SWEN 304 Project 1

Question 1

1. A list of the primary keys and foreign keys for each relation, along with a brief justification for your choice of keys and foreign keys.

Banks

Attributes: (BankName, City, NoAccounts, Security)
Primary Key: (BankName, City)
Foreign Key: N/A

Justification: Bank branches are specified by the names of each bank and the city that they are located in, therefore making the primary key above a suitable one (due to no duplicates). There is no foreign key as none of its attributes refer to another relation.

Robberies

Attributes: (BankName, City, Date, Amount)
Primary Key: (BankName, City, Date)
Foreign Key: (BankName, City)

Justification: (BankName, City) is a primary key in the Banks relation, therefore making it a foreign key here. Additionally, the chosen primary key uniquely defines each robbery that takes place, therefore making it a suitable primary because it doesn't contain duplicates..

Plans

Attributes: (BankName, City, PlannedDates, NoRobbers)
Primary Key: (BankName, City, PlannedDates)
Foreign Key: (BankName, City)

Justification: (BankName, City) is a primary key in the Banks relation, therefore making it a foreign key here. Additionally, the chosen primary key uniquely defines each plan, therefore making it a suitable primary because it doesn't contain duplicates.

Robbers

Attributes: (RobberId, NickName, Age, NoYears)
Primary Key: (RobberId)

Foreign Key: N/A

Justification: The RobberID attribute uniquely identifies each Robber in the Robbers relation, therefore making it a suitable primary key. Because none of its attributes refer to another relation, it indicates that there is no foreign key.

Skills

Attributes: (SkillId, Description)
Primary Key: (BankName, City)
Foreign Key: N/A

Justification: The SkillId attribute uniquely identifies each Skill in the relation, therefore making it a suitable primary key. There is no foreign key due to the lack of an attribute that refers to another relation.

HasSkills

Attributes: (RobberId, SkillId, Preference, Grade)
Primary Key: (RobberId, SkillId)
Foreign Key: (RobberId, SkillId)

Justification: (RobberId, SkillId) is a suitable primary key because it uniquely identifies each tuple in the HasSkills relation. Additionally, both of these attributes point to the Robbers and Skills relation respectively, therefore making it a foreign key as well.

HasAccounts

Attributes: (BankName, City, RobberId)
Primary Key: (BankName, City, RobberId)
Foreign Key: (BankName, City, RobberId)

Justification: This relation requires all three attributes to determine which bank in which city (i.e the bank branch) the robber has an account on. As a result, it uniquely identifies each tuple in the relation, therefore making it a suitable primary key. Additionally, all three attributes point to other relations, therefore making it a foreign key as well.

Accomplices

Attributes: (BankName, City, RobberId, Date, Share)
Primary Key: (BankName, City, RobberId, Date)
Foreign Key: (BankName, City, RobberId, Date)

Justification: This relation requires all of the given attributes to determine each robber, bank name, city, date of the robbery and share from the robbery that accomplices can get. As a result, it uniquely identifies each tuple in the relation, therefore making it a suitable primary key. Additionally, three of the four attributes point to other relations, therefore making it a foreign key as well.

2. A list of all your CREATE TABLE statements.

NOTE: I wrote my statements in a .sql file prior to putting them on the command line.

Banks

```
swen304project1romanematt=> CREATE TABLE Banks(
swen304project1romanematt(>     BankName CHAR(30) NOT NULL,
swen304project1romanematt(>     City CHAR(30) NOT NULL,
swen304project1romanematt(>     NoAccounts INT DEFAULT 0,
swen304project1romanematt(>     Security CHAR(15),
swen304project1romanematt(>     CONSTRAINT BanksPK PRIMARY KEY (BankName, City)
swen304project1romanematt(> );
CREATE TABLE
swen304project1romanematt=>
```

Robberies

```
swen304project1romanematt=> CREATE TABLE Robberies(
swen304project1romanematt(>     BankName CHAR(30) NOT NULL,
swen304project1romanematt(>     City CHAR(30) NOT NULL,
swen304project1romanematt(>     Date DATE NOT NULL,
swen304project1romanematt(>     Amount int NOT NULL,
swen304project1romanematt(>     CONSTRAINT RobberiesPK Primary Key (BankName, City, Date),
swen304project1romanematt(>     FOREIGN KEY (BankName, City)
swen304project1romanematt(>     REFERENCES Banks(BankName, City)
swen304project1romanematt(> );
CREATE TABLE
swen304project1romanematt=>
```

Plans

```
swen304project1romanematt=> CREATE TABLE Plans(
swen304project1romanematt(>     BankName CHAR(30) NOT NULL,
swen304project1romanematt(>     City CHAR(30) NOT NULL,
swen304project1romanematt(>     NoRobbers int,
swen304project1romanematt(>     PlannedDate DATE NOT NULL,
swen304project1romanematt(>     CONSTRAINT PlansPK PRIMARY KEY (BankName, City, PlannedDate),
swen304project1romanematt(>     FOREIGN KEY (BankName, City) REFERENCES Banks(BankName, City)
swen304project1romanematt(> );
CREATE TABLE
swen304project1romanematt=>
```

Robbers

```
swen304project1romanematt=> CREATE TABLE Robbers(
swen304project1romanematt(>     RobberId int NOT NULL UNIQUE,
swen304project1romanematt(>     Nickname CHAR(25) NOT NULL,
swen304project1romanematt(>     Age int NOT NULL,
swen304project1romanematt(>     NoYears int,
swen304project1romanematt(>     CONSTRAINT RobbersPK PRIMARY KEY (RobberId),
swen304project1romanematt(>     CONSTRAINT RobberNoYears CHECK (Age >= NoYears)
swen304project1romanematt(> );
CREATE TABLE
swen304project1romanematt=>
```

Skills

```
swen304project1romanematt=> CREATE TABLE Skills(
swen304project1romanematt(>     SkillId int NOT NULL UNIQUE CONSTRAINT SkillsPK PRIMARY KEY,
swen304project1romanematt(>     Description CHAR(25) NOT NULL
swen304project1romanematt(> );
CREATE TABLE
swen304project1romanematt=>
```

HasSkills

```
swen304project1romanematt=> CREATE TABLE HasSkills (
swen304project1romanematt(>     RobberId int NOT NULL,
swen304project1romanematt(>     SkillId int NOT NULL,
swen304project1romanematt(>     Preference int NOT NULL,
swen304project1romanematt(>     Grade CHAR(3) NOT NULL,
swen304project1romanematt(>     CONSTRAINT HasSkillsPK PRIMARY KEY (RobberId, SkillId),
swen304project1romanematt(>     FOREIGN KEY (RobberId) REFERENCES Robbers(RobberId),
swen304project1romanematt(>     FOREIGN KEY (SkillId) REFERENCES Skills(SkillId)
swen304project1romanematt(> );
CREATE TABLE
swen304project1romanematt=>
```

HasAccounts

```
swen304project1romanematt=> CREATE TABLE HasAccounts (
swen304project1romanematt(>     RobberId int NOT NULL,
swen304project1romanematt(>     BankName CHAR(30) NOT NULL,
swen304project1romanematt(>     City CHAR(30) NOT NULL,
swen304project1romanematt(>     CONSTRAINT HasAccountsPK PRIMARY KEY (RobberId, BankName, City),
swen304project1romanematt(>     CONSTRAINT HasAccounts2FK FOREIGN KEY (RobberId) REFERENCES Robbers(RobberId),
swen304project1romanematt(>     CONSTRAINT HasAccounts1FK FOREIGN KEY (BankName, City) REFERENCES Banks(BankName, City)
swen304project1romanematt(> );
CREATE TABLE
swen304project1romanematt=>
```

HasAccomplices

```
swen304project1romanematt=> CREATE TABLE Accomplices (
swen304project1romanematt(>     RobberId int NOT NULL,
swen304project1romanematt(>     BankName CHAR(30) NOT NULL,
swen304project1romanematt(>     City CHAR(30) NOT NULL,
swen304project1romanematt(>     Date DATE NOT NULL,
swen304project1romanematt(>     Share REAL NOT NULL,
swen304project1romanematt(>     CONSTRAINT AccomplicesPK PRIMARY KEY (RobberId, BankName, City, Date),
swen304project1romanematt(>     CONSTRAINT AccomplicesFK FOREIGN KEY (RobberId) REFERENCES Robbers(RobberId),
swen304project1romanematt(>     CONSTRAINT HasAccounts1FK FOREIGN KEY (BankName, City, Date) REFERENCES Robberies(BankName, City, Date)

swen304project1romanematt(> );
CREATE TABLE
swen304project1romanematt=>
```

Additional Constraints

Banks

Ensuring that the number of bank accounts is not below zero

```
swen304project1romanematt=> ALTER TABLE Banks
swen304project1romanematt-> ADD CONSTRAINT bankAccountNumConstraint CHECK(NoAccounts >= 0);
ALTER TABLE
swen304project1romanematt=>
```

Categorizing levels of security

```
swen304project1romanematt=> ALTER TABLE Banks
swen304project1romanematt-> ADD CONSTRAINT bankSecConstraint CHECK(
swen304project1romanematt(>         Security = 'excellent'
swen304project1romanematt(>         OR Security = 'very good'
swen304project1romanematt(>         OR Security = 'weak'
swen304project1romanematt(>     );
ALTER TABLE
swen304project1romanematt=>
```

Robberies

Changing amount type to REAL instead of int as initially declared

```
swen304project1romanematt=> ALTER TABLE Robberies
swen304project1romanematt-> ALTER COLUMN Amount TYPE REAL;
ALTER TABLE
swen304project1romanematt=>
```

Robbers

Added constraint to the RobberNoYears attribute that their prison time cannot be greater than the age

```
swen304project1romanematt(>     CONSTRAINT RobberNoYears CHECK (Age >= NoYears)
```

3. A justification for your choice of actions on delete or on update for each foreign key.

If I wanted to update the content of the foreign key, I would need to ensure that the content I want to update is within the foreign key's original table. If this is the case, I can update the foreign key. For example, if I wanted to update the type of a foreign key, it would also require me to update the type of the foreign key in the relation that it is referencing to. This is inconvenient and unnecessarily complicated.

4. A brief justification for your choice of attribute constraints (other than the basic data).

In addition to the primary key and foreign key constraints, I have added a few others to my database.

For example, my Banks relation has a bankAccountNumConstraint to ensure it doesn't fall below zero. Additionally, I added the bankSecConstraint to categorize security levels (not defined in instructions).

In the Robbers relation, I set the RobberNoYears constraint to such that the number of years in prison cannot be greater than the age, as that's unrealistic.

Question 2

1. A description of how you performed all the data conversion, for example, a sequence of the PostgreSQL statements that accomplished the conversion.

NOTE: My datafiles was stored in my account's Documents directory

Populating Banks

```
swen304project1romanematt=> \copy Banks(BankName, City, NoAccounts, Security) FROM ~/Documents/datafiles/banks_22.data
COPY 20
```

Populating Robberies

```
swen304project1romanematt=> \copy Robberies(BankName, City, Date, Amount) FROM ~/Documents/datafiles/robberies_22.data
COPY 21
```

Populating Plans

```
swen304project1romanematt=> \copy Plans(BankName, City, PlannedDate, NoRobbers) FROM ~/Documents/datafiles/plans_22.data

COPY 11
swen304project1romanematt=>
```

Populating Robbers

Setting the Robber ID

```
swen304project1romanematt=> CREATE SEQUENCE Robbers_RobberId_seq AS integer;
CREATE SEQUENCE
```

```
swen304project1romanematt=> ALTER TABLE Robbers
swen304project1romanematt-> ALTER COLUMN RobberId SET DEFAULT nextval('Robbers_RobberId_seq');
E Robbers_RobberId_seq OWNED BY Robbers.RobberId;ALTER TABLE
swen304project1romanematt=> ALTER SEQUENCE Robbers_RobberId_seq OWNED BY Robbers.RobberId;
ALTER SEQUENCE
swen304project1romanematt=>
```

Copying the data

```
swen304project1romanematt=> \copy Robbers(Nickname, Age, NoYears) FROM ~/Documents/datafiles/robbers_22.data
COPY 24
swen304project1romanematt=>
```

Resultant relation

```
swen304project1romanematt=> Select * From Robbers
swen304project1romanematt-> \g
 robberid |         nickname          | age | noyears
----------+---------------------------+-----+---------
        1 | Al Capone                 |  31 |       2
        2 | Bugsy Malone              |  42 |      15
        3 | Lucky Luchiano            |  42 |      15
        4 | Anastazia                 |  48 |      15
        5 | Mimmy The Mau Mau         |  18 |       0
        6 | Tony Genovese             |  28 |      16
        7 | Dutch Schulz              |  64 |      31
        8 | Clyde                     |  20 |       0
        9 | Calamity Jane             |  44 |       3
       10 | Bonnie                    |  19 |       0
       11 | Meyer Lansky              |  34 |       6
       12 | Moe Dalitz                |  41 |       3
       13 | Mickey Cohen              |  24 |       3
       14 | Kid Cann                  |  14 |       0
       15 | Boo Boo Hoff              |  54 |      13
       16 | King Solomon              |  74 |      43
       17 | Bugsy Siegel              |  48 |      13
       18 | Vito Genovese             |  66 |       0
       19 | Mike Genovese             |  35 |       0
       20 | Longy Zwillman            |  35 |       6
       21 | Waxey Gordon              |  15 |       0
       22 | Greasy Guzik              |  25 |       1
       23 | Lepke Buchalter           |  25 |       1
       24 | Sonny Genovese            |  39 |       0
(24 rows)
```

Populating Skills

Setting the SkillID

```
swen304project1romanematt=> CREATE SEQUENCE Skills_SkillId_seq AS integer;
CREATE SEQUENCE
swen304project1romanematt=> ALTER TABLE Skills ALTER COLUMN SkillId SET DEFAULT nextval('Skills_SkillId_seq');
ALTER TABLE
swen304project1romanematt=> ALTER SEQUENCE Skills_SkillId_seq OWNED BY Skills.SkillId;
ALTER SEQUENCE
swen304project1romanematt=>
```

Creating a temporary relation

```
swen304project1romanematt=> CREATE TABLE TempSkills (
swen304project1romanematt(>     RobberNameTemp CHAR(25) NOT NULL,
swen304project1romanematt(>     DescTemp CHAR(25) NOT NULL,
swen304project1romanematt(>     Preference int NOT NULL,
swen304project1romanematt(>     Grade CHAR(3) NOT NULL
swen304project1romanematt(> );
CREATE TABLE
swen304project1romanematt=>
```

Then copying data to the temporary table

```
swen304project1romanematt=> CREATE TABLE TempSkills (
swen304project1romanematt(>     RobberNameTemp CHAR(25) NOT NULL,
swen304project1romanematt(>     DescTemp CHAR(25) NOT NULL,
swen304project1romanematt(>     Preference int NOT NULL,
swen304project1romanematt(>     Grade CHAR(3) NOT NULL
swen304project1romanematt(> );
CREATE TABLE
swen304project1romanematt=> \copy TempSkills(RobberNameTemp, DescTemp, Preference, Grade) FROM ~/Documents/datafiles/has
skills_22.data
COPY 38
swen304project1romanematt=>
```

Extracting the Description column and adding it into the Skills relation

```
swen304project1romanematt=> INSERT INTO Skills(Description)
swen304project1romanematt-> SELECT DescTemp FROM TempSkills
swen304project1romanematt-> \g
INSERT 0 38
```

Resultant relation

```
swen304project1romanematt=> INSERT INTO Skills(Description)
swen304project1romanematt-> SELECT DescTemp FROM TempSkills
swen304project1romanematt-> \g
INSERT 0 38
swen304project1romanematt=> SELECT * FROM Skills
swen304project1romanematt-> \g
 skillid |        description
---------+---------------------------
       1 | Planning
       2 | Safe-Cracking
       3 | Preaching
       4 | Planning
       5 | Driving
       6 | Guarding
       7 | Preaching
       8 | Planning
       9 | Explosives
      10 | Driving
      11 | Guarding
      12 | Gun-Shooting
      13 | Lock-Picking
      14 | Scouting
      15 | Planning
      16 | Lock-Picking
      17 | Driving
      18 | Preaching
      19 | Lock-Picking
      20 | Money Counting
      21 | Planning
      22 | Driving
      23 | Guarding
      24 | Driving
      25 | Lock-Picking
      26 | Driving
      27 | Safe-Cracking
      28 | Money Counting
      29 | Money Counting
      30 | Safe-Cracking
      31 | Explosives
      32 | Safe-Cracking
      33 | Lock-Picking
      34 | Eating
      35 | Scouting
      36 | Cooking
      37 | Eating
      38 | Gun-Shooting
(38 rows)

swen304project1romanematt=>
```

```
swen304project1romanematt=> SELECT description, COUNT(*) FROM Skills GROUP BY description HAVING
 COUNT(*) >1;
       description        | count
--------------------------+-------
 Driving                  |     6
 Money Counting           |     3
 Safe-Cracking            |     4
 Planning                 |     5
 Lock-Picking             |     5
 Eating                   |     2
 Guarding                 |     3
 Scouting                 |     2
 Gun-Shooting             |     2
 Explosives               |     2
 Preaching                |     3
(11 rows)

swen304project1romanematt=>
```

As shown above, the Skills table contains many duplicate tuples. To resolve the problem, I opted to delete the duplicates but keep the latest of each tuple:

```
swen304project1romanematt=> DELETE FROM Skills WHERE skills.skillid NOT IN (SELECT * FROM (SELECT MAX(skills.skillid) FROM Skills GROUP BY description)AS d_alias);
DELETE 26
```

This is how the relation looks now:

```
swen304project1romanematt=> SELECT * FROM Skills
swen304project1romanematt-> \g
 skillid |        description
---------+---------------------------
      18 | Preaching
      21 | Planning
      23 | Guarding
      26 | Driving
      29 | Money Counting
      31 | Explosives
      32 | Safe-Cracking
      33 | Lock-Picking
      35 | Scouting
      36 | Cooking
      37 | Eating
      38 | Gun-Shooting
(12 rows)

swen304project1romanematt=>
```

Of course, the id's are now incorrect so I resolved this by doing the following:

```
swen304project1romanematt=> CREATE SEQUENCE Skills_SkillId_seq2 AS integer;
CREATE SEQUENCE
swen304project1romanematt=> ALTER SEQUENCE Skills_SkillId_seq2 OWNED BY Skills.SkillId;
ALTER SEQUENCE
swen304project1romanematt=>
```

After making updates, the relation looks like this:

```
swen304project1romanematt=> update skills set SkillID = nextval('Skills_SkillId_seq2')
swen304project1romanematt-> \g
UPDATE 12
swen304project1romanematt=> SELECT * FROM Skills
swen304project1romanematt-> \g
 skillid |        description
---------+---------------------------
       1 | Preaching
       2 | Planning
       3 | Guarding
       4 | Driving
       5 | Money Counting
       6 | Explosives
       7 | Safe-Cracking
       8 | Lock-Picking
       9 | Scouting
      10 | Cooking
      11 | Eating
      12 | Gun-Shooting
(12 rows)

swen304project1romanematt=>
```

Populating HasSkills

Extracting the RobberID and SkillID column, adding it to HasSkills table

```
swen304project1romanematt=> INSERT INTO HasSkills(RobberId,SkillId,Preference,Grade)
swen304project1romanematt-> SELECT r.RobberId,s.SkillId,ts.Preference,ts.Grade
swen304project1romanematt-> FROM TempSkills ts,Robbers r,Skills s
swen304project1romanematt-> WHERE r.Nickname=ts.RobberNameTemp AND s.Description=ts.DescTemp
swen304project1romanematt-> \g
INSERT 0 38
swen304project1romanematt=>
```

Result:

```
swen304project1romanematt=> SELECT * FROM HasSkills
swen304project1romanematt-> \g
 robberid | skillid | preference | grade
----------+---------+------------+-------
        1 |       1 |          3 | A+
        1 |       7 |          2 | C+
        1 |       2 |          1 | A+
        2 |       6 |          1 | A
        3 |       4 |          2 | B+
        3 |       8 |          1 | B+
        4 |       3 |          1 | A
        5 |       4 |          2 | C
        5 |       2 |          1 | A+
        6 |      11 |          1 | B+
        7 |       4 |          2 | C+
        7 |       8 |          1 | A+
        8 |       2 |          3 | C
        8 |       9 |          2 | C+
        8 |       8 |          1 | C+
        9 |      12 |          1 | B
       10 |       1 |          1 | B
       11 |       7 |          1 | A+
       12 |       7 |          1 | A
       13 |       5 |          1 | B+
       14 |       5 |          1 | B
       15 |       2 |          1 | A+
       16 |       2 |          1 | A
       17 |       3 |          2 | C+
       17 |       4 |          1 | A+
       18 |      11 |          3 | A+
       18 |      10 |          2 | A
       18 |       9 |          1 | B+
       19 |       5 |          1 | C
       20 |       4 |          1 | C
       21 |      12 |          1 | C
       22 |       8 |          2 | C
       22 |       1 |          1 | A+
       23 |       3 |          2 | C
       23 |       4 |          1 | A
       24 |       8 |          3 | B
       24 |       7 |          2 | C+
       24 |       6 |          1 | B
(38 rows)

swen304project1romanematt=>
```

Populating HasAccounts

Creating a temporary relation

```
swen304project1romanematt=>     CREATE TABLE TempHasAccounts (
swen304project1romanematt(>         RobberNameTemp CHAR(25) NOT NULL,
swen304project1romanematt(>         BankName CHAR(30) NOT NULL,
swen304project1romanematt(>         City CHAR(30) NOT NULL
swen304project1romanematt(>     );
CREATE TABLE
swen304project1romanematt=>
```

Copying data into the relation

```
swen304project1romanematt=> \copy TempHasAccounts(RobberNameTemp, BankName, City) FROM ~/Docu
ments/datafiles/hasaccounts_22.data
COPY 31
swen304project1romanematt=>
```

Extracting the Robber ID and putting it in the HasAccounts relation

```
swen304project1romanematt=> INSERT INTO HasAccounts(RobberId,BankName,City)
swen304project1romanematt-> SELECT r.RobberId,ta.BankName,ta.City
swen304project1romanematt-> FROM TempHasAccounts ta,Robbers r
swen304project1romanematt-> WHERE r.Nickname=ta.RobberNameTemp
swen304project1romanematt-> \g
INSERT 0 31
swen304project1romanematt=>
```

Result:

```
swen304project1romanematt=> select * from hasaccounts
swen304project1romanematt-> \g
 robberid |            bankname            |              city
----------+--------------------------------+--------------------------------
        1 | Bad Bank                       | Chicago
        1 | Inter-Gang Bank                | Evanston
        1 | NXP Bank                       | Chicago
        2 | Loanshark Bank                 | Chicago
        2 | Loanshark Bank                 | Deerfield
        3 | NXP Bank                       | Chicago
        3 | Bankrupt Bank                  | Evanston
        4 | Loanshark Bank                 | Evanston
        5 | Inter-Gang Bank                | Evanston
        5 | Loanshark Bank                 | Evanston
        7 | Inter-Gang Bank                | Chicago
        8 | Penny Pinchers                 | Evanston
        9 | PickPocket Bank                | Chicago
        9 | PickPocket Bank                | Evanston
        9 | Bad Bank                       | Chicago
        9 | Dollar Grabbers                | Chicago
       11 | Penny Pinchers                 | Evanston
       12 | Dollar Grabbers                | Evanston
       12 | Gun Chase Bank                 | Evanston
       13 | Gun Chase Bank                 | Burbank
       14 | PickPocket Bank                | Evanston
       15 | PickPocket Bank                | Deerfield
       17 | PickPocket Bank                | Chicago
       18 | Bad Bank                       | Chicago
       18 | Gun Chase Bank                 | Evanston
--More--
```

Populating Accomplices

Creating a temporary relation

```
swen304project1romanematt=> CREATE TABLE TempAccomplices (
swen304project1romanematt(> RobberNameTemp CHAR(25) NOT NULL,
swen304project1romanematt(> BankName CHAR(30) NOT NULL,
swen304project1romanematt(> City CHAR(30) NOT NULL ,
swen304project1romanematt(> DateTemp DATE NOT NULL,
swen304project1romanematt(> Share REAL NOT NULL
swen304project1romanematt(> );
CREATE TABLE
swen304project1romanematt=>
```

Copying data and adding it to the temporary relation

```
swen304project1romanematt=> \copy TempAccomplices(RobberNameTemp,BankName,City,DateTemp,Share) F
ROM ~/Documents/datafiles/accomplices_22.data
COPY 76
swen304project1romanematt=>
```

Extracting Robber ID column and adding it to the Accomplices table

```
swen304project1romanematt=> INSERT INTO Accomplices(RobberId,BankName,City,Date,Share)
swen304project1romanematt-> SELECT r.RobberId,ta.BankName,ta.City,ta.DateTemp,ta.Share
swen304project1romanematt-> FROM TempAccomplices ta,Robbers r
swen304project1romanematt-> WHERE r.Nickname=ta.RobberNameTemp
swen304project1romanematt-> \g
INSERT 0 76
swen304project1romanematt=>
```

Resultant relation:

```
swen304project1romanematt=> select * from accomplices
swen304project1romanematt-> \g
 robberid |           bankname           |             city             |    date    |  shar
e
----------+------------------------------+------------------------------+------------+------
---
        1 | Bad Bank                     | Chicago                      | 2017-02-02 |    30
10
        1 | NXP Bank                     | Chicago                      | 2019-01-08 |    64
06
        1 | Loanshark Bank               | Evanston                     | 2019-02-28 |    49
97
        1 | Loanshark Bank               | Chicago                      | 2019-03-30 |    42
01
        1 | Inter-Gang Bank              | Evanston                     | 2016-02-16 |   121
03
        1 | Inter-Gang Bank              | Evanston                     | 2018-02-14 |    87
69
        2 | NXP Bank                     | Chicago                      | 2019-01-08 |    23
00
        3 | Penny Pinchers               | Evanston                     | 2016-08-30 |   165
00
        3 | Loanshark Bank               | Evanston                     | 2019-02-28 |    49
97
        3 | Loanshark Bank               | Chicago                      | 2017-11-09 |    82
00
        3 | Loanshark Bank               | Chicago                      | 2019-03-30 |    42
01
        3 | Inter-Gang Bank              | Evanston                     | 2018-02-14 |    87
69
        4 | Penny Pinchers               | Evanston                     | 2016-08-30 |   165
00
        4 | NXP Bank                     | Chicago                      | 2019-01-08 | 6408.
32
--More--
```

Having done all of the above, I deleted the temporary relations that were necessary in copying data to the Skills, HasAccounts and Accomplices relations:

```
swen304project1romanematt=> drop table tempskills
swen304project1romanematt-> \g
DROP TABLE
swen304project1romanematt=> drop table hasaccounts
swen304project1romanematt-> \g
DROP TABLE
swen304project1romanematt=> drop table tempaccomplices
swen304project1romanematt-> \g
DROP TABLE
swen304project1romanematt=>
```

2. A brief description of the order in which you have implemented the tables of the RobbersGang database. Justify your answer.

I implemented my RobbersGang database as I did based on which relation had a foreign key pointing to another relation. For example, the Banks, Robbers and Skills relations all contained attributes that the rest of the relations rely on. It was, therefore, important that I implemented these first. Creating the HasSkills, HasAccounts and Accomplices table came afterward, once I had implemented the relations in which they had been referenced by foreign keys.

Question 3

1. Insert the following tuple into the Skills table:

a. (21, ‘Driving’)

```
INSERT INTO Skills (SkillID, Description)
VALUES(21,'Driving');
```

Violates UNIQUE constraint of the primary key

```
swen304project1romanematt=> INSERT INTO Skills (SkillID, Description)
swen304project1romanematt-> VALUES(21,'Driving');
ERROR:  duplicate key value violates unique constraint "skillspk"
DETAIL:  Key (skillid)=(21) already exists.
swen304project1romanematt=>
```

2. Insert the following tuples into the Banks table:

a. (‘Loanshark Bank’, ‘Evanston’, 100, ‘very good’)

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

Violates UNIQUE constraint of the primary key

```
swen304project1romanematt=> INSERT INTO Banks (BankName, City, NoAccounts, Security )
swen304project1romanematt-> VALUES('Loanshark Bank', 'Evanston', 100,'very good');
ERROR:  duplicate key value violates unique constraint "bankspk"
DETAIL:  Key (bankname, city)=(Loanshark Bank              , Evanston
already exists.
swen304project1romanematt=>
```

b. (‘EasyLoan Bank’, ‘Evanston’, -5, ‘excellent’)

```
INSERT INTO Banks (BankName, City, NoAccounts, Security )
VALUES('EasyLoan Bank', 'Evanston', -5,'excellent');
```

Violates CHECK constraint that the number of banks cannot be below 0

```
swen304project1romanematt=> INSERT INTO Banks (BankName, City, NoAccounts, Security )
swen304project1romanematt-> VALUES('EasyLoan Bank', 'Evanston', -5,'excellent');
ERROR:  new row for relation "banks" violates check constraint "bankaccountnumconstraint"
DETAIL:  Failing row contains (EasyLoan Bank               , Evanston
-5, excellent      ).
swen304project1romanematt=>
```

c. ('EasyLoan Bank', 'Evanston', 100, 'poor')

```
INSERT INTO Banks (BankName, City, NoAccounts, Security )
VALUES('EasyLoan Bank', 'Evanston', 100,'poor');
```

Violates CHECK constraint that security level must be as follows:

- Excellent
- Very good
- Good
- Weak

```
swen304project1romanematt=> INSERT INTO Banks (BankName, City, NoAccounts, Security )
swen304project1romanematt-> VALUES('EasyLoan Bank', 'Evanston', 100,'poor');
ERROR:  new row for relation "banks" violates check constraint "banksecconstraint"
DETAIL:  Failing row contains (EasyLoan Bank                , Evanston
100, poor          ).
swen304project1romanematt=>
```

3. Insert the following tuple into the Robberies table:

a. ('NXP Bank', 'Chicago', '2019-01-08', 1000)

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

Violates UNIQUE constraint of the primary key

```
swen304project1romanematt=> INSERT INTO Robberies (BankName, City, Date, Amount )
swen304project1romanematt-> VALUES('NXP Bank', 'Chicago','2019-01-08' ,1000);
ERROR:  duplicate key value violates unique constraint "robberiespk"
DETAIL:  Key (bankname, city, date)=(NXP Bank                    , Chicago
    , 2019-01-08) already exists.
swen304project1romanematt=>
```

4. Delete the following tuple from the Skills table:

a. (1, 'Driving')

```
DELETE FROM Skills
WHERE SkillID = 1 AND Description='Driving'
```

```
swen304project1romanematt=> DELETE FROM Skills
swen304project1romanematt-> WHERE SkillID = 1 AND Description='Driving'
swen304project1romanematt-> \g
DELETE 0
```

Deleting it this way doesn't work because in my relation, the Driving skill has a Skill ID of 4, so I retry to delete the tuple using its actual ID

```
swen304project1romanematt=> DELETE FROM Skills
swen304project1romanematt-> WHERE SkillID = 4 AND Description='Driving'
swen304project1romanematt-> \g
ERROR:  update or delete on table "skills" violates foreign key constraint "hasskills_skillid_f
key" on table "hasskills"
DETAIL:  Key (skillid)=(4) is still referenced from table "hasskills".
swen304project1romanematt=>
```

As shown above, my second attempt throws an error because it violates the foreign key constraint.

5. Delete the following tuples from the Banks table:

a. ('PickPocket Bank', 'Evanston', 2000, 'very good')

```
DELETE FROM Banks
WHERE BankName="PickPocket Bank" AND City="Evanston" AND NoAccounts="2000"
AND Security="very good"
```

Violates FOREIGN KEY constraint as the content of the FOREIGN key must be within the relation that the FOREIGN KEY is referencing to.

```
swen304project1romanematt=> DELETE FROM Banks
swen304project1romanematt-> WHERE BankName='PickPocket Bank' AND City='Evanston' AND NoAccounts=
2000 AND Security='very good'
swen304project1romanematt-> \g
ERROR:  update or delete on table "banks" violates foreign key constraint "robberies_bankname_fk
ey" on table "robberies"
DETAIL:  Key (bankname, city)=(PickPocket Bank              , Evanston                    ) i
s still referenced from table "robberies".
swen304project1romanematt=>
```

6. Delete the following tuple from the Robberies table:

a. ('Loanshark Bank', 'Chicago', '', '')

```
DELETE FROM Robberies
WHERE BankName="Loanshark Bank" AND City="Chicago" AND Date="" AND
Amount=""
```

Violates TYPE constraint that the datatype must match

```
swen304project1romanematt=> DELETE FROM Robberies
swen304project1romanematt-> WHERE BankName="Loanshark Bank" AND City="Chicago" AND Date="" AND A
mount=""
swen304project1romanematt-> \g
ERROR:  zero-length delimited identifier at or near """"
LINE 2: ...Name="Loanshark Bank" AND City="Chicago" AND Date="" AND Amo...
                                                             ^
swen304project1romanematt=>
```

In the following two tasks, we assume that there is a robber with Id 3, but no robber with Id

333.

7. Insert the following tuples into the Robbers table:

a. (1, ‘Shotgun’, 70, 0)

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

Violates UNIQUE constraint for RobberID

```
swen304project1romanematt=> INSERT INTO Robbers (RobberId, Nickname, Age, NoYears)
swen304project1romanematt-> VALUES (1, 'Shotgun', 70, 0);
ERROR:  duplicate key value violates unique constraint "robberspk"
DETAIL:  Key (robberid)=(1) already exists.
swen304project1romanematt=>
```

b. (333, ‘Jail Mouse’, 25, 35)

```
INSERT INTO Robbers (RobberId, Nickname, Age, NoYears)
VALUES (333, 'Jail Mouse', 25, 35);
```

Violates CHECK constraint that prison time cannot be greater than age

```
swen304project1romanematt=> INSERT INTO Robbers (RobberId, Nickname, Age, NoYears)
swen304project1romanematt-> VALUES (333, 'Jail Mouse', 25, 35);
ERROR:  new row for relation "robbers" violates check constraint "robbernoyears"
DETAIL:  Failing row contains (333, Jail Mouse                , 25, 35).
swen304project1romanematt=>
```

8. Insert the following tuples into the HasSkills table:

a. (1, 7, 1, ‘A+’)

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

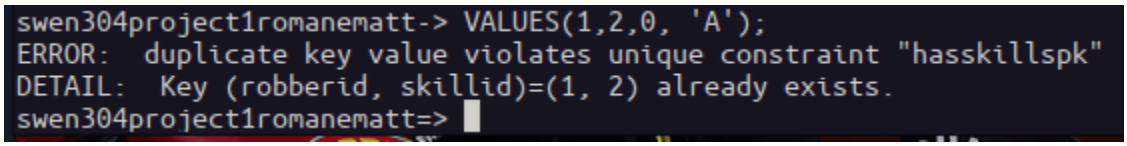
Violates UNIQUE constraint of the primary key

```
swen304project1romanematt=> INSERT INTO HasSkills (RobberId, SkillId, Preference, Grade)
swen304project1romanematt-> VALUES(1,7,1, 'A+');
ERROR:  duplicate key value violates unique constraint "hasskillspk"
DETAIL:  Key (robberid, skillid)=(1, 7) already exists.
swen304project1romanematt=>
```

b. (1, 2, 0, ‘A’)

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

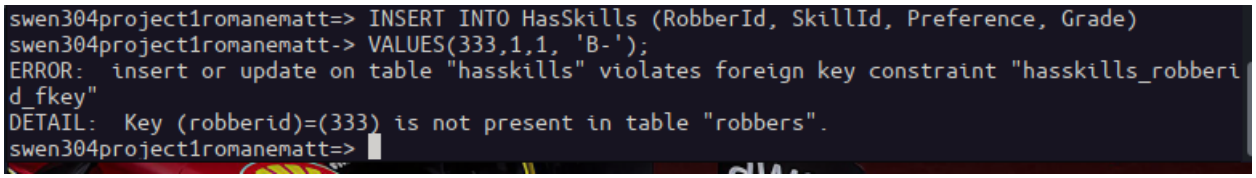
Violates UNIQUE constraint of the primary key



c. (333, 1, 1, ‘B-’)

```
INSERT INTO HasSkills (RobberId, SkillId, Preference, Grade)
VALUES(333,1,1, 'B-');
```

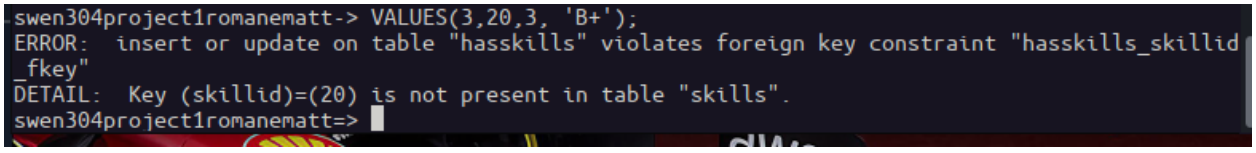
Violates FOREIGN KEY constraint as the content of the FOREIGN key must be within the relation that the FOREIGN KEY is referencing to.



d. (3, 20, 3, ‘B+’)

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

Violates FOREIGN KEY constraint as the content of the FOREIGN key must be within the relation that the FOREIGN KEY is referencing to.



In the following task, we assume that Al Capone has the robber Id 1. If Al Capone has a different Id in your database, then please change the first entry in the following tuple to be your Id of Al Capone.

9. Delete the following tuple from the Robbers table:

a. (1, 'Al Capone', 31, 2);

```
DELETE FROM Robbers
Where RobberId = 1 AND Nickname = 'Al Capone' AND Age=31 AND NoYears = 2
```

Violates FOREIGN KEY constraint as the content of the FOREIGN key must be within the relation that the FOREIGN KEY is referencing to.

```
swen304project1romanematt-> \g
ERROR:  update or delete on table "robbers" violates foreign key constraint "hasskills_robberid_
fkey" on table "hasskills"
DETAIL:  Key (robberid)=(1) is still referenced from table "hasskills".
swen304project1romanematt=>
```

Question 4

1. Retrieve BankName and City of all banks that have never been robbed.

```
SELECT b.BankName ,b.City
FROM Banks b
WHERE NOT EXISTS(SELECT 1 FROM Robberies r WHERE r.BankName = b.BankName
AND r.City = b.City)
```

```
swen304project1romanematt=> SELECT b.BankName ,b.City
swen304project1romanematt-> FROM Banks b
swen304project1romanematt-> WHERE NOT EXISTS(SELECT 1 FROM Robberies r WHERE r.BankName = b.Bank
Name
swen304project1romanematt(> AND r.City = b.City)
swen304project1romanematt-> \g
          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)

swen304project1romanematt=>
```

2. Retrieve RobberId, Nickname, Age, and all skill descriptions of all robbers who are older than 40 years old.

```
SELECT r.RobberId ,r.NickName,r.Age,s.Description
FROM Robbers r ,HasSkills hs,Skills s
WHERE r.Age >= 40 AND r.RobberId = hs.RobberId AND
s.SkillId=hs.SkillId
```

```
swen304project1romanematt=> SELECT r.RobberId ,r.NickName,r.Age,s.Description
swen304project1romanematt-> FROM Robbers r ,HasSkills hs,Skills s
swen304project1romanematt-> WHERE r.Age >= 40 AND r.RobberId = hs.RobberId AND
swen304project1romanematt-> s.SkillId=hs.SkillId;
 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 rows)

swen304project1romanematt=>
```

3. Retrieve BankName and city of all banks where Al Capone has an account. The answer should list every bank at most once.

```
SELECT DISTINCT b.BankName, b.City
FROM Banks b NATURAL JOIN HasAccounts hs NATURAL JOIN Robbers r
WHERE Hs.RobberId = r.RobberId AND r.NickName = 'Al Capone'
```

```
swen304project1romanematt=> SELECT DISTINCT b.BankName, b.City
swen304project1romanematt-> FROM Banks b NATURAL JOIN HasAccounts hs NATURAL JOIN Robbers r
swen304project1romanematt-> WHERE Hs.RobberId = r.RobberId AND r.NickName = 'Al Capone'
swen304project1romanematt-> \g
           bankname           |             city
------------------------------+------------------------------
 Bad Bank                     | Chicago
 Inter-Gang Bank              | Evanston
 NXP Bank                     | Chicago
(3 rows)

swen304project1romanematt=>
```

4. Retrieve BankName and City and NoAccounts of all banks that have no branch in Chicago. The answer should be sorted in increasing order of the number of accounts.

```
SELECT b.BankName ,b.City,b.NoAccounts
FROM Banks b
WHERE b.BankName NOT IN (SELECT b1.BankName FROM Banks b1 WHERE b1.City =
'Chicago')
ORDER BY b.NoAccounts ASC
```

```
swen304project1romanematt=> SELECT b.BankName ,b.City,b.NoAccounts
swen304project1romanematt-> FROM Banks b
swen304project1romanematt-> WHERE b.BankName NOT IN (SELECT b1.BankName FROM Banks b1 WHERE b1.C
ity =
swen304project1romanematt(> 'Chicago')
swen304project1romanematt-> ORDER BY b.NoAccounts ASC
swen304project1romanematt-> \g
           bankname           |             city             | noaccounts
------------------------------+------------------------------+------------
 Gun Chase Bank               | Burbank                      |       1999
 Bankrupt Bank                | Evanston                     |     444000
 Gun Chase Bank               | Evanston                     |     656565
(3 rows)

swen304project1romanematt=>
```

5. Retrieve RobberId, Nickname and individual total “earnings” of those robbers who have earned more than $40,000 by robbing banks. The answer should be sorted in decreasing order of the total earnings.

```
SELECT r.RobberId ,r.NickName,Rc.TotalEarnings
FROM Robbers r JOIN(SELECT RobberId , SUM(Share) AS TotalEarnings FROM
Accomplices GROUP BY RobberId) Rc
ON Rc.RobberId = r.RobberId
WHERE Rc.TotalEarnings >=40000
ORDER BY Rc.TotalEarnings DESC
```

```
swen304project1romanematt=> SELECT r.RobberId ,r.NickName,Rc.TotalEarnings
swen304project1romanematt-> FROM Robbers r JOIN(SELECT RobberId , SUM(Share) AS TotalEarnings FROM
(swen304project1romanematt(> Accomplices GROUP BY RobberId) Rc
swen304project1romanematt-> ON Rc.RobberId = r.RobberId
swen304project1romanematt-> WHERE Rc.TotalEarnings >=40000
swen304project1romanematt-> ORDER BY Rc.TotalEarnings DESC;
s robberid |         nickname          | totalearnings
----------+---------------------------+---------------
         5 | Mimmy The Mau Mau         |         70000
        15 | Boo Boo Hoff              |       61447.6
        16 | King Solomon              |       59725.8
        17 | Bugsy Siegel              |       52601.1
         3 | Lucky Luchiano            |         42667
        10 | Bonnie                    |         40085
(6 rows)

swen304project1romanematt=>
```

6. Retrieve RobberId, NickName, and the Number of Years in prison for all robbers who were in prison for more than ten years.

```
Select r.RobberId,r.NickName,r.NoYears
From Robbers r
Where r.NoYears>10
```

```
swen304project1romanematt=> Select r.RobberId,r.NickName,r.NoYears
swen304project1romanematt-> From Robbers r
swen304project1romanematt-> Where r.NoYears>10
swen304project1romanematt-> \g
 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)

swen304project1romanematt=>
```

7. Retrieve RobberId, Nickname and the Number of Years not spent in prison for all robbers who spent more than half of their life in prison.

```
Select r.RobberId,r.NickName,(r.Age-r.NoYears) AS YearsNotInPrison
From Robbers r
Where r.NoYears>(r.age/2)
```

```
swen304project1romanematt=> Select r.RobberId,r.NickName,(r.Age-r.NoYears) AS YearsNotInPrison
swen304project1romanematt-> From Robbers r
swen304project1romanematt-> Where r.NoYears>(r.age/2)
swen304project1romanematt-> \g
 robberid |         nickname          | yearsnotinprison
----------+---------------------------+------------------
        6 | Tony Genovese             |               12
       16 | King Solomon              |               31
(2 rows)

swen304project1romanematt=>
```

8. Retrieve the Description of all skills together with RobberId and NickName of all robbers who possess this skill. The answer should be ordered by skill description.

```
Select r.RobberId,r.NickName,s.Description
From Robbers r NATURAL JOIN Skills s NATURAL JOIN HasSkills hs
ORDER BY s.Description
```

```
swen304project1romanematt=> Select r.RobberId,r.NickName,s.Description
swen304project1romanematt-> From Robbers r NATURAL JOIN Skills s NATURAL JOIN HasSkills hs
swen304project1romanematt-> ORDER BY s.Description
swen304project1romanematt-> \g
 robberid |         nickname          |        description
----------+---------------------------+---------------------------
       18 | Vito Genovese             | Cooking
       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            | Driving
        6 | Tony Genovese             | Eating
       18 | Vito Genovese             | Eating
       24 | Sonny Genovese            | Explosives
        2 | Bugsy Malone              | Explosives
        4 | Anastazia                 | Guarding
       17 | Bugsy Siegel              | Guarding
       23 | Lepke Buchalter           | Guarding
        9 | Calamity Jane             | Gun-Shooting
       21 | Waxey Gordon              | Gun-Shooting
        8 | Clyde                     | Lock-Picking
        3 | Lucky Luchiano            | Lock-Picking
        7 | Dutch Schulz              | Lock-Picking
       22 | Greasy Guzik              | Lock-Picking
       24 | Sonny Genovese            | Lock-Picking
       13 | Mickey Cohen              | Money Counting
       14 | Kid Cann                  | Money Counting
       19 | Mike Genovese             | Money Counting
       15 | Boo Boo Hoff              | Planning
        8 | Clyde                     | Planning
        5 | Mimmy The Mau Mau         | Planning
        1 | Al Capone                 | Planning
       16 | King Solomon              | Planning
       22 | Greasy Guzik              | Preaching
       10 | Bonnie                    | Preaching
        1 | Al Capone                 | Preaching
        1 | Al Capone                 | Safe-Cracking
       24 | Sonny Genovese            | Safe-Cracking
       12 | Moe Dalitz                | Safe-Cracking
       11 | Meyer Lansky              | Safe-Cracking
        8 | Clyde                     | Scouting
       18 | Vito Genovese             | Scouting
(38 rows)

swen304project1romanematt=>
```

Question 5

1. Retrieve BankName and City of all banks that were not robbed in the year, in which there were robbery plans for that bank.

```
Select p.BankName,p.City
From Plans p JOIN
```

```
(SELECT b.BankName ,b.City
FROM Banks b
WHERE NOT EXISTS(SELECT 1 FROM Robberies r WHERE r.BankName = b.BankName
AND r.City = b.City) ) nr
ON p.BankName = nr.BankName AND p.City = nr.City
```

```
swen304project1romanematt=> Select p.BankName,p.City
swen304project1romanematt-> From Plans p JOIN
swen304project1romanematt-> (SELECT b.BankName ,b.City
swen304project1romanematt(> FROM Banks b
swen304project1romanematt(> WHERE NOT EXISTS(SELECT 1 FROM Robberies r WHERE r.BankName = b.Ban
kName
swen304project1romanematt(> AND r.City = b.City) ) nr
swen304project1romanematt-> ON p.BankName = nr.BankName AND p.City = nr.City
swen304project1romanematt-> \g
           bankname           |             city
------------------------------+------------------------------
 Loanshark Bank               | Deerfield
 Dollar Grabbers              | Chicago
 PickPocket Bank              | Deerfield
 Hidden Treasure              | Chicago
 PickPocket Bank              | Deerfield
(5 rows)

swen304project1romanematt=>
```

2. Retrieve RobberId and Nickname of all robbers who never robbed the banks at which they have an account.

```
Select DISTINCT r.RobberId,r.NickName
From Robbers r , HasAccounts ha ,Accomplices a
WHERE r.RobberId =ha.RobberId AND ha.RobberId = a.RobberId AND ha.BankName
=
a.BankName AND ha.City = a.City
```

```
swen304project1romanematt=> Select DISTINCT r.RobberId,r.NickName
swen304project1romanematt-> From Robbers r , HasAccounts ha ,Accomplices a
swen304project1romanematt-> WHERE r.RobberId =ha.RobberId AND ha.RobberId = a.RobberId AND ha.B
ankName =
swen304project1romanematt-> a.BankName AND ha.City = a.City
swen304project1romanematt-> \g
 robberid |         nickname
----------+--------------------------
        1 | Al Capone
        5 | Mimmy The Mau Mau
        8 | Clyde
       11 | Meyer Lansky
       17 | Bugsy Siegel
       18 | Vito Genovese
       20 | Longy Zwillman
       22 | Greasy Guzik
(8 rows)

swen304project1romanematt=>
```

3. Retrieve RobberId, Nickname, and Description of the first preferred skill of all robbers who have two or more skills.

```sql
SELECT r.RobberId, r.Nickname, s.Description
FROM Robbers r NATURAL JOIN HasSkills h NATURAL JOIN Skills s
GROUP BY r.RobberId, r.Nickname, h.Preference, s.Description
HAVING h.Preference = 1;
```

```
swen304project1romanematt=> SELECT r.RobberId, r.Nickname, s.Description
swen304project1romanematt-> FROM Robbers r NATURAL JOIN HasSkills h NATURAL JOIN Skills s
swen304project1romanematt-> GROUP BY r.RobberId, r.Nickname, h.Preference, s.Description
swen304project1romanematt-> HAVING h.Preference = 1;
 robberid |         nickname          |        description
----------+---------------------------+---------------------------
        1 | Al Capone                 | Planning
        2 | Bugsy Malone              | Explosives
        3 | Lucky Luchiano            | Lock-Picking
        4 | Anastazia                 | Guarding
        5 | Mimmy The Mau Mau         | Planning
        6 | Tony Genovese             | Eating
        7 | Dutch Schulz              | Lock-Picking
        8 | Clyde                     | Lock-Picking
        9 | Calamity Jane             | Gun-Shooting
       10 | Bonnie                    | Preaching
       11 | Meyer Lansky              | Safe-Cracking
       12 | Moe Dalitz                | Safe-Cracking
       13 | Mickey Cohen              | Money Counting
       14 | Kid Cann                  | Money Counting
       15 | Boo Boo Hoff              | Planning
       16 | King Solomon              | Planning
       17 | Bugsy Siegel              | Driving
       18 | Vito Genovese             | Scouting
       19 | Mike Genovese             | Money Counting
       20 | Longy Zwillman            | Driving
       21 | Waxey Gordon              | Gun-Shooting
       22 | Greasy Guzik              | Preaching
       23 | Lepke Buchalter           | Driving
       24 | Sonny Genovese            | Explosives
(24 rows)

swen304project1romanematt=>
```

4. Retrieve BankName, City and Date of all robberies in the city that observes the highest Share among all robberies.

```
Select r.BankName, r.City, r.Date, r.Amount
From Robberies r Natural JOIN (Select City, Max(Amount) AS MaxAmount From
Robberies
Group By BankName, City) hs
Where r.Amount = hs.MaxAmount;
```

```
swen304project1romanematt=> Select r.BankName, r.City, r.Date, r.Amount
swen304project1romanematt-> From Robberies r Natural JOIN (Select City, Max(Amount) AS MaxAmount From Robberies
swen304project1romanematt(> Group By BankName, City) hs
swen304project1romanematt-> Where r.Amount = hs.MaxAmount;
           bankname           |             city             |    date    | amount
------------------------------+------------------------------+------------+---------
 NXP Bank                     | Chicago                      | 2019-01-08 | 34302.3
 Penny Pinchers               | Chicago                      | 2016-08-30 |     900
 Penny Pinchers               | Evanston                     | 2016-08-30 | 99000.8
 Gun Chase Bank               | Evanston                     | 2016-04-30 | 18131.3
 PickPocket Bank              | Evanston                     | 2016-03-30 | 2031.99
 Loanshark Bank               | Chicago                      | 2017-11-09 |   41000
 PickPocket Bank              | Chicago                      | 2015-09-21 |    2039
 Loanshark Bank               | Evanston                     | 2016-04-20 |   20880
 Inter-Gang Bank              | Evanston                     | 2017-03-13 |   92620
 Dollar Grabbers              | Evanston                     | 2017-11-08 |    4380
 Bad Bank                     | Chicago                      | 2017-02-02 |    6020
(11 rows)
```

5. Retrieve BankName and City of all banks that were robbed by all robbers.

```
Select c1.BankName,c1.City
From Robbers r Natural JOIN
(select a.BankName,a.City,count( a.RobberId) AS c
From (Select BankName,City,RobberId From Accomplices) a
Group by BankName,City) c1 JOIN (Select Count(*) as c From Robbers )c2
ON c1.c = c2.c
```

```
swen304project1romanematt=> Select c1.BankName,c1.City
swen304project1romanematt-> From Robbers r Natural JOIN
swen304project1romanematt-> (select a.BankName,a.City,count( a.RobberId) AS c
swen304project1romanematt(> From (Select BankName,City,RobberId From Accomplices) a
swen304project1romanematt(> Group by BankName,City) c1 JOIN (Select Count(*) as c From Robbers
)c2
swen304project1romanematt-> ON c1.c = c2.c
swen304project1romanematt-> \g
 bankname | city
----------+------
(0 rows)

swen304project1romanematt=>
```

Question 6

1. 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 Approach

Nested Query

```
SELECT a.RobberId, a.NickName, SUM (b.share) AS Earnings FROM Accomplices
b
   INNER JOIN Robbers a ON a.RobberId=b.RobberId
   WHERE a.NoYears=0
   GROUP BY a.RobberId, a.NickName
   ORDER BY Earnings DESC
```

```
swen304project1romanematt=> SELECT a.RobberId, a.NickName, SUM (b.share) AS Earnings FROM Accomplic
es b
swen304project1romanematt->     INNER JOIN Robbers a ON a.RobberId=b.RobberId
swen304project1romanematt->     WHERE a.NoYears=0
swen304project1romanematt->     GROUP BY a.RobberId, a.NickName
swen304project1romanematt->     ORDER BY Earnings DESC;
 robberid |         nickname          | earnings
----------+---------------------------+----------
        5 | Mimmy The Mau Mau         |    70000
       10 | Bonnie                    |    40085
        8 | Clyde                     |    31800
       21 | Waxey Gordon              |  16447.1
       24 | Sonny Genovese            |    13664
       18 | Vito Genovese             |     6800
       14 | Kid Cann                  |     1790
(7 rows)

swen304project1romanematt=>
```

2. 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 Approach

Creating view SecurityAmount:

```
CREATE VIEW SecurityAmount as (
SELECT b.BankName, b.City, b.Security, r.Amount
FROM Banks b NATURAL JOIN Robberies r
ORDER BY b.Security);
```

```
swen304project1romanematt=> CREATE VIEW SecurityAmount as (
swen304project1romanematt(> SELECT b.BankName, b.City, b.Security, r.Amount
swen304project1romanematt(> FROM Banks b NATURAL JOIN Robberies r
swen304project1romanematt(> ORDER BY b.Security);
CREATE VIEW
swen304project1romanematt=>
```

Creating view RobberAmount

```
CREATE VIEW RobberAmount as (
SELECT Security, COUNT(Security) AS NumberRobberies,
AVG(Amount) AS AverageAmount
FROM SecurityAmount
GROUP BY Security
ORDER BY NumberRobberies DESC);
```

```
swen304project1romanematt=> CREATE VIEW RobberAmount as (
swen304project1romanematt(> SELECT Security, COUNT(Security) AS NumberRobberies,
swen304project1romanematt(> AVG(Amount) AS AverageAmount
swen304project1romanematt(> FROM SecurityAmount
swen304project1romanematt(> GROUP BY Security
swen304project1romanematt(> ORDER BY NumberRobberies DESC);
CREATE VIEW
swen304project1romanematt=>
```

Result:

```
swen304project1romanematt=> select * from robberamount;
    security     | numberrobberies |  averageamount
-----------------+-----------------+------------------
 excellent       |              12 | 39238.0831705729
 weak            |               4 |           2299.5
 very good       |               3 | 12292.4269205729
 good            |               2 |             3980
(4 rows)

swen304project1romanematt=>
```

Nested Query

```
SELECT Security AS SecurityLevel, COUNT(Security) AS NumberRobberies,
AVG(Amount) AS AverageAmount
FROM (SELECT b.BankName, b.City, b.Security, r.Amount
FROM Robberies r NATURAL JOIN Banks b) AS sec
GROUP BY Security
ORDER BY NumberRobberies DESC
```

```
swen304project1romanematt=> SELECT Security AS SecurityLevel, COUNT(Security) AS NumberRobbe
ries,
swen304project1romanematt-> AVG(Amount) AS AverageAmount
swen304project1romanematt-> FROM (SELECT b.BankName, b.City, b.Security, r.Amount
swen304project1romanematt(> FROM Robberies r NATURAL JOIN Banks b) AS sec
swen304project1romanematt-> GROUP BY Security
swen304project1romanematt-> ORDER BY NumberRobberies DESC;
  securitylevel  | numberrobberies |  averageamount
-----------------+-----------------+------------------
 excellent       |              12 | 39238.0831705729
 weak            |               4 |           2299.5
 very good       |               3 | 12292.4269205729
 good            |               2 |             3980
(4 rows)

swen304project1romanematt=>
```