

SWEN304 Project 1

300492683 Royan Andree Saril

Question 1: Defining the Database

Bank:

Primary Key: {BankName, City} I've chosen these two because the banks are specified by the bank names and the city where each branch is located.

Foreign Key: {} no foreign key

Attribute Constraint: BankName and City are set to NOT NULL because they are the primary key in the Bank table. To ensure that the security levels are within Security, I used the constraint CHECK.

Robberies:

Primary Key: {BankName, City, Date} which is combined with three attributes. Date is needed to compose the primary key, if not then we cannot distinguish if a bank has been robbed more than once on the same date.

Foreign Key: {BankName, City} \subseteq Bank{BankName, City}

Attribute Constraint: BankName and City are set to NOT NULL because both are used as primary keys in the Robberies table and are referencing the Bank table. Date is also NOT NULL as it is part of the primary key.

DELETE RESTRICT is used to avoid violation of data inconsistency.

UPDATE CASCADE is used to avoid data inconsistency, for example if a bank name is changed all robberies referenced to the bank should be updated as well.

Robbers:

Primary Key: {RobberID}

Foreign Key: {} no foreign key

Attribute Constraint: RobberID's created in a column which consists of unique integers and values that are NOT NULL, therefore I used serial to generate these unique IDs. I created constraints for Age and PrisonYears to ensure that the robbers were over 0 years old and that their number years spent in prison is less than their age and greater than or equal to 0 years.

Plans:

Primary Key: {BankName, City, Date}

Foreign Key: {BankName, City} \subseteq Banks{BankName, City}

Attribute Constraint: BankName, City and Date are set to NOT NULL because they are the primary key. To ensure that there is at least one robber that robs a bank, I used constraint CHECK.

DELETE RESTRICT is used to avoid violation of data inconsistency.

UPDATE CASCADE is used to avoid data inconsistency when there have been changes made with references towards that table.

Skills:

Primary Key: {SkillId}

Foreign Key: {} no foreign key

Attribute Constraint: SkillId is unique as well as set to NOT NULL, therefore I used serial to generate these unique IDs.

HasSkills:

Primary Key: {SkillId, RobberId}

Foreign Key: {RobberId} \subseteq Robbers{RobberId}

{SkillId} \subseteq Skills{SkillId}

Robbers table connects to the Skills table, as the RobberId in HasSkills is a subset of RobberId in Robbers table.

Attribute Constraint: RobberId and SkillId are set to not null. Constraint CHECK is used to ensure that the Grade is in the correct order, and CHECK to make sure that the Preference is between 1-3.

DELETE RESTRICT is used to avoid violation of data inconsistency, for example if RobberId or SkillId is deleted in either the Robbers or Skills table then it should also be deleted in the HasSkills table.

UPDATE RESTRICT is used to ensure that there are no updates to the RobberIds in the Robbers table.

HasAccounts:

Primary Key: {RobberId, BankName, City} – We are able to get each member's accounts if we know their RobberId, BankName and City.

Foreign Key: {RobberId} \subseteq Robbers{RobberId}

{BankName, City} \subseteq Bank{BankName, City}

These foreign keys will connect to HasAccounts table with Bank and Robbers.

Attribute Constraint: RobberId, BankName and City are set to NOT NULL as they are the primary key and store important information.

DELETE RESTRICT is used to avoid violation of data inconsistency, where if RobberId is deleted in the Robbers table however HasAccounts table references RobberId should not be allowed.

UPDATE RESTRICT is used to ensure that there are no updates to the RobberIds in the Robbers table.

Accomplices:

Primary Key: {RobberId, Date, BankName, City} – If we know these attributes of a gang member then we are able to know their accomplices.

Foreign Key: {RobberId} \subseteq Robbers{RobberId}

{BankName, City, Date} \subseteq Bank{BankName, City, Date}

These foreign keys are created as an Accomplice should be a Robber who exists in the Robbers table and all data from an Accomplice should refer to the robberies that exist in the Robberies table.

Attribute Constraint: RobberId, Date, BankName and City are set to NOT NULL as they make up the primary key. Constraint CHECK to ensure that all shares is greater than 0.

DELETE RESTRICT is used to avoid violation of data inconsistency, where if RobberId is deleted in the Robbers table however Accompliacs table references RobberId should not be allowed.

UPDATE RESTRICT is used to ensure that there are no updates to the RobberId in the Robbers table. Date, BankName and City are also applied and updated in the Banks table first.

psql -d project1

```
CREATE TABLE Bank(BankName TEXT NOT NULL, City TEXT NOT NULL, NoAccounts INTEGER, Security TEXT CONSTRAINT SecurityLevel CHECK(Security IN('excellent','very good','good','weak')), CONSTRAINT BANK_PK PRIMARY KEY(BankName, City));
```

```
CREATE TABLE Robberies(BankName TEXT NOT NULL, City TEXT NOT NULL, Date DATE NOT NULL, Amount DECIMAL(12,2), CONSTRAINT Robberies_PK PRIMARY KEY(BankName, City, Date), CONSTRAINT Robberies_FK FOREIGN KEY(BankName, City) references Bank(BankName, City) ON DELETE RESTRICT ON UPDATE CASCADE);
```

```
CREATE TABLE Plans(BankName TEXT NOT NULL, City TEXT NOT NULL, NoRobbers INTEGER CONSTRAINT NoRobber CHECK(NoRobbers > 0), PlannedDate DATE NOT NULL, CONSTRAINT Plans_PK PRIMARY KEY(BankName, City, PlannedDate), CONSTRAINT Plans_FK FOREIGN KEY(BankName, City) references Bank(BankName, City) ON DELETE RESTRICT ON UPDATE CASCADE);
```

```
CREATE TABLE Robbers(RobberId SERIAL, NickName TEXT, Age INTEGER CONSTRAINT AgeCheck CHECK(Age >= 0), NoYears INTEGER CONSTRAINT PrisonYears CHECK(NoYears < Age AND NoYears >= 0), PRIMARY KEY(RobberId));
```

```
CREATE TABLE Skills(SkillId SERIAL, Description TEXT, PRIMARY KEY(SkillId));
```

```
CREATE TABLE HasSkills(RobberID INTEGER NOT NULL, SkillId INTEGER NOT NULL, Preference INTEGER CONSTRAINT PreferenceRank CHECK(Preference > 0 AND Preference <= 3), Grade CHAR(2), CONSTRAINT GradeId CHECK(Grade IN('A+', 'A', 'A-', 'B+', 'B', 'B-', 'C+', 'C')), CONSTRAINT HasSkills_PK PRIMARY KEY(RobberId, SkillId), CONSTRAINT HasSkill_FKrobbers FOREIGN KEY(RobberId) references Robbers(RobberId) ON DELETE RESTRICT ON UPDATE RESTRICT, CONSTRAINT HasSkills_FKskills FOREIGN KEY(SkillId) references Skills(SkillId) ON DELETE RESTRICT ON UPDATE RESTRICT);
```

```
CREATE TABLE HasAccounts(RobberId INTEGER NOT NULL, BankName TEXT NOT NULL, City TEXT NOT NULL, CONSTRAINT HasAccounts_PK PRIMARY KEY(RobberId, BankName, City), CONSTRAINT HasAccounts_FKbank FOREIGN KEY(BankName, City) references Bank(BankName, City) ON DELETE RESTRICT ON UPDATE RESTRICT, CONSTRAINT HasAccount_FKrobbers FOREIGN KEY(RobberId) references Robbers(RobberId) ON DELETE RESTRICT ON UPDATE CASCADE);
```

```
CREATE TABLE Accomplices(RobberId INTEGER NOT NULL, BankName TEXT NOT NULL, City TEXT NOT NULL, Date DATE NOT NULL, Share INTEGER CONSTRAINT Shares CHECK(Share > 0), CONSTRAINT Accomplices_PK PRIMARY KEY(RobberId, BankName, City, Date), CONSTRAINT Accomplices_FKrobbers FOREIGN KEY(RobberId) references Robbers(RobberId) ON DELETE RESTRICT ON UPDATE CASCADE, CONSTRAINT Accomplices_FKbank FOREIGN KEY(BankName, City) references Bank(BankName, City) ON DELETE RESTRICT ON UPDATE CASCADE);
```

Question 2: Populating your Database with Data

The order of how I implemented the tables of the database was sort of mirroring as what was being assigned in the handout. Where the Bank, Plans, Robberies and Robbers were straightforward to insert the data as these did not need much to generate. In the handout then, it mentions that HasSkills, HasAccounts and Accomplices will need temporary

relations and joins. Thus, that is the order that I followed from the handout.

Bank Table:

```
project1=> \copy Bank(BankName, City, NoAccounts, Security) FROM /home/sarilroya/swen304_project1/datafiles_21/banks_21.data
COPY 20
project1=> select * from Bank;
```

bankname	city	noaccounts	security
NXP Bank	Chicago	1593311	very good
Bankrupt Bank	Evanston	444000	weak
Loan shark Bank	Evanston	7654321	excellent
Loan shark Bank	Deerfield	3456789	very good
Loan shark 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)

Plans Table:

```
project1=> \copy Plans(BankName, City, PlannedDate, NoRobbers) FROM /home/sarilroya/swen304_project1/datafiles_21/plans_21.data
COPY 11
project1=> select * from Plans;
```

bankname	city	norobbers	planneddate
NXP Bank	Chicago	5	2019-10-30
Loan shark 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)

Robberies Table:

```
project1=> \copy Robberies(BankName, City, Date, Amount) FROM /home/sarilroya/swen304_project1/datafiles_21/robberies_21.data
COPY 21
project1=> select * from Robberies;
```

bankname	city	date	amount
NXP Bank	Chicago	2019-01-08	34302.30
Loan shark Bank	Evanston	2019-02-28	19990.00
Loan shark 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	99800.80
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
Loan shark 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
Loan shark Bank	Chicago	2017-11-09	41000.00
Penny Pinchers	Evanston	2019-05-30	13000.40
PickPocket Bank	Chicago	2015-09-21	2039.00
Loan shark 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)

Robbers Table:

CREATE TABLE robbersTEMP(NickName TEXT NOT NULL, Age INT NOT NULL, NoYears INT NOT NULL)

```
project1=> \copy robbersTEMP(NickName, Age, NoYears) FROM /home/sarilroya/swen304_project1/datafiles_21/robbers_21.data
COPY 24
project1=> INSERT INTO Robbers(SELECT nextval('robbers_robberid_seq'), * FROM robbersTEMP);
INSERT 0 24
project1=> DROP TABLE robbersTEMP;
DROP TABLE
project1=> select * from robbers;
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)
```

HasSkills Table:

CREATE TABLE hasskillsTEMP(Nickname TEXT NOT NULL, Description TEXT NOT NULL, Preference INT NOT NULL, Grade CHAR(2) NOT NULL);

\copy hasskillsTEMP(Nickname, Description, Preference, Grade) FROM /home/sarilroya/swen304_project1/datafiles_21/hasskills_21.data

CREATE VIEW skillsDISTINCT AS SELECT DISTINCT Description FROM hasskillsTEMP;

INSERT INTO Skills(SELECT nextval('skills_skillid_seq'), Description FROM skillsDISTINCT);

DROP VIEW skillsDISTINCT;

INSERT INTO HasSkills(SELECT r.RobberId, s.SkillId, temp.Preference, temp.Grade FROM Robbers r, Skills s, hasskillsTEMP temp WHERE temp.Nickname = r.Nickname AND temp.Description = s.Description);

DROP TABLE hasskillsTEMP;

```
project1=> CREATE VIEW skillsDISTINCT AS SELECT DISTINCT Description FROM hasskillsTEMP;
CREATE VIEW
project1=> INSERT INTO Skills(SELECT nextval('skills_skillid_seq'), Description FROM skillsDISTINCT);
INSERT 0 12
project1=> DROP VIEW skillsDISTINCT;
DROP VIEW
project1=> INSERT INTO HasSkills(SELECT r.RobberId, s.SkillId, temp.Preference, temp.Grade FROM Robbers r, Skills s, hasskillsTEMP temp
WHERE temp.Nickname = r.Nickname AND temp.Description = s.Description);
INSERT 0 38
project1=> DROP TABLE hasskillsTEMP;
DROP TABLE
project1=>
```

```

project1=> select * from HasSkills;
robberid | skillid | preference | grade
-----|-----|-----|-----
18 | 4 | 2 | A
23 | 9 | 1 | A
7 | 9 | 2 | C+
17 | 9 | 1 | A+
5 | 9 | 2 | C
3 | 9 | 2 | B+
20 | 9 | 1 | C
6 | 10 | 1 | B+
18 | 10 | 3 | A+
24 | 1 | 1 | B
2 | 1 | 1 | A
4 | 2 | 1 | A
17 | 2 | 2 | C+
23 | 2 | 2 | C
21 | 5 | 1 | C
9 | 5 | 1 | B
22 | 6 | 2 | C
7 | 6 | 1 | A+
3 | 6 | 1 | B+
8 | 6 | 1 | C+
24 | 6 | 3 | B
19 | 12 | 1 | C
13 | 12 | 1 | B+
14 | 12 | 1 | B
16 | 3 | 1 | A
8 | 3 | 3 | C
5 | 3 | 1 | A+
15 | 3 | 1 | A+
1 | 3 | 1 | A+
22 | 8 | 1 | A+
1 | 8 | 3 | A+
10 | 8 | 1 | B
12 | 7 | 1 | A
11 | 7 | 1 | A+
1 | 7 | 2 | C+
24 | 7 | 2 | C+
18 | 11 | 1 | B+
8 | 11 | 2 | C+
(38 rows)

```

```

project1=> select * from skills;
skillid | 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)

```

HasAccounts Table:

```

CREATE TABLE hasaccountsTEMP(Nickname TEXT NOT NULL, BankName TEXT NOT
NULL, City TEXT NOT NULL);

```

```

\copy hasaccountsTEMP(Nickname, BankName, City) FROM

```

```

/home/sarilroya/swen304_project1/datafiles_21/hasaccounts_21.data

```

```

INSERT INTO HasAccounts(SELECT r.RobberId, temp.BankName, temp.City FROM
Robbers r, hasaccountsTEMP temp WHERE temp.Nickname = r.Nickname);

```

```

DROP TABLE hasaccountsTEMP;

```

```

project1=> select * from HasAccounts;
robberid | bankname | city
-----|-----|-----
1 | Bad Bank | Chicago
1 | Inter-Gang Bank | Evanston
1 | NXP Bank | Chicago
2 | Loan shark Bank | Chicago
2 | Loan shark Bank | Deerfield
3 | NXP Bank | Chicago
3 | Bankrupt Bank | Evanston
4 | Loan shark Bank | Evanston
5 | Inter-Gang Bank | Evanston
5 | Loan shark 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
19 | Gun Chase Bank | Burbank
20 | PickPocket Bank | Evanston
21 | PickPocket Bank | Evanston
22 | PickPocket Bank | Chicago
23 | Hidden Treasure | Chicago
24 | Hidden Treasure | Chicago
(31 rows)

```

Accomplices Table:

```

CREATE TABLE accomplicesTEMP(Nickname TEXT NOT NULL, BankName TEXT NOT
NULL, City TEXT NOT NULL, Date DATE NOT NULL, Share DECIMAL(15,2) NOT NULL);

```

```

\copy accomplicesTEMP(Nickname, BankName, City, Date, Share) FROM

```

```

/home/sarilroya/swen304_project1/datafiles_21/accomplices_21.data

```



```
INSERT INTO Accomplices(SELECT r.RobberId, temp.BankName, temp.City, temp.date,
temp.Share FROM Robbers r, accomplicesTEMP temp WHERE temp.Nickname =
r.Nickname);
```

```
project1=> select * from Accomplices;
```

robberid	bankname	city	date	share
1	Bad Bank	Chicago	2017-02-02	3010
1	NXP Bank	Chicago	2019-01-08	6406
1	Loan shark Bank	Evanston	2019-02-28	4997
1	Loan shark Bank	Chicago	2019-03-30	4201
1	Inter-Gang Bank	Evanston	2016-02-16	12103
1	Inter-Gang Bank	Evanston	2018-02-14	8769
2	NXP Bank	Chicago	2019-01-08	2300
3	Penny Pinchers	Evanston	2016-08-30	16500
3	Loan shark Bank	Evanston	2019-02-28	4997
3	Loan shark Bank	Chicago	2017-11-09	8200
3	Loan shark Bank	Chicago	2019-03-30	4201
3	Inter-Gang Bank	Evanston	2018-02-14	8769
4	Penny Pinchers	Evanston	2016-08-30	16500
4	NXP Bank	Chicago	2019-01-08	6406
4	Loan shark Bank	Chicago	2019-03-30	4201
4	Inter-Gang Bank	Evanston	2018-02-14	8769
4	Gun Chase Bank	Evanston	2016-04-30	3291
5	Inter-Gang Bank	Evanston	2017-03-13	60000
5	Loan shark Bank	Evanston	2016-04-20	16000
7	Penny Pinchers	Chicago	2016-08-30	450
7	Loan shark Bank	Evanston	2017-04-20	2749
7	Inter-Gang Bank	Evanston	2018-02-14	8769
7	Gun Chase Bank	Evanston	2016-04-30	3282
8	Penny Pinchers	Evanston	2016-08-30	16500
8	Penny Pinchers	Chicago	2016-08-30	450
8	Loan shark Bank	Evanston	2017-04-20	2747
8	Inter-Gang Bank	Evanston	2016-02-16	12103
10	Penny Pinchers	Evanston	2016-08-30	16500
10	Loan shark Bank	Chicago	2017-11-09	8200
10	Inter-Gang Bank	Evanston	2016-02-16	12103
10	Gun Chase Bank	Evanston	2016-04-30	3282
11	Penny Pinchers	Evanston	2017-10-30	3000
12	PickPocket Bank	Evanston	2016-03-30	32
13	Dollar Grabbers	Evanston	2017-11-09	2000
14	Dollar Grabbers	Evanston	2017-06-20	1790
15	Inter-Gang Bank	Evanston	2017-03-13	30000
15	PickPocket Bank	Chicago	2018-02-28	119
15	Penny Pinchers	Evanston	2017-10-30	3001
15	Penny Pinchers	Evanston	2019-05-30	3250
15	Loan shark Bank	Chicago	2019-03-30	4201

Question 3: Checking your Database

1. Insert the following tuples into the Banks table:

a. ('Loan shark Bank', 'Evanston', 100, 'very good')

```
project1=> INSERT INTO Bank
project1=> VALUES('Loan shark Bank', 'Evanston', 100, 'very good');
ERROR: duplicate key value violates unique constraint "bank_pk"
DETAIL: Key (bankname, city)=(Loan shark Bank, Evanston) already exists.
project1=>
```

b. ('EasyLoan Bank', 'Evanston', -5, 'excellent')

```
project1=> INSERT INTO Bank
project1=> VALUES('EasyLoan Bank', 'Evanston', -5, 'excellent');
INSERT 0 1
project1=>
```

Unfortunately there is no error message when inserting this tuple into the Bank table. This means that my database was not created correctly. The constraint should be an ERROR for a CHECK constraint.

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

```
project1=> INSERT INTO Bank
project1=> VALUES('EasyLoan Bank', 'Evanston', 100, 'poor');
ERROR: new row for relation "bank" violates check constraint "securitylevel"
DETAIL: Failing row contains (EasyLoan Bank, Evanston, 100, poor).
project1=>
```

2. Insert the following tuple into the Skills table:

• (21, 'Driving')

```
project1=> INSERT INTO Skills
project1=> VALUES(21, 'Driving');
INSERT 0 1
project1=>
```

There is no error message when inserting this tuple into the Skills table. The constraint should be an ERROR for a CHECK constraint.

3. Insert the following tuple into the Robberies table:

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

```
project1=> INSERT INTO Robberies
project1-> VALUES('NXP Bank', 'Chicago', '2019-01-08', 1000);
ERROR: duplicate key value violates unique constraint "robberies_pk"
DETAIL: Key (bankname, city, date)=(NXP Bank, Chicago, 2019-01-08) already exists.
project1=>
```

4. Delete the following tuples from the Banks table:

- ('PickPocket Bank', 'Evanston', 2000, 'very good')

```
project1=>
project1=> DELETE FROM Bank
project1-> WHERE BankName = 'PickPocket Bank' AND City = 'Evanston' AND NoAccounts = 2000 AND Security = 'very good';
ERROR: update or delete on table "bank" violates foreign key constraint "robberies_fk" on table "robberies"
DETAIL: Key (bankname, city)=(PickPocket Bank, Evanston) is still referenced from table "robberies".
project1=>
```

5. Delete the following tuple from the Robberies table:

- ('Loanshark Bank', 'Chicago', "", "")

```
project1=> DELETE FROM Robberies
project1-> WHERE BankName = 'Loanshark Bank' AND City = 'Chicago' AND Date = '' AND Amount = '';
ERROR: syntax error at or near "Bank'"
LINE 2: WHERE BankName = 'Loanshark Bank' AND City = 'Chicago' AND D...
                        ^
project1=>
```

In the following two tasks, we assume that there is a robber with Id 3, but no robber with Id 666.

6. Insert the following tuples into the Robbers table:

- a. (1, 'Shotgun', 70, 0)

```
project1=> INSERT INTO Robbers
project1-> VALUES(1, 'Shotgun', 70, 0);
ERROR: duplicate key value violates unique constraint "robbers_pkey"
DETAIL: Key (robberid)=(1) already exists.
project1=>
```

- b. (666, 'Jail Mouse', 25, 35)

```
project1=> INSERT INTO Robbers
project1-> VALUES(666, 'Jail Mouse', 25, 35);
ERROR: new row for relation "robbers" violates check constraint "prisonyears"
DETAIL: Failing row contains (666, Jail Mouse, 25, 35).
project1=>
```

7. Insert the following tuples into the HasSkills table:

- a. (1, 7, 1, 'A+')

```
project1=> INSERT INTO HasSkills
project1-> VALUES(1, 7, 1, 'A+');
ERROR: duplicate key value violates unique constraint "hasskills_pk"
DETAIL: Key (robberid, skillid)=(1, 7) already exists.
project1=>
```

- b. (1, 2, 0, 'A')


```
project1=> INSERT INTO HasSkills
project1-> VALUES(1, 2, 0, 'A');
ERROR: new row for relation "hasskills" violates check constraint "preferencerank"
DETAIL: Failing row contains (1, 2, 0, A ).
project1=>
```

c. (666, 1, 1, 'B-')

```
project1=> INSERT INTO HasSkills
project1-> VALUES(666, 1, 1, 'B-');
ERROR: insert or update on table "hasskills" violates foreign key constraint "hasskill_fkrobbers"
DETAIL: Key (robberid)=(666) is not present in table "robbers".
project1=>
```

d. (3, 20, 3, 'B+')

```
project1=> INSERT INTO HasSkills
project1-> VALUES(3, 20, 3, 'B+');
ERROR: insert or update on table "hasskills" violates foreign key constraint "hasskills_fkskills"
DETAIL: Key (skillid)=(20) is not present in table "skills".
project1=>
```

8. Delete the following tuple from the Skills table:

- (7, 'Planning')

```
project1=> DELETE FROM Skills
project1-> WHERE SkillId=7 AND Description='Planning';
DELETE 0
project1=>
```

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:

- (1, 'Al Capone', 31, 2).

```
project1=> DELETE FROM Robbers
project1-> WHERE RobberId=1 AND Nickname='Al Capone' AND Age=31 AND NoYears=2;
ERROR: update or delete on table "robbers" violates foreign key constraint "hasskill_fkrobbers" on table "hasskills"
DETAIL: Key (robberid)=(1) is still referenced from table "hasskills".
project1=>
```

Question 4: Simple Database Queries

1. Retrieve RobberId, Nickname, Age, and all skill descriptions of all robbers who are between 20 and 40 years old.

```
SELECT DISTINCT r.RobberId, r.Nickname, r.Age, s.description
FROM Robbers r NATURAL JOIN Skills s WHERE r.age >=20 and r.age <= 40;
```

```

project1=> SELECT DISTINCT r.RobberId, r.Nickname, r.Age, s.description
project1-> FROM Robbers r NATURAL JOIN Skills s WHERE r.age >=20 and r.age <= 40;

```

robberid	nickname	age	description
24	Sonny Genovese	39	Scouting
6	Tony Genovese	28	Money Counting
6	Tony Genovese	28	Explosives
22	Greasy Guzik	25	Guarding
1	Al Capone	31	Driving
8	Clyde	20	Safe-Cracking
11	Meyer Lansky	34	Money Counting
23	Lepke Buchalter	25	Cooking
24	Sonny Genovese	39	Planning
11	Meyer Lansky	34	Gun-Shooting
1	Al Capone	31	Eating
22	Greasy Guzik	25	Money Counting
22	Greasy Guzik	25	Driving
20	Longy Zwillman	35	Guarding
6	Tony Genovese	28	Preaching
23	Lepke Buchalter	25	Safe-Cracking
20	Longy Zwillman	35	Driving
13	Mickey Cohen	24	Preaching
6	Tony Genovese	28	Cooking
23	Lepke Buchalter	25	Money Counting
20	Longy Zwillman	35	Explosives
1	Al Capone	31	Money Counting
24	Sonny Genovese	39	Preaching
19	Mike Genovese	35	Money Counting
23	Lepke Buchalter	25	Explosives
8	Clyde	20	Planning
20	Longy Zwillman	35	Cooking
11	Meyer Lansky	34	Driving
11	Meyer Lansky	34	Explosives
23	Lepke Buchalter	25	Driving
24	Sonny Genovese	39	Eating
11	Meyer Lansky	34	Eating
20	Longy Zwillman	35	Lock-Picking
24	Sonny Genovese	39	Gun-Shooting
24	Sonny Genovese	39	Driving

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

```

SELECT BankName, City
FROM Bank
WHERE NoAccounts = 0;

```

```

project1=> SELECT BankName, City
project1-> FROM Bank
project1-> WHERE NoAccounts = 0;

```

bankname	city
(0 rows)	

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 BankName, City
FROM Bank NATURAL JOIN HasAccounts NATURAL JOIN Robbers WHERE Nickname='Al Capone';

```

```

project1=> SELECT DISTINCT BankName, City FROM Bank NATURAL JOIN HasAccounts NATURAL JOIN Robbers WHERE Nickname = 'Al Capone';

```

bankname	city
Bad Bank	Chicago
Inter-Gang Bank	Evanston
NXP Bank	Chicago

(3 rows)

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

```

SELECT BankName, City, NoAccounts
FROM Bank
WHERE BankName NOT IN (SELECT BankName FROM Bank WHERE City = 'Deerfield')
ORDER BY NoAccounts ASC;

```

```

project1=> SELECT BankName, City, NoAccounts
project1-> FROM Bank
project1-> WHERE BankName NOT IN (SELECT BankName FROM Bank WHERE City = 'Deerfield') ORDER BY NoAccounts ASC;

```

bankname	city	noaccounts
EasyLoan Bank	Evanston	-5
Gun Chase Bank	Burbank	1999
Outside Bank	Chicago	5000
Bad Bank	Chicago	6000
Dollar Grabbers	Chicago	56005
Inter-Gang Bank	Chicago	100000
Penny Pinchers	Evanston	130013
Penny Pinchers	Chicago	156165
Bankrupt Bank	Evanston	444000
Inter-Gang Bank	Evanston	555555
Gun Chase Bank	Evanston	656565
NXP Bank	Evanston	656565
Dollar Grabbers	Evanston	909090
Hidden Treasure	Chicago	999999
NXP Bank	Chicago	1593311

(15 rows)

```

project1=>

```

5. Retrieve RobberId, Nickname and individual total “earnings” of those robbers who have earned more than \$30,000 by robbing banks. The answer should be sorted in decreasing order of the total earnings.

```

SELECT RobberId, Nickname, earnings
FROM (SELECT RobberId, SUM(Share) AS earnings FROM Accomplices GROUP BY
RobberId) AS totalEarnings NATURAL JOIN Robbers
WHERE Earnings > 30000 ORDER BY Earnings DESC;

```

```

project1=> SELECT RobberId, Nickname, earnings
project1-> FROM (SELECT RobberId, SUM(Share) AS earnings FROM Accomplices GROUP BY RobberId) AS totalEarnings NATURAL JOIN Robbers
project1-> WHERE Earnings > 30000 ORDER BY Earnings DESC;

```

robberid	nickname	earnings
5	Mimmy The Mau Mau	70000
15	Boo Boo Hoff	61448
16	King Solomon	59726
17	Bugsy Siegel	52601
3	Lucky Luchiano	42667
10	Bonnie	40085
1	Al Capone	39486
4	Anastazia	39169
8	Clyde	31800

(9 rows)

```

project1=>

```

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

```

SELECT RobberId, Nickname, NoYears
FROM Robbers
WHERE NoYears > 10
ORDER BY RobberId;

```

```

project1=> SELECT RobberId, Nickname, NoYears
project1-> FROM Robbers
project1-> WHERE NoYears > 10 ORDER BY RobberId;

```

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)

```

project1=>

```

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 RobberId, Nickname, NoYears
FROM Robbers
WHERE NoYears > (Age/2);
```

```
[project1=> SELECT RobberId, Nickname, NoYears
[project1=> FROM Robbers
[project1=> WHERE NoYears> (Age/2);
  robberid |      nickname      | noyears
-----+-----+-----
          6 | Tony Genovese      |      16
          16 | King Solomon       |      43
(2 rows)
```

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 RobberId, Nickname, Description
FROM Robbers NATURAL JOIN HasSkills NATURAL JOIN Skills
ORDER BY Description;
```

```
project1=> SELECT RobberId, Nickname, Description
project1-> FROM Robbers NATURAL JOIN HasSkills NATURAL JOIN Skills ORDER BY Description;
  robberid |      nickname      | description
-----+-----+-----
          18 | Vito Genovese      | Cooking
          23 | Lepke Buchalter    | Driving
           7 | Dutch Schulz       | Driving
          17 | Bugsy Siegel       | Driving
           5 | Mimmy The Mau Mau  | Driving
           3 | Lucky Luchiano     | Driving
          20 | Longy Zwillman     | 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
          21 | Waxey Gordon       | Gun-Shooting
           9 | Calamity Jane      | Gun-Shooting
          22 | Greasy Guzik       | Lock-Picking
           7 | Dutch Schulz       | Lock-Picking
           3 | Lucky Luchiano     | Lock-Picking
           8 | Clyde              | Lock-Picking
          24 | Sonny Genovese     | Lock-Picking
          19 | Mike Genovese      | Money Counting
          13 | Mickey Cohen       | Money Counting
          14 | Kid Cann           | Money Counting
...skipping 1 line
           8 | Clyde              | Planning
           5 | Mimmy The Mau Mau  | Planning
          15 | Boo Boo Hoff       | Planning
           1 | Al Capone           | Planning
          22 | Greasy Guzik       | Preaching
           1 | Al Capone           | Preaching
          10 | Bonnie             | Preaching
          12 | Moe Dalitz         | Safe-Cracking
          11 | Meyer Lansky       | Safe-Cracking
           1 | Al Capone           | Safe-Cracking
          24 | Sonny Genovese     | Safe-Cracking
          18 | Vito Genovese      | Scouting
           8 | Clyde              | Scouting
(38 rows)
project1=>
```

Question 5: Complex Database Queries Part A)

For each of the following tasks, you are asked to construct SQL queries.

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 r.BankName, r.City
FROM Robberies r NATURAL JOIN Plans p
GROUP BY r.BankName, r.City, r.Date, p.PlannedDate
HAVING (DATE_PART('year', p.PlannedDate) - DATE_PART('year', r.Date) = 0);
```

```

project1=> SELECT r.BankName, r.City
project1-> FROM Robberies r NATURAL JOIN Plans p
project1-> GROUP BY r.BankName, r.City, r.Date, p.PlannedDate
project1-> HAVING (DATE_PART('year', p.PlannedDate) - DATE_PART('year', r.Date) = 0);
  bankname | city
-----+-----
  NXP Bank | Chicago
  NXP Bank | Chicago
(2 rows)

project1=>

```

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

```

SELECT r.RobberId, r.Nickname
FROM Bank b NATURAL JOIN Robbers r NATURAL JOIN HasAccounts h
WHERE b.NoAccounts = 0
GROUP BY r.RobberId, r.Nickname;

```

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

```

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;

```

```

project1=> SELECT r.RobberId, r.Nickname, s.Description
project1-> FROM Robbers r NATURAL JOIN HasSkills h NATURAL JOIN Skills s
project1-> GROUP BY r.RobberId, r.Nickname, h.Preference, s.Description
project1-> 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
--More--

```

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

```

SELECT a.BankName, a.City, a.Date
FROM Accomplices a
JOIN ( SELECT City, MAX(Share) AS HighestShare
      FROM Accomplices
      GROUP BY City)
HighestShare ON (HighestShare.City = a.City);

```



```

project1=> SELECT a.BankName, a.City, a.Date
project1-> FROM Accomplices a
project1-> JOIN ( SELECT City , MAX(Share) AS HighestShare
project1(> FROM Accomplices
project1(> GROUP BY City)
project1-> HighestShare ON (HighestShare.City = a.City);

```

bankname	city	date
Bad Bank	Chicago	2017-02-02
NXP Bank	Chicago	2019-01-08
Loanshark Bank	Evanston	2019-02-28
Loanshark Bank	Chicago	2019-03-30
Inter-Gang Bank	Evanston	2016-02-16
Inter-Gang Bank	Evanston	2018-02-14
NXP Bank	Chicago	2019-01-08
Penny Pinchers	Evanston	2016-08-30
Loanshark Bank	Evanston	2019-02-28
Loanshark Bank	Chicago	2017-11-09
Loanshark Bank	Chicago	2019-03-30
Inter-Gang Bank	Evanston	2018-02-14
Penny Pinchers	Evanston	2016-08-30
NXP Bank	Chicago	2019-01-08
Loanshark Bank	Chicago	2019-03-30
Inter-Gang Bank	Evanston	2018-02-14
Gun Chase Bank	Evanston	2016-04-30
Inter-Gang Bank	Evanston	2017-03-13
Loanshark Bank	Evanston	2016-04-20
Penny Pinchers	Chicago	2016-08-30
Loanshark Bank	Evanston	2017-04-20
Inter-Gang Bank	Evanston	2018-02-14
Gun Chase Bank	Evanston	2016-04-30

Below is another query which ensures that the cities shown above, really contain the highest shares.

```

SELECT City, MAX(Share) AS HighestShare
FROM Accomplices
GROUP BY City;

```

```

project1=> SELECT City, MAX(Share) AS HighestShare
project1-> FROM Accomplices
project1-> GROUP BY City;

```

city	highestshare
Evanston	60000
Chicago	8200

(2 rows)

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

```

SELECT r.BankName, r.City
FROM Bank b NATURAL JOIN Robberies r
WHERE b.NoAccounts > 1;

```



```

project1=> SELECT r.BankName, r.City
project1-> FROM Bank b NATURAL JOIN Robberies r
project1-> WHERE b.NoAccounts > 1;
  bankname      | city
-----+-----
 NXP Bank       | Chicago
 Loanshark Bank | Evanston
 Loanshark Bank | Chicago
 Inter-Gang Bank | Evanston
 Penny Pinchers | Chicago
 Penny Pinchers | Evanston
 Gun Chase Bank | Evanston
 PickPocket Bank | Evanston
 PickPocket Bank | Chicago
 Loanshark Bank | Evanston
 Inter-Gang Bank | Evanston
 Penny Pinchers | Evanston
 PickPocket Bank | Evanston
 Loanshark Bank | Chicago
 Penny Pinchers | Evanston
 PickPocket Bank | Chicago
 Loanshark Bank | Evanston
 Inter-Gang Bank | Evanston
 Dollar Grabbers | Evanston
 Dollar Grabbers | Evanston
 Bad Bank       | Chicago
(21 rows)

project1=>

```

6. The police department wants to know which robbers are most likely to attack a particular bank branch. Robbing bank branches with a certain security level might require certain skills. For example, maybe every robbery of a branch with “excellent” security requires a robber with “Explosives” skill.

Construct a view containing the Security level, the Skills (if any) that appear in every single bank robbery with that security level, and the Nicknames of all the robbers in the database who possess each of those skills.

STEP WISE APPROACH:

```

CREATE VIEW RobberId as (
SELECT DISTINCT a.RobberId, b.Security
FROM Accomplices a NATURAL JOIN Bank b
ORDER BY b.Security);

```

```

CREATE VIEW SkillsId as (
SELECT h.RobberId, r.Security
FROM HasSkills h NATURAL JOIN RobberId r);

```

```

CREATE VIEW Descriptions as (
SELECT s.Security, skill.Description
FROM SkillsId s NATURAL JOIN Skills skill);

```

```

CREATE VIEW Nickname as (
SELECT des.Security, des.Description, r.Nickname
FROM Robbers r NATURAL JOIN Descriptions des
GROUP BY des.Security, des.Description, r.Nickname
ORDER BY des.Security ASC);

```

```

select * from Nickname;

```

```

project1> CREATE VIEW Nickname as (
project1> SELECT des.Security, des.Description, r.Nickname
project1> FROM Robbers r NATURAL JOIN Descriptions des
project1> GROUP BY des.Security, des.Description, r.Nickname
project1> ORDER BY des.Security ASC);
CREATE VIEW
project1> SELECT * FROM Nickname;
security | description | nickname
-----|-----|-----
excellent | Cooking | Al Capone
excellent | Cooking | Anastasia
excellent | Cooking | Bonnie
excellent | Cooking | Boo Boo Hoff
excellent | Cooking | Buggy Malone
excellent | Cooking | Buggy Siegel
excellent | Cooking | Calamity Jane
excellent | Cooking | Clyde
excellent | Cooking | Dutch Schulz
excellent | Cooking | Greasy Guzik
excellent | Cooking | Kid Cann
excellent | Cooking | King Solomon
excellent | Cooking | Lepke Buchalter
excellent | Cooking | Longy Zwillman
excellent | Cooking | Lucky Luchiano
excellent | Cooking | Meyer Lansky
excellent | Cooking | Mickey Cohen
excellent | Cooking | Mike Genovese
excellent | Cooking | Mimmy The Mau Mau
excellent | Cooking | Moe Dalitz
excellent | Cooking | Sonny Genovese
excellent | Cooking | Tony Genovese
excellent | Cooking | Vito Genovese
...skipping 1 line
weak | Cooking | King Solomon
weak | Cooking | Lepke Buchalter
weak | Cooking | Longy Zwillman
weak | Cooking | Lucky Luchiano
weak | Cooking | Meyer Lansky
weak | Cooking | Mickey Cohen
weak | Cooking | Mike Genovese
weak | Cooking | Mimmy The Mau Mau
weak | Cooking | Moe Dalitz
weak | Cooking | Sonny Genovese
weak | Cooking | Tony Genovese
weak | Cooking | Vito Genovese

```

SINGLE NESTED QUERY:

```

SELECT secL.Security, secL.Description, r.Nickname
FROM Robbers r NATURAL JOIN (
    SELECT sec.Security, sec.RobberId, s.Description
    FROM (SELECT h.Robberid, h.SkillId, lev.Security
          FROM HasSkills h JOIN (
              SELECT DISTINCT a.RobberId, b.security
              FROM Bank b NATURAL JOIN Accomplices a) lev ON
              lev.RobberId = h.RobberId) sec
    NATURAL JOIN Skills s) secL
ORDER BY secL.Security ASC;

```

```

project1> SELECT secL.Security, secL.Description, r.Nickname
project1> FROM Robbers r NATURAL JOIN (SELECT sec.Security, sec.RobberId, s.Description
project1> FROM (SELECT h.Robberid, h.SkillId, lev.Security
project1> FROM HasSkills h JOIN (SELECT DISTINCT a.RobberId, b.Security
project1> FROM Bank b NATURAL JOIN Accomplices a) lev ON lev.RobberId = h.RobberId) sec NATURAL JOIN Skills s) secL
project1> ORDER BY secL.Security ASC;
security | description | nickname
-----|-----|-----
excellent | Planning | Al Capone
excellent | Safe-Cracking | Al Capone
excellent | Preaching | Al Capone
excellent | Lock-Picking | Lucky Luchiano
excellent | Driving | Lucky Luchiano
excellent | Guarding | Anastasia
excellent | Planning | Mimmy The Mau Mau
excellent | Driving | Mimmy The Mau Mau
excellent | Lock-Picking | Dutch Schulz
excellent | Driving | Dutch Schulz
excellent | Planning | Clyde
excellent | Lock-Picking | Clyde
excellent | Scouting | Clyde
excellent | Preaching | Bonnie
excellent | Safe-Cracking | Meyer Lansky
excellent | Planning | Boo Boo Hoff
excellent | Planning | King Solomon
excellent | Guarding | Buggy Siegel
excellent | Driving | Buggy Siegel
excellent | Driving | Longy Zwillman
excellent | Gun-Shooting | Waxey Gordon
excellent | Lock-Picking | Greasy Guzik
excellent | Preaching | Greasy Guzik
excellent | Explosives | Sonny Genovese
...skipping 1 line
excellent | Safe-Cracking | Sonny Genovese
good | Eating | Vito Genovese
good | Money Counting | Mickey Cohen
good | Money Counting | Kid Cann
good | Scouting | Vito Genovese
good | Cooking | Vito Genovese
very good | Lock-Picking | Sonny Genovese
very good | Safe-Cracking | Moe Dalitz
very good | Planning | King Solomon
very good | Planning | Al Capone
very good | Safe-Cracking | Al Capone
very good | Preaching | Al Capone
very good | Driving | Lepke Buchalter
very good | Explosives | Sonny Genovese
very good | Guarding | Lepke Buchalter
very good | Explosives | Buggy Malone
very good | Guarding | Anastasia
very good | Driving | Longy Zwillman
very good | Safe-Cracking | Sonny Genovese
weak | Preaching | Greasy Guzik
weak | Planning | Clyde

```

7. 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.

STEP WISE APPROACH:

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

```
CREATE VIEW robberAmt as (  
SELECT Security, COUNT(Security) AS NumberRobberies, ROUND(AVG(Amount),2 ) AS  
AverageAmount  
FROM securityAmt  
GROUP BY Security  
ORDER BY NumberRobberies DESC);
```

```
select * from robberAmt;
```

```
project1=> select * from robberAmt;  
security | numberrobberies | averageamount  
-----+-----+-----  
excellent | 12 | 39238.08  
weak | 4 | 2299.50  
very good | 3 | 12292.43  
good | 2 | 3980.00  
(4 rows)  
  
project1=> 
```

SINGLE NESTED QUERY:

```
SELECT Security AS SecurityLevel, COUNT(Security) AS NumberRobberies,  
ROUND(AVG(Amount), 2) AS AverageAmount  
FROM (SELECT b.BankName, b.City, b.Security, r.Amount  
FROM Robberies r NATURAL JOIN Bank b) AS sec  
GROUP BY Security  
ORDER BY NumberRobberies DESC;
```

```
project1=> SELECT Security AS SecurityLevel, COUNT(Security) AS NumberRobberies, ROUND(AVG(Amount), 2) AS AverageAmount  
project1-> FROM (SELECT b.BankName, b.City, b.Security, r.Amount  
project1-> FROM Robberies r NATURAL JOIN Bank b) AS sec  
project1-> GROUP BY Security  
project1-> ORDER BY NumberRobberies DESC;  
securitylevel | numberrobberies | averageamount  
-----+-----+-----  
excellent | 12 | 39238.08  
weak | 4 | 2299.50  
very good | 3 | 12292.43  
good | 2 | 3980.00  
(4 rows)  
  
project1=> 
```