The Failed Relationship Review Inc.

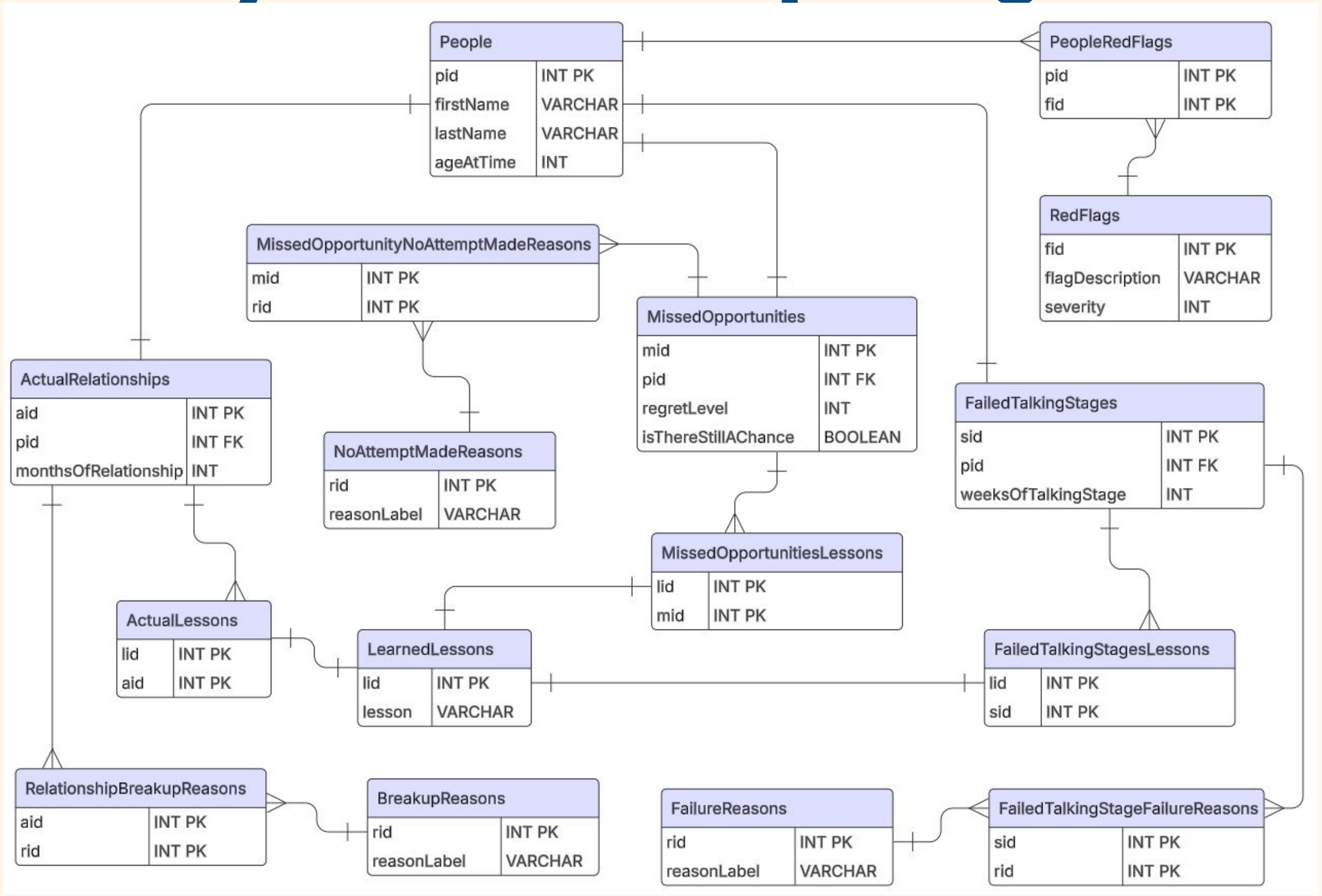
A Database Design Proposal
By Andrew Casamento

Executive Summary

The Failed Relationship Review Inc. is an organization dedicated to helping individuals learn from their failed relationship experiences. The purpose of this database is to organize all of one’s relationship “Ls”, if you will. Organizing this data using sound database design will allow for further analysis of failed relationships by expert analysts at The Failed Relationship Review Inc.

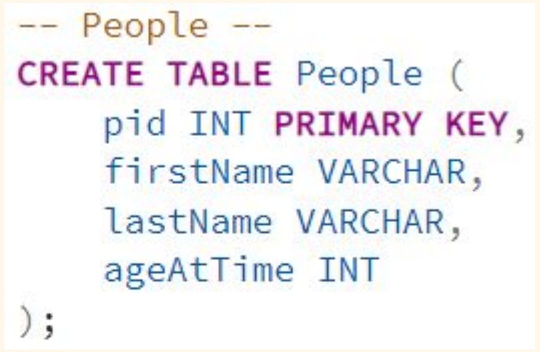
This report shows the design of the database, examples of helpful views that can be created for easier analysis of the data, how to query for certain data, and a variety of stored procedures and triggers that can be created to help make it easier to for failed relationship reviewers to conduct analysis of the data in the database. Also included are some database user roles with specific permissions granted that could make sense for this organization.

Entity Relationship Diagram



Create Table Statements

The ‘People’ table lists all of the people that the subject has had some sort of failed relationship with and basic information about each one.

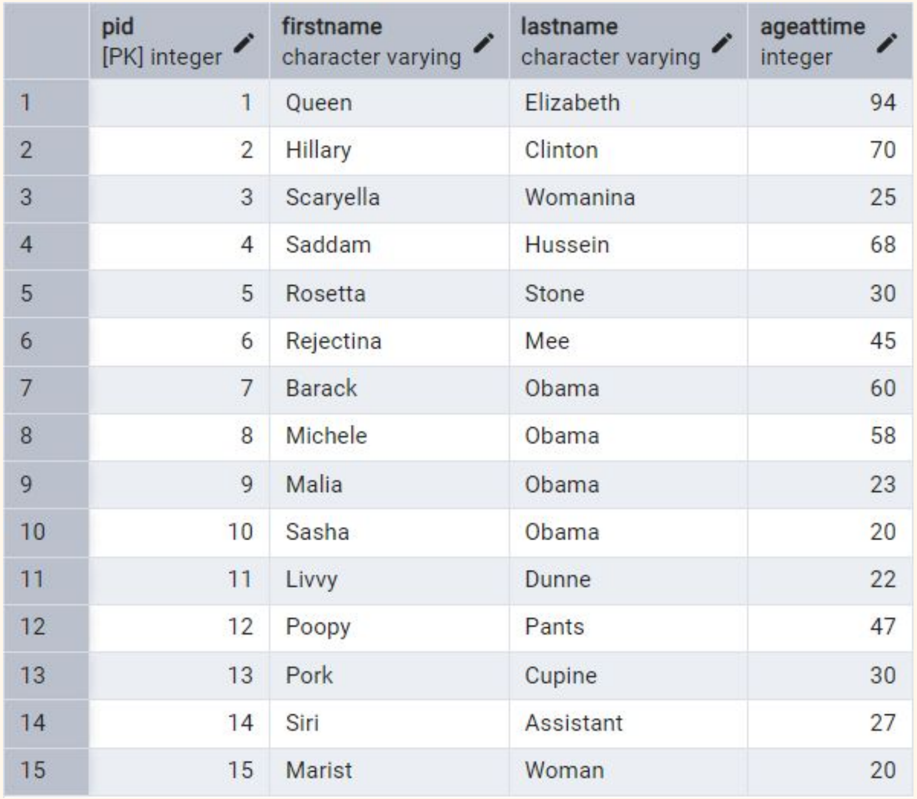


```
-- People --
CREATE TABLE People (
    pid INT PRIMARY KEY,
    firstName VARCHAR,
    lastName VARCHAR,
    ageAtTime INT
);
```

<u>FUNCTION DEPENDENCIES</u>

pid → firstName, lastName, ageAtTime

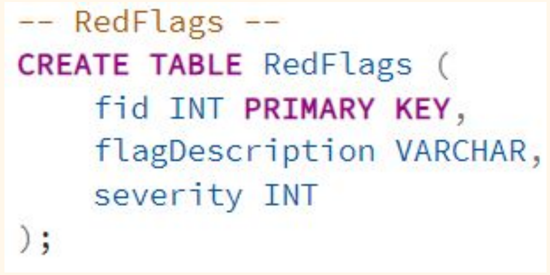
The People table with test data:



	pid [PK] integer	firstname character varying	lastname character varying	ageattime integer
1	1	Queen	Elizabeth	94
2	2	Hillary	Clinton	70
3	3	Scaryella	Womanina	25
4	4	Saddam	Hussein	68
5	5	Rosetta	Stone	30
6	6	Rejectina	Mee	45
7	7	Barack	Obama	60
8	8	Michele	Obama	58
9	9	Malia	Obama	23
10	10	Sasha	Obama	20
11	11	Livvy	Dunne	22
12	12	Poopy	Pants	47
13	13	Pork	Cupine	30
14	14	Siri	Assistant	27
15	15	Marist	Woman	20

Create Table Statements

The ‘RedFlags’ table lists common red flags with the severity level of that red flag on a scale of 1-10. The severity is determined by the subject.

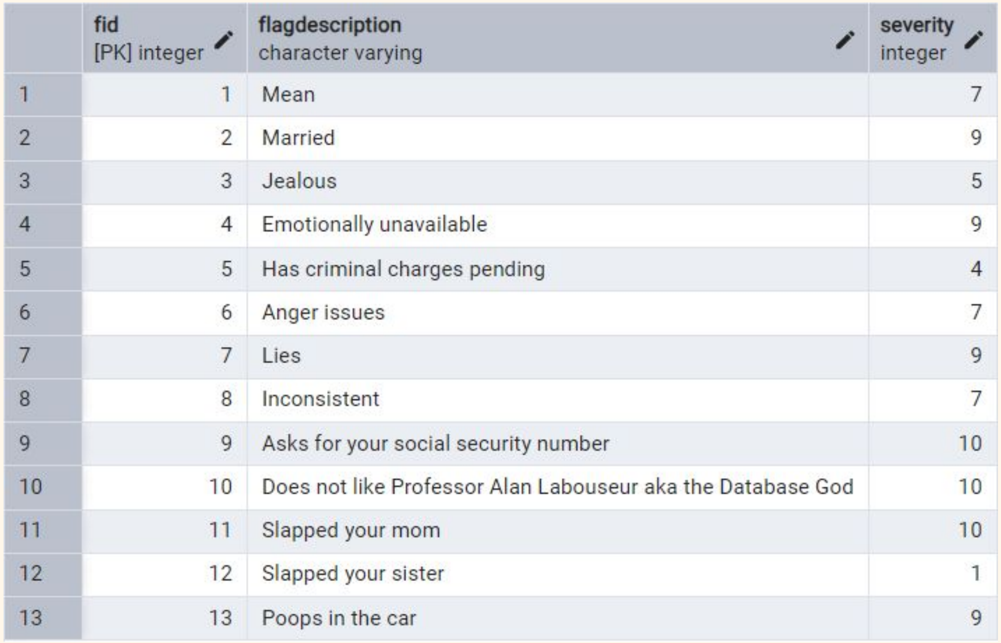


```
-- RedFlags --
CREATE TABLE RedFlags (
    fid INT PRIMARY KEY,
    flagDescription VARCHAR,
    severity INT
);
```

FUNCTION DEPENDENCIES

fid → flagDescription, severity

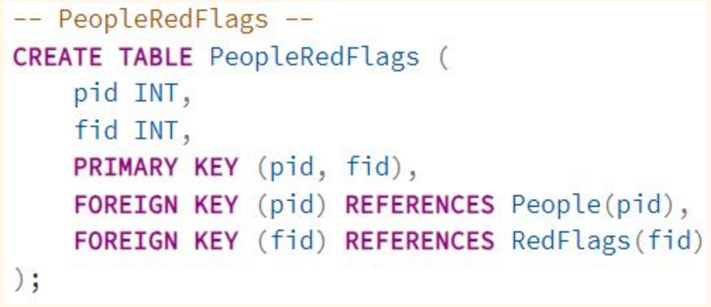
The RedFlags table with test data:



	fid [PK] integer	flagdescription character varying	severity integer
1	1	Mean	7
2	2	Married	9
3	3	Jealous	5
4	4	Emotionally unavailable	9
5	5	Has criminal charges pending	4
6	6	Anger issues	7
7	7	Lies	9
8	8	Inconsistent	7
9	9	Asks for your social security number	10
10	10	Does not like Professor Alan Labouseur aka the Database God	10
11	11	Slapped your mom	10
12	12	Slapped your sister	1
13	13	Poops in the car	9

Create Table Statements

The ‘PeopleRedFlags’ table is a mapping table for the ‘People’ table and the ‘RedFlags’ table. It ties together which people have which red flag(s).

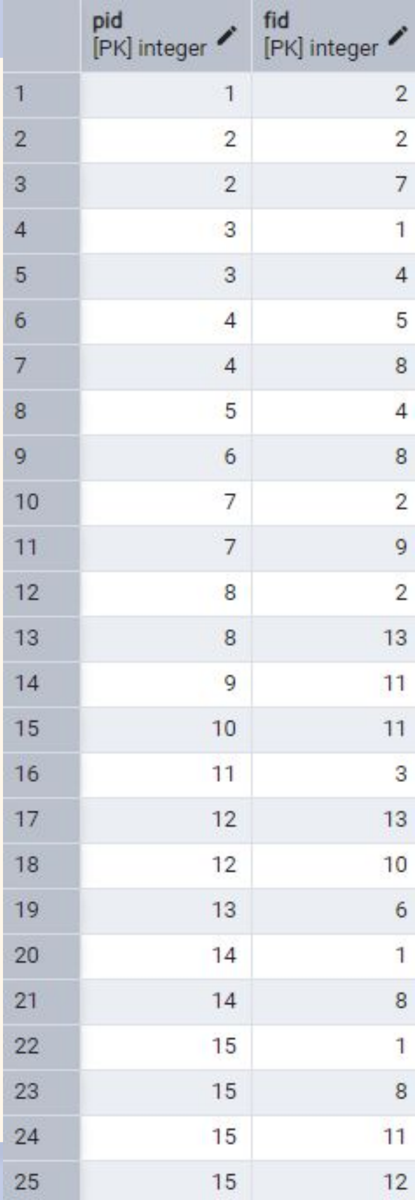


```
-- PeopleRedFlags --
CREATE TABLE PeopleRedFlags (
    pid INT,
    fid INT,
    PRIMARY KEY (pid, fid),
    FOREIGN KEY (pid) REFERENCES People(pid),
    FOREIGN KEY (fid) REFERENCES RedFlags(fid)
);
```

FUNCTION DEPENDENCIES

$\{pid, fid\} \rightarrow \varnothing$

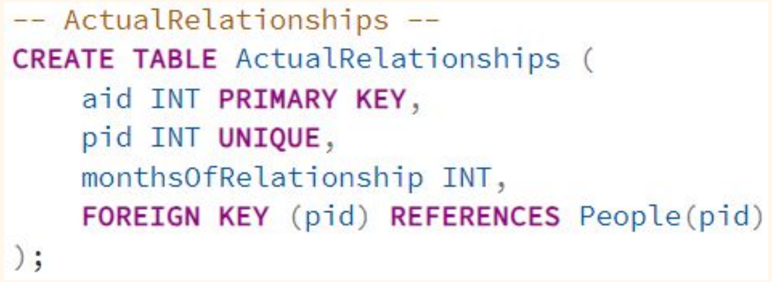
The ‘PeopleRedFlags’ table with test data:



	pid [PK] integer	fid [PK] integer
1	1	2
2	2	2
3	2	7
4	3	1
5	3	4
6	4	5
7	4	8
8	5	4
9	6	8
10	7	2
11	7	9
12	8	2
13	8	13
14	9	11
15	10	11
16	11	3
17	12	13
18	12	10
19	13	6
20	14	1
21	14	8
22	15	1
23	15	8
24	15	11
25	15	12

Create Table Statements

The 'ActualRelationships' table lists all relationships that made it to girlfriend/boyfriend status with the number of months the relationship lasted and who it was with.

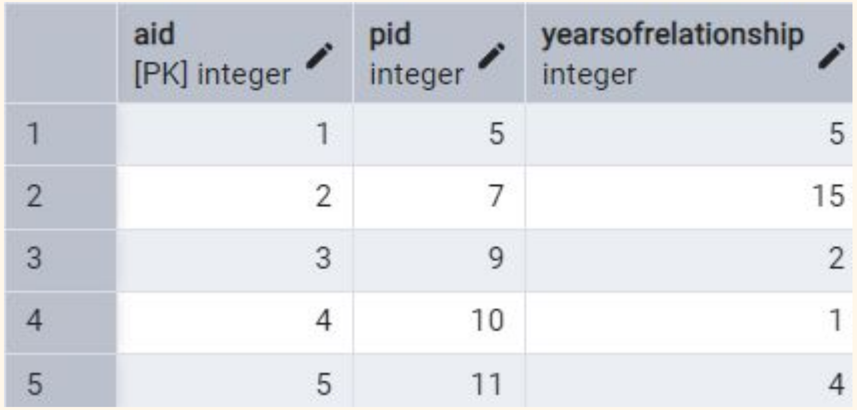


```
-- ActualRelationships --
CREATE TABLE ActualRelationships (
    aid INT PRIMARY KEY,
    pid INT UNIQUE,
    monthsOfRelationship INT,
    FOREIGN KEY (pid) REFERENCES People(pid)
);
```

<u>FUNCTION DEPENDENCIES</u>

aid → pid, monthsOfRelationship

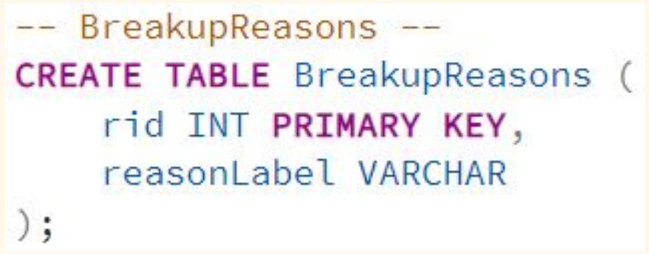
The 'ActualRelationships' table with test data:



	aid [PK] integer	pid integer	yearsofrelationship integer
1	1	5	5
2	2	7	15
3	3	9	2
4	4	10	1
5	5	11	4

Create Table Statements

The 'BreakupReasons' table lists all of the reasons an actual relationship ended.

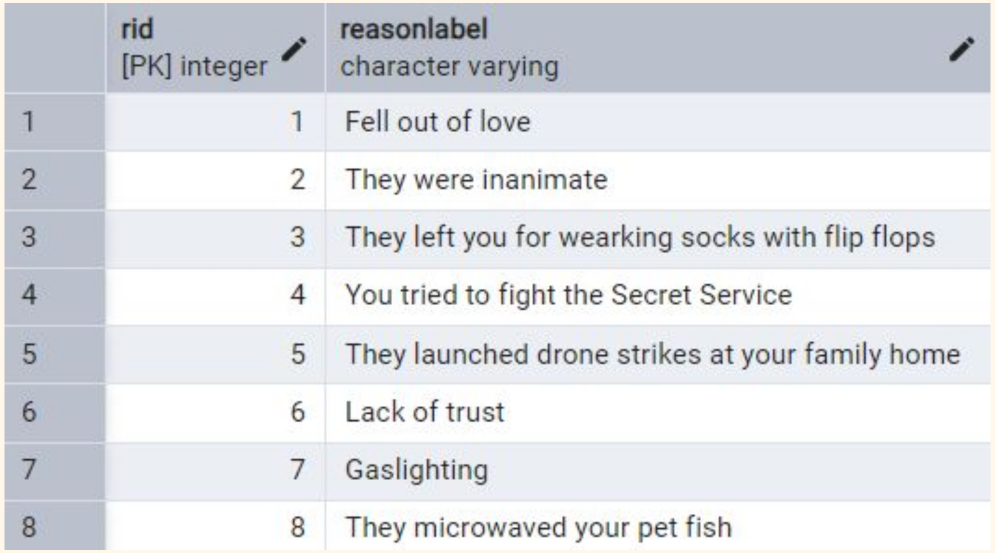


```
-- BreakupReasons --
CREATE TABLE BreakupReasons (
    rid INT PRIMARY KEY,
    reasonLabel VARCHAR
);
```

FUNCTION DEPENDENCIES

rid → reasonLabel

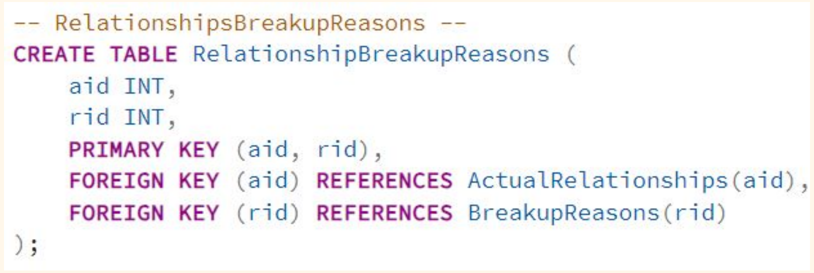
The 'BreakupReasons' table with test data:



	rid [PK] integer	reasonlabel character varying
1	1	Fell out of love
2	2	They were inanimate
3	3	They left you for wearking socks with flip flops
4	4	You tried to fight the Secret Service
5	5	They launched drone strikes at your family home
6	6	Lack of trust
7	7	Gaslighting
8	8	They microwaved your pet fish

Create Table Statements

The ‘RelationshipBreakupReasons’ table is another mapping table that takes an actual relationship and maps it with one or more reasons the relationship ended.

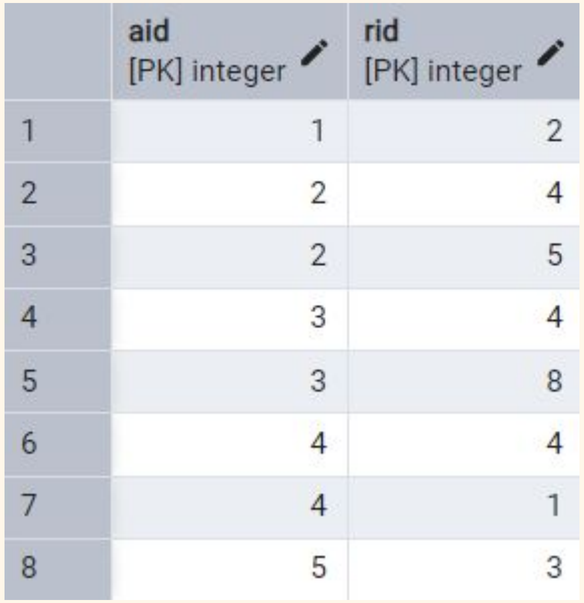


```sql
-- RelationshipsBreakupReasons --
CREATE TABLE RelationshipBreakupReasons (
    aid INT,
    rid INT,
    PRIMARY KEY (aid, rid),
    FOREIGN KEY (aid) REFERENCES ActualRelationships(aid),
    FOREIGN KEY (rid) REFERENCES BreakupReasons(rid)
);
```

FUNCTION DEPENDENCIES

$$\{aid, rid\} \rightarrow \varnothing$$

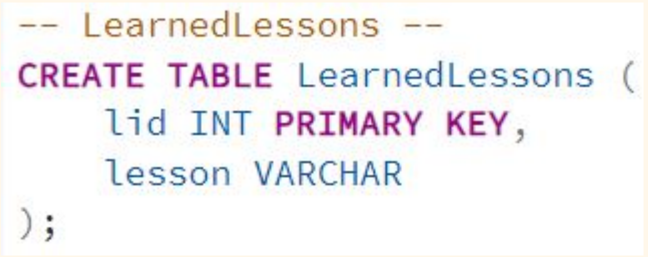
The ‘RelationshipBreakupReasons’ table with test data:



	aid [PK] integer	rid [PK] integer
1	1	2
2	2	4
3	2	5
4	3	4
5	3	8
6	4	4
7	4	1
8	5	3

Create Table Statements

The ‘LearnedLessons’ table lists all possible lessons learned from all failed relationship attempts.

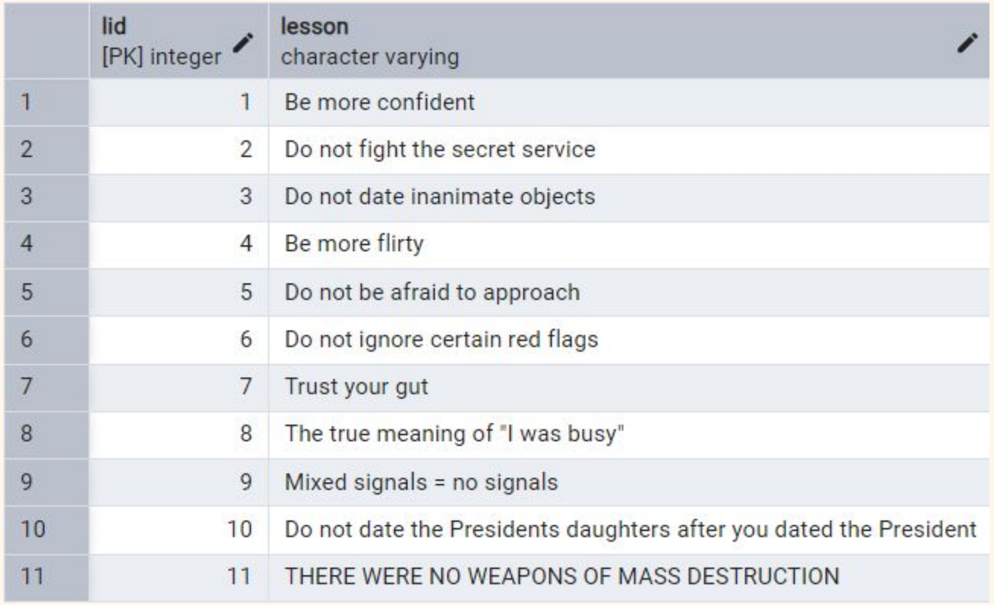


```sql
-- LearnedLessons --
CREATE TABLE LearnedLessons (
    lid INT PRIMARY KEY,
    lesson VARCHAR
);
```

FUNCTION DEPENDENCIES

lid → lesson

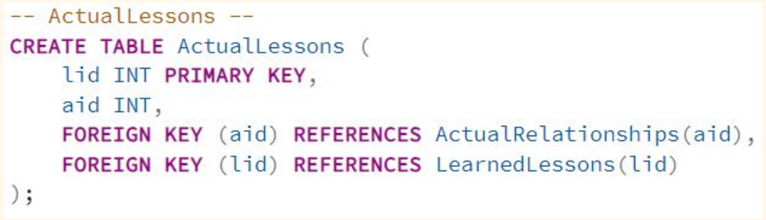
The ‘LearnedLessons’ table with test data:



	lid [PK] integer	lesson character varying
1	1	Be more confident
2	2	Do not fight the secret service
3	3	Do not date inanimate objects
4	4	Be more flirty
5	5	Do not be afraid to approach
6	6	Do not ignore certain red flags
7	7	Trust your gut
8	8	The true meaning of "I was busy"
9	9	Mixed signals = no signals
10	10	Do not date the Presidents daughters after you dated the President
11	11	THERE WERE NO WEAPONS OF MASS DESTRUCTION

Create Table Statements

The 'ActualLessons' table is a mapping table. It takes an actual relationship and matches it with the lesson that was learned from it by the subject.

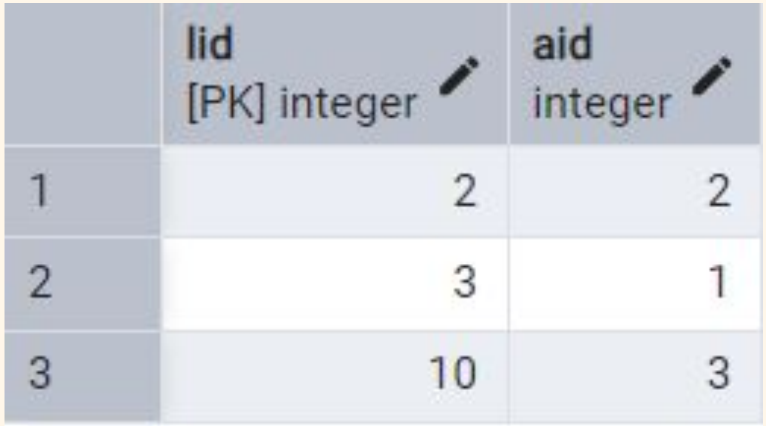


```
-- ActualLessons --
CREATE TABLE ActualLessons (
    lid INT PRIMARY KEY,
    aid INT,
    FOREIGN KEY (aid) REFERENCES ActualRelationships(aid),
    FOREIGN KEY (lid) REFERENCES LearnedLessons(lid)
);
```

FUNCTION DEPENDENCIES

{lid, aid} → ∅

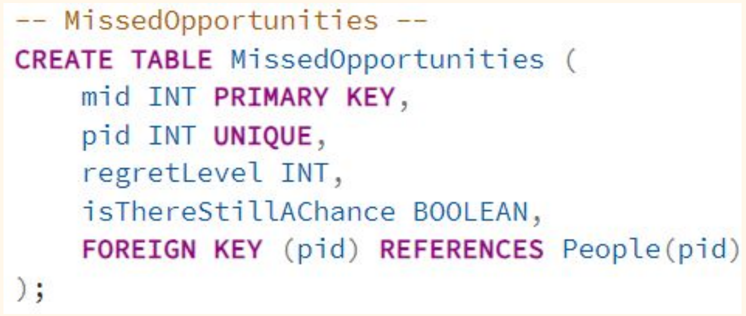
The 'ActualLessons' table with test data:



	lid [PK] integer	aid integer
1	2	2
2	3	1
3	10	3

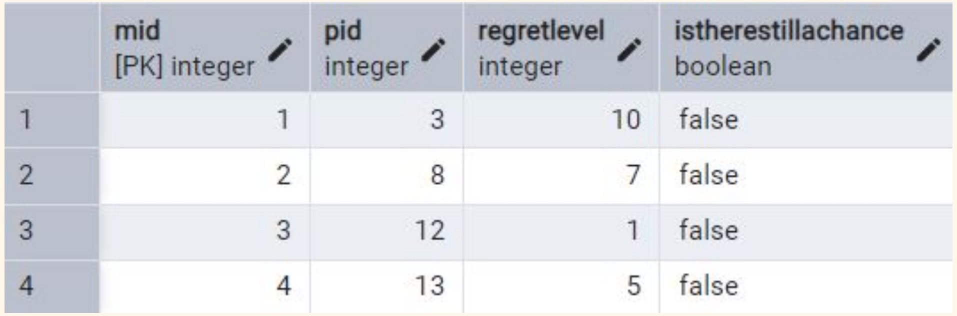
Create Table Statements

The ‘MissedOpportunities’ table lists all of the missed relationship opportunities that the subject has had because they did not attempt to even start a relationship. The table includes the subject’s regret level on a scale of 1-10 and if there is still a chance with the missed opportunity person.



```
-- MissedOpportunities --
CREATE TABLE MissedOpportunities (
    mid INT PRIMARY KEY,
    pid INT UNIQUE,
    regretLevel INT,
    isThereStillAChance BOOLEAN,
    FOREIGN KEY (pid) REFERENCES People(pid)
);
```

The ‘MissedOpportunities’ table with test data:



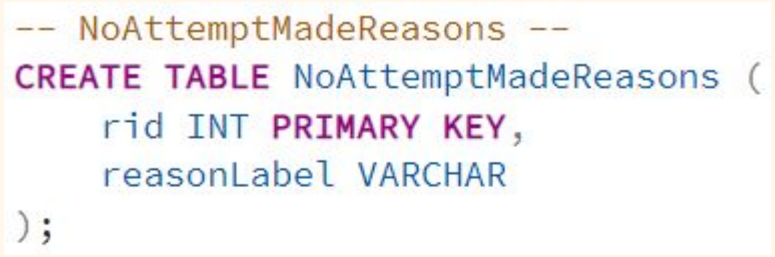
	mid [PK] integer	pid integer	regretlevel integer	istherestillachance boolean
1	1	3	10	false
2	2	8	7	false
3	3	12	1	false
4	4	13	5	false

<u>FUNCTION DEPENDENCIES</u>

mid → pid, regretLevel, isThereStillAChance

Create Table Statements

The ‘NoAttemptMadeReasons’ table lists all of the reasons (or excuses) that the subject did not attempt to start a relationship with a person.

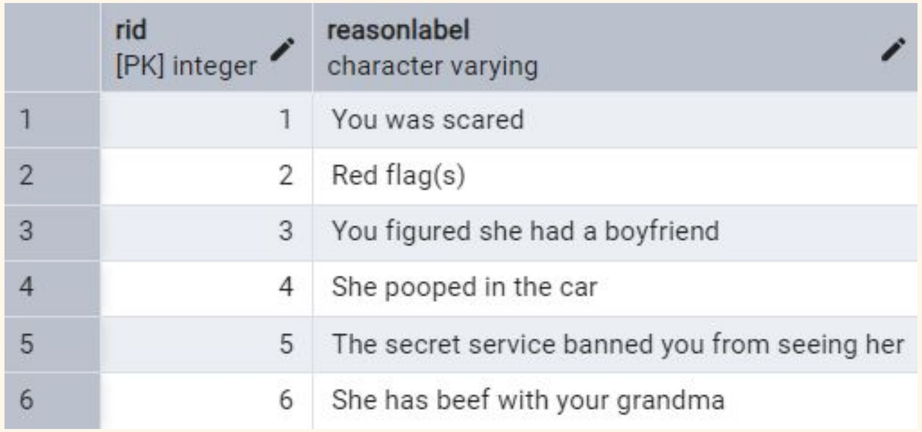


```
-- NoAttemptMadeReasons --
CREATE TABLE NoAttemptMadeReasons (
    rid INT PRIMARY KEY,
    reasonLabel VARCHAR
);
```

FUNCTION DEPENDENCIES

rid → reasonLabel

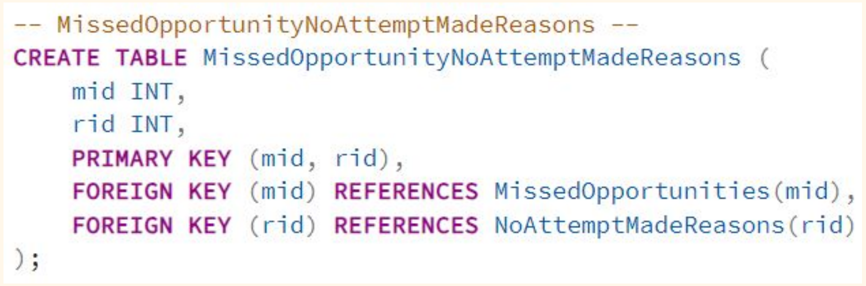
The ‘NoAttemptMadeReasons’ table with test data:



	rid [PK] integer	reasonlabel character varying
1	1	You was scared
2	2	Red flag(s)
3	3	You figured she had a boyfriend
4	4	She pooped in the car
5	5	The secret service banned you from seeing her
6	6	She has beef with your grandma

Create Table Statements

The 'MissedOpportunityNoAttemptMadeReasons' table is a mapping table that maps a missed relationship opportunity to a reason that no attempt was made to start a relationship.

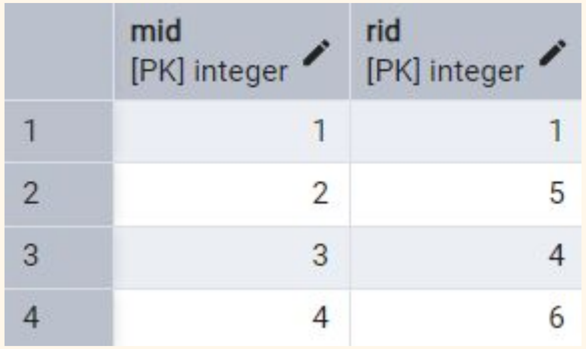


```
-- MissedOpportunityNoAttemptMadeReasons --
CREATE TABLE MissedOpportunityNoAttemptMadeReasons (
    mid INT,
    rid INT,
    PRIMARY KEY (mid, rid),
    FOREIGN KEY (mid) REFERENCES MissedOpportunities(mid),
    FOREIGN KEY (rid) REFERENCES NoAttemptMadeReasons(rid)
);
```

FUNCTION DEPENDENCIES

$\{mid, rid\} \rightarrow \varnothing$

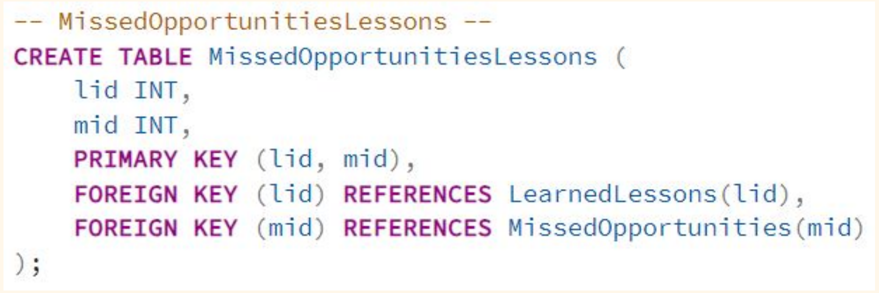
'MissedOpportunityNoAttemptMadeReasons' table with test data:



	mid [PK] integer	rid [PK] integer
1	1	1
2	2	5
3	3	4
4	4	6

Create Table Statements

The ‘MissedOpportunitiesLessons’ table is a mapping table. It takes an a missed opportunity and matches it with the lesson that was learned from it by the subject.

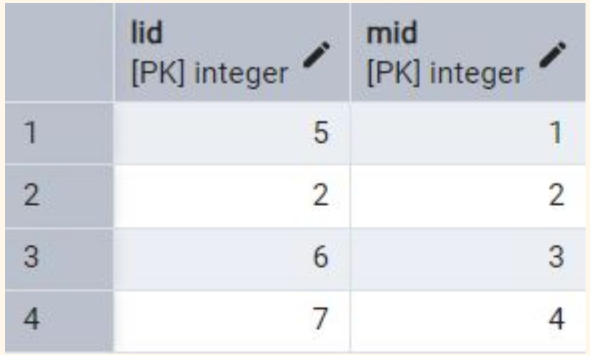


```
-- MissedOpportunitiesLessons --
CREATE TABLE MissedOpportunitiesLessons (
    lid INT,
    mid INT,
    PRIMARY KEY (lid, mid),
    FOREIGN KEY (lid) REFERENCES LearnedLessons(lid),
    FOREIGN KEY (mid) REFERENCES MissedOpportunities(mid)
);
```

<u>**FUNCTION DEPENDENCIES**</u>

$\{mid, lid\} \rightarrow \varnothing$

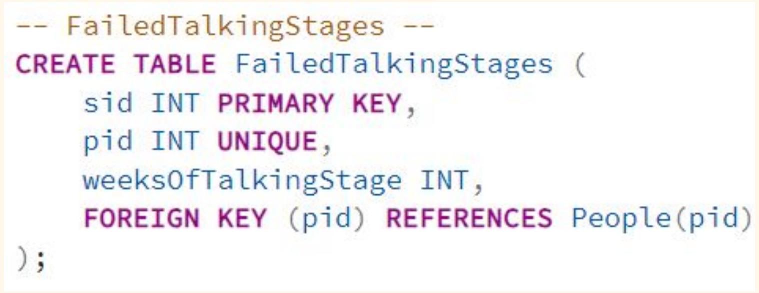
‘MissedOpportunitiesLessons’ table with test data:



	lid [PK] integer	mid [PK] integer
1	5	1
2	2	2
3	6	3
4	7	4

Create Table Statements

The ‘FailedTalkingStages’ table lists all of the failed talking stages experienced by the poor lonely subject, who it was with, and the number of weeks it lasted.

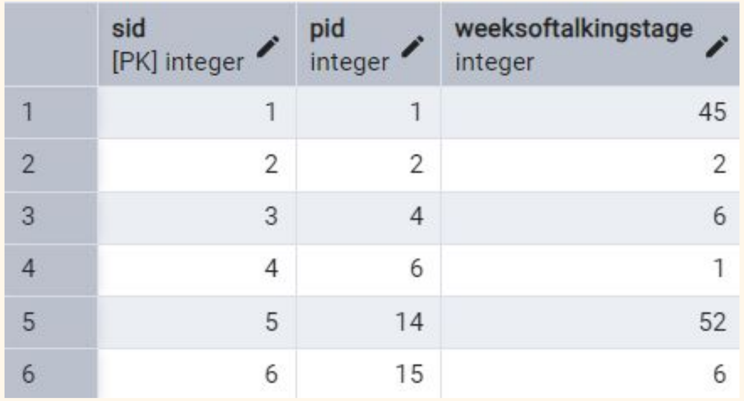


```
-- FailedTalkingStages --
CREATE TABLE FailedTalkingStages (
    sid INT PRIMARY KEY,
    pid INT UNIQUE,
    weeksOfTalkingStage INT,
    FOREIGN KEY (pid) REFERENCES People(pid)
);
```

FUNCTION DEPENDENCIES

sid → pid, weeksOfTalkingStage

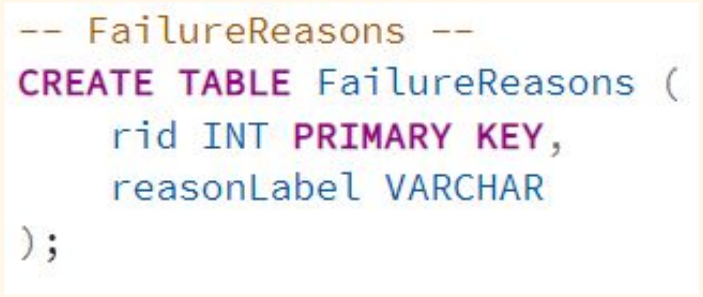
The ‘FailedTalkingStages’ table with test data:



	sid [PK] integer	pid integer	weeksoftalkingstage integer
1	1	1	45
2	2	2	2
3	3	4	6
4	4	6	1
5	5	14	52
6	6	15	6

Create Table Statements

The ‘FailureReasons’ table lists all of the reasons that a talking stage failed.

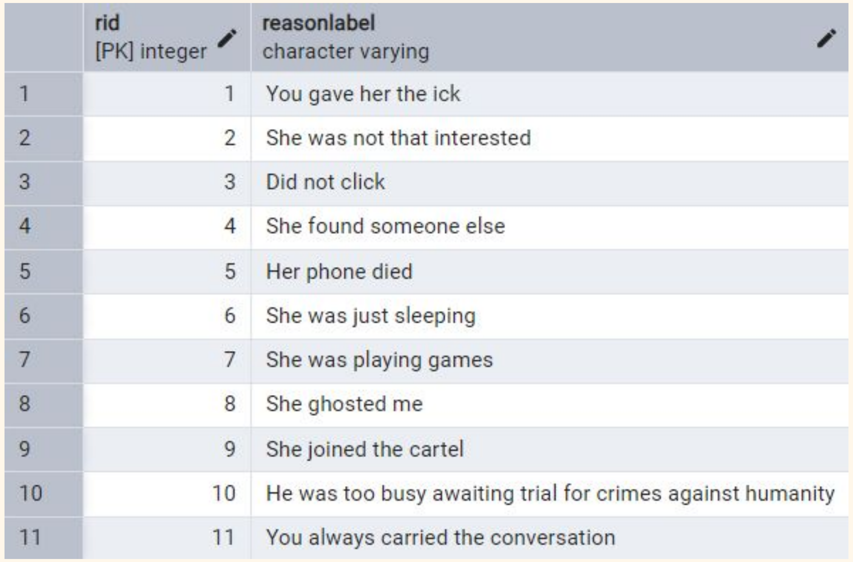


```
-- FailureReasons --
CREATE TABLE FailureReasons (
    rid INT PRIMARY KEY,
    reasonLabel VARCHAR
);
```

<u>FUNCTION DEPENDENCIES</u>

rid → reasonLabel

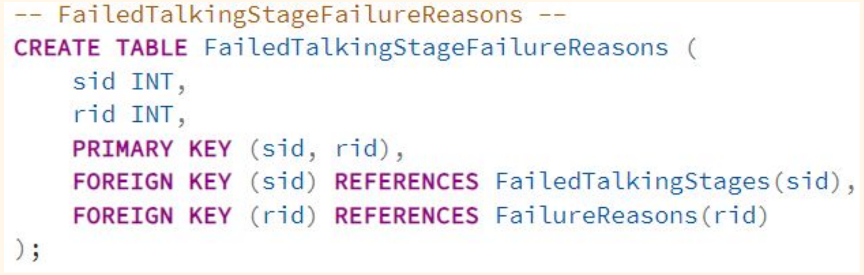
The ‘FailureReasons’ table with test data:



	rid [PK] integer	reasonlabel character varying
1	1	You gave her the ick
2	2	She was not that interested
3	3	Did not click
4	4	She found someone else
5	5	Her phone died
6	6	She was just sleeping
7	7	She was playing games
8	8	She ghosted me
9	9	She joined the cartel
10	10	He was too busy awaiting trial for crimes against humanity
11	11	You always carried the conversation

Create Table Statements

The ‘FailedTalkingStageFailureReasons’ table is a mapping table. It maps failed talking stages to reason(s) the talking stage failed.

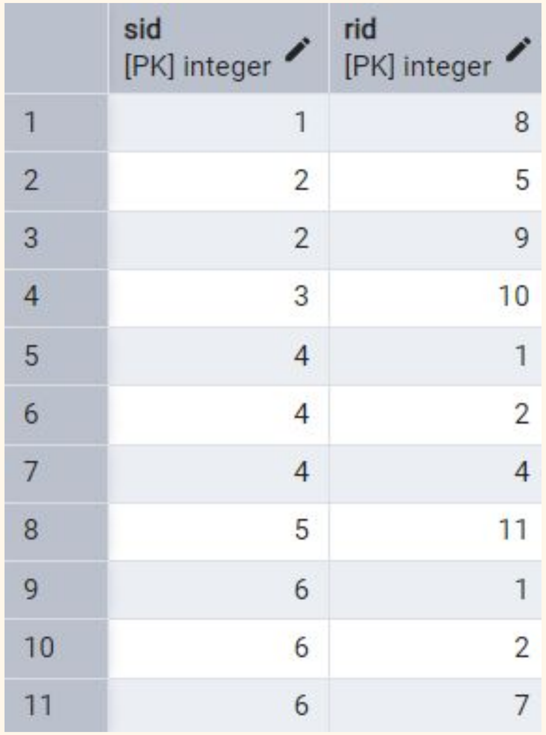


```
-- FailedTalkingStageFailureReasons --
CREATE TABLE FailedTalkingStageFailureReasons (
    sid INT,
    rid INT,
    PRIMARY KEY (sid, rid),
    FOREIGN KEY (sid) REFERENCES FailedTalkingStages(sid),
    FOREIGN KEY (rid) REFERENCES FailureReasons(rid)
);
```

FUNCTION DEPENDENCIES

$\{sid, rid\} \rightarrow \varnothing$

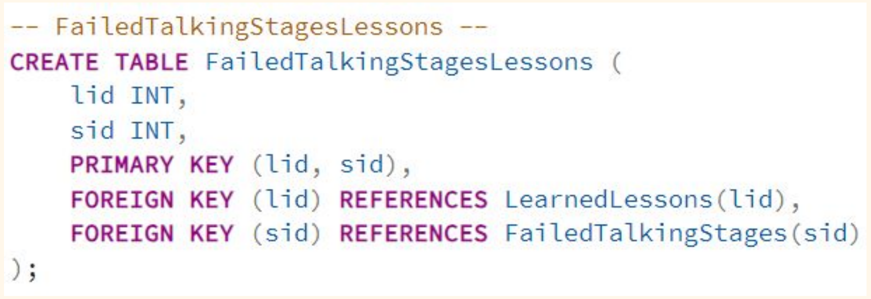
The ‘FailedTalkingStageFailureReasons’ table with test data:



	sid [PK] integer	rid [PK] integer
1	1	8
2	2	5
3	2	9
4	3	10
5	4	1
6	4	2
7	4	4
8	5	11
9	6	1
10	6	2
11	6	7

Create Table Statements

The ‘FailedTalkingStagesLessons’ table is a mapping table. It maps failed talking stages to a lesson that was learned from it.

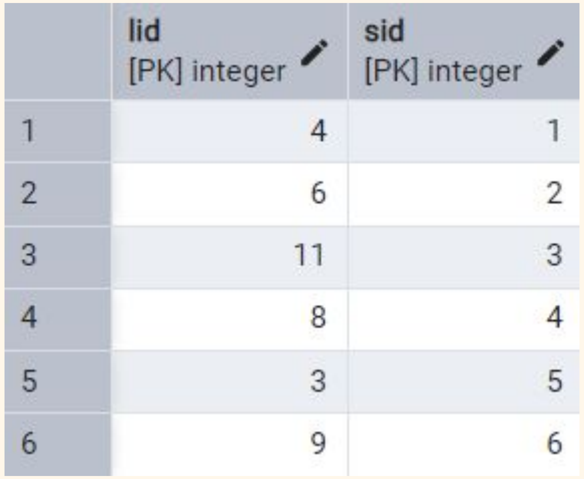


```
-- FailedTalkingStagesLessons --
CREATE TABLE FailedTalkingStagesLessons (
    lid INT,
    sid INT,
    PRIMARY KEY (lid, sid),
    FOREIGN KEY (lid) REFERENCES LearnedLessons(lid),
    FOREIGN KEY (sid) REFERENCES FailedTalkingStages(sid)
);
```

<u>FUNCTION DEPENDENCIES</u>

$\{sid, lid\} \rightarrow \varnothing$

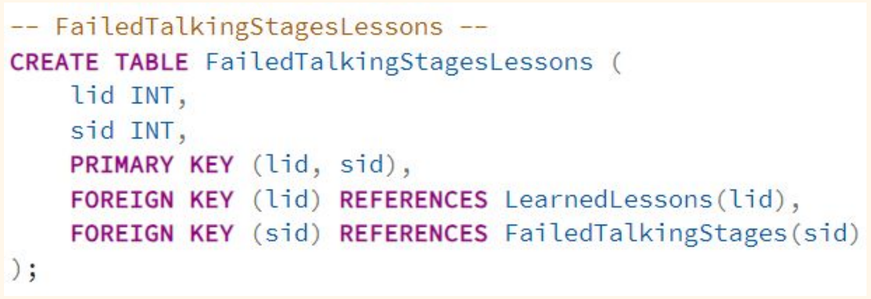
The ‘FailedTalkingStagesLessons’ table with test data:



	lid [PK] integer	sid [PK] integer
1	4	1
2	6	2
3	11	3
4	8	4
5	3	5
6	9	6

Create Table Statements

The ‘FailedTalkingStagesLessons’ table is a mapping table. It maps failed talking stages to a lesson that was learned from it.

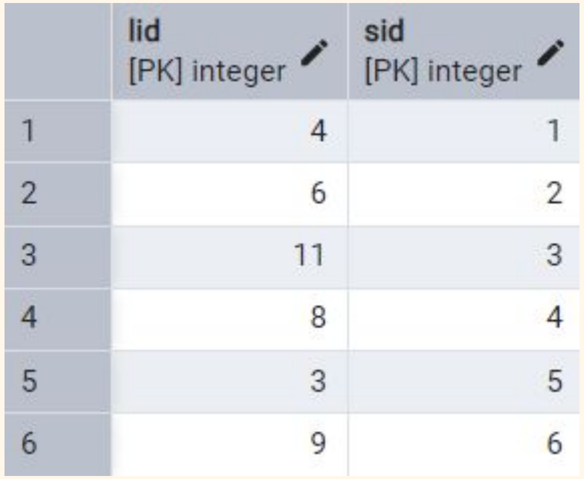


```
-- FailedTalkingStagesLessons --
CREATE TABLE FailedTalkingStagesLessons (
    lid INT,
    sid INT,
    PRIMARY KEY (lid, sid),
    FOREIGN KEY (lid) REFERENCES LearnedLessons(lid),
    FOREIGN KEY (sid) REFERENCES FailedTalkingStages(sid)
);
```

FUNCTION DEPENDENCIES

{sid, lid} → ∅

The ‘FailedTalkingStagesLessons’ table with test data:

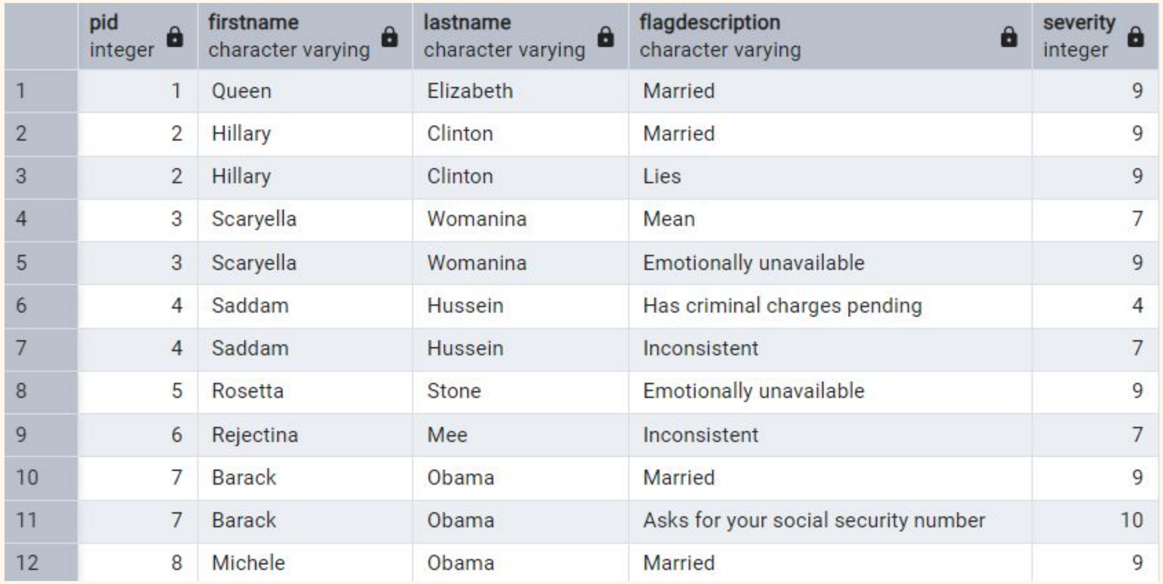


	lid [PK] integer	sid [PK] integer
1	4	1
2	6	2
3	11	3
4	8	4
5	3	5
6	9	6

Views

‘PeopleWithRedFlags’ view to show each person and the red flag(s) associated with them:

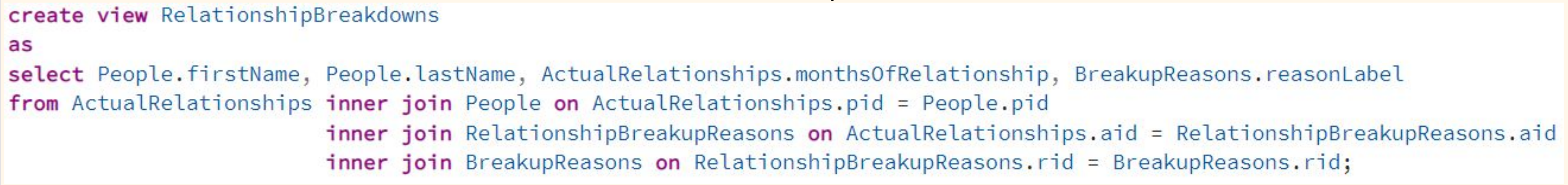
```
create view PeopleWithRedFlags
as
select People.pid, People.firstName, People.lastName, RedFlags.flagDescription, RedFlags.severity
from People inner join PeopleRedFlags on People.pid = PeopleRedFlags.pid
            inner join RedFlags on PeopleRedFlags.fid = RedFlags.fid;
```



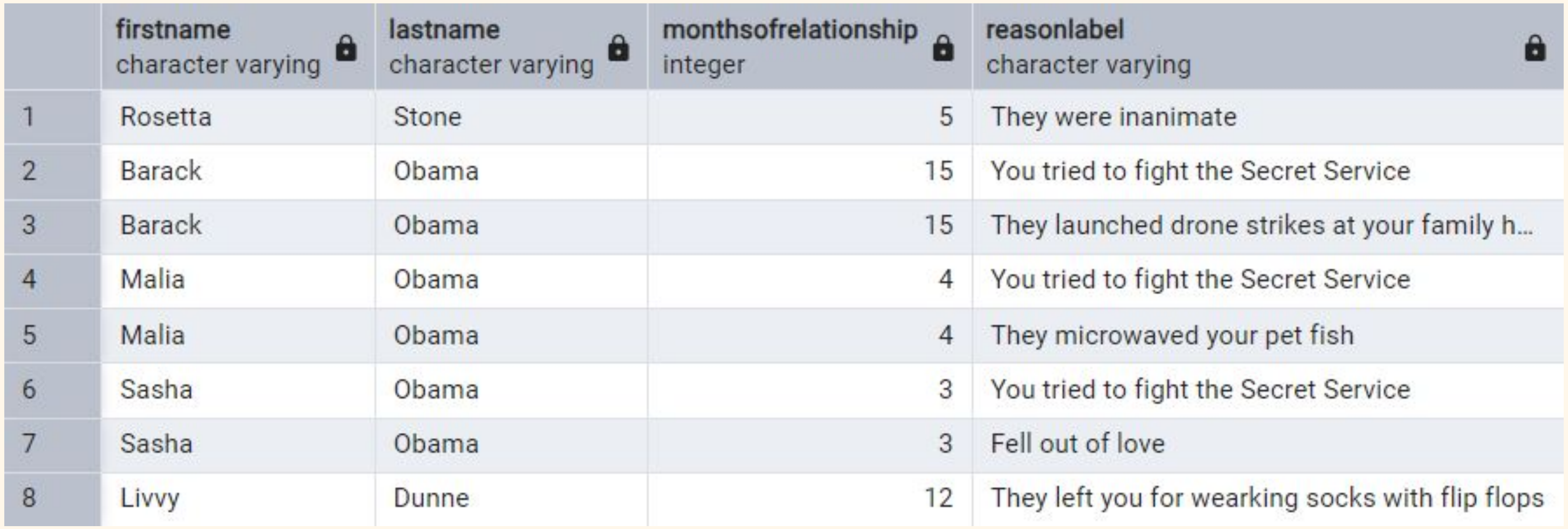
	pid integer	firstname character varying	lastname character varying	flagdescription character varying	severity integer
1	1	Queen	Elizabeth	Married	9
2	2	Hillary	Clinton	Married	9
3	2	Hillary	Clinton	Lies	9
4	3	Scaryella	Womanina	Mean	7
5	3	Scaryella	Womanina	Emotionally unavailable	9
6	4	Saddam	Hussein	Has criminal charges pending	4
7	4	Saddam	Hussein	Inconsistent	7
8	5	Rosetta	Stone	Emotionally unavailable	9
9	6	Rejectina	Mee	Inconsistent	7
10	7	Barack	Obama	Married	9
11	7	Barack	Obama	Asks for your social security number	10
12	8	Michele	Obama	Married	9

Views

‘RelationshipBreakdowns’ view to show each person, length of relationship, and reason for the breakup:



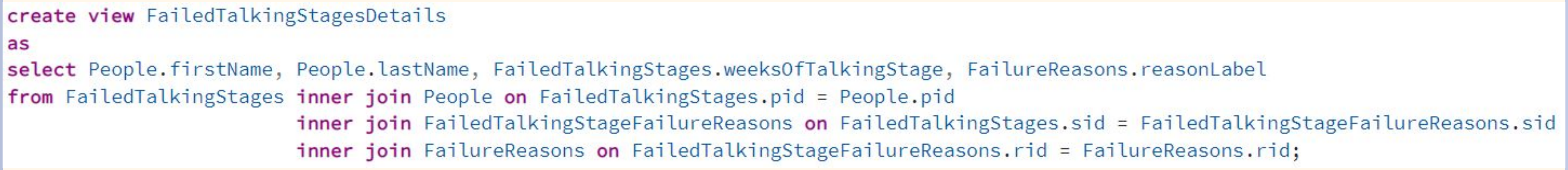
```
create view RelationshipBreakdowns
as
select People.firstName, People.lastName, ActualRelationships.monthsOfRelationship, BreakupReasons.reasonLabel
from ActualRelationships inner join People on ActualRelationships.pid = People.pid
                        inner join RelationshipBreakupReasons on ActualRelationships.aid = RelationshipBreakupReasons.aid
                        inner join BreakupReasons on RelationshipBreakupReasons.rid = BreakupReasons.rid;
```



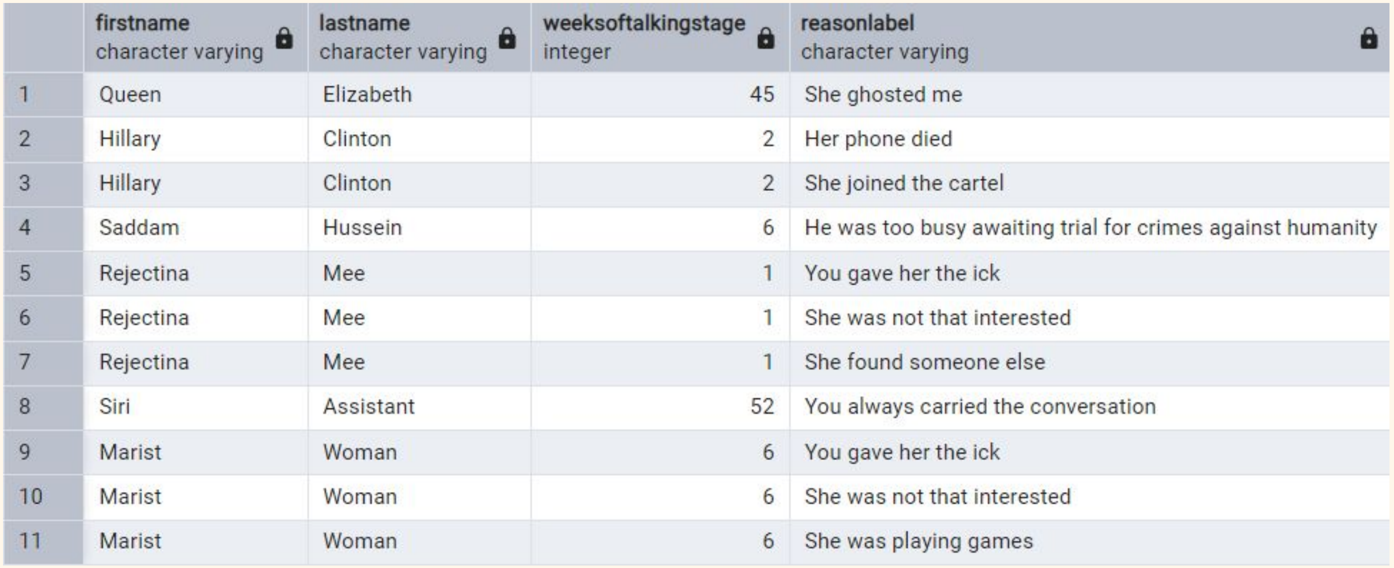
	firstname character varying	lastname character varying	monthsofrelationship integer	reasonlabel character varying
1	Rosetta	Stone	5	They were inanimate
2	Barack	Obama	15	You tried to fight the Secret Service
3	Barack	Obama	15	They launched drone strikes at your family h...
4	Malia	Obama	4	You tried to fight the Secret Service
5	Malia	Obama	4	They microwaved your pet fish
6	Sasha	Obama	3	You tried to fight the Secret Service
7	Sasha	Obama	3	Fell out of love
8	Livvy	Dunne	12	They left you for wearking socks with flip flops

Views

‘FailedTalkingStagesDetails’ view to show each failed talking stage, length of the failed talking stage, and why it failed:

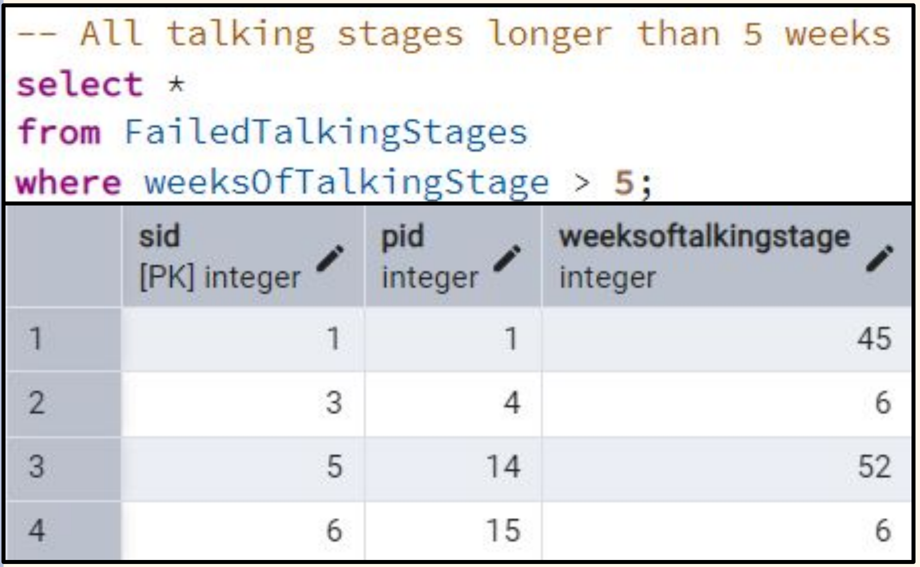


```sql
create view FailedTalkingStagesDetails
as
select People.firstName, People.lastName, FailedTalkingStages.weeksOfTalkingStage, FailureReasons.reasonLabel
from FailedTalkingStages inner join People on FailedTalkingStages.pid = People.pid
                         inner join FailedTalkingStageFailureReasons on FailedTalkingStages.sid = FailedTalkingStageFailureReasons.sid
                         inner join FailureReasons on FailedTalkingStageFailureReasons.rid = FailureReasons.rid;
```



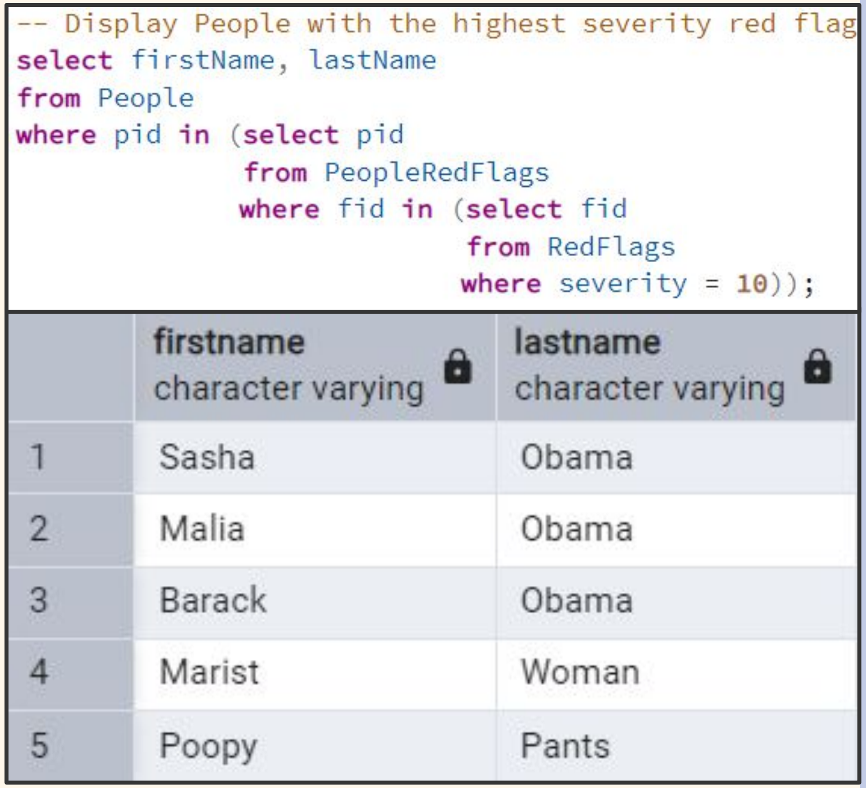
	firstname character varying	lastname character varying	weeksoftalkingstage integer	reasonlabel character varying
1	Queen	Elizabeth	45	She ghosted me
2	Hillary	Clinton	2	Her phone died
3	Hillary	Clinton	2	She joined the cartel
4	Saddam	Hussein	6	He was too busy awaiting trial for crimes against humanity
5	Rejectina	Mee	1	You gave her the ick
6	Rejectina	Mee	1	She was not that interested
7	Rejectina	Mee	1	She found someone else
8	Siri	Assistant	52	You always carried the conversation
9	Marist	Woman	6	You gave her the ick
10	Marist	Woman	6	She was not that interested
11	Marist	Woman	6	She was playing games

Reports



```sql
-- All talking stages longer than 5 weeks
select *
from FailedTalkingStages
where weeksOfTalkingStage > 5;
```

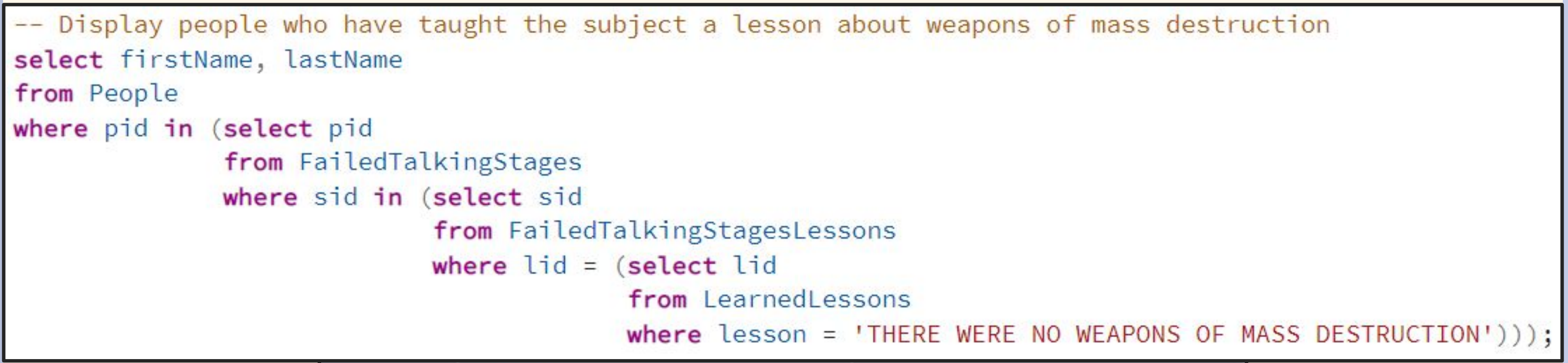
	sid [PK] integer	pid integer	weeksoftalkingstage integer
1	1	1	45
2	3	4	6
3	5	14	52
4	6	15	6



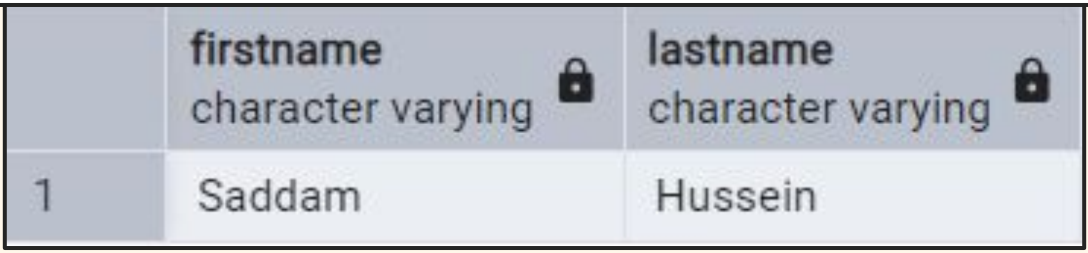
```sql
-- Display People with the highest severity red flag
select firstName, lastName
from People
where pid in (select pid
              from PeopleRedFlags
             where fid in (select fid
                           from RedFlags
                          where severity = 10));
```

	firstname character varying	lastname character varying
1	Sasha	Obama
2	Malia	Obama
3	Barack	Obama
4	Marist	Woman
5	Poopy	Pants

Reports

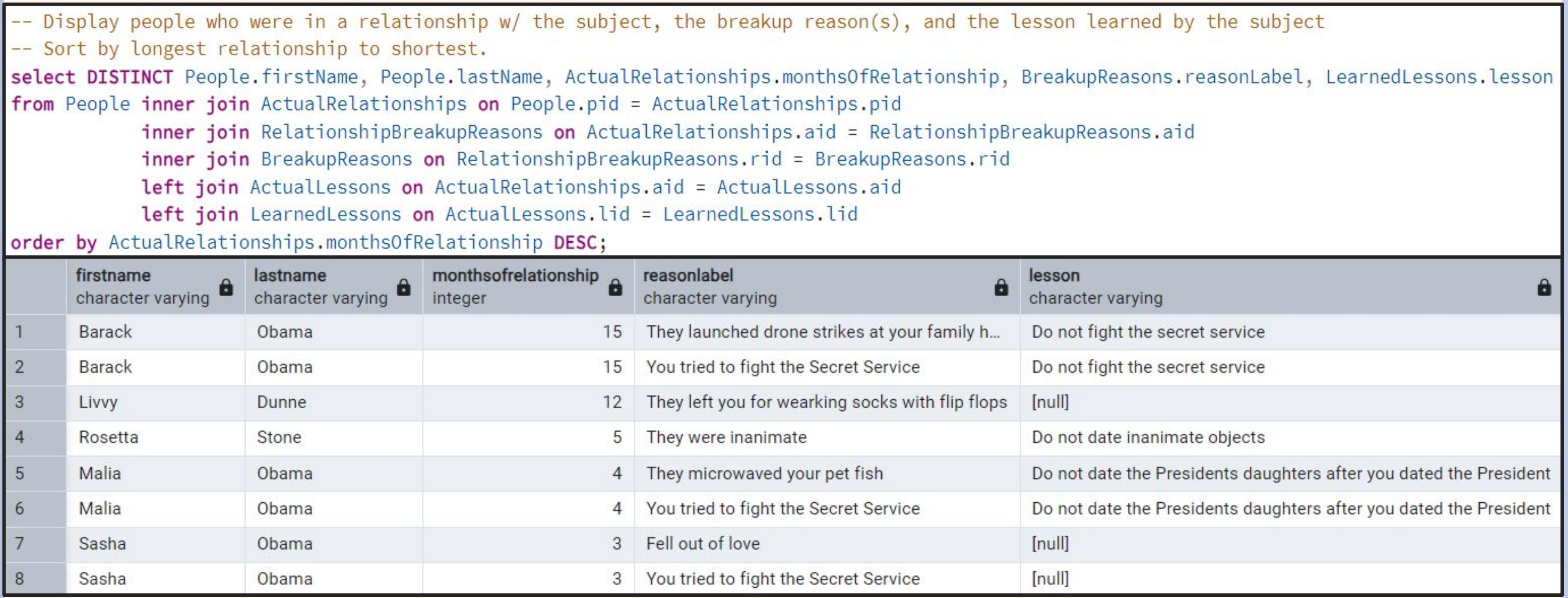


```sql
-- Display people who have taught the subject a lesson about weapons of mass destruction
select firstName, lastName
from People
where pid in (select pid
              from FailedTalkingStages
              where sid in (select sid
                            from FailedTalkingStagesLessons
                            where lid = (select lid
                                         from LearnedLessons
                                         where lesson = 'THERE WERE NO WEAPONS OF MASS DESTRUCTION')));
```



	firstname character varying	lastname character varying
1	Saddam	Hussein

Reports

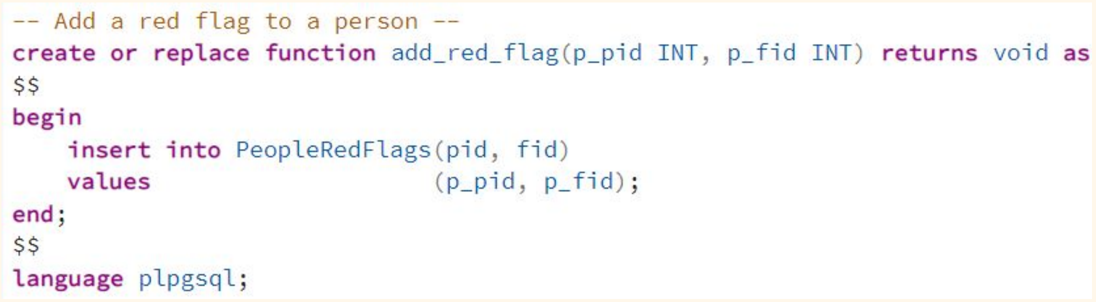


```sql
-- Display people who were in a relationship w/ the subject, the breakup reason(s), and the lesson learned by the subject
-- Sort by longest relationship to shortest.
select DISTINCT People.firstName, People.lastName, ActualRelationships.monthsOfRelationship, BreakupReasons.reasonLabel, LearnedLessons.lesson
from People inner join ActualRelationships on People.pid = ActualRelationships.pid
            inner join RelationshipBreakupReasons on ActualRelationships.aid = RelationshipBreakupReasons.aid
            inner join BreakupReasons on RelationshipBreakupReasons.rid = BreakupReasons.rid
            left join ActualLessons on ActualRelationships.aid = ActualLessons.aid
            left join LearnedLessons on ActualLessons.lid = LearnedLessons.lid
order by ActualRelationships.monthsOfRelationship DESC;
```

	firstname character varying	lastname character varying	monthsofrelationship integer	reasonlabel character varying	lesson character varying
1	Barack	Obama	15	They launched drone strikes at your family h...	Do not fight the secret service
2	Barack	Obama	15	You tried to fight the Secret Service	Do not fight the secret service
3	Livvy	Dunne	12	They left you for wearking socks with flip flops	[null]
4	Rosetta	Stone	5	They were inanimate	Do not date inanimate objects
5	Malia	Obama	4	They microwaved your pet fish	Do not date the Presidents daughters after you dated the President
6	Malia	Obama	4	You tried to fight the Secret Service	Do not date the Presidents daughters after you dated the President
7	Sasha	Obama	3	Fell out of love	[null]
8	Sasha	Obama	3	You tried to fight the Secret Service	[null]

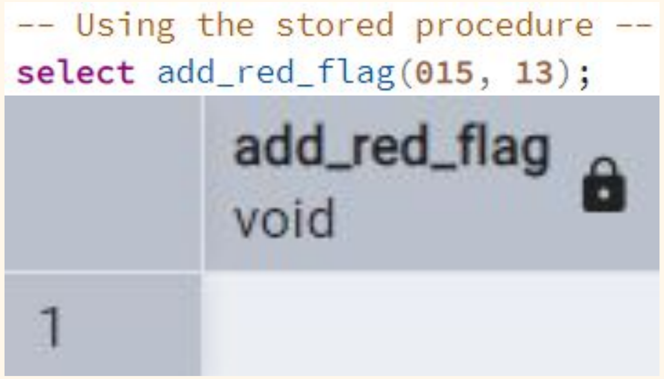
Stored Procedures

This stored procedure adds a red flag to a person.



```
-- Add a red flag to a person --
create or replace function add_red_flag(p_pid INT, p_fid INT) returns void as
$$
begin
    insert into PeopleRedFlags(pid, fid)
    values                    (p_pid, p_fid);
end;
$$
language plpgsql;
```

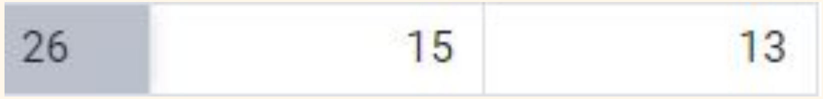
Testing the stored procedure:



```
-- Using the stored procedure --
select add_red_flag(015, 13);
```

	add_red_flag void
1	

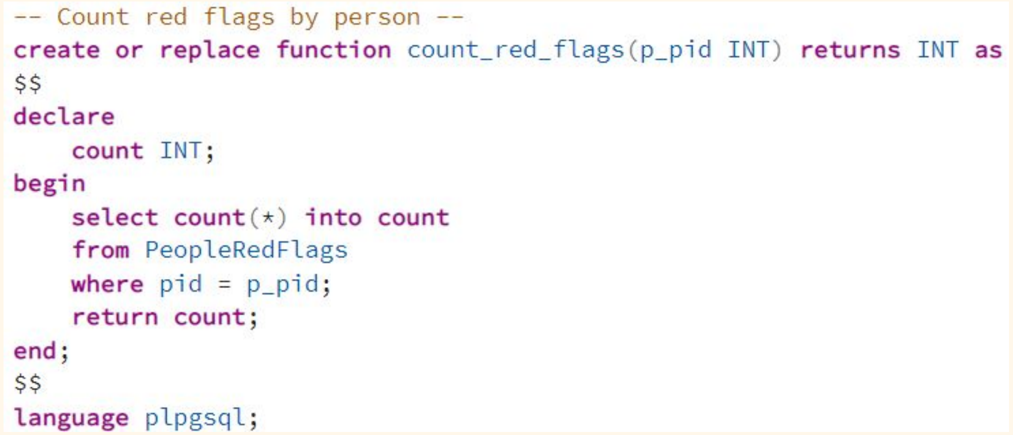
Row added to the 'PeopleRedFlags' table:



26	15	13

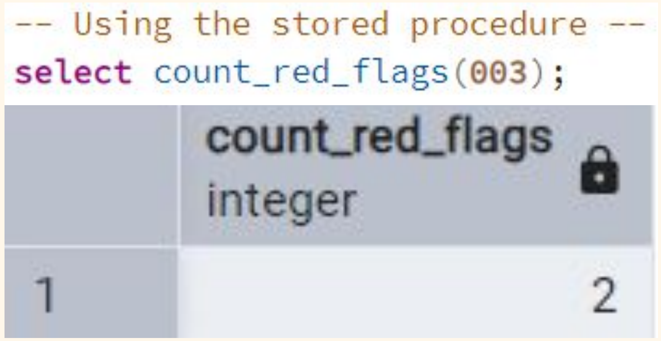
Stored Procedures

This stored procedure displays the number of red flags a person has.



```
-- Count red flags by person --
create or replace function count_red_flags(p_pid INT) returns INT as
$$
declare
    count INT;
begin
    select count(*) into count
    from PeopleRedFlags
    where pid = p_pid;
    return count;
end;
$$
language plpgsql;
```

Testing the stored procedure:

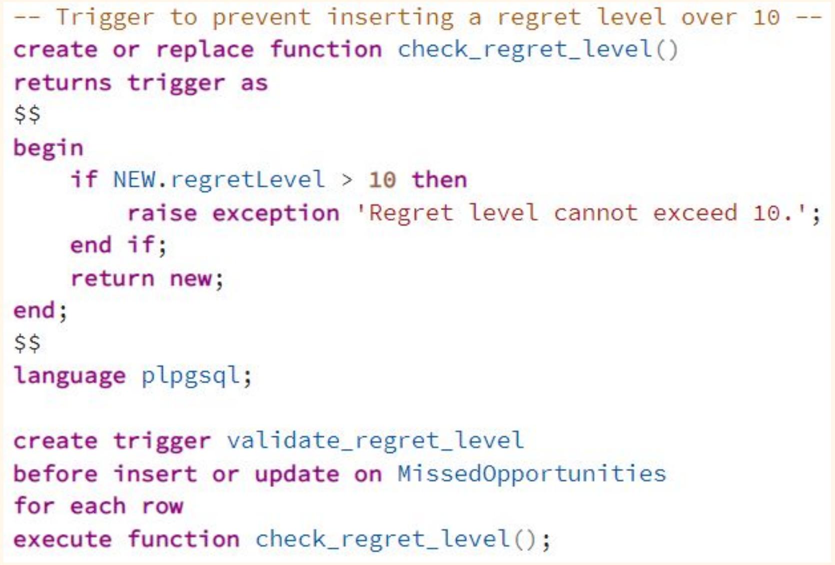


```
-- Using the stored procedure --
select count_red_flags(003);
```

	count_red_flags integer
1	2

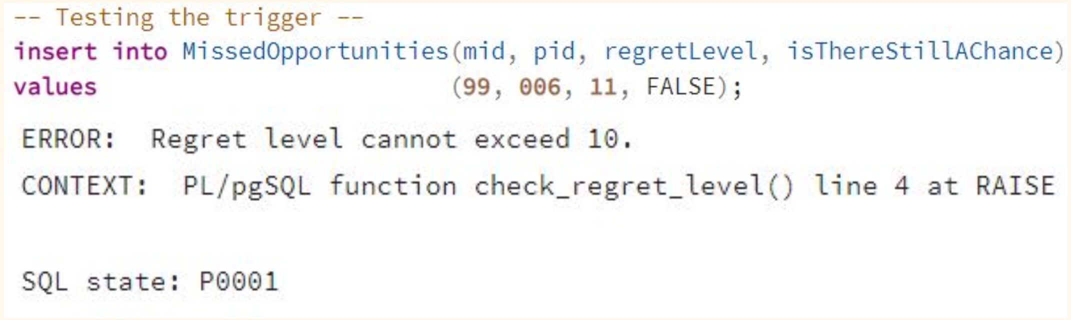
Triggers

Creating a trigger to prevent inserting a regret level over 10 in the ‘MissedOpportunities’ table



```sql
-- Trigger to prevent inserting a regret level over 10 --
create or replace function check_regret_level()
returns trigger as
$$
begin
    if NEW.regretLevel > 10 then
        raise exception 'Regret level cannot exceed 10.';
    end if;
    return new;
end;
$$
language plpgsql;

create trigger validate_regret_level
before insert or update on MissedOpportunities
for each row
execute function check_regret_level();
```

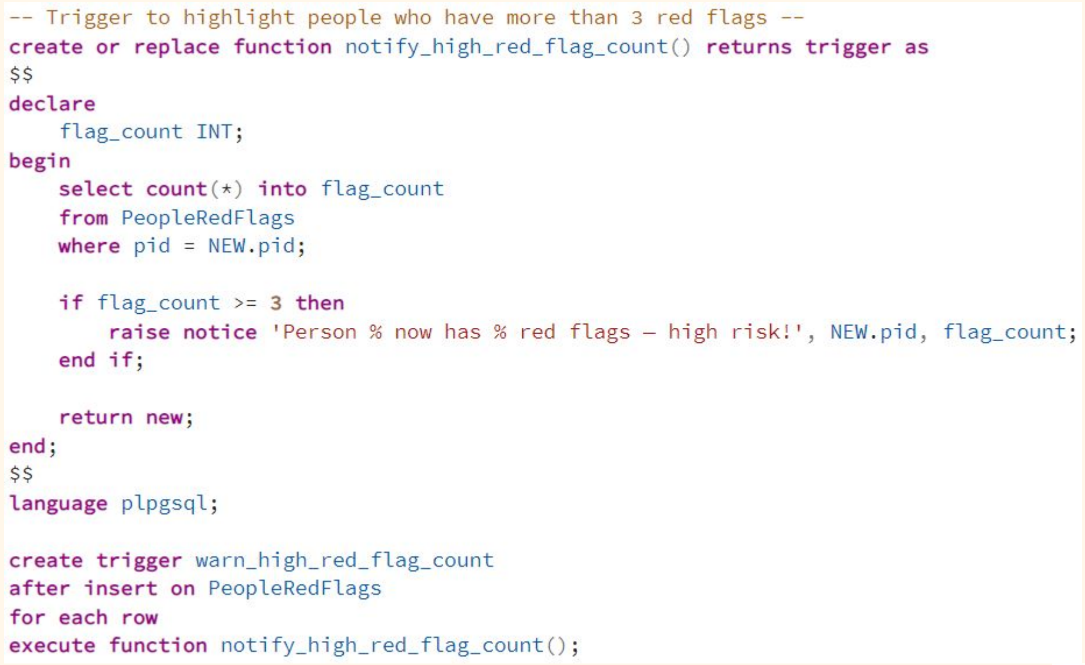
Testing the trigger to make sure it does not allow regret levels over 10 into the table



```sql
-- Testing the trigger --
insert into MissedOpportunities(mid, pid, regretLevel, isThereStillAChance)
values                         (99, 006, 11, FALSE);
```

```
ERROR:  Regret level cannot exceed 10.
CONTEXT:  PL/pgSQL function check_regret_level() line 4 at RAISE


SQL state: P0001
```

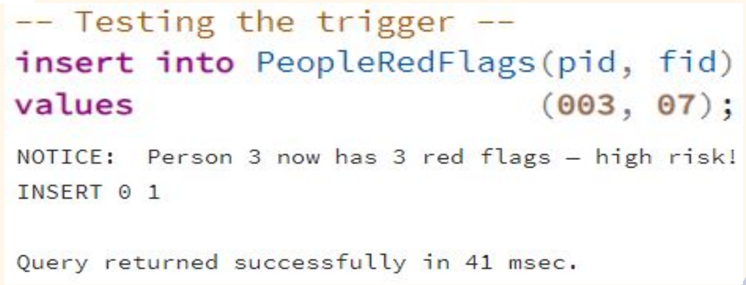
Trigger to highlight people who have more than 3 red flags as high alert



```
-- Trigger to highlight people who have more than 3 red flags --
create or replace function notify_high_red_flag_count() returns trigger as
$$
declare
    flag_count INT;
begin
    select count(*) into flag_count
    from PeopleRedFlags
    where pid = NEW.pid;

    if flag_count >= 3 then
        raise notice 'Person % now has % red flags – high risk!', NEW.pid, flag_count;
    end if;

    return new;
end;
$$
language plpgsql;

create trigger warn_high_red_flag_count
after insert on PeopleRedFlags
for each row
execute function notify_high_red_flag_count();
```

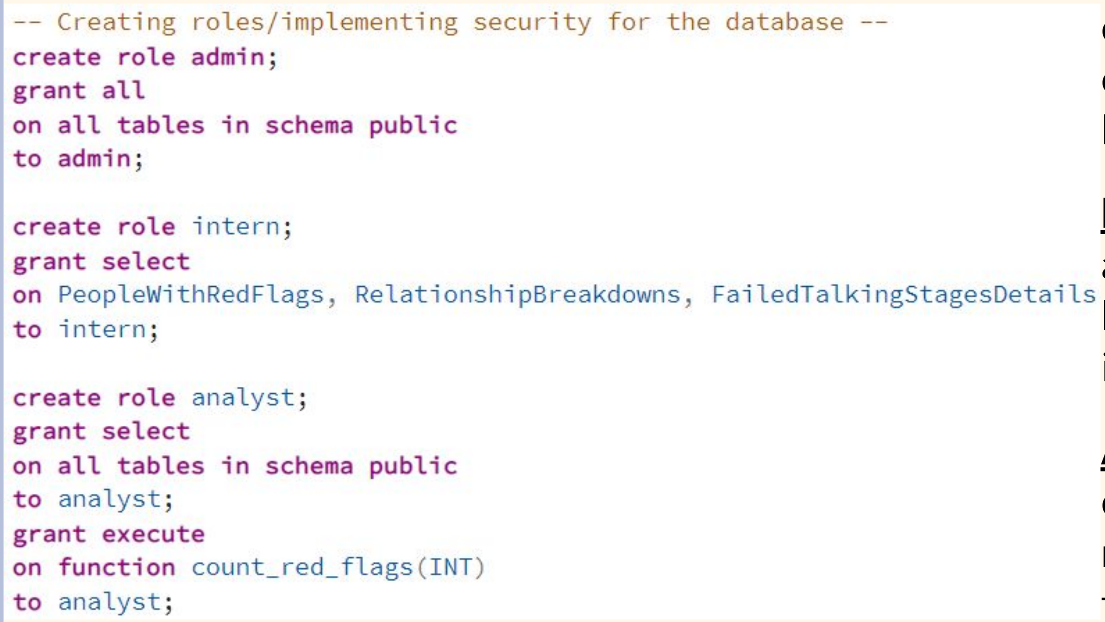
Triggers

Testing the trigger to make sure it gives a notice when a person accumulates 3 or more red flags



```
-- Testing the trigger --
insert into PeopleRedFlags(pid, fid)
values                    (003, 07);

NOTICE:  Person 3 now has 3 red flags – high risk!
INSERT 0 1

Query returned successfully in 41 msec.
```

Security



```
-- Creating roles/implementing security for the database --
create role admin;
grant all
on all tables in schema public
to admin;

create role intern;
grant select
on PeopleWithRedFlags, RelationshipBreakdowns, FailedTalkingStagesDetails
to intern;

create role analyst;
grant select
on all tables in schema public
to analyst;
grant execute
on function count_red_flags(INT)
to analyst;
```

Admin: This role grants access to all aspects of the database for developers, Database Administrators, or the owner of The Failed Relationship Review Inc.

Intern: You would not want to give full access to an intern of the company, so I implemented read only access for interns.

Analyst: Because analyst might explore relationships in raw data or run reports across multiple tables, they need to be able to read everything and have access to custom logic such as count_red_flags(INT).

Implementations Notes/Known Problems/Future Enhancements

- After designing the database and loading test data, I realized that I could have made primary keys such as pid, aid, sid, etc auto-incrementing.
- I also realized that I maybe should have set up a many-to-many relationship between each failed relationship type (ActualRelationships, MissedOpportunities, FailedTalkingStages) and the LearnedLessons table.
- Additionally, I forgot to add check constraints - which is why I created a trigger that does something similar for regretLevel in the MissedOpportunities table.
- Another future enhancement could be to create another table to to show a full timeline of someone's interactions (talking, dating, ghosted, etc) for better timeline analysis.
- Lastly, another future enhancement would be to create a table for the high risk people that I created a trigger for. It could be more useful to have this in a table than just as a notice.