SWEN 304
Database System Engineering

Project 1, David Thomsen, 300052209

Question 1: Defining the Database

Table: Banks

Constraint: positive_accounts
Justification: It is impossible for a bank to have less than zero accounts.

Constraint: security_range
Justification: Banks can only have one of these four security levels. If a bank was inserted with the security level ‘poor’, this would be missed if a search was done for banks with a ‘weak’ security level, even though they might be equivalent.

Constraint: bank_pk (primary key)
Justification: Bank Names are not unique, there can be banks in different cities with the same name. City is not unique, there can be multiple banks with different names in the same city. But there can only be one bank of each Bank Name and City, so this is a minimal primary key.

```
CREATE TABLE banks (
  bank_name CHAR(32) NOT NULL,
  city CHAR(32) NOT NULL,
  no_accounts integer NOT NULL DEFAULT 0,
  security CHAR(16) NOT NULL,
  CONSTRAINT positive_accounts CHECK (no_accounts >= 0),
  CONSTRAINT security_range
    CHECK (security in ('weak','good','very good','excellent')),
  CONSTRAINT bank_pk PRIMARY KEY (bank_name, city)
);
```

Table: Robberies

Constraint: positive_amount
Justification: Cannot rob less than zero money from the bank. This would just be an anonymous deposit.

Constraint: robberies_pk (primary key)
Justification: The primary key for Banks is Bank Name and City, so these must be part of the primary key. In addition, the same Bank can be robbed on different Dates, so date must also be part of the primary key.

Constraint: robberies_fk (foreign key)
Justification: Each robbery corresponds to a particular Bank, and each Bank has a key of Bank Name and City, so these must be part of the foreign key. Robberies should not restrict Banks from being deleted, so I didn't use Restrict on Delete. I used Cascade for deletion because to a robbery with no reference to a bank is of no interest. I used Cascade for updating because a Bank with a different Bank Name is still the same bank.
Assumption: A robbery with only a date and amount is of no interest.

```
CREATE TABLE robberies (
  bank_name CHAR(32) NOT NULL,
  city CHAR(32) NOT NULL,
  date DATE NOT NULL,
  amount numeric NOT NULL,
  CONSTRAINT positive_amount CHECK (amount >= 0),
  CONSTRAINT robberies_pk PRIMARY KEY (bank_name, city, date),
  CONSTRAINT robberies_fk FOREIGN KEY (bank_name, city)
    REFERENCES banks (bank_name, city) ON UPDATE CASCADE ON DELETE
CASCADE
);
```

Table: Plans

Constraint: positive_no_robbers
Justification: A bank can only be robbed by one or more robbers.

Constraint: plans_pk (primary key)
Justification: Bank Name and City are part of the primary key of Banks. Because a Bank can be planned to be robbed on more than one occasion in the future, I also include Planned Date to create a unique key set.
Assumption: A bank cannot be planned to be robbed twice on the same day.

Constraint: plans_fk (foreign key)
Justification: This table refers to Banks, which has Bank Name and City as a primary key, so these columns are used as a foreign key here. When a Bank changes its Bank Name or City, these changes cascade because it is still the same Bank that the gang plans to rob. When a Bank is delete, the plans must also be deleted because the gang cannot plan to rob a Bank that no longer exists.

Notes: I have allowed Planned Date to be Null because a robber might be able to Plan to rob a bank at some point in the future, but not have a set time yet.

```
CREATE TABLE plans (
  bank_name CHAR(32) NOT NULL,
  city CHAR(32) NOT NULL,
  planned_date DATE NULL,
  no_robbers integer NOT NULL DEFAULT 1,
  CONSTRAINT positive_no_robbers CHECK (no_robbers >= 1),
  CONSTRAINT plans_pk PRIMARY KEY (bank_name, city, planned_date),
  CONSTRAINT plans_fk FOREIGN KEY (bank_name, city)
    REFERENCES banks (bank_name, city) ON UPDATE CASCADE ON DELETE
CASCADE
);
```

Table: Robbers

Constraint: positive_age
Justification: A robber cannot have an age of less than zero years.
Assumption: There is no minimum age for a bank robber. If a robber has just been born, it can still take a share of a robbery if it is taken along for the robbery.

Constraint: no_years_less_than_age
Justification: A robber cannot have been in jail for longer than they have been alive.
Assumption: Babies can go to jail.

```
CREATE TABLE robbers (
  robber_id SERIAL PRIMARY KEY,
  nickname CHAR(32) NOT NULL,
  age integer NOT NULL,
  no_years integer NOT NULL DEFAULT 0,
  CONSTRAINT positive_age CHECK (age >= 0),
  CONSTRAINT no_years_less_than_age CHECK (no_years < age)
);
```

Table: Skills

Constraint: unique_description
Justification: Descriptions must be unique.

```
CREATE TABLE skills (
  skill_id SERIAL PRIMARY KEY,
  description CHAR(32) NOT NULL,
  CONSTRAINT unique_description UNIQUE (description)
);
```

Table: Has Skills

Constraint: preference_range
Justification: Preference can only be the range [1,2,3].

Constraint: grade_range
Justification: Grade must in the set ['C-','C','C+','B-','B','B+','A-','A','A+'].

Constraint: has_skills_pk (primary key)
Justification: A robber can have multiple skills but they cannot have skills with the same preference. It would also have been valid to have a primary key of (robber_id, skill_id), but redundant to have a primary key of (robber_id, skill_id, preference)

Constraint: has_skills_robber_fk (foreign key)
Justification: The robber_id refers to a particular Robber in the Robbers table. When a Robber is deleted, their Has Skills should be as well.

Constraint: has_skills_skill_fk (foreign key)
Justification: The skill_id refers to a particular Skill on the Skills table. On Delete Restrict means that a skill cannot be deleted while a Robber still has it.

Constraint: unique_robber_skill
Justification: A robber cannot have the same skill twice on their list of preferences.

Constraint: unique_robber_preference
Justification: A robber cannot have the preference twice.

```
CREATE TABLE has_skills (
  robber_id integer NOT NULL,
  skill_id integer NOT NULL,
  preference integer NOT NULL,
  grade CHAR(2) NOT NULL,
  CONSTRAINT preference_range CHECK (preference BETWEEN 1 AND 3),
  CONSTRAINT grade_range
    CHECK (Grade IN ('C-', 'C', 'C+', 'B-', 'B', 'B+', 'A-', 'A', 'A+')),
  CONSTRAINT has_skills_pk PRIMARY KEY (robber_id, preference),
  CONSTRAINT has_skills_robber_fk FOREIGN KEY (robber_id)
    REFERENCES robbers (robber_id) ON DELETE CASCADE,
  CONSTRAINT has_skills_skill_fk FOREIGN KEY (skill_id)
    REFERENCES skills (skill_id) ON DELETE RESTRICT,
  CONSTRAINT unique_robber_skill UNIQUE (robber_id, skill_id),
  CONSTRAINT unique_robber_preference UNIQUE (robber_id, preference)
);
```

Table: Has Accounts

Constraint: has_account_pk (primary key)
Justification: A Robber can have multiple accounts at different Banks branch, and different Bank branches can have accounts belonging to multiple Robbers, but a Robber can only have one account at each Bank Branch.

Constraint: has_accounts_robber_fk (foreign key)
Justification: The Robber Id refers to a row in the Robbers table. The Robbers cannot be deleted while they still have an account open, so On Delete is restricted.

Constraint: has_accounts_bank_fk (foreign key)
Justification: The Bank Name and City refer to the primary key in the Banks table. On Update, the changes from Bank should cascade to the Has Accounts table. A Bank should not be able to be deleted while a Robber still has an account there, so On Delete is restricted.

```
CREATE TABLE has_accounts (
  robber_id integer NOT NULL,
  bank_name CHAR(32) NOT NULL,
  city CHAR(32) NOT NULL,
  CONSTRAINT has_account_pk PRIMARY KEY (robber_id, bank_name,
city),
  CONSTRAINT has_accounts_robber_fk FOREIGN KEY (robber_id)
    REFERENCES robbers (robber_id) ON DELETE RESTRICT,
  CONSTRAINT has_accounts_bank_fk FOREIGN KEY (bank_name, city)
    REFERENCES banks (bank_name, city) ON UPDATE CASCADE ON DELETE
RESTRICT
);
```

Table: Accomplices

Constraint: positive_share
Justification: A Robber can only have a positive share from a bank robbery. It is possible that their Share is zero, however.

Constraint: accomplices_pk (primary key)
Justification: This has a long primary key because it is possible for multiple Robbers to take a share for a Robbery, it is possible for for a multiple Bank branches to be robbed on the same day by the same Robber(s), and it is possible for the same Bank to be robbed on different days by the same Robber(s). No part of this key can be removed.

Constraint: accomplices_robber_fk (foreign key)
Justification: Robber Id refers to a column in the Robbers table. When a Robber is deleted, that deletion should cascade into this table.

Constraint: accomplices_bank_fk (foreign key)

Justification: The share that an Accomplice took from a particular Bank branch on a particular Day refers to a particular Robbery in the Robberies table. When a Robbery updates its details, this change should Cascade so that this table refers to the new name of the bank on the Banks table. When a Robbery is deleted, this should cascade and the share in the Accomplices should be deleted.

```
CREATE TABLE accomplices (
  robber_id integer NOT NULL,
  bank_name CHAR(32) NOT NULL,
  city CHAR(32) NOT NULL,
  robbery_date DATE NOT NULL,
  share numeric NOT NULL
  CONSTRAINT
    positive_share CHECK (share >= 0),
  CONSTRAINT accomplices_pk
    PRIMARY KEY (robber_id, bank_name, city, robbery_date),
  CONSTRAINT accomplices_robber_fk
    FOREIGN KEY (robber_id) REFERENCES robbers (robber_id) ON
DELETE CASCADE,
  CONSTRAINT accomplices_bank_fk
    FOREIGN KEY (bank_name, city, robbery_date)
    REFERENCES robberies (bank_name, city, date)
    ON UPDATE CASCADE ON DELETE CASCADE
);
```

Question 2: Populating your Database with Data

Step 1: Banks
Justification: Banks do not have any foreign keys so they should be added first.

Step 2: Robberies
Step 3: Plans
Justification: Robberies and Plans also only reference Banks as a foreign key so they can be added second and third.

Step 4: Robbers
Justification: Robbers also don't have any foreign keys, I have added them fourth because they require an extra step to upload.

Step 5: Has Skills
Justification: Has Skills references Robbers as a foreign key, so must be done after Robbers.

Step 6: Accomplices
Justifications: Accomplices references both Robbers and Robberies as foreign keys, so must

be done after those two steps.

Step 7: Has Accounts references both Robbers and Banks, so must be done after those steps.

```
\COPY banks FROM ~/Pro1/banks_16.data
\COPY robberies FROM ~/Pro1/robberies_16.data
\COPY plans FROM ~/Pro1/plans_16.data

CREATE TABLE robbers_temp (
  nickname CHAR(32) NOT NULL,
  age integer NOT NULL,
  no_years integer NOT NULL
);

\COPY robbers_temp FROM ~/Pro1/robbers_16.data

INSERT INTO robbers (nickname, age, no_years) SELECT nickname, age,
no_years
  FROM robbers_temp;

DROP TABLE robbers_temp;

CREATE TABLE has_skills_temp (
  nickname CHAR(32) NOT NULL,
  description CHAR(32) NOT NULL,
  preference integer NOT NULL,
  grade CHAR(2) NOT NULL
);

\copy has_skills_temp from ~/Pro1/hasskills_16.data

INSERT INTO skills (description) SELECT DISTINCT description FROM
has_skills_temp;

INSERT INTO has_skills
  SELECT r.robber_id, s.skill_id, h_s_t.preference, h_s_t.grade
  FROM robbers r, skills s, has_skills_temp h_s_t
  WHERE r.nickname = h_s_t.nickname
  AND s.description = h_s_t.description;

DROP TABLE has_skills_temp;

CREATE TABLE accomplices_temp (
  nickname CHAR(32) NOT NULL,
  bank_name CHAR(32) NOT NULL,
  city CHAR(32) NOT NULL,
```

```
  robbery_date DATE NOT NULL,
  share numeric NOT NULL
);

\copy accomplices_temp from ~/Pro1/accomplices_16.data

INSERT INTO accomplices
  SELECT r.robber_id, a_t.bank_name, a_t.city, a_t.robbery_date,
a_t.share
  FROM robbers r, accomplices_temp a_t
  WHERE r.nickname = a_t.nickname;

DROP TABLE accomplices_temp;

CREATE TABLE has_accounts_temp (
  nickname CHAR(32) NOT NULL,
  bank_name CHAR(32) NOT NULL,
  city CHAR(32) NOT NULL
);

\copy has_accounts_temp from ~/Pro1/hasaccounts_16.data

INSERT INTO has_accounts
  SELECT r.robber_id, h_a_t.bank_name, h_a_t.city
  FROM robbers r, has_accounts_temp h_a_t
  WHERE r.nickname = h_a_t.nickname;

DROP TABLE has_accounts_temp;
```

Question 3: Checking your Database

1(a)

```
INSERT INTO banks VALUES ('Loanshark Bank', 'Evanston', 100, 'very
good');
```

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

This primary key already exists in the Banks table.

1(b)

```
INSERT INTO banks VALUES ('Loanshark Bank', 'Evanston', 100, 'very
good');
```

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

Bank cannot have less than zero accounts.

1(c)

```
INSERT INTO banks VALUES ('EasyLoan Bank', 'Evanston', 100,
'poor');
```

```
ERROR:  new row for relation "banks" violates check constraint
"security_range"
DETAIL:  Failing row contains (EasyLoan Bank                    ,
Evanston                        , 5, poor            ).
```

Attempting to set up a bank with ‘poor’ security, which is not in the range ['weak','good','very good','excellent'].

2(a)

```
INSERT INTO skills VALUES (20, 'Guarding');
```

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

All of the Descriptions for the Skills must be unique, and ‘Guarding’ already exists.

3(a)

```
INSERT INTO robbers VALUES (1, 'Shotgun', 70, 0);
```

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

A Robber with the primary key ‘1’ already exists. We should instead try inserting the values (‘Shotgun’, 70, 0) and let the table generate the ID.

3(b)

```
INSERT INTO robbers VALUES (333, 'Jail Mouse', 25, 35);
```

```
ERROR:  new row for relation "robbers" violates check constraint
"no_years_less_than_age"
DETAIL:  Failing row contains (333, Jail Mouse
, 25, 35).
```

‘Jail Mouse’ cannot have spent more years in prison than he has been alive.

4(a)

```
INSERT INTO has_skills VALUES (333, 1, 1, 'B-');
```

```
ERROR:  insert or update on table "has_skills" violates foreign key
constraint "has_skills_robber_fk"
DETAIL:  Key (robber_id)=(333) is not present in table "robbers".
```

There is no Robber with the ID ‘333’.

4(b)

```
INSERT INTO has_skills VALUES (3, 20, 3, 'B+');
```

```
ERROR:  insert or update on table "has_skills" violates foreign key
constraint "has_skills_skill_fk"
DETAIL:  Key (skill_id)=(20) is not present in table "skills".
```

There is no Skill with the ID ‘20’.

4(c)

```
INSERT INTO has_skills VALUES (1, 9, 1, 'A+');
```

```
ERROR:  duplicate key value violates unique constraint
"has_skills_pk"
DETAIL:  Key (robber_id, preference)=(1, 1) already exists.
```

The Robber with the ID ‘1’ already has the Preference ‘1’.

4(d)

```
INSERT INTO has_skills VALUES (1, 2, 0, 'A');
```

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

Preference must be in range 1 to 3, cannot be zero.

5(a)

```
INSERT INTO robberies VALUES ('NXP Bank', 'Chicago', '2009-01-08',
1000);
```

```
ERROR:  duplicate key value violates unique constraint
"robberies_pk"
DETAIL:  Key (bank_name, city, date)=(NXP Bank
, Chicago                          , 2009-01-08) already exists.
```

A Robbery for this Bank branch on this Day already exists.

6(a)

```
DELETE FROM banks WHERE bank_name='PickPocket Bank' AND
city='Evanston' AND no_accounts=2000 AND security='very good';
```

```
ERROR:  update or delete on table "banks" violates foreign key
constraint "has_accounts_bank_fk" on table "has_accounts"
DETAIL:  Key (bank_name, city)=(PickPocket Bank                  ,
Evanston                         ) is still referenced from table
"has_accounts".
```

Cannot delete Bank while Robber(s) still have Account(s) there.

6(b)

```
DELETE FROM banks WHERE bank_name='Gun Chase Bank' AND
city='Evanston' AND no_accounts=656565 AND security='excellent';
```

```
ERROR:  update or delete on table "banks" violates foreign key
constraint "has_accounts_bank_fk" on table "has_accounts"
DETAIL:  Key (bank_name, city)=(Gun Chase Bank                   ,
Evanston                         ) is still referenced from table
```

```
"has_accounts".
```

Cannot delete Bank while Robber(s) still have Account(s) there.

7(a)

```
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 "has_accounts_robber_fk" on table "has_accounts"
DETAIL:  Key (robber_id)=(1) is still referenced from table
"has_accounts".
```

Cannot delete a Robber while they still have an Account at a Bank

8(a)

```
DELETE FROM skills WHERE skill_id=1 AND description='Driving';
```

```
DELETE 0
```

Tuple does not match as skill_id 1 has description ‘Safe-Cracking, so nothing is deleted.

Question 4: Simple Database Queries

1

```
SELECT bank_name, security FROM banks
WHERE city='Chicago' AND no_accounts>9000;
```

```
            bank_name              |     security
-----------------------------------+------------------
 NXP Bank                          | very good
 Loanshark Bank                    | excellent
 Inter-Gang Bank                   | excellent
 Penny Pinchers                    | weak
 Dollar Grabbers                   | very good
 PickPocket Bank                   | weak
 Hidden Treasure                   | excellent
(7 rows)
```

2

```
SELECT DISTINCT (bank_name) FROM banks NATURAL JOIN has_accounts
NATURAL JOIN robbers WHERE nickname='Calamity Jane';
```

```
           bank_name
---------------------------------
 Bad Bank
 Dollar Grabbers
 PickPocket Bank
(3 rows)
```

3

```
SELECT bank_name, city FROM banks
WHERE bank_name NOT IN (SELECT bank_name FROM banks
WHERE city='Chicago')
ORDER BY no_accounts;
```

```
           bank_name            |               city
--------------------------------+---------------------------------
 Gun Chase Bank                 | Deerfield
 Bankrupt Bank                  | Evanston
 Gun Chase Bank                 | Evanston
(3 rows)
```

4

```
SELECT bank_name, city FROM robberies ORDER BY date LIMIT 1;
```

```
           bank_name            |               city
--------------------------------+---------------------------------
 Loanshark Bank                 | Evanston
(1 row)
```

5

```
SELECT robber_id, nickname, earnings FROM (
  SELECT robber_id, SUM(share) AS earnings
  FROM accomplices GROUP BY robber_id
) AS total_earnings NATURAL JOIN robbers
WHERE earnings>30000 ORDER BY earnings DESC;
```

```
 robber_id |             nickname             | earnings
```

```
----------+-----------------------------------+----------
         5 | Mimmy The Mau Mau                 |    70000
        15 | Boo Boo Hoff                      | 61447.61
        16 | King Solomon                      |  59725.8
        17 | Bugsy Siegel                      |  52601.1
         3 | Lucky Luchiano                    |    42667
        10 | Bonnie                            |    40085
         1 | Al Capone                         |    39486
         4 | Anastazia                         | 39169.62
         8 | Clyde                             |    31800
(9 rows)
```

6

```
SELECT description, robber_id, nickname
FROM robbers NATURAL JOIN has_skills NATURAL JOIN skills
ORDER BY 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)
```

7

```
SELECT robber_id, nickname, no_years FROM robbers
WHERE no_years > 3;
```

```
 robber_id |             nickname              | no_years
-----------+-----------------------------------+----------
         2 | Bugsy Malone                      |       15
         3 | Lucky Luchiano                    |       15
         4 | Anastazia                         |       15
         6 | Tony Genovese                     |       16
         7 | Dutch Schulz                      |       31
        11 | Meyer Lansky                      |        6
        15 | Boo Boo Hoff                      |       13
        16 | King Solomon                      |       43
        17 | Bugsy Siegel                      |       13
        20 | Longy Zwillman                    |        6
(10 rows)
```

8

```
SELECT robber_id, nickname, (age - no_years)
AS no_years_not_in_prison
FROM robbers WHERE no_years>(age/2);
```

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

Question 5: Complex Database Queries

1

```
CREATE VIEW robber_total AS
SELECT robber_id, SUM(share) AS
earnings
FROM accomplices GROUP BY
robber_id;

CREATE VIEW earning AS SELECT
robber_id, earnings
FROM robber_total
WHERE earnings >
(SELECT SUM(share) FROM
accomplices)/
(SELECT COUNT(robber_id) FROM
robbers);

SELECT nickname FROM robbers
NATURAL JOIN earning
WHERE no_years = 0 ORDER BY
earnings DESC;
```

```
SELECT nickname FROM robbers
NATURAL JOIN(
  SELECT robber_id, earnings
  FROM(
    SELECT robber_id, SUM(share)
    AS earnings
    FROM accomplices GROUP BY
    robber_id
  ) AS robber_total
  WHERE earnings >
  (SELECT SUM(share) FROM
  accomplices)/
  (SELECT COUNT(robber_id) FROM
  robbers)
) AS earning
WHERE no_years = 0 ORDER BY
earnings DESC;
```

```
             nickname
-------------------------------
-
 Mimmy The Mau Mau
 Bonnie
 Clyde
(3 rows)
```

```
             nickname
-------------------------------
-
 Mimmy The Mau Mau
 Bonnie
 Clyde
(3 rows)
```

2

```
CREATE VIEW
average_robberies_per_security
AS SELECT bank_name, city,
amount, security
FROM robberies NATURAL JOIN
banks;

SELECT security, COUNT(security)
AS number, ROUND(AVG(amount),2)
AS amount FROM
```

```
SELECT security, COUNT(security)
AS number, ROUND(AVG(amount),2)
AS amount FROM(
  SELECT bank_name, city,
amount,
  security FROM robberies
NATURAL
  JOIN banks
) AS
average_robberies_per_security
```

average_robberies_per_security GROUP BY security;	GROUP BY security;
security | number | amount ---+---+--- weak | 4 | 2299.50 good | 2 | 3980.00 excellent | 12 |39238.08 very good | 3 |12292.43 (4 rows)	security | number | amount ---+---+--- weak | 4 | 2299.50 good | 2 | 3980.00 excellent | 12 |39238.08 very good | 3 |12292.43 (4 rows)

3

CREATE VIEW bank_security AS SELECT security, robber_id FROM banks NATURAL JOIN accomplices GROUP BY bank_name, city, robber_id; CREATE VIEW securities AS SELECT security, skill_id FROM bank_security NATURAL JOIN has_skills GROUP BY security, skill_id; CREATE VIEW security_descriptions AS SELECT security, skill_id, description FROM securities NATURAL JOIN skills; CREATE VIEW security_descriptions_robbers AS SELECT security, description, robber_id FROM security_descriptions NATURAL JOIN has_skills; SELECT security, description, nickname	SELECT security, description, nickname FROM (SELECT security, description, robber_id FROM (SELECT security, skill_id, description FROM (SELECT security, skill_id FROM (SELECT security, robber_id FROM banks NATURAL JOIN accomplices GROUP BY bank_name, city, robber_id) AS bank_security NATURAL JOIN has_skills GROUP BY security, skill_id) AS securities NATURAL JOIN skills) AS security_descriptions NATURAL JOIN has_skills) AS security_descriptions_robbers NATURAL JOIN robbers WHERE security_descriptions_robbers.ro

```
FROM
security_descriptions_robbers
NATURAL JOIN robbers
WHERE
security_descriptions_robbers.ro
bber_id = robbers.robber_id
GROUP BY security, description,
nickname
ORDER BY security, description;
```

```
bber_id = robbers.robber_id
GROUP BY security, description,
nickname
ORDER BY security, description;
```

```
     security     |
description            |
nickname
------------------+-------------
---------------------+----------
------------------------
 excellent        | Driving
| Bugsy Siegel
 excellent        | Driving
| Dutch Schulz
 excellent        | Driving
| Lepke Buchalter
 excellent        | Driving
| Longy Zwillman
 excellent        | Driving
| Lucky Luchiano
 excellent        | Driving
| Mimmy The Mau Mau
 excellent        | Explosives
| Bugsy Malone
 excellent        | Explosives
| Sonny Genovese
 excellent        | Guarding
| Anastazia
 excellent        | Guarding
| Bugsy Siegel
 excellent        | Guarding
| Lepke Buchalter
 excellent        | Gun-Shooting
| Calamity Jane
 excellent        | Gun-Shooting
| Waxey Gordon
 excellent        | Lock-Picking
| Clyde
 excellent        | Lock-Picking
| Dutch Schulz
 excellent        | Lock-Picking
```

```
     security     |
description            |
nickname
------------------+-------------
---------------------+----------
------------------------
 excellent        | Driving
| Bugsy Siegel
 excellent        | Driving
| Dutch Schulz
 excellent        | Driving
| Lepke Buchalter
 excellent        | Driving
| Longy Zwillman
 excellent        | Driving
| Lucky Luchiano
 excellent        | Driving
| Mimmy The Mau Mau
 excellent        | Explosives
| Bugsy Malone
 excellent        | Explosives
| Sonny Genovese
 excellent        | Guarding
| Anastazia
 excellent        | Guarding
| Bugsy Siegel
 excellent        | Guarding
| Lepke Buchalter
 excellent        | Gun-Shooting
| Calamity Jane
 excellent        | Gun-Shooting
| Waxey Gordon
 excellent        | Lock-Picking
| Clyde
 excellent        | Lock-Picking
| Dutch Schulz
 excellent        | Lock-Picking
```

```
| Greasy Guzik
 excellent        | Lock-Picking
| Lucky Luchiano
 excellent        | Lock-Picking
| Sonny Genovese
 excellent        | Planning
| Al Capone
 excellent        | Planning
| Boo Boo Hoff
 excellent        | Planning
| Clyde
 excellent        | Planning
| King Solomon
 excellent        | Planning
| Mimmy The Mau Mau
 excellent        | Preaching
| Al Capone
 excellent        | Preaching
| Bonnie
 excellent        | Preaching
| Greasy Guzik
 excellent        |
Safe-Cracking
| Al Capone
 excellent        |
Safe-Cracking
| Meyer Lansky
 excellent        |
Safe-Cracking
| Moe Dalitz
 excellent        |
Safe-Cracking
| Sonny Genovese
 excellent        | Scouting
| Clyde
 excellent        | Scouting
| Vito Genovese
 good             | Cooking
| Vito Genovese
 good             | Eating
| Tony Genovese
 good             | Eating
| Vito Genovese
 good             | Money
Counting                  | Kid
Cann
 good             | Money
```


```
| Greasy Guzik
 excellent        | Lock-Picking
| Lucky Luchiano
 excellent        | Lock-Picking
| Sonny Genovese
 excellent        | Planning
| Al Capone
 excellent        | Planning
| Boo Boo Hoff
 excellent        | Planning
| Clyde
 excellent        | Planning
| King Solomon
 excellent        | Planning
| Mimmy The Mau Mau
 excellent        | Preaching
| Al Capone
 excellent        | Preaching
| Bonnie
 excellent        | Preaching
| Greasy Guzik
 excellent        |
Safe-Cracking
| Al Capone
 excellent        |
Safe-Cracking
| Meyer Lansky
 excellent        |
Safe-Cracking
| Moe Dalitz
 excellent        |
Safe-Cracking
| Sonny Genovese
 excellent        | Scouting
| Clyde
 excellent        | Scouting
| Vito Genovese
 good             | Cooking
| Vito Genovese
 good             | Eating
| Tony Genovese
 good             | Eating
| Vito Genovese
 good             | Money
Counting                  | Kid
Cann
 good             | Money
```

Counting	Counting
Mickey Cohen	Mickey Cohen
good | Money	good | Money
Counting	Counting
Mike Genovese	Mike Genovese
good | Scouting	good | Scouting
Clyde	Clyde
good | Scouting	good | Scouting
Vito Genovese	Vito Genovese
very good | Driving	very good | Driving
Bugsy Siegel	Bugsy Siegel
very good | Driving	very good | Driving
Dutch Schulz	Dutch Schulz
very good | Driving	very good | Driving
Lepke Buchalter	Lepke Buchalter
very good | Driving	very good | Driving
Longy Zwillman	Longy Zwillman
very good | Driving	very good | Driving
Lucky Luchiano	Lucky Luchiano
very good | Driving	very good | Driving
Mimmy The Mau Mau	Mimmy The Mau Mau
very good | Explosives	very good | Explosives
Bugsy Malone	Bugsy Malone
very good | Explosives	very good | Explosives
Sonny Genovese	Sonny Genovese
very good | Guarding	very good | Guarding
Anastazia	Anastazia
very good | Guarding	very good | Guarding
Bugsy Siegel	Bugsy Siegel
very good | Guarding	very good | Guarding
Lepke Buchalter	Lepke Buchalter
very good | Lock-Picking	very good | Lock-Picking
Clyde	Clyde
very good | Lock-Picking	very good | Lock-Picking
Dutch Schulz	Dutch Schulz
very good | Lock-Picking	very good | Lock-Picking
Greasy Guzik	Greasy Guzik
very good | Lock-Picking	very good | Lock-Picking
Lucky Luchiano	Lucky Luchiano
very good | Lock-Picking	very good | Lock-Picking
Sonny Genovese	Sonny Genovese
very good | Planning	very good | Planning
Al Capone	Al Capone
very good | Planning	very good | Planning
Boo Boo Hoff	Boo Boo Hoff
very good | Planning	very good | Planning
Clyde	Clyde

very good	Planning	King Solomon
very good	Planning	Mimmy The Mau Mau
very good	Preaching	Al Capone
very good	Preaching	Bonnie
very good	Preaching	Greasy Guzik
very good	Safe-Cracking	Al Capone
very good	Safe-Cracking	Meyer Lansky
very good	Safe-Cracking	Moe Dalitz
very good	Safe-Cracking	Sonny Genovese
weak	Cooking	Vito Genovese
weak	Driving	Bugsy Siegel
weak	Driving	Dutch Schulz
weak	Driving	Lepke Buchalter
weak	Driving	Longy Zwillman
weak	Driving	Lucky Luchiano
weak	Driving	Mimmy The Mau Mau
weak	Eating	Tony Genovese
weak	Eating	Vito Genovese
weak	Explosives	Bugsy Malone
weak	Explosives	Sonny Genovese
weak	Guarding	Anastazia
weak	Guarding	

very good	Planning	King Solomon
very good	Planning	Mimmy The Mau Mau
very good	Preaching	Al Capone
very good	Preaching	Bonnie
very good	Preaching	Greasy Guzik
very good	Safe-Cracking	Al Capone
very good	Safe-Cracking	Meyer Lansky
very good	Safe-Cracking	Moe Dalitz
very good	Safe-Cracking	Sonny Genovese
weak	Cooking	Vito Genovese
weak	Driving	Bugsy Siegel
weak	Driving	Dutch Schulz
weak	Driving	Lepke Buchalter
weak	Driving	Longy Zwillman
weak	Driving	Lucky Luchiano
weak	Driving	Mimmy The Mau Mau
weak	Eating	Tony Genovese
weak	Eating	Vito Genovese
weak	Explosives	Bugsy Malone
weak	Explosives	Sonny Genovese
weak	Guarding	Anastazia
weak	Guarding	

```
| Bugsy Siegel
 weak             | Guarding
| Lepke Buchalter
 weak             | Lock-Picking
| Clyde
 weak             | Lock-Picking
| Dutch Schulz
 weak             | Lock-Picking
| Greasy Guzik
 weak             | Lock-Picking
| Lucky Luchiano
 weak             | Lock-Picking
| Sonny Genovese
 weak             | Planning
| Al Capone
 weak             | Planning
| Boo Boo Hoff
 weak             | Planning
| Clyde
 weak             | Planning
| King Solomon
 weak             | Planning
| Mimmy The Mau Mau
 weak             | Preaching
| Al Capone
 weak             | Preaching
| Bonnie
 weak             | Preaching
| Greasy Guzik
 weak             |
Safe-Cracking
| Al Capone
 weak             |
Safe-Cracking
| Meyer Lansky
 weak             |
Safe-Cracking
| Moe Dalitz
 weak             |
Safe-Cracking
| Sonny Genovese
 weak             | Scouting
| Clyde
 weak             | Scouting
| Vito Genovese
(101 rows)
```


```
| Bugsy Siegel
 weak             | Guarding
| Lepke Buchalter
 weak             | Lock-Picking
| Clyde
 weak             | Lock-Picking
| Dutch Schulz
 weak             | Lock-Picking
| Greasy Guzik
 weak             | Lock-Picking
| Lucky Luchiano
 weak             | Lock-Picking
| Sonny Genovese
 weak             | Planning
| Al Capone
 weak             | Planning
| Boo Boo Hoff
 weak             | Planning
| Clyde
 weak             | Planning
| King Solomon
 weak             | Planning
| Mimmy The Mau Mau
 weak             | Preaching
| Al Capone
 weak             | Preaching
| Bonnie
 weak             | Preaching
| Greasy Guzik
 weak             |
Safe-Cracking
| Al Capone
 weak             |
Safe-Cracking
| Meyer Lansky
 weak             |
Safe-Cracking
| Moe Dalitz
 weak             |
Safe-Cracking
| Sonny Genovese
 weak             | Scouting
| Clyde
 weak             | Scouting
| Vito Genovese
(101 rows)
```

5

```
CREATE VIEW chicago AS
SELECT city, ROUND(AVG(share),2)
AS average_robbery
FROM accomplices WHERE
city='Chicago' group by city;

CREATE VIEW not_chicago AS
SELECT city, ROUND(AVG(share),2)
AS average_robberY FROM
accomplices
WHERE city!='Chicago' group by
city ORDER BY average_robbery
DESC LIMIT 1;

SELECT city, average_robbery
FROM chicago UNION
SELECT city, average_robbery
FROM not_chicago;
```

```
   city   | average_robbery
----------+-----------------
 Chicago  | 4221.41
 Evanston | 8255.16
(2 rows)
```

```
SELECT city, average_robbery
FROM (
  SELECT city,
  ROUND(AVG(share),2) AS
  average_robbery
  FROM accomplices
  WHERE city='Chicago'
  group by city
) AS chicago UNION (
  SELECT city,
  ROUND(AVG(share),2) AS
  average_robbery
  FROM accomplices
  WHERE city!='Chicago'
  group by city
  ORDER BY average_robbery DESC
  LIMIT 1
);
```

```
   city   | average_robbery
----------+-----------------
 Chicago  | 4221.41
 Evanston | 8255.16
(2 rows)
```