Swen 304 Project 1

Cameron Richards 300363518

1 - Database Design

Relations

Banks

Attributes: BankName, City, NoAccounts, Security
Attribute Constraints: NoAccounts > 0
Primary Key: {BankName, City}
Foreign Keys: None

Because the local business rules dictate that no bank in the same city may have the same name, this provides a guarantee that this key will always be unique, and no ID will need to be generated.

Robberies

Attributes: BankName, City, Amount, Date
Attribute Constraints: Amount > 0
Primary Key: {BankName, City, Date}
Foreign Keys:
{BankName, City} REFERENCES Banks{BankName, City}
Any combination of {BankName, City} entered into the relations should match with an entry in the Banks relation, as it should be an actual and known bank.

Update/Deletion Protocol:
Cascade will be used for update and delete events, as information about robberies on no longer existing banks is probably not very useful.

Plans

Attributes: BankName, City, NoRobbers, PlannedDate
Attribute Constraints: NoRobbers > 0; as a planned robbery must have at least one robber to carry it out.
Primary Key: {BankName, City, PlannedDate}
As we are dealing with one gang of robbers, we can assume that there will only be one possible robbery on a given bank on a given date.

Foreign Keys:
(BankName,City) REFERENCES Banks (BankName,City)
Any planned robbery must be upon an actual bank which should be stored in the reference of all banks.

Update/Deletion Protocol: Cascade will be used for update and delete events, as information about planned robberies on no longer existing banks is irrelevant.

Robbers

Attributes: RobberId, Nickname, Age, NoYears
Attribute Constraints: Age > 0 (Robber must be older than 0), NoYears >= 0 (NoYears spent in prison cannot be negative), Age > NoYears (You must be older than the amount of time you have spent in prison.)
Primary Key: {RobberId}

This is an incremented integer, guaranteed to be unique for all entries and is a perfect primary key.

Foreign Keys: None

Skills

Attributes: SkillId, Description
Attribute Constraints: Description should be UNIQUE to avoid storing the same skill multiple times under different IDS
Primary Key: {SkillId}

This is an incremented integer, guaranteed to be unique for all entries and is a perfect primary key.

Foreign Keys:

HasSkills

Attributes: RobberId, SkillId, Preference, Grade

Attribute Constraints: Preference > 0 (Cannot be 0 ranked preference),

Primary Key: {RobberId, SkillId}

Any robber can only have each skill once.

Foreign Keys:

(SkillId) REFERENCES Skills(SkillId)
Any skill referenced in the relation should be a valid and recorded skill in the skills relation.

(RobberId) REFERENCES Robbers(RobberId)
Any robber referenced in the relation should be a valid and recorded robber in the robber relation.

Update/Deletion Protocol: Cascade for deletion, as if a robber or skill is no longer stored in their respective relations, it is not relevant to continue to store them in this relation. Restrict for Update as incrementally generated IDs should remain static and unchanged.

HasAccounts

Attributes: RobberId, BankName, City

Attribute Constraints:

Primary Key: {RobberId, BankName, City}

Foreign Keys:

{RobberId} REFERENCES Robbers{RobberId}
Any robber referenced should be stored in the reference of all known robbers.

(BankName,City) REFERENCES Banks(BankName,City))
Banks that are referenced to in the hasAccounts table must actually exist in the Banks table

Update/Deletion Protocol: Cascade for Deletion for both foreign keys as HasAccounts should only reference valid and stored Banks and Robbers. Cascade for update for {BankName, City} as changes to either of these fields should be reflected in this HasAccounts as any accounts held by a robber should reference actual Banks with correct information. Restrict on update for {RobberId} as incrementally generated IDs should remain static.

Accomplices

Attributes: RobberId, BankName, City, Date, Share

Attribute Constraints: Share >= 0.

Primary Key: {RobberId, BankName, City, Date}

As we need to correctly identify a particular robbery to identify the Accomplices, we can use the primary key from robberies {BankName,City,Date} along with {RobberID} as an accomplice can only have one participation in a particular robbery.

Foreign Keys:

{RobberId} REFERENCES Robbers{RobberId}
Any robber referenced should be stored in the reference of all known robbers.

(BankName,City) REFERENCES Banks(BankName,City))
Banks that are referenced to in the hasAccounts table must actually exist in the Banks table

Update/Deletion Protocol: Cascade for Deletion for both foreign keys as HasAccounts should only reference valid and stored Banks and Robbers. Cascade for update for {BankName, City} as changes to either of these fields should be reflected in this HasAccounts relation to ensure accomplices reference actual Banks with correct information. Restrict on update for {RobberId} as incrementally generated IDs should remain static.

2 - Population

I started first with the tables which could be populated directly from their respective files (Banks, Robbers, Robberies and Plans. From there I poulated all the skills tables using all the distinct skills from the temporary view tempSkills, before populating the hasSkills table as this table stores the skillIds which are generated within the skills table. Similarly, I used a temporary table to assist in populating hasAccounts as this requires the robberID generated within the robber table. Finally I populated the accomplices table.

Question 3

1

```
INSERT INTO Skills VALUES (21,'Driving');
--Should Fail, Description not unique
```

```
psql:/Users/richards/Desktop/University/Swen304/Project1/304P1Q3.sql:5: ERROR:  duplicate key value violates unique constraint "skills_description_key"
DETAIL:  Key (description)=(Driving           ) already exists.
```

2

```
INSERT INTO Banks (BankName,City,NoAccounts,Security)
Values ('Loanshark Bank','Evanston',100,'very good');
--Failing Correctly, Primary Key Constraint
```

```
psql:/Users/richards/Desktop/University/Swen304/Project1/304P1Q3.sql:12: ERROR:  duplicate key value violates unique constraint "banks_pkey"
DETAIL:  Key (bankname, city)=(Loanshark Bank      , Evanston           ) already exists.
```

```
INSERT INTO Banks (BankName,City,NoAccounts,Security)
Values ('EasyLoan Bank','Evanston',-5,'excellent');
--Failing Correctly, Negative Account Contraint
```

```
psql:/Users/richards/Desktop/University/Swen304/Project1/304P1Q3.sql:16: ERROR:  new row for relation "banks" violates check constraint "accounts_positive"
DETAIL:  Failing row contains (EasyLoan Bank       , Evanston           , -5, excellent          ).
```

```
INSERT INTO Banks (BankName,City,NoAccounts,Security)
Values ('EasyLoan Bank','Evanston',100,'poor');
--Not yet Failing
```

(Should fail as not valid security rating)

```
INSERT 0 1
```

3

```
INSERT INTO Robberies VALUES ('NXP Bank','Chicago','2019-01-08',1000);
--Failing Correctly, Primary Key already exists
```

```
psql:/Users/richards/Desktop/University/Swen304/Project1/304P1Q3.sql:27: ERROR:  duplicate key value violates unique constraint "robberies_pkey"
DETAIL:  Key (bankname, city, daterobbed)=(NXP Bank           , Chicago            , 2019-01-08         ) already exists.
```

4

```
DELETE FROM Skills
WHERE SkillId = 1
AND Description = 'Driving';
```

```
DELETE 0
```

5

```sql
DELETE FROM Banks
WHERE BankName = 'PickPocket Bank'
AND City = 'Evanston'
AND NoAccounts = 2000
AND Security = 'very good';
--Failing Correctly, still referenced from accomplices
```

```
psql:/Users/richards/Desktop/University/Swen304/Project1/304P1Q3.sql:46: ERROR:  update or delete on table "robberies" violates foreign key constraint "accomplices_bankname_city_robberydate_fkey" on table "accomplices"
DETAIL:  Key (bankname, city, daterobbed)=(PickPocket Bank      , Evanston            , 2016-03-30          ) is still referenced
```

6

```sql
DELETE FROM Banks
WHERE BankName = 'Loanshark Bank'
AND City = 'Chicago';
```

```
from table "accomplices".
psql:/Users/richards/Desktop/University/Swen304/Project1/304P1Q3.sql:56: ERROR:  update or delete on table "robberies" violates foreign key constraint "accomplices_bankname_city_robberydate_fkey" on table "accomplices"
DETAIL:  Key (bankname, city, daterobbed)=(Loanshark Bank       , Chicago             , 2017-11-09          ) is still referenced
```

7

```sql
INSERT INTO Robbers VALUES (1,'Shotgun',70,0);
--Failing Correctly, Robber Id = 1 already exists
```

```
from table "accomplices".
psql:/Users/richards/Desktop/University/Swen304/Project1/304P1Q3.sql:63: ERROR:  duplicate key value violates unique constraint "robbers_pkey"
DETAIL:  Key (robberid)=(1) already exists.
```

```sql
INSERT INTO Robbers VALUES (333,'Jail Mouse', 25, 35);
--Failing Correctly, PrisonTime < Age Constraint
```

```
psql:/Users/richards/Desktop/University/Swen304/Project1/304P1Q3.sql:66: ERROR:  new row for relation "robbers" violates check constraint "age_more_than_prison"
DETAIL:  Failing row contains (333, Jail Mouse          , 25, 35).
```

8

```sql
INSERT INTO HasSkills VALUES (1, 7, 1, 'A+');
```

(Primary key constraint)

```
psql:/Users/richards/Desktop/University/Swen304/Project1/304P1Q3.sql:73: ERROR:  duplicate key value violates unique constraint "hasskills_pkey"
DETAIL:  Key (robberid, skillid)=(1, 7) already exists.
```

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

```
INSERT 0 1
```

```sql
INSERT INTO HasSkills VALUES (333, 1, 1, 'B-');
--Failing Correctly, no robber with ID = 333
```

```
psql:/Users/richards/Desktop/University/Swen304/Project1/304P1Q3.sql:78: ERROR:  insert or update on table "hasskills" violates foreign key constraint "hasskills_robberid_fkey"
DETAIL:  Key (robberid)=(333) is not present in table "robbers".
```

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

```
psql:/Users/richards/Desktop/University/Swen304/Project1/304P1Q3.sql:81: ERROR:  insert or update on table "hasskills" violates foreign key constraint "hasskills_skillid_fkey"
DETAIL:  Key (skillid)=(20) is not present in table "skills".
```

Question 4

Task 1

```sql
--Select all banks that have not been robbed



SELECT Banks.BankName, Banks.City
FROM Banks
left outer join Robberies
on Robberies.City = Banks.City and Banks.BankName = Robberies.BankName
WHERE Robberies.City IS NULL and Robberies.bankname IS NULL;
```

```
      bankname       |         city
----------------------+----------------------
 Bankrupt Bank        | Evanston
 Loanshark Bank       | Deerfield
 Inter-Gang Bank      | Chicago
 NXP Bank             | Evanston
 Dollar Grabbers      | Chicago
 Gun Chase Bank       | Burbank
 PickPocket Bank      | Deerfield
 Hidden Treasure      | Chicago
 Outside Bank         | Chicago
(9 rows)
```

Task 2

```sql
--Retrieve RobberId, Nickname, Age, and all skill descriptions of all robbers who are

```

```
SELECT r.robberid as robber_id,
r.nickname as nickname,
r.age as individual_earnings
FROM (SELECT * FROM
(SELECT robberid,
SUM(share) as earnings
FROM accomplices
GROUP BY robberid)
as robbery_earnings
WHERE earnings > 40000) as i
JOIN Robbers r
ON i.robberid = r.robberid
ORDER BY individual_earnings DESC;
```

```
 robber_id |       nickname        | individual_earnings
-----------+----------------------+---------------------
        16 | King Solomon         |                  74
        15 | Boo Boo Hoff         |                  54
        17 | Bugsy Siegel         |                  48
         3 | Lucky Luchiano       |                  42
        10 | Bonnie               |                  19
         5 | Mimmy The Mau Mau    |                  18
(6 rows)
```

Task 3

```
--Retrieve BankName and city of all banks where Al Capone has an account.

select distinct bankname, city
from hasaccounts
where robberid =
(select robberid
from robbers
where nickname = 'Al Capone');
```

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

Task 4

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

```sql
SELECT bankname,city,noaccounts
FROM Banks
where bankname NOT IN (
SELECT bankname
from Banks
where city = 'Chicago')
ORDER BY noaccounts;
```

```
      bankname       |         city         | noaccounts
----------------------+----------------------+------------
 Gun Chase Bank       | Burbank              |       1999
 Bankrupt Bank        | Evanston             |     444000
 Gun Chase Bank       | Evanston             |     656565
(3 rows)
```

Task 5

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


SELECT x.robberid as robber_id,
r.nickname as nickname,
x.earnings as individual_earnings
from (SELECT * FROM
(SELECT robberid,
SUM(share) as earnings
from accomplices
GROUP BY robberid)
as robbery_earnings
WHERE earnings > 40000) as x
join Robbers r
on x.robberid = r.robberid
ORDER BY individual_earnings DESC;
```

```
 robber_id |       nickname       | individual_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
(6 rows)
```

Task 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;
```

```
 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)
```

Task 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,(Age - NoYears) as Years_Out_Prison
from Robbers
where NoYears > Age / 2;
```

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

Task 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 skills.description as description,
r.robberid as robberid,
r.nickname as nickname
FROM (select skills.description as description,
hasskills.robberid as robber_id
from hasskills
join skills
ON hasskills.skillid = skills.skillid
ORDER BY description) as skills
JOIN Robbers r
ON r.robberid = skills.robber_id;
```

```
     description      | robberid |       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
: 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
: 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)
```

Question 5

Task 1

Code

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

Create VIEW robbery_years as(
SELECT SUBSTRING(DateRobbed, 1, 4) AS ExtractString, BankName, City
FROM Robberies);


Create VIEW plan_years as(
SELECT SUBSTRING(PlannedDate, 1, 4) AS ExtractString, BankName, City
FROM Plans);

Select distinct plan_years.BankName , plan_years.City
from plan_years
left outer join robbery_years
on plan_years.City = robbery_years.City and plan_years.BankName =
robbery_years.BankName
where robbery_years.City is null and robbery_years.BankName is null;
```

Output

```
      bankname       |        city
----------------------+----------------------
 Hidden Treasure      | Chicago
 PickPocket Bank      | Deerfield
 Loanshark Bank       | Deerfield
 Dollar Grabbers      | Chicago
(4 rows)
```

Task 2

Code

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

Create VIEW robbed_self as(
select accomplices.RobberId, accomplices.bankname
from accomplices
join HasAccounts
on HasAccounts.bankname = accomplices.bankname and HasAccounts.city = accomplices.city
and accomplices.RobberId = HasAccounts.RobberId);

Create VIEW didnt_rob as(
SELECT robbers.RobberId, robbers.NickName
FROM robbers
WHERE NOT EXISTS
   (SELECT *
    FROM robbed_self
    WHERE robbers.RobberId = robbed_self.RobberId)
);

select * from didnt_rob;
```

Output

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

Task 3

Code

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

CREATE VIEW multi_skilled as(
   SELECT count(*), RobberId
   FROM HasSkills
   GROUP BY RobberId
   HAVING COUNT(*) > 1
);

Create VIEW best_skill as(
SELECT HasSkills.RobberID, HasSkills.SkillId
FROM HasSkills
JOIN multi_skilled
on HasSkills.RobberId = multi_skilled.RobberId
where HasSkills.Preference = 1);

Create VIEW skill_desc as(
SELECT best_skill.RobberID, Skills.Description
FROM best_skill
JOIN skills
on best_skill.SkillId = skills.SkillId );

SELECT skill_desc.RobberId, robbers.Nickname, skill_desc.Description
FROM skill_desc
JOIN robbers
on robbers.RobberId= skill_desc.RobberId
```

Output

```
 robberid |       nickname        |      description
----------+----------------------+----------------------
       22 | Greasy Guzik         | Preaching
        3 | Lucky Luchiano       | Lock-Picking
       17 | Bugsy Siegel         | Driving
        5 | Mimmy The Mau Mau    | Planning
        7 | Dutch Schulz         | Lock-Picking
       24 | Sonny Genovese       | Explosives
        1 | Al Capone            | Planning
       18 | Vito Genovese        | Scouting
       23 | Lepke Buchalter      | Driving
```

```
        8 | Clyde               | Lock-Picking
(10 rows)
```

Task 4

Code

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


Create VIEW biggest_share as(
select bankname, city, share
from accomplices
order by share desc
LIMIT 1);

select robberies.bankname, robberies.city, robberies.DateRobbed
from robberies
join biggest_share
on robberies.city = biggest_share.city;
```

Output

```
      bankname       |         city         |      daterobbed
----------------------+----------------------+----------------------
 Loanshark Bank       | Evanston             | 2019-02-28
 Inter-Gang Bank      | Evanston             | 2018-02-14
 Penny Pinchers       | Evanston             | 2016-08-30
 Gun Chase Bank       | Evanston             | 2016-04-30
 PickPocket Bank      | Evanston             | 2016-03-30
 Loanshark Bank       | Evanston             | 2017-04-20
 Inter-Gang Bank      | Evanston             | 2016-02-16
 Penny Pinchers       | Evanston             | 2017-10-30
 PickPocket Bank      | Evanston             | 2018-01-30
 Penny Pinchers       | Evanston             | 2019-05-30
 Loanshark Bank       | Evanston             | 2016-04-20
 Inter-Gang Bank      | Evanston             | 2017-03-13
 Dollar Grabbers      | Evanston             | 2017-11-08
 Dollar Grabbers      | Evanston             | 2017-06-28
(14 rows)
```

Task 5

Code

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

Create VIEW distinct_accomplicies as(
Select distinct RobberId, BankName, City
from accomplices);


SELECT count(*), BankName, City
FROM distinct_accomplicies
GROUP BY BankName, City
HAVING COUNT(*) =
(SELECT COUNT(RobberId)
FROM robbers);
```

Output

```
 count | bankname | city
-------+----------+------
(0 rows)
```

(No cases where a bank has robbed by every robbery stored)

Question 6

Task 1

Step Wise:

```
CREATE VIEW robberies_for_each_robber as (
select robberid,
COUNT(robberid) as total_robberies,
SUM(share) as total_earnings
from accomplices
GROUP BY robberid);

CREATE VIEW average_robberies as (
SELECT AVG (total_robberies) as average_robberies
FROM robberies_for_each_robber);

CREATE VIEW active_robbers as (
select * from robberies_for_each_robber
WHERE total_robberies >
(select average_robberies
from average_robberies));

CREATE VIEW nicknames as (
select nickname
from active_robbers a
JOIN robbers r
ON r.robberid = a.robberid
WHERE r.noyears = 0
ORDER BY total_earnings DESC);

select * from nicknames;
```

Postgre Output

```
      nickname
----------------------
 Bonnie
 Clyde
 Sonny Genovese
(3 rows)
```

Nested:

```
select nickname
from
(select *
from (select robberid,
COUNT(robberid) as total_robberies,
SUM(share) as total_earnings
from accomplices
GROUP BY robberid) as robberies_for_each_robber
WHERE total_robberies > (SELECT AVG (total_robberies)
FROM (select robberid,
COUNT(robberid) as total_robberies,
SUM(share) as total_earnings
from accomplices
GROUP BY robberid) as robberies_for_each_robber)) as active_robbers
JOIN robbers
ON robbers.robberid = active_robbers.robberid
WHERE robbers.noyears = 0
ORDER BY total_earnings DESC;
```

Postgre Output

```
   nickname
----------------------
 Bonnie
 Clyde
 Sonny Genovese
(3 rows)
```

Task 2

Stepwise

```
CREATE VIEW robberiesSecurity as (
SELECT b.bankname as bankname,
b.city as city,
b.security as security,
r.amount as amount
FROM Banks b
JOIN Robberies r
ON b.bankname = r.bankname
AND b.city = r.city
ORDER BY b.security);


CREATE VIEW finalview as (
SELECT security as security_level,
COUNT(security) as total_number_of_robberies,
AVG(amount) as average_amount_stolen
FROM robberiesSecurity
GROUP BY security
ORDER BY total_number_of_robberies DESC);


SELECT * FROM finalview;
```

Output

```
   security_level    | total_number_of_robberies | average_amount_stolen
----------------------+---------------------------+------------------------
 excellent           |                        12 |     39238.083333333333
 weak                |                         4 |  2299.5000000000000000
 very good           |                         3 | 12292.4266666666666667
 good                |                         2 |  3980.0000000000000000
(4 rows)
```

Nested

```
SELECT security as security_level,
COUNT(security) as total_number_of_robberies,
AVG(amount) as average_amount_stolen
```

```sql
FROM (SELECT b.bankname as bankname,
b.city as city,
b.security as security,
r.amount as amount
FROM Banks b
JOIN Robberies r
ON b.bankname = r.bankname
AND b.city = r.city
ORDER BY b.security) as robberiesGroupedBySecurity
GROUP BY security
ORDER BY total_number_of_robberies DESC;
```

Output

```
   security_level    | total_number_of_robberies | average_amount_stolen
----------------------+---------------------------+------------------------
 excellent            |                        12 |     39238.083333333333
 weak                 |                         4 |  2299.5000000000000000
 very good            |                         3 | 12292.4266666666666667
 good                 |                         2 |  3980.0000000000000000
(4 rows)
```