau Mau
 Driving         |     3 | Lucky Luchiano
 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    |     21 | Waxey Gordon
 Gun-Shooting    |     9 | Calamity Jane
 Lock-Picking    |     22 | Greasy Guzik
 Lock-Picking    |     7 | Dutch Schulz
 Lock-Picking    |     3 | Lucky Luchiano
 Lock-Picking    |     8 | Clyde
 Lock-Picking    |     24 | Sonny Genovese
 Money Counting |     19 | Mike Genovese
```

```
 Money Counting |     13 | Mickey Cohen
 Money Counting |     14 | Kid Cann
 Planning       |     16 | King Solomon
 Planning       |     8 | Clyde
 Planning       |     5 | Mimmy The Mau Mau
 Planning       |     15 | Boo Boo Hoff
 Planning       |     1 | Al Capone
 Preaching      |     22 | Greasy Guzik
 Preaching      |     1 | Al Capone
 Preaching      |     10 | Bonnie
 Safe-Cracking  |     12 | Moe Dalitz
 Safe-Cracking  |     11 | Meyer Lansky
 Safe-Cracking  |     1 | Al Capone
 Safe-Cracking  |     24 | Sonny Genovese
 Scouting       |     18 | Vito Genovese
 Scouting       |     8 | Clyde
(38 rows)
```

Q5

Part A)

1.

```sql
SELECT BankName, City
FROM Banks
WHERE (BankName, City) NOT IN
   (SELECT BankName, City
   FROM Robberies
   WHERE (BankName, City, EXTRACT(YEAR FROM Date)) IN
       (SELECT BankName, City, EXTRACT(YEAR FROM PlannedDate)
       FROM Plans)
   );
```

```
     bankname    |   city
-----------------+-----------
 Bankrupt Bank   | Evanston
 Loanshark Bank  | Evanston
 Loanshark Bank  | Deerfield
 Loanshark Bank  | Chicago
 Inter-Gang Bank | Chicago
 Inter-Gang Bank | Evanston
 NXP Bank        | Evanston
 Penny Pinchers  | Chicago
 Dollar Grabbers | Chicago
 Penny Pinchers  | Evanston
 Dollar Grabbers | Evanston
 Gun Chase Bank  | Evanston
 Gun Chase Bank  | Burbank
 PickPocket Bank | Evanston
 PickPocket Bank | Deerfield
```

```
 PickPocket Bank | Chicago
 Hidden Treasure | Chicago
 Bad Bank        | Chicago
 Outside Bank    | Chicago
(19 rows)
```

2.

```
SELECT DISTINCT RobberId, Nickname
FROM Robbers, Banks
WHERE (RobberId, BankName, City) IN (SELECT RobberId, BankName, City FROM
HasAccounts)
   AND (RobberId, BankName, City) NOT IN (SELECT RobberId, BankName, City
FROM Accomplices);
```

```
 robberid |      nickname
----------+-----------------
       14 | Kid Cann
       13 | Mickey Cohen
       18 | Vito Genovese
       24 | Sonny Genovese
       19 | Mike Genovese
       2 | Bugsy Malone
       12 | Moe Dalitz
       21 | Waxey Gordon
       7 | Dutch Schulz
       15 | Boo Boo Hoff
       4 | Anastazia
       9 | Calamity Jane
       3 | Lucky Luchiano
       23 | Lepke Buchalter
(14 rows)
```

3.

```
SELECT RobberId, Nickname, Description
FROM Robbers, Skills
WHERE (RobberId, SkillId, 1) IN (SELECT RobberId, SkillId, Preference FROM
HasSkills)
   AND RobberId IN (SELECT RobberId FROM HasSkills 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 rows)
```

4.

```
SELECT BankName, City, Date
FROM Robberies
WHERE (BankName, City, Date) IN
     (SELECT BankName, City, Date FROM Accomplices WHERE Share = (SELECT
MAX(share) FROM Accomplices));
--could simplify to only the select statement in the where section but that
--would be selecting from Accomplices not Robberies
```

```
      bankname    |    city    |     date
------------------+------------+------------
 Inter-Gang Bank | Evanston | 2017-03-13
(1 row)
```

5.

Part B)

6.

7.