**Name:** Rhea D'Souza **Std. ID:** 300627012 **UID:** dsouzrhea

# Project 1
## SWEN304

## QUESTION 1: Defining your Database [15 Marks]

**You are expected to design the database with appropriate choices of:**

- *Keys.* Choose appropriate attributes or sets of attributes to be keys and decide on the primary key.
- *Foreign keys*. Determine all foreign keys and decide what should be done if the tuple referred to is deleted or modified.
- *Attribute constraints.* Choose suitable basic data types and additional constraints, such as NOT NULL constraints, CHECK constraints or DEFAULT values.

**Your answer to Question 1 should include:**

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.
2. A list of all your CREATE TABLE statements.
3. A justification for your choice of actions on delete or on update for each foreign key.
4. A brief justification for your choice of attribute constraints (other than the basic data).

```sql
CREATE TABLE Banks (
    bank_name VARCHAR(100) NOT NULL,
    city VARCHAR(100) NOT NULL,
    no_accounts INT CHECK (no_accounts > 0),
    sec_level VARCHAR(20) CHECK (sec_level IN ('weak', 'good', 'very good', 'excellent')),
    PRIMARY KEY (bank_name, city)
);
```

**Keys:**
Bank_name and city are the primary key/candidate keys for this table as they create unique combinations based off the assumptions that are given in the assignment brief.

**Foreign Keys + Update and Delete:**
The Banks table does not have any foreign keys as it is an originating node which most of the other tables depend upon due to its primary key.

**Attribute Constraints:**

- The no_accounts check to make sure that at least one account is there with the bank for it to be in the table otherwise there is no point in the bank being added as it wouldn't attract any robbers due to it likely making little to no business.
- The security level attribute has a constraint on the things that are considered valid inputs to make sure that only a specific set of words can be inserted to reflect the information this attribute is meant to store.
- The only other attribute constraints given are NOT NULL to the primary key values as to prevent any violation of them, and a character limit to the varying character type columns to prevent an overload or incorrect information from being stored.

```sql
CREATE TABLE Robberies (
    bank_name VARCHAR(20) NOT NULL,
    city VARCHAR(20) NOT NULL,
    date_robbed DATE NOT NULL,
    amount DECIMAL(20, 2),
    PRIMARY KEY (bank_name, city, date_robbed),
    FOREIGN KEY (bank_name, city) REFERENCES Banks(bank_name, city)
);
```

**Keys:**
Robberies takes the bank_name, city and date_robbed primary keys in order to prevent against duplicated in the multiple robberies made to the same bank.
**Foreign Keys + Update and Delete:**
The bank_name and city are selected are you
**Attribute Constraints:**

- Has not null attribute checks on all the primary key attributes which to prevent duplicate instances due to NULL being considered a valid input.

```sql
CREATE TABLE RobberyPlans (
    bank_name VARCHAR(20) NOT NULL,
    city VARCHAR(20) NOT NULL,
    no_robbers INT CHECK (no_robbers >= 1),
    planned_date DATE NOT NULL,
    PRIMARY KEY (bank_name, city, planned_date),
    FOREIGN KEY (bank_name, city) REFERENCES Banks(bank_name, city)
);
```

**Keys:**
Bank_name, city and planned_date is the primary key as it is highly unlikely for multiple robbery plans to take place at the same bank and on the same day.
**Foreign Keys + Update and Delete:**
On delete and update of the Banks parent instance it will error as it may impact the actual information that is meant to be related to this instance and may delete important information by accident.
**Attribute Constraints:**

- No_robbers have a check to make sure that at least 1 robber is a part of the plan because it wouldn't make sense for a plan to be created in the first place if there is no robber planning to perform it.
- All other attributes were set to not null, though this was likely unnecessary because it would violate a primary key constraint if it was null or a foreign key constraint if the value of the attribute did not exist within the referenced tables.

```sql
CREATE TABLE Robbers (
    robber_id SERIAL PRIMARY KEY,
    nickname VARCHAR(20),
    age INT CHECK (age >= 0),
    no_years INT CHECK (no_years >= 0)
);
```

**Keys:**
Robber_id is the primary key and it is set to have a serial input as to auto-increment and generate the robber_id value so that all primary keys are unique.
**Foreign Keys + Update and Delete:**
No foreign keys, so this is irrelevant.
**Attribute Constraints:**

- I have come to realise upon hindsight, right before handing this in that I should have made sure the years in prison should have had the check to ensure that the inputted value is between 0 and their given age as that would have made the inputted information more in touch with reality. However, I only checked to ensure the number of years and age of the robber is greater than or equal to 0 in so that no negative values would be inputted into the database. I did attempt to put age as greater than or equal to 18 as that is usually when people develop

their criminal record but there was a 14-year-old robber with in the datafiles so I changed it to 0 or more.

```
CREATE TABLE Skills (
    skill_id SERIAL PRIMARY KEY,
    description VARCHAR(20) NOT NULL UNIQUE
);
```

**Keys:**
Skill_id is the only primary key attribute and it made to be SERIAL so that the skill attribute auto-increments with each insertion of a skill description.
**Foreign Keys + Update and Delete:**
There are no foreign keys so, this is irrelevant.
**Attribute Constraints:**

- Description is set to NOT NULL and UNIQUE as the Skills Table is basically used as a map between the skill-id and the actual skill that is used and for this to be consistent the Skills table needs to take unique values for both.

```
CREATE TABLE HasSkills (
    robber_id INT NOT NULL,
    skill_id INT NOT NULL,
    preference INT CHECK (preference >= 1 AND preference <= 3),
    grade CHAR(2) CHECK (grade IN ('A+', 'A', 'A-', 'B+', 'B', 'B-', 'C+', 'C', 'C-', 'D', 'E', 'F', 'K', 'P', 'G', 'J', 'L', '
    PRIMARY KEY (robber_id, skill_id),
    FOREIGN KEY (robber_id) REFERENCES Robbers(robber_id),
    FOREIGN KEY (skill_id) REFERENCES Skills(skill_id)
);
```

**Keys:**
Robber_id and skill_id are the primary key as they are both unique on their own and because this table is used to match skills to the robbers who are allow to have multiple skills and we don't want them to have the same skill saved twice for the same robber, but we do want to allow for different robbers to have the same skill.
**Foreign Keys + Update and Delete:**
The foreign keys do not have any cascade or restriction on them other than the default ones as to prevent the deletion of the child skill instances in HasSkills. This will also help to prevent the accidental deletion of a skill if a child instance goes under the radar when deleting a Skill in the skills table.
**Attribute Constraints:**

- Grade has a CHAR(2) constraint for length AND check for a set of valid input values which are reflective of the VUW grading system, to ensure the correct information is being inputted into this attribute.
- Preference has a check to make sure the preference number inputted is between the values of 1 and 3, this was only assumed based on the values withing the files.
- robber_id and skill_id have Not Null constraint, to ensure the value is not left null, though this would likely be prevented anyways due to them being primary keys and foreign keys.

```
CREATE TABLE HasAccounts (
    robber_id INT,
    bank_name VARCHAR(20),
    city VARCHAR(20),
    PRIMARY KEY (robber_id, bank_name, city),
    FOREIGN KEY (robber_id) REFERENCES Robbers(robber_id),
    FOREIGN KEY (bank_name, city) REFERENCES Banks(bank_name, city)
);
```

**Keys:**
Primary keys are robber_id, bank_name and city. Bank_name and city will have unique instances but because a robber may have multiple accounts with different banks, and because there is a chance that a bank branch may have multiple robbers with an account with them; it is having all 3 of them is necessary to create unique instances.
**Foreign Keys + Update and Delete:**

Robber_id is from the Robbers table and bank_name and city are from the Banks table. These both have a foreign key constraint to prevent their parent instance from being deleted if a child instance within the HasAccounts table exists. This is important especially for the Banks table as many tables rely on its primary keys as foreign keys.

**Attribute Constraints:**

- Should have put a NOT NULL constraint on them all in order to ensure there are no duplicates occur within the table, though this is likely already prevented by the key constraints.

```
CREATE TABLE Accomplices (
    robber_id INT,
    bank_name VARCHAR(20),
    city VARCHAR(20),
    date_robbed DATE,
    robbery_share DECIMAL(10,2) CHECK (robbery_share >= 0),
    PRIMARY KEY (robber_id, bank_name, city, date_robbed),
    FOREIGN KEY (robber_id) REFERENCES Robbers(robber_id),
    FOREIGN KEY (bank_name, city, date_robbed) REFERENCES Robberies(bank_name, city, date_robbed) ON DELETE CASCADE ON UPDATE C
);
```

**Keys:**

Takes all foreign keys as a primary key as they are all meant to be unique, and therefore the combination of all should help define unique instances.

Robber_id and bank_name, city, and date_robbed are the primary key as I don't want to log a robber twice for the same robbery, but I do want to be able to have multiple robbers be logged for the same robbery and I do want to log the robber for any other robbery that they may have been an accomplice in.

**Foreign Keys + Update and Delete:**

Foreign keys are robber_id from Robbers and bank_name, city and date_robbed all from the Robberies table as accomplices are mean to be linked to a robbery.

- I did not implement any update and deletion constraints for the Robbers foreign key because I do not think the instance within the Robbers should be able to be deleted if there are child instances that exist as valuable information would be lost if the child instances were deleted in a cascade deletion.
- I did however, allow for the Robberies to cascade delete and update the Accomplices instances as if there is no Robbery, then that instance of an accomplice is not needed due to the Robbery not taking place. It would also result in inconsistencies in the database system if the child instances were not deleted with it.

**Attribute Constraints:**

- Checked the robbery_share attribute to ensure that none of the inputted values are negative as it would be unrealistic for robbers to lose money rather than gain money.
- Another constraint that I could have added that I noticed only hindsight is a check on the date to make sure the date that was inputted was not set in the future. However, I feel like this might have taken too much time to implement as I am not familiar with creating dynamic constraints.
- In addition to these checks, I think a good attribute constraint that I did not end up implementing would be to CHECK the 'amount' attribute in the Robberies database and ensure that the Accomplices instances that are linked to that do not exceed have their 'robbery_share' attribute summing to a value greater than the amount stolen from a robbery.

## QUESTION 2: Populating your Database [15 Marks]

**Name:** Rhea D'Souza **Std. ID:** 300627012 **UID:** dsouzrhea

Banks, Robbers and Skills are the parents of most of the other tables therefore they are the ones that are created before their dependent tables are created. I decided to populate Banks and then it's dependent table, Robberies as the HasAccounts file needed to derive information from this table. Following this I populated Robbers as it was the next simplest command to just copy the data over, and then I populated RobberyPlans, though that could have been performed the other way round as RobberyPlans is only dependent on Banks.

After that I took on the more complicated tables to populate, with Skills as HasSkills is dependent on Skills and Robbers. This was followed by the population of HasSkills as their information was derived from the same file and a temporary table was already created. Following the population of the order of the remaining tables doesn't matter as all dependent parent tables have already been created, but I populated HasAccounts and Accomplices last as they derived information from the Banks, Robberies and Robbers Tables.

**Populating Banks:**

```
project1_dsouzrhea=> \copy Banks FROM ./datafiles/banks_24.data
COPY 20
project1_dsouzrhea=> SELECT * FROM Banks
project1_dsouzrhea-> ;
    bank_name    |   city    | no_accounts | sec_level
-----------------+-----------+-------------+-----------
 NXP Bank        | Chicago   |     1593311 | very good
 Bankrupt Bank   | Evanston  |      444000 | weak
 Loanshark Bank  | Evanston  |     7654321 | excellent
 Loanshark Bank  | Deerfield |     3456789 | very good
 Loanshark Bank  | Chicago   |      121212 | excellent
 Inter-Gang Bank | Chicago   |      100000 | excellent
 Inter-Gang Bank | Evanston  |      555555 | excellent
 NXP Bank        | Evanston  |      656565 | excellent
 Penny Pinchers  | Chicago   |      156165 | weak
 Dollar Grabbers | Chicago   |       56005 | very good
 Penny Pinchers  | Evanston  |      130013 | excellent
 Dollar Grabbers | Evanston  |      909090 | good
 Gun Chase Bank  | Evanston  |      656565 | excellent
 Gun Chase Bank  | Burbank   |        1999 | weak
 PickPocket Bank | Evanston  |        2000 | very good
 PickPocket Bank | Deerfield |        6565 | excellent
 PickPocket Bank | Chicago   |      130013 | weak
 Hidden Treasure | Chicago   |      999999 | excellent
 Bad Bank        | Chicago   |        6000 | weak
 Outside Bank    | Chicago   |        5000 | good
(20 rows)
```

Data was copied directly from the file into the table.

**Populating Robberies:**

```
project1_dsouzrhea=> \copy Robberies FROM ./datafiles/robberies_24.data
COPY 21
project1_dsouzrhea=> SELECT * FROM Robberies
project1_dsouzrhea-> ;
    bank_name    |   city   | date_robbed |  amount
-----------------+----------+-------------+----------
 NXP Bank        | Chicago  | 2019-01-08  | 34302.30
 Loanshark Bank  | Evanston | 2019-02-28  | 19990.00
 Loanshark Bank  | Chicago  | 2019-03-30  | 21005.00
 Inter-Gang Bank | Evanston | 2018-02-14  | 52619.00
 Penny Pinchers  | Chicago  | 2016-08-30  |   900.00
 Penny Pinchers  | Evanston | 2016-08-30  | 99000.80
 Gun Chase Bank  | Evanston | 2016-04-30  | 18131.30
 PickPocket Bank | Evanston | 2016-03-30  |  2031.99
 PickPocket Bank | Chicago  | 2018-02-28  |   239.00
 Loanshark Bank  | Evanston | 2017-04-20  | 10990.00
 Inter-Gang Bank | Evanston | 2016-02-16  | 72620.00
 Penny Pinchers  | Evanston | 2017-10-30  |  9000.50
 PickPocket Bank | Evanston | 2018-01-30  |   542.99
 Loanshark Bank  | Chicago  | 2017-11-09  | 41000.00
 Penny Pinchers  | Evanston | 2019-05-30  | 13000.40
 PickPocket Bank | Chicago  | 2015-09-21  |  2039.00
 Loanshark Bank  | Evanston | 2016-04-20  | 20880.00
 Inter-Gang Bank | Evanston | 2017-03-13  | 92620.00
 Dollar Grabbers | Evanston | 2017-11-08  |  4380.00
 Dollar Grabbers | Evanston | 2017-06-28  |  3580.00
 Bad Bank        | Chicago  | 2017-02-02  |  6020.00
(21 rows)
```

Data was copied directly from the file into the table.

**Populating Robbers:**

**Name:** Rhea D'Souza **Std. ID:** 300627012 **UID:** dsouzrhea

```
project1_dsouzrhea=> \copy Robbers(nickname, age, no_years) FROM './datafiles/robbers_24.data' WITH DELIMITER E'\t';
COPY 24
project1_dsouzrhea=> SELECT * FROM Robbers
project1_dsouzrhea-> ;
 robber_id |      nickname      | age | no_years
-----------+--------------------+-----+----------
         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)
```

Data was copied directly from the file into the table, but with specified order because of the way my table was constructed, and with a delimiter specifying that tabs separate the attributes due to spaces between certain attributes.

**Populating RobberyPlans (`Plans` in the project brief):**

```
project1_dsouzrhea=> \copy RobberyPlans(bank_name, city, planned_date, no_robbers) FROM './datafiles/plans_24.data' WITH DELIMITER E'\t';
COPY 11
project1_dsouzrhea=> SELECT * RobberyPlans
project1_dsouzrhea-> ;
ERROR:  syntax error at or near "RobberyPlans"
LINE 1: SELECT * RobberyPlans
                 ^
project1_dsouzrhea=> SELECT * FROM RobberyPlans;
    bank_name    |   city    | no_robbers | planned_date
-----------------+-----------+------------+--------------
 NXP Bank        | Chicago   |          5 | 2019-10-30
 Loanshark Bank  | Deerfield |          4 | 2019-11-15
 Inter-Gang Bank | Evanston  |          4 | 2019-12-31
 Dollar Grabbers | Chicago   |          3 | 2019-12-10
 Gun Chase Bank  | Evanston  |          6 | 2019-10-30
 PickPocket Bank | Deerfield |          6 | 2019-12-15
 PickPocket Bank | Chicago   |          2 | 2020-03-10
 Hidden Treasure | Chicago   |          5 | 2020-01-11
 NXP Bank        | Chicago   |          5 | 2019-10-10
 Bad Bank        | Chicago   |          2 | 2020-02-02
 PickPocket Bank | Deerfield |          6 | 2019-11-30
(11 rows)
```

Data was copied directly from the file into the table, but with specified order because of the way my table was constructed, and with a delimiter specifying that tabs separate the attributes due to spaces between certain attributes.

**Populating Skills:**

```
project1_dsouzrhea=> CREATE TEMP TABLE temp_has_skills (
project1_dsouzrhea(>     robber_nickname VARCHAR(100),
project1_dsouzrhea(>     description VARCHAR(100),
project1_dsouzrhea(>     preference INT,
project1_dsouzrhea(>     grade CHAR(2)
project1_dsouzrhea(> );
CREATE TABLE
```

Created a temporary table to hold the values for the hasskills_24.data doc because that way it would be easier to create instances using a single SQL statement as I need to get the robber_nickname from the Robbers table.

```
project1_dsouzrhea=> \copy temp_has_skills FROM './datafiles/hasskills_24.data'
COPY 38
project1_dsouzrhea=> SELECT description FROM temp_has_skills;
   description
-----------------
 Planning
 Safe-Cracking
 Preaching
 Planning
 Driving
 Guarding
 Preaching
 Planning
 Explosives
 Driving
 Guarding
 Gun-Shooting
 Lock-Picking
 Scouting
 Planning
 Lock-Picking
 Driving
 Preaching
 Lock-Picking
 Money Counting
 Planning
 Driving
 Guarding
 Driving
 Lock-Picking
 Driving
 Safe-Cracking
```

I then copied the data into the temporary table and tested a query to see whether it would be possible to extract only the description for the Skills table.

```
project1_dsouzrhea=> INSERT INTO Skills(description) SELECT description FROM temp_has_skills;
ERROR:  duplicate key value violates unique constraint "skills_description_key"
DETAIL:  Key (description)=(Planning) already exists.
project1_dsouzrhea=> INSERT INTO Skills(description) SELECT DISTINCT description FROM temp_has_skills;
INSERT 0 12
```

I then attempted to extract the data but failed to take into consideration the UNIQUE constraint I put on the description, so I had to query it again made sure to add the DISTINCT keyword to only extract the unique descriptions. The resulting table is below. Once the table has been populated the temporary table is dropped.
(Note: I deleted and recreated the Skills table to fix the skill_id incrementation as it is set to SERIAL making it autoincrement, and due to the previous 4 being deleted it started skill_id at 5. So, this was done for consistency though should not have any effect on the data yet as it the Skills table does not get referenced yet.)

```
project1_dsouzrhea=> SELECT * FROM skills;
 skill_id |  description
----------+----------------
        1 | Explosives
        2 | Guarding
        3 | Planning
        4 | Cooking
        5 | Gun-Shooting
        6 | Lock-Picking
        7 | Safe-Cracking
        8 | Preaching
        9 | Driving
       10 | Eating
       11 | Scouting
       12 | Money Counting
(12 rows)
```

**Name:** Rhea D'Souza **Std. ID:** 300627012 **UID:** dsouzrhea

## Populating HasSkills:

```
project1_dsouzrhea=> INSERT INTO HasSkills (robber_id, skill_id, preference, grade)
project1_dsouzrhea-> SELECT r.robber_id, s.skill_id, hs.preference, hs.grade
project1_dsouzrhea-> FROM temp_has_skills hs
project1_dsouzrhea-> JOIN Robbers r ON r.nickname = hs.robber_nickname
project1_dsouzrhea-> JOIN Skills s ON s.description = hs.description;
INSERT 0 38
project1_dsouzrhea=> DROP TABLE temp_has_skills CASCADE;
DROP TABLE
```

The temp_has_skills was used to insert the preference and grade. Additionally, it was used to retrieve the robber_id using the robbers nickname and skill_id using the skill description. Once the table has been populated the temporary table is dropped.

```
project1_dsouzrhea=> SELECT * FROM HasSkills;
 robber_id | skill_id | preference | grade
-----------+----------+------------+-------
         1 |       12 |          3 | A+
         1 |       11 |          2 | C+
         1 |        7 |          1 | A+
         2 |        5 |          1 | A
         3 |       13 |          2 | B+
         3 |       10 |          1 | B+
```

## Populating HasAccount:

```
project1_dsouzrhea=> CREATE TEMP TABLE temp_has_accounts (
project1_dsouzrhea(>   nickname VARCHAR(20),
project1_dsouzrhea(>   bank_name VARCHAR(20),
project1_dsouzrhea(>   city VARCHAR(20)
project1_dsouzrhea(> );
CREATE TABLE
project1_dsouzrhea=> \copy temp_has_accounts from ./datafiles/hasaccounts_24.data WITH DELIMITER E'\t';
COPY 31
project1_dsouzrhea=> INSERT INTO HasAccounts (robber_id, bank_name, city)
project1_dsouzrhea->  SELECT r.robber_id, b.bank_name, b.city
project1_dsouzrhea->  FROM temp_has_accounts ha
project1_dsouzrhea->  JOIN Robbers r ON ha.nickname = r.nickname
project1_dsouzrhea->  JOIN Banks b ON ha.bank_name = b.bank_name AND ha.city = b.city;
INSERT 0 31
project1_dsouzrhea=> DROP TABLE temp_has_accounts CASCADE;
DROP TABLE
```

A temporary table was created, and information was inserted into the HasAccount table using the robber's nickname to derive the robber_id from the Robbers table. Once the table has been populated the temporary table is dropped.

```
project1_dsouzrhea=> SELECT * FROM HasAccounts;
 robber_id |    bank_name     |   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
```

## Populating Accomplices:

**Name:** Rhea D'Souza **Std. ID:** 300627012 **UID:** dsouzrhea

```
project1_dsouzrhea=> DROP TABLE temp_has_accounts CASCADE;
DROP TABLE
project1_dsouzrhea=>  CREATE TEMP TABLE temp_accomplices (
project1_dsouzrhea(>    nickname VARCHAR(20),
project1_dsouzrhea(>    bank_name VARCHAR(20),
project1_dsouzrhea(>    city VARCHAR(20),
project1_dsouzrhea(>    date_robbed DATE,
project1_dsouzrhea(>    robbery_share DECIMAL(10, 2)
project1_dsouzrhea(>  );
CREATE TABLE
project1_dsouzrhea=> \copy temp_accomplices from ./datafiles/accomplices_24.data WITH DELIMITER E'\t';
COPY 76
project1_dsouzrhea=>  INSERT INTO Accomplices (robber_id, bank_name, city, date_robbed, robbery_share)
project1_dsouzrhea->  SELECT r.robber_id, a.bank_name, a.city, a.date_robbed, a.robbery_share
project1_dsouzrhea->  FROM temp_accomplices a
project1_dsouzrhea->  JOIN Robbers r ON a.nickname = r.nickname;
INSERT 0 76
```

A temporary table was created and then used to populate the table by deriving the robber_id value from the Robbers table using the robber's nickname and transfer the other information which doesn't need to be derived into the relevant instances. Once the table has been populated the temporary table is dropped.

```
project1_dsouzrhea=> SELECT * FROM Accomplices;
 robber_id |    bank_name    |   city   | date_robbed | robbery_share
-----------+-----------------+----------+-------------+---------------
         1 | Bad Bank        | Chicago  | 2017-02-02  |       3010.00
         1 | NXP Bank        | Chicago  | 2019-01-08  |       6406.00
         1 | Loanshark Bank  | Evanston | 2019-02-28  |       4997.00
         1 | Loanshark Bank  | Chicago  | 2019-03-30  |       4201.00
         1 | Inter-Gang Bank | Evanston | 2016-02-16  |      12103.00
         1 | Inter-Gang Bank | Evanston | 2018-02-14  |       8769.00
         2 | NXP Bank        | Chicago  | 2019-01-08  |       2300.00
         3 | Penny Pinchers  | Evanston | 2016-08-30  |      16500.00
```

**Name:** Rhea D'Souza **Std. ID:** 300627012 **UID:** dsouzrhea

## QUESTION 3: Checking your Database [10 Marks]

The changes made to the database from Q1-Q3 have been stored in the 'swen304_p1_Q1AndQ3_final.sql' file that was submitted (this includes the population of the database)

**Inserting into Skills:**

```
project1_dsouzrhea=> INSERT INTO Skills VALUES (21, 'Driving');
ERROR:  duplicate key value violates unique constraint "skills_description_key"
DETAIL:  Key (description)=(Driving) already exists.
```

Violated a candidate/primary key constraint.

**Inserting into Banks:**

```
project1_dsouzrhea=> INSERT INTO Banks(bank_name, city, no_accounts, sec_level) VALUES ('Loanshark Bank', 'Evanston', 100, 'very good');
ERROR:  duplicate key value violates unique constraint "banks_pkey"
DETAIL:  Key (bank_name, city)=(Loanshark Bank, Evanston) already exists.
```

Violated a primary key constraint.

**Inserting into Robberies:**

```
project1_dsouzrhea=> INSERT INTO Robberies (bank_name, city, date_robbed, amount)
project1_dsouzrhea-> VALUES ('NXP Bank', 'Chicago', '2019-01-08', 1000);
ERROR:  duplicate key value violates unique constraint "robberies_pkey"
DETAIL:  Key (bank_name, city, date_robbed)=(NXP Bank, Chicago, 2019-01-08) already exists.
```

Violated a primary key constraint.

**Deleting from Skills:**

```
project1_dsouzrhea=> DELETE FROM Skills WHERE skill_id = 1 AND description = 'Driving';
DELETE 0
project1_dsouzrhea=> DELETE FROM Skills WHERE description = 'Driving';
ERROR:  update or delete on table "skills" violates foreign key constraint "hasskills_skill_id_fkey" on table "hasskills"
DETAIL:  Key (skill_id)=(13) is still referenced from table "hasskills".
project1_dsouzrhea=> DELETE FROM Skills WHERE skill_id = 1;
DELETE 0
project1_dsouzrhea=> select * from skills
project1_dsouzrhea-> ;
 skill_id |  description
----------+---------------
        5 | Explosives
        6 | Guarding
        7 | Planning
        8 | Cooking
        9 | Gun-Shooting
       10 | Lock-Picking
       11 | Safe-Cracking
       12 | Preaching
       13 | Driving
       14 | Eating
       15 | Scouting
       16 | Money Counting
(12 rows)

project1_dsouzrhea=> DELETE FROM Skills WHERE skill_id = 5;
ERROR:  update or delete on table "skills" violates foreign key constraint "hasskills_skill_id_fkey" on table "hasskills"
DETAIL:  Key (skill_id)=(5) is still referenced from table "hasskills".
```

Violated a foreign key constraint.

**Deleting from Banks:**

```
project1_dsouzrhea=> DELETE FROM Banks WHERE bank_name = 'PickPocket Bank' AND city = 'Evanston' AND no_accounts = 2000 AND sec_level = 'very good';
ERROR:  update or delete on table "banks" violates foreign key constraint "robberies_bank_name_city_fkey" on table "robberies"
DETAIL:  Key (bank_name, city)=(PickPocket Bank, Evanston) is still referenced from table "robberies".
```

Violated a foreign key constraint.

**Delete from Robberies:**

```
project1_dsouzrhea=> DELETE FROM Robberies WHERE bank_name = 'Loanshark Bank' AND city = 'Chicago' AND date_robbed = '' AND amount = '';
ERROR:  invalid input syntax for type date: ""
LINE 1: ...hark Bank' AND city = 'Chicago' AND date_robbed = '' AND amo...
                                                             ^
```

Not sure if this was the intent but it was not able to be deleted as the syntax was incorrect due to amount being a decimal type.

**Insert into Robbers:**

```
project1_dsouzrhea=> INSERT INTO Robbers (robber_id, nickname, age, no_years) VALUES (1, 'Shotgun', 70, 0);
ERROR:  duplicate key value violates unique constraint "robbers_pkey"
DETAIL:  Key (robber_id)=(1) already exists.
```

Violated the primary key constraint.

```
project1_dsouzrhea=> INSERT INTO Robbers (robber_id, nickname, age, no_years) VALUES (999, 'Jail Mouse', 25, 35);
INSERT 0 1
project1_dsouzrhea=> REMOVE FROM Robbers WHERE robber_id = 999;
ERROR:  syntax error at or near "REMOVE"
LINE 1: REMOVE FROM Robbers WHERE robber_id = 999;
        ^
project1_dsouzrhea=> DELETE FROM Robbers WHERE robber_id = 999;
DELETE 1
project1_dsouzrhea=> ALTER TABLE Robbers ALTER COLUMN robber_id ADD GENERATED ALWAYS AS IDENTITY;
ERROR:  column "robber_id" of relation "robbers" already has a default value
project1_dsouzrhea=> ALTER TABLE Robbers ALTER COLUMN robber_id DROP DEFAULT;
ALTER TABLE
project1_dsouzrhea=> ALTER TABLE Robbers ALTER COLUMN robber_id ADD GENERATED ALWAYS AS IDENTITY;
ALTER TABLE
project1_dsouzrhea=> INSERT INTO Robbers (robber_id, nickname, age, no_years) VALUES (999, 'Jail Mouse', 25, 35);
ERROR:  cannot insert a non-DEFAULT value into column "robber_id"
DETAIL:  Column "robber_id" is an identity column defined as GENERATED ALWAYS.
HINT:  Use OVERRIDING SYSTEM VALUE to override.
```

Didn't error the first time because I set it to SERIAL and that made sure to allow for override of the default value, so I changed the constraint to prevent the default value from being changed to something other than the default auto incremented one.

**Insert into HasSkills:**

```
project1_dsouzrhea=> INSERT INTO HasSkills VALUES (1, 7, 1, 'A+');
ERROR:  duplicate key value violates unique constraint "hasskills_pkey"
DETAIL:  Key (robber_id, skill_id)=(1, 7) already exists.
project1_dsouzrhea=> INSERT INTO HasSkills 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 ).
project1_dsouzrhea=> INSERT INTO HasSkills VALUES (999, 1, 1, 'B-');
ERROR:  insert or update on table "hasskills" violates foreign key constraint "hasskills_robber_id_fkey"
DETAIL:  Key (robber_id)=(999) is not present in table "robbers".
project1_dsouzrhea=> INSERT INTO HasSkills VALUES (3, 20, 3, 'B+');
ERROR:  insert or update on table "hasskills" violates foreign key constraint "hasskills_skill_id_fkey"
DETAIL:  Key (skill_id)=(20) is not present in table "skills".
```

First one violated primary key constraint.
Second one violated the preference constraint that I set in place.
And the third and fourth ones violated the robber_id foreign key constraint.

**Delete from Robbers:**

```
project1_dsouzrhea=> DELETE FROM robbers where robber_id = 1 AND nickname = 'Al Capone' AND age = 31 AND no_years = 2;
ERROR:  update or delete on table "robbers" violates foreign key constraint "hasskills_robber_id_fkey" on table "hasskills"
DETAIL:  Key (robber_id)=(1) is still referenced from table "hasskills".
```

Violated a foreign key constraint.

**Name:** Rhea D'Souza **Std. ID:** 300627012 **UID:** dsouzrhea

## QUESTION 4: Simple Database Queries [24 Marks]

### BankName and City of banks that have been NOT been robbed:

```
project1_dsouzrhea=> SELECT DISTINCT b.bank_name, b.city
project1_dsouzrhea-> FROM Banks b
project1_dsouzrhea-> LEFT JOIN Robberies r ON b.bank_name = r.bank_name AND b.city = r.city
project1_dsouzrhea-> WHERE r.bank_name IS NULL;
    bank_name    |   city
-----------------+-----------
 Gun Chase Bank  | Burbank
 EasyLoan Bank   | Evanston
 Inter-Gang Bank | Chicago
 Hidden Treasure | Chicago
 PickPocket Bank | Deerfield
 Bankrupt Bank   | Evanston
 Outside Bank    | Chicago
 Loanshark Bank  | Deerfield
 Dollar Grabbers | Chicago
 NXP Bank        | Evanston
(10 rows)
```

### RobberID, Nickname and Years NOT spent in prison:

```
project1_dsouzrhea=> SELECT robber_id, nickname, (age - no_years) AS years_out_of_prison FROM Robbers;
 robber_id |     nickname      | years_out_of_prison
-----------+-------------------+---------------------
         1 | Al Capone         |                  29
         2 | Bugsy Malone      |                  27
         3 | Lucky Luchiano    |                  27
         4 | Anastazia         |                  33
         5 | Mimmy The Mau Mau |                  18
         6 | Tony Genovese     |                  12
         7 | Dutch Schulz      |                  33
         8 | Clyde             |                  20
         9 | Calamity Jane     |                  41
        10 | Bonnie            |                  19
        11 | Meyer Lansky      |                  28
        12 | Moe Dalitz        |                  38
        13 | Mickey Cohen      |                  21
        14 | Kid Cann          |                  14
        15 | Boo Boo Hoff      |                  41
        16 | King Solomon      |                  31
        17 | Bugsy Siegel      |                  35
        18 | Vito Genovese     |                  66
        19 | Mike Genovese     |                  35
        20 | Longy Zwillman    |                  29
        21 | Waxey Gordon      |                  15
        22 | Greasy Guzik      |                  24
        23 | Lepke Buchalter   |                  24
        24 | Sonny Genovese    |                  39
(24 rows)
```

### RobberID, Nickname, Age and SkillDescription for those that are not younger than 35:

```
project1_dsouzrhea=> SELECT r.robber_id, r.nickname, r.age, s.description
project1_dsouzrhea-> FROM Robbers r
project1_dsouzrhea-> JOIN HasSkills hs ON r.robber_id = hs.robber_id
project1_dsouzrhea-> JOIN Skills s ON hs.skill_id = s.skill_id
project1_dsouzrhea-> WHERE r.age >= 35;
 robber_id |    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
        19 | Mike Genovese  |  35 | Money Counting
        20 | Longy Zwillman |  35 | Driving
        24 | Sonny Genovese |  39 | Lock-Picking
        24 | Sonny Genovese |  39 | Safe-Cracking
        24 | Sonny Genovese |  39 | Explosives
(20 rows)
```

**Retrieve a set of all banks that Al Capone has an account with:**

```
project1_dsouzrhea=> SELECT DISTINCT ha.bank_name, ha.city
project1_dsouzrhea-> FROM HasAccounts ha
project1_dsouzrhea-> JOIN Robbers r ON ha.robber_id = r.robber_id AND r.nickname = 'Al Capone';
    bank_name    |   city
-----------------+----------
 Bad Bank        | Chicago
 Inter-Gang Bank | Evanston
 NXP Bank        | Chicago
(3 rows)
```

**Retrieve all the robbers that have earned a total of $50000 or more from the shares they earnt from each robbery they participated in:**

```
project1_dsouzrhea=> SELECT r.robber_id, r.nickname, SUM(a.robbery_share) AS total_share
project1_dsouzrhea-> FROM Robbers r
project1_dsouzrhea-> JOIN Accomplices a ON r.robber_id = a.robber_id
project1_dsouzrhea-> GROUP BY r.robber_id, r.nickname
project1_dsouzrhea-> HAVING SUM(a.robbery_share) >= 50000
project1_dsouzrhea-> ORDER BY total_share DESC;
 robber_id |     nickname      | total_share
-----------+-------------------+-------------
         5 | Mimmy The Mau Mau |    70000.00
        15 | Boo Boo Hoff      |    61447.61
        16 | King Solomon      |    59725.80
        17 | Bugsy Siegel      |    52601.10
(4 rows)
```

**Retrieve all Skill descriptions with their associated robber_id and nicknames of the robbers that have those skills and order them by skill description:**

**Name:** Rhea D'Souza **Std. ID:** 300627012 **UID:** dsouzrhea

```
project1_dsouzrhea=> SELECT s.description, r.robber_id, r.nickname
project1_dsouzrhea-> FROM Skills s
project1_dsouzrhea-> JOIN HasSkills hs ON s.skill_id = hs.skill_id
project1_dsouzrhea-> JOIN Robbers r ON hs.robber_id = r.robber_id
project1_dsouzrhea-> ORDER BY s.description;
  description   | robber_id |     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)
```

**Name:** Rhea D'Souza **Std. ID:** 300627012 **UID:** dsouzrhea

# QUESTION 5: Complex Database Queries [20 Marks]

## Nested approach:

### 1.1

```
project1_dsouzrhea=> SELECT DISTINCT rb.bank_name, rb.city
project1_dsouzrhea-> FROM Robberies rb
project1_dsouzrhea-> WHERE (rb.bank_name, rb.city) NOT IN (
project1_dsouzrhea(>         SELECT rb.bank_name, rb.city
project1_dsouzrhea(>         FROM Robbers r
project1_dsouzrhea(>         JOIN Accomplices a ON r.robber_id = a.robber_id
project1_dsouzrhea(>         WHERE rb.bank_name = a.bank_name AND rb.city = a.city
project1_dsouzrhea(> );
 bank_name | city
-----------+------
(0 rows)
```

### 2.1

```
project1_dsouzrhea=> SELECT r.robber_id, r.nickname, s.description AS preferred_skill
project1_dsouzrhea-> FROM Robbers r
project1_dsouzrhea-> JOIN (
project1_dsouzrhea(>     SELECT robber_id, skill_id, preference, ROW_NUMBER() OVER (PARTITION BY robber_id ORDER BY preference) AS rank
project1_dsouzrhea(>     FROM HasSkills
project1_dsouzrhea(> ) hs ON r.robber_id = hs.robber_id
project1_dsouzrhea-> JOIN Skills s ON hs.skill_id = s.skill_id
project1_dsouzrhea-> WHERE hs.rank = 1 AND r.robber_id IN (
project1_dsouzrhea(>     SELECT robber_id
project1_dsouzrhea(>     FROM HasSkills
project1_dsouzrhea(>     GROUP BY robber_id
project1_dsouzrhea(>     HAVING COUNT(*) >= 2
project1_dsouzrhea(> );
 robber_id |     nickname      | preferred_skill
-----------+-------------------+-----------------
         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)
```

### 3.1

```
project1_dsouzrhea=> SELECT bank_name, city
project1_dsouzrhea-> FROM RobberyPlans
project1_dsouzrhea-> WHERE (bank_name, city) NOT IN (
project1_dsouzrhea(>     SELECT bank_name, city
project1_dsouzrhea(>     FROM Robberies
project1_dsouzrhea(>     WHERE EXTRACT(YEAR FROM RobberyPlans.planned_date) = EXTRACT(YEAR FROM Robberies.date_robbed
project1_dsouzrhea(> );
    bank_name    |   city
-----------------+-----------
 Loanshark Bank  | Deerfield
 Inter-Gang Bank | Evanston
 Dollar Grabbers | Chicago
 Gun Chase Bank  | Evanston
 PickPocket Bank | Deerfield
 PickPocket Bank | Chicago
 Hidden Treasure | Chicago
 Bad Bank        | Chicago
 PickPocket Bank | Deerfield
(9 rows)
```

### 4.1

```
project1_dsouzrhea=> SELECT robber_id, nickname
project1_dsouzrhea-> FROM Robbers
project1_dsouzrhea-> WHERE robber_id NOT IN (
project1_dsouzrhea(>     SELECT DISTINCT h.robber_id
project1_dsouzrhea(>     FROM HasAccounts h
project1_dsouzrhea(>     INNER JOIN Robberies r ON h.bank_name = r.bank_name AND h.city = r.city
project1_dsouzrhea(> );
 robber_id |    nickname
-----------+-----------------
         6 | Tony Genovese
         7 | Dutch Schulz
        10 | Bonnie
        13 | Mickey Cohen
        15 | Boo Boo Hoff
        16 | King Solomon
        19 | Mike Genovese
        23 | Lepke Buchalter
        24 | Sonny Genovese
(9 rows)
```

## Stepwise approach:

### 1.2

```
project1_dsouzrhea=> CREATE VIEW TotalRobbers AS
project1_dsouzrhea-> SELECT COUNT(DISTINCT robber_id) AS total_robbers
project1_dsouzrhea-> FROM Robbers;
CREATE VIEW
project1_dsouzrhea=> CREATE VIEW RobberiesPerBank AS
project1_dsouzrhea-> SELECT bank_name, city, COUNT(DISTINCT robber_id) AS robbers_count
project1_dsouzrhea-> FROM Accomplices
project1_dsouzrhea-> GROUP BY bank_name, city;
CREATE VIEW
project1_dsouzrhea=> CREATE VIEW BanksRobbedByAll AS
project1_dsouzrhea-> SELECT bank_name, city
project1_dsouzrhea-> FROM RobberiesPerBank
project1_dsouzrhea-> WHERE robbers_count = (SELECT total_robbers FROM TotalRobbers);
CREATE VIEW
project1_dsouzrhea=> SELECT * FROM BanksRobbedByAll;
 bank_name | city
-----------+------
(0 rows)
```

### 2.2

```
project1_dsouzrhea=> CREATE VIEW RobberSkillsCount AS
project1_dsouzrhea-> SELECT robber_id, COUNT(*) AS num_skills
project1_dsouzrhea-> FROM HasSkills
project1_dsouzrhea-> GROUP BY robber_id;
CREATE VIEW
project1_dsouzrhea=>
project1_dsouzrhea=> CREATE VIEW RobbersWithMultipleSkills AS
project1_dsouzrhea-> SELECT robber_id
project1_dsouzrhea-> FROM RobberSkillsCount
project1_dsouzrhea-> WHERE num_skills >= 2;
CREATE VIEW
project1_dsouzrhea=>
project1_dsouzrhea=> CREATE VIEW FirstPreferredSkills AS
project1_dsouzrhea-> SELECT robber_id, skill_id
project1_dsouzrhea-> FROM HasSkills
project1_dsouzrhea-> WHERE preference = 1;
CREATE VIEW
project1_dsouzrhea=>
project1_dsouzrhea=> SELECT r.robber_id, r.nickname, s.description
project1_dsouzrhea-> FROM Robbers r
project1_dsouzrhea-> JOIN FirstPreferredSkills f ON r.robber_id = f.robber_id
project1_dsouzrhea-> JOIN Skills s ON f.skill_id = s.skill_id
project1_dsouzrhea-> WHERE r.robber_id IN (SELECT robber_id FROM RobbersWithMultipleSkills);
 robber_id |     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)
```

**Name:** Rhea D'Souza **Std. ID:** 300627012 **UID:** dsouzrhea

## 3.2

```
project1_dsouzrhea=> CREATE VIEW PlannedYear AS
project1_dsouzrhea-> SELECT bank_name, city, EXTRACT(YEAR FROM planned_date) AS planned_year
project1_dsouzrhea-> FROM RobberyPlans;
CREATE VIEW
project1_dsouzrhea=>
project1_dsouzrhea=> CREATE VIEW RobberyYear AS
project1_dsouzrhea-> SELECT bank_name, city, EXTRACT(YEAR FROM date_robbed) AS robbed_year
project1_dsouzrhea-> FROM Robberies;
CREATE VIEW
project1_dsouzrhea=>
project1_dsouzrhea=> CREATE VIEW BanksNotRobbed AS
project1_dsouzrhea-> SELECT p.bank_name, p.city
project1_dsouzrhea-> FROM PlannedYear p
project1_dsouzrhea-> LEFT JOIN RobberyYear r ON p.bank_name = r.bank_name AND p.city = r.city AND p.planned_year = r.robbed_yea
project1_dsouzrhea-> WHERE r.bank_name IS NULL AND r.city IS NULL;
CREATE VIEW
project1_dsouzrhea=>
project1_dsouzrhea=> SELECT bank_name, city
project1_dsouzrhea-> FROM BanksNotRobbed;
    bank_name    |   city
-----------------+-----------
 Loanshark Bank  | Deerfield
 Inter-Gang Bank | Evanston
 Dollar Grabbers | Chicago
 Gun Chase Bank  | Evanston
 PickPocket Bank | Deerfield
 PickPocket Bank | Chicago
 Hidden Treasure | Chicago
 Bad Bank        | Chicago
 PickPocket Bank | Deerfield
(9 rows)
```

## 4.2

```
project1_dsouzrhea=> CREATE VIEW RobbedRobbers AS
project1_dsouzrhea-> SELECT DISTINCT h.robber_id
project1_dsouzrhea-> FROM HasAccounts h
project1_dsouzrhea-> INNER JOIN Robberies r ON h.bank_name = r.bank_name AND h.city = r.city;
CREATE VIEW
project1_dsouzrhea=> CREATE VIEW RobbersNotRobbedBanks AS
project1_dsouzrhea-> SELECT robber_id, nickname
project1_dsouzrhea-> FROM Robbers
project1_dsouzrhea-> WHERE robber_id NOT IN (
project1_dsouzrhea(>     SELECT robber_id FROM RobbedRobbers
project1_dsouzrhea(> );
CREATE VIEW
project1_dsouzrhea=> SELECT * FROM RobbersNotRobbedBanks;
 robber_id |    nickname
-----------+-----------------
         6 | Tony Genovese
         7 | Dutch Schulz
        10 | Bonnie
        13 | Mickey Cohen
        15 | Boo Boo Hoff
        16 | King Solomon
        19 | Mike Genovese
        23 | Lepke Buchalter
        24 | Sonny Genovese
(9 rows)
```

**Name:** Rhea D'Souza **Std. ID:** 300627012 **UID:** dsouzrhea

## QUESTION 6: Even More Database Queries [16 Marks]

**Nested Approach:**

TASK 1

```
project1_dsouzrhea=> SELECT r.nickname
project1_dsouzrhea-> FROM Robbers r
project1_dsouzrhea-> JOIN Accomplices a ON r.robber_id = a.robber_id
project1_dsouzrhea-> WHERE r.robber_id IN (
project1_dsouzrhea(>     SELECT robber_id
project1_dsouzrhea(>     FROM Accomplices
project1_dsouzrhea(>     GROUP BY robber_id
project1_dsouzrhea(>     HAVING COUNT(*) > (
project1_dsouzrhea(>         SELECT AVG(num_robberies)
project1_dsouzrhea(>         FROM (
project1_dsouzrhea(>             SELECT COUNT(*) AS num_robberies
project1_dsouzrhea(>             FROM Accomplices
project1_dsouzrhea(>             GROUP BY robber_id
project1_dsouzrhea(>         ) AS avg_robberies
project1_dsouzrhea(>     ) AND r.no_years = 0
project1_dsouzrhea(> )
project1_dsouzrhea-> GROUP BY r.nickname
project1_dsouzrhea-> ORDER BY SUM(a.robbery_share) DESC;
    nickname
----------------
 Bonnie
 Clyde
 Sonny Genovese
(3 rows)

project1_dsouzrhea=> CREATE VIEW ActiveRobbersWithoutPrison AS
project1_dsouzrhea-> SELECT r.nickname
project1_dsouzrhea-> FROM Robbers r
project1_dsouzrhea-> JOIN Accomplices a ON r.robber_id = a.robber_id
project1_dsouzrhea-> WHERE r.robber_id IN (
project1_dsouzrhea(>     SELECT robber_id
project1_dsouzrhea(>     FROM Accomplices
project1_dsouzrhea(>     GROUP BY robber_id
project1_dsouzrhea(>     HAVING COUNT(*) > (
project1_dsouzrhea(>         SELECT AVG(num_robberies)
project1_dsouzrhea(>         FROM (
project1_dsouzrhea(>             SELECT COUNT(*) AS num_robberies
project1_dsouzrhea(>             FROM Accomplices
project1_dsouzrhea(>             GROUP BY robber_id
project1_dsouzrhea(>         ) AS avg_robberies
project1_dsouzrhea(>     ) AND r.no_years = 0
project1_dsouzrhea(> )
project1_dsouzrhea-> GROUP BY r.nickname
project1_dsouzrhea-> ORDER BY SUM(a.robbery_share) DESC;
CREATE VIEW
project1_dsouzrhea=> SELECT * FROM  ActiveRobbersWithoutPrison;
    nickname
----------------
 Bonnie
 Clyde
 Sonny Genovese
(3 rows)
```

**Name:** Rhea D'Souza **Std. ID:** 300627012 **UID:** dsouzrhea

TASK 2

```
project1_dsouzrhea=> CREATE VIEW SecurityLevelStats AS
project1_dsouzrhea-> SELECT rb.sec_level, COUNT(rb.bank_name) AS total_robberies, AVG(rb.amount) AS average_amount
project1_dsouzrhea-> FROM (
project1_dsouzrhea(>     SELECT b.bank_name, b.city, b.sec_level, r.amount
project1_dsouzrhea(>     FROM Robberies r
project1_dsouzrhea(>     JOIN Banks b ON r.bank_name = b.bank_name AND r.city = b.city
project1_dsouzrhea(> ) AS rb
project1_dsouzrhea-> JOIN (SELECT bank_name, city, sec_level FROM Banks) AS b ON rb.bank_name = b.bank_name AND rb.city = b.city
project1_dsouzrhea-> GROUP BY rb.sec_level;
CREATE VIEW
project1_dsouzrhea=>
project1_dsouzrhea=> SELECT * FROM SecurityLevelStats;
 sec_level | total_robberies |     average_amount
-----------+-----------------+------------------------
 weak      |               4 |  2299.5000000000000000
 good      |               2 |  3980.0000000000000000
 very good |               3 | 12292.4266666666666667
 excellent |              12 |     39238.083333333333
(4 rows)
```

## Sequential Approach:

TASK 1

```
project1_dsouzrhea=> CREATE VIEW CountRobberies AS
project1_dsouzrhea-> SELECT robber_id, COUNT(*) AS num_robberies
project1_dsouzrhea-> FROM Accomplices
project1_dsouzrhea-> GROUP BY robber_id;
CREATE VIEW
project1_dsouzrhea=>
project1_dsouzrhea=> CREATE VIEW AverageRobberies AS
project1_dsouzrhea-> SELECT AVG(num_robberies) AS avg_robberies
project1_dsouzrhea-> FROM CountRobberies;
CREATE VIEW
project1_dsouzrhea=>
project1_dsouzrhea=> CREATE VIEW RobbersAboveAverage AS
project1_dsouzrhea-> SELECT c.robber_id
project1_dsouzrhea-> FROM CountRobbersRobberies c
project1_dsouzrhea-> GROUP BY c.robber_id, c.num_robberies
project1_dsouzrhea-> HAVING c.num_robberies > (SELECT avg_robberies FROM AverageRobberies);
CREATE VIEW
project1_dsouzrhea=>
project1_dsouzrhea=> CREATE VIEW ActiveRobbersWithoutPrison AS
project1_dsouzrhea-> SELECT r.robber_id, r.nickname
project1_dsouzrhea-> FROM Robbers r
project1_dsouzrhea-> JOIN RobbersAboveAverage ra ON r.robber_id = ra.robber_id
project1_dsouzrhea-> WHERE r.no_years = 0;
CREATE VIEW
project1_dsouzrhea=>
project1_dsouzrhea=> CREATE VIEW ActiveRobbersSorted AS
project1_dsouzrhea-> SELECT a.nickname AS total_earnings
project1_dsouzrhea-> FROM ActiveRobbersWithoutPrison a
project1_dsouzrhea-> JOIN Accomplices ac ON a.robber_id = ac.robber_id
project1_dsouzrhea-> GROUP BY a.robber_id, a.nickname
project1_dsouzrhea-> ORDER BY SUM(ac.robbery_share) DESC;
CREATE VIEW
project1_dsouzrhea=>
project1_dsouzrhea=> SELECT * FROM ActiveRobbersSorted;
 total_earnings
----------------
 Bonnie
 Clyde
 Sonny Genovese
(3 rows)
```

TASK 2

```
project1_dsouzrhea=> CREATE VIEW RobberiesWithSecurityLevels AS
project1_dsouzrhea-> SELECT b.bank_name, b.city, b.sec_level, r.amount
project1_dsouzrhea-> FROM Robberies r
project1_dsouzrhea-> JOIN Banks b ON r.bank_name = b.bank_name AND r.city = b.city;
CREATE VIEW
project1_dsouzrhea=>
project1_dsouzrhea=> CREATE VIEW DistinctBanks AS
project1_dsouzrhea-> SELECT bank_name, city, sec_level
project1_dsouzrhea-> FROM Banks;
CREATE VIEW
project1_dsouzrhea=>
project1_dsouzrhea=> CREATE VIEW SecurityLevelStats AS
project1_dsouzrhea-> SELECT
project1_dsouzrhea->     rb.sec_level,
project1_dsouzrhea->     COUNT(rb.bank_name) AS total_robberies,
project1_dsouzrhea->     AVG(rb.amount) AS average_amount
project1_dsouzrhea-> FROM RobberiesWithSecurityLevels rb
project1_dsouzrhea-> JOIN DistinctBanks b ON rb.bank_name = b.bank_name AND rb.city = b.city
project1_dsouzrhea-> GROUP BY rb.sec_level;
CREATE VIEW
project1_dsouzrhea=>
project1_dsouzrhea=> SELECT * FROM SecurityLevelStats;
 sec_level | total_robberies |     average_amount
-----------+-----------------+------------------------
 weak      |               4 |  2299.5000000000000000
 good      |               2 |  3980.0000000000000000
 very good |               3 | 12292.4266666666666667
 excellent |              12 |     39238.083333333333
(4 rows)
```