USA GYMNASTICS

USAG Regionals

Database Design Proposal

Mastered by Madeline Atwood

Table of Contents

Executive Summary

This database has been created for the USAG Regional 2020 competition- specifically for region 6 because go New York! This database will work for all the 13 sections of regionals through the United States. Since every competition in the USA has the same layout since they all follow the same guidelines, this database may in fact be used for every competition as small as a local state meet to as large as the US National competition. This database is designed in PostgreSQL but is transferable to any other SQL designed database if necessary.

This document and the following information outlines a database to hold all of the data for any competition taking place in the United States that follow USAG guidelines. The design of this database is used to show the framework for the amount of data any given gym hosting a meet may need to prepare to work with and report back to USA Gymnastics. Any gym that chooses to host a gymnastics competition (not matter how large) should be using this database not only for organizational reasons but also for safety reasons- and should return the data back to the headquarters of USAG.

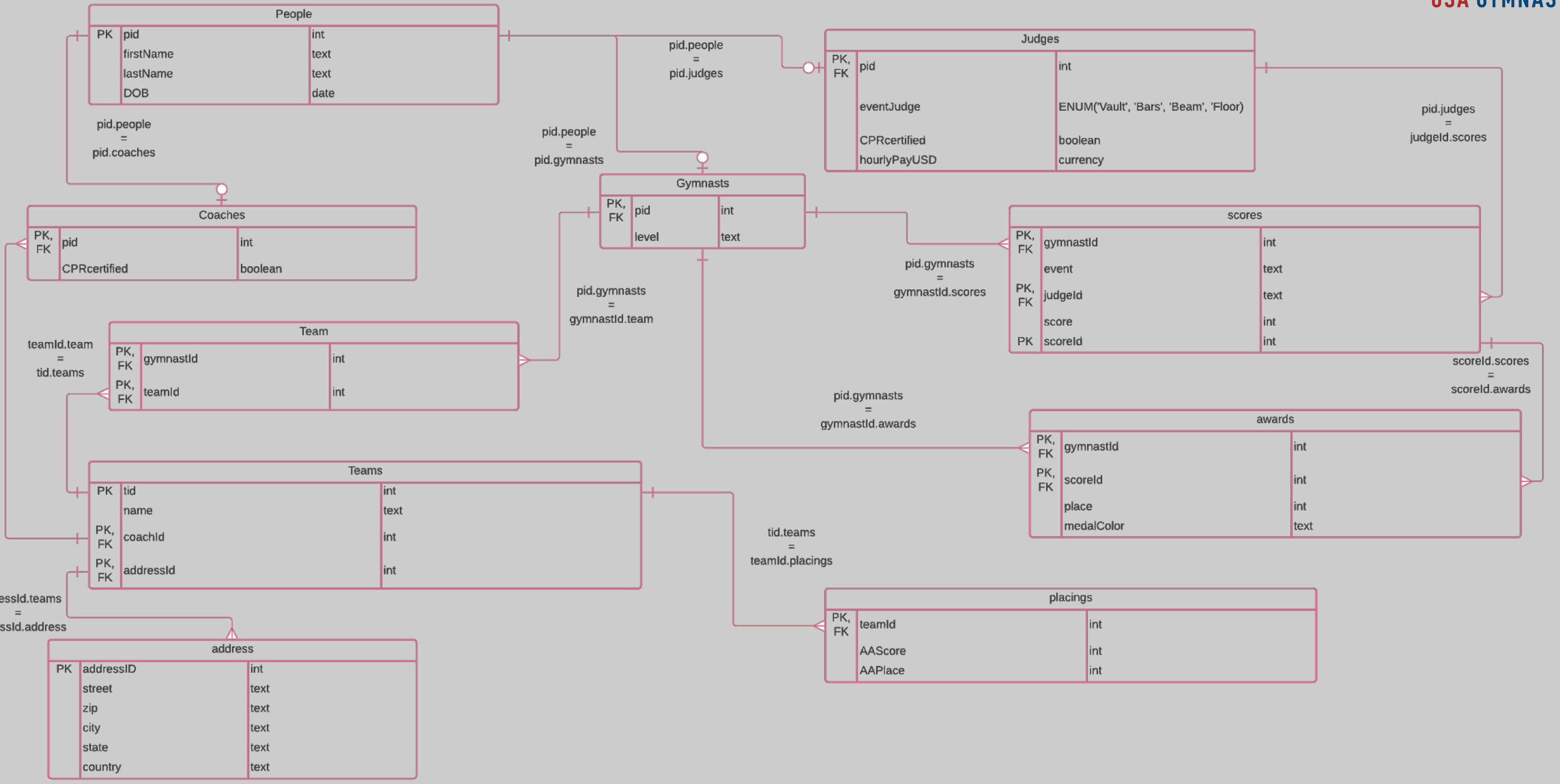
All of the data and names used in this database are fictional but after extensive research this database is accurate to any USAG gymnastics competition and may be accurately used if found necessary by any gym/ meet host.

USA GYMNASTICS

USAG Regionals

Welcome to the annual USA Gymnastics Regional Competition! Only the best gymnasts and team have qualified for this meet at their USAG state meet. Being in the top 10 of gymnasts and top 5 of teams from your home state has qualified you for regionals which in turn will hopefully qualify you for Nationals, and then Worlds! Chalk up gymnasts!

Entity Relationship Diagram for USAG Regionals


pid.people = pid.judges
pid.people = pid.coaches
pid.people = pid.gymnasts
pid.judges = judgeId.scores
pid.gymnasts = gymnastId.scores
pid.gymnasts = gymnastId.team
pid.coaches = coachId.teams
teamId.team = tid.teams
scoreId.scores = scoreId.awards
pid.gymnasts = gymnastId.awards
tid.teams = teamId.placings
addressId.teams = addressId.address

People

PK	pid	int
	firstName	text
	lastName	text
	DOB	date

Judges

PK, FK	pid	int
	eventJudge	ENUM('Vault', 'Bars', 'Beam', 'Floor)
	CPRcertified	boolean
	hourlyPayUSD	currency

Gymnasts

PK, FK	pid	int
	level	text

Coaches

PK, FK	pid	int
	CPRcertified	boolean

scores

PK, FK	gymnastId	int
	event	text
PK, FK	judgeId	text
	score	int
PK	scoreId	int

Team

PK, FK	gymnastId	int
PK, FK	teamId	int

awards

PK, FK	gymnastId	int
PK, FK	scoreId	int
	place	int
	medalColor	text

Teams

PK	tid	int
	name	text
PK, FK	coachId	int
PK, FK	addressId	int

placings

PK, FK	teamId	int
	AAScore	int
	AAPlace	int

address

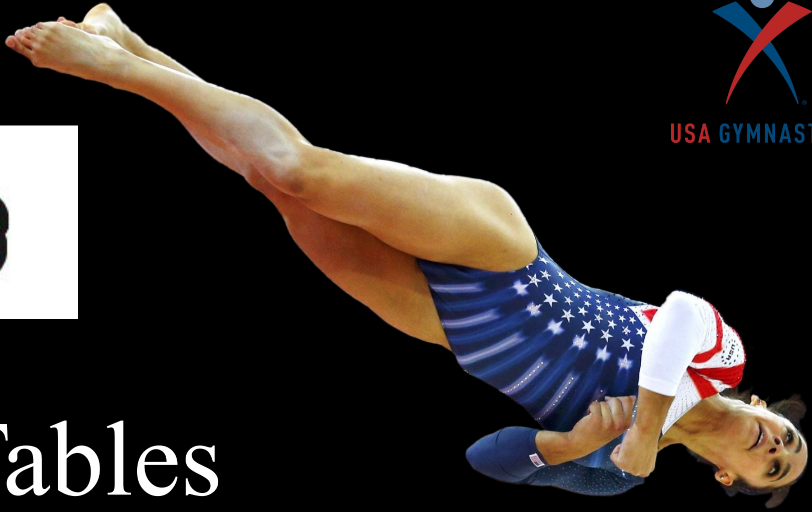
PK	addressID	int
	street	text
	zip	text
	city	text
	state	text
	country	text

USA GYMNASTICS.

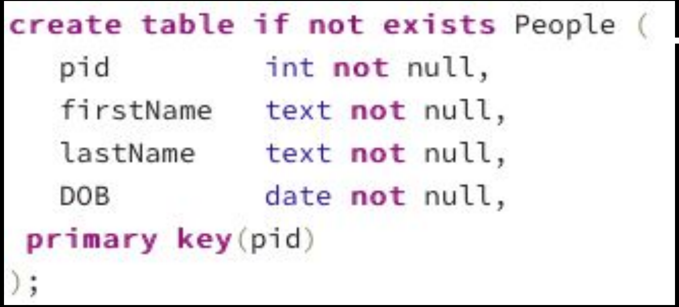
USAG Regionals

USA GYMNASTICS

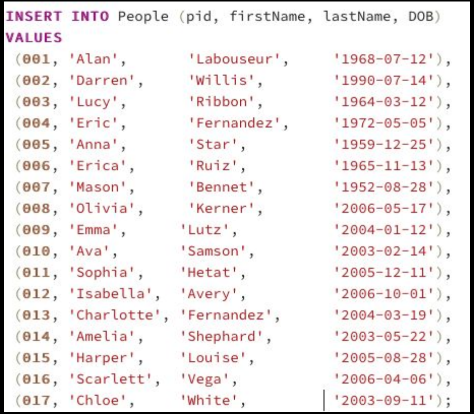
Database Tables



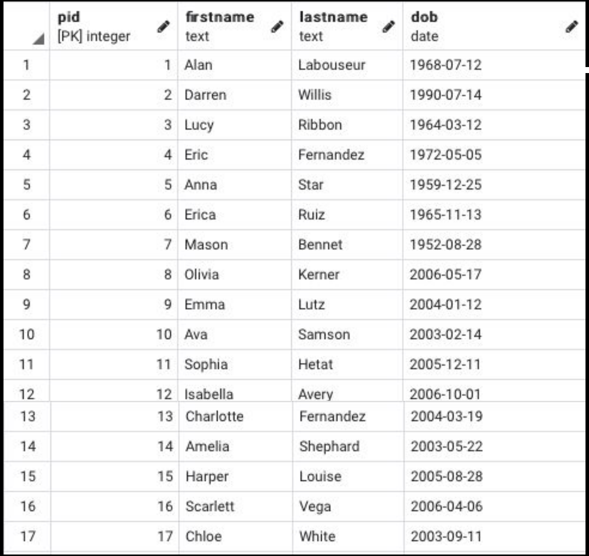
People Table



```
create table if not exists People (
   pid          int not null,
   firstName    text not null,
   lastName     text not null,
   DOB          date not null,
 primary key(pid)
);
```



```
INSERT INTO People (pid, firstName, lastName, DOB)
VALUES
 (001, 'Alan',       'Labouseur',   '1968-07-12'),
 (002, 'Darren',     'Willis',      '1990-07-14'),
 (003, 'Lucy',       'Ribbon',      '1964-03-12'),
 (004, 'Eric',       'Fernandez',   '1972-05-05'),
 (005, 'Anna',       'Star',        '1959-12-25'),
 (006, 'Erica',      'Ruiz',        '1965-11-13'),
 (007, 'Mason',      'Bennet',      '1952-08-28'),
 (008, 'Olivia',     'Kerner',      '2006-05-17'),
 (009, 'Emma',      'Lutz',         '2004-01-12'),
 (010, 'Ava',       'Samson',       '2003-02-14'),
 (011, 'Sophia',    'Hetat',        '2005-12-11'),
 (012, 'Isabella',  'Avery',        '2006-10-01'),
 (013, 'Charlotte', 'Fernandez',    '2004-03-19'),
 (014, 'Amelia',    'Shephard',     '2003-05-22'),
 (015, 'Harper',    'Louise',       '2005-08-28'),
 (016, 'Scarlett',  'Vega',         '2006-04-06'),
 (017, 'Chloe',     'White',        '2003-09-11');
```



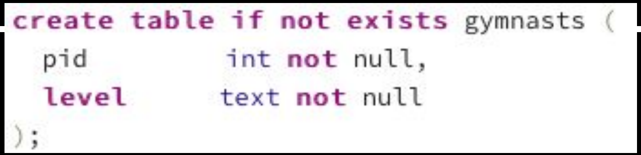
	pid [PK] integer	firstname text	lastname text	dob date
1	1	Alan	Labouseur	1968-07-12
2	2	Darren	Willis	1990-07-14
3	3	Lucy	Ribbon	1964-03-12
4	4	Eric	Fernandez	1972-05-05
5	5	Anna	Star	1959-12-25
6	6	Erica	Ruiz	1965-11-13
7	7	Mason	Bennet	1952-08-28
8	8	Olivia	Kerner	2006-05-17
9	9	Emma	Lutz	2004-01-12
10	10	Ava	Samson	2003-02-14
11	11	Sophia	Hetat	2005-12-11
12	12	Isabella	Avery	2006-10-01
13	13	Charlotte	Fernandez	2004-03-19
14	14	Amelia	Shephard	2003-05-22
15	15	Harper	Louise	2005-08-28
16	16	Scarlett	Vega	2006-04-06
17	17	Chloe	White	2003-09-11

Functional Dependencies

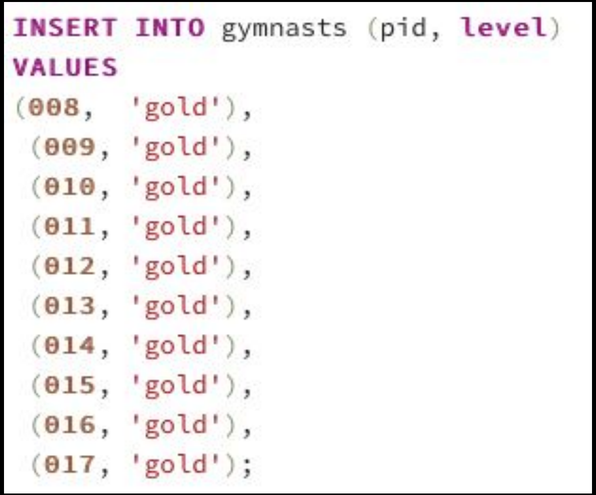
pid → firstName, lastName, DOB

Anybody who is a judge, coach or gymnast can be found in this table. This table is the basis for all of the people in the competition- all should also be found in the generalized USAG database that would originally allow them to enter this competition grounds.

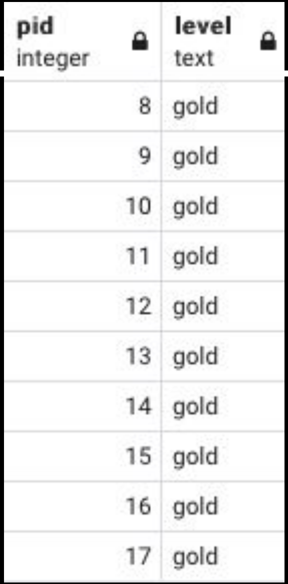
Gymnasts Table



```
create table if not exists gymnasts (
  pid          int not null,
  level        text not null
);
```



```
INSERT INTO gymnasts (pid, level)
VALUES
(008,  'gold'),
 (009, 'gold'),
 (010, 'gold'),
 (011, 'gold'),
 (012, 'gold'),
 (013, 'gold'),
 (014, 'gold'),
 (015, 'gold'),
 (016, 'gold'),
 (017, 'gold');
```



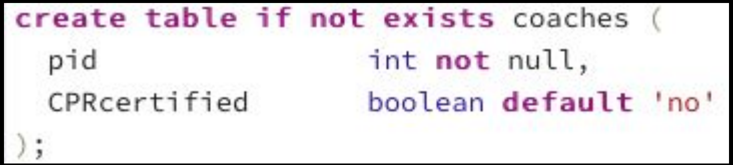
pid integer	level text
8	gold
9	gold
10	gold
11	gold
12	gold
13	gold
14	gold
15	gold
16	gold
17	gold

Functional Dependencies

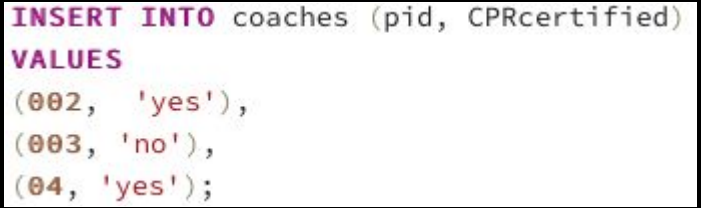
pid → level

This is the gymnasts table- which will simply tell us the level of the gymnast based off of their pid. Please not that in this table the level is all gold because this regional competition is just for gold level gymnasts- this does not need to be the case for all future data sets.

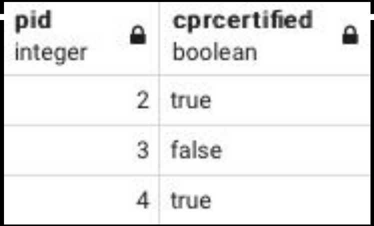
Coaches Table



```
create table if not exists coaches (
  pid                    int not null,
  CPRcertified           boolean default 'no'
);
```



```
INSERT INTO coaches (pid, CPRcertified)
VALUES
(002,  'yes'),
(003, 'no'),
(04, 'yes');
```



pid integer	cprcertified boolean
2	true
3	false
4	true

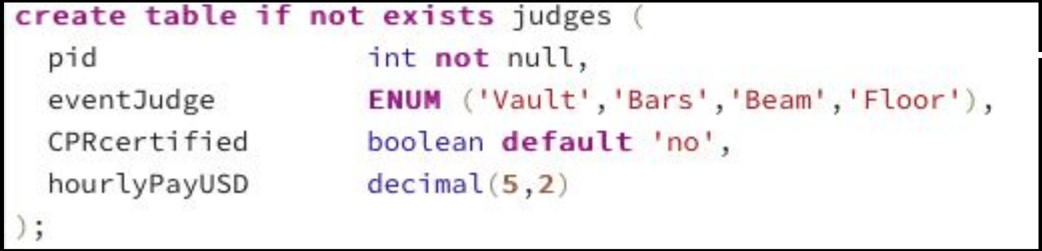
Functional Dependencies

pid → CPRcertified

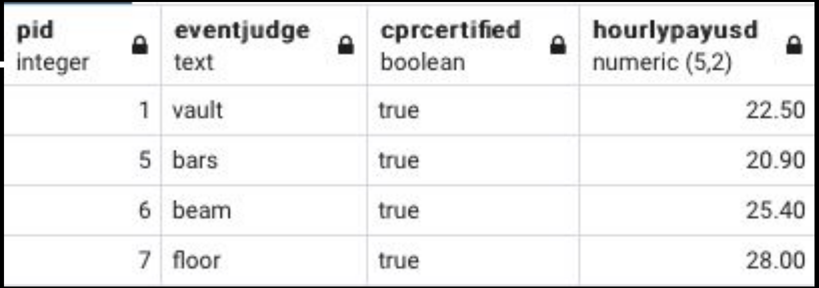
Please note that CPRcerfitied column is boolean in the system to avoid confusion that was ultimately be fatal if misinterpreted. As a result of this, the data can be printed differently if confusing to viewers.

USA GYMNASTICS

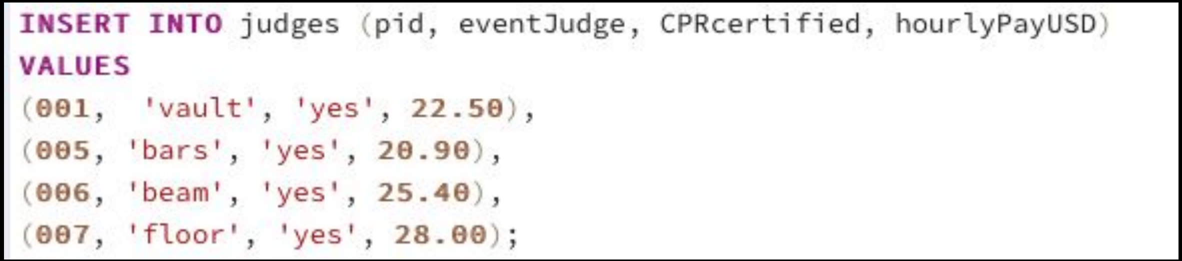
Judges Table



```
create table if not exists judges (
  pid                  int not null,
  eventJudge           ENUM ('Vault','Bars','Beam','Floor'),
  CPRcertified         boolean default 'no',
  hourlyPayUSD         decimal(5,2)
);
```



pid integer	eventjudge text	cprcertified boolean	hourlypayusd numeric (5,2)
1	vault	true	22.50
5	bars	true	20.90
6	beam	true	25.40
7	floor	true	28.00



```
INSERT INTO judges (pid, eventJudge, CPRcertified, hourlyPayUSD)
VALUES
(001,  'vault', 'yes', 22.50),
(005, 'bars', 'yes', 20.90),
(006, 'beam', 'yes', 25.40),
(007, 'floor', 'yes', 28.00);
```

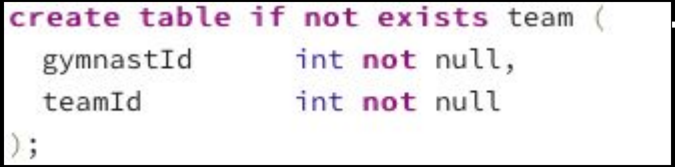
Functional Dependencies

pid → eventJudge, CPRcertified, hourlyPayUSD

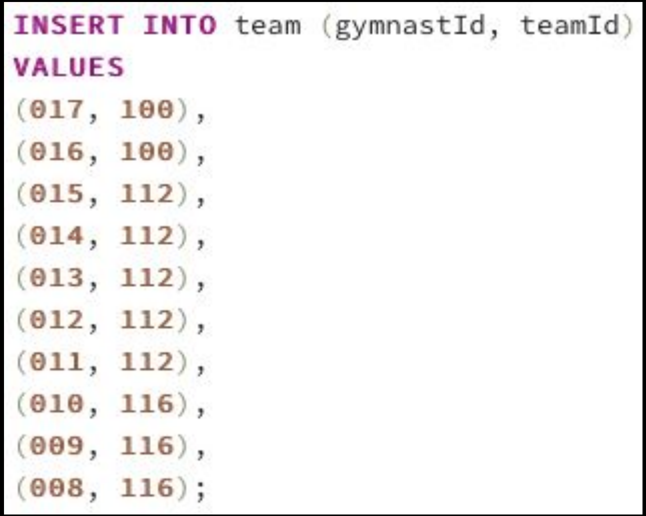
This table consists of more information specific to the judges. A judge should only be judging one event per competition to prevent biases. CPR training is part of being a judge so all should be trained. Hourly pay will be represented in US dollars.

-

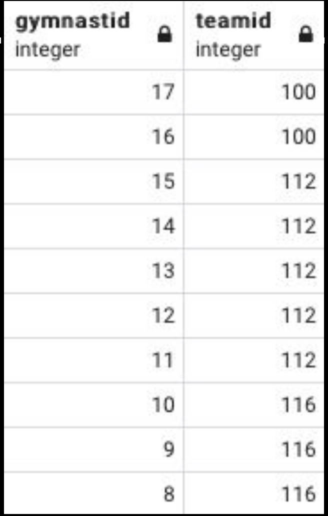
Team Table



```
create table if not exists team (
  gymnastId         int not null,
  teamId            int not null
);
```



```
INSERT INTO team (gymnastId, teamId)
VALUES
(017, 100),
(016, 100),
(015, 112),
(014, 112),
(013, 112),
(012, 112),
(011, 112),
(010, 116),
(009, 116),
(008, 116);
```



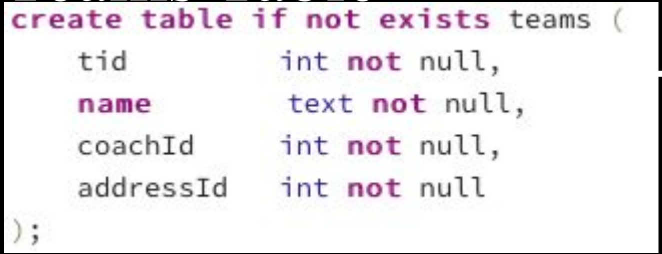
gymnastid integer	teamid integer
17	100
16	100
15	112
14	112
13	112
12	112
11	112
10	116
9	116
8	116

Functional Dependencies

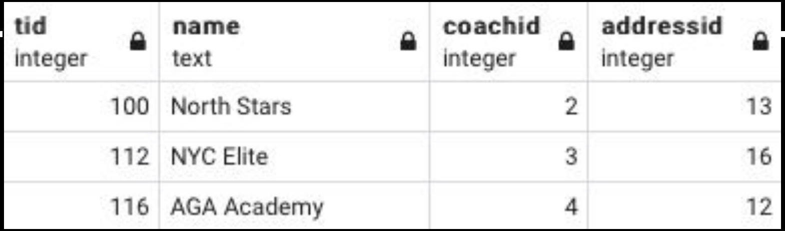
There are no functional dependencies because this is a composite key- both of these keys are foreign keys.

This table shows what gymnast is on what team.

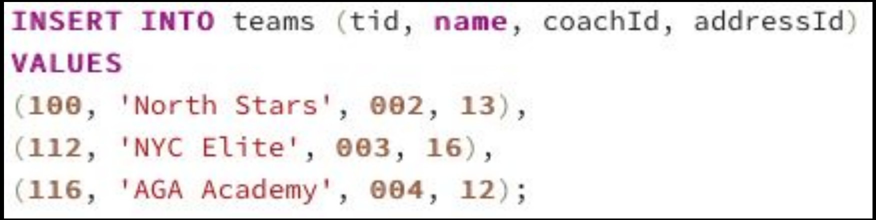
Teams Table



```
create table if not exists teams (
    tid         int not null,
    name        text not null,
    coachId     int not null,
    addressId   int not null
);
```



tid integer	name text	coachid integer	addressid integer
100	North Stars	2	13
112	NYC Elite	3	16
116	AGA Academy	4	12



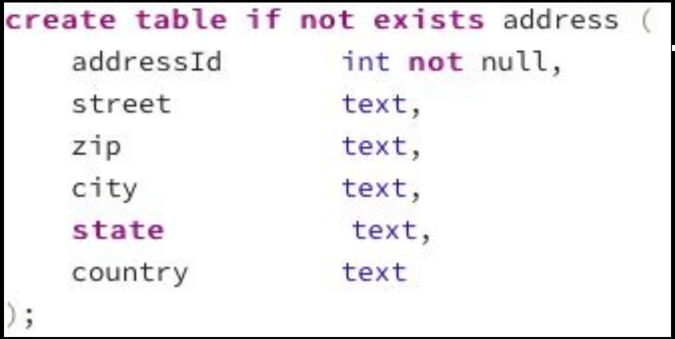
```
INSERT INTO teams (tid, name, coachId, addressId)
VALUES
(100, 'North Stars', 002, 13),
(112, 'NYC Elite', 003, 16),
(116, 'AGA Academy', 004, 12);
```

Functional Dependencies

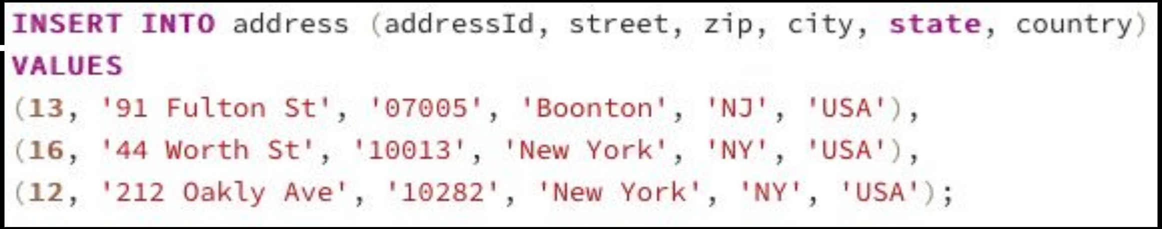
tid → name, coachId, addressId

This table gives the general information that is identified by the team ID number. There is only one coach permitted per team to avoid overcrowding on the floor as per the new covid guidelines. Gymnasts from each team can be found on the team composite table.

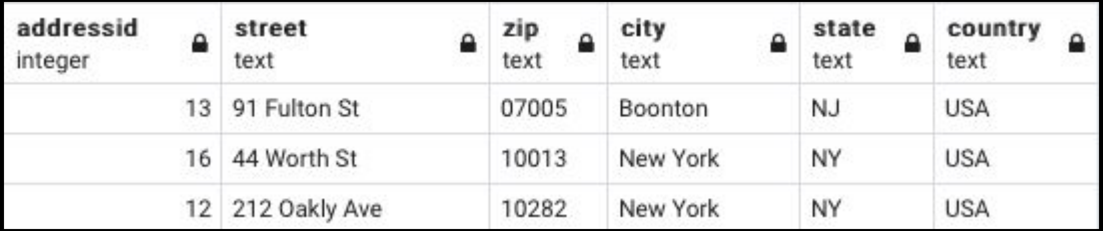
Address Table



```
create table if not exists address (
    addressId       int not null,
    street          text,
    zip             text,
    city            text,
    state            text,
    country         text
);
```



```
INSERT INTO address (addressId, street, zip, city, state, country)
VALUES
(13, '91 Fulton St', '07005', 'Boonton', 'NJ', 'USA'),
(16, '44 Worth St', '10013', 'New York', 'NY', 'USA'),
(12, '212 Oakly Ave', '10282', 'New York', 'NY', 'USA');
```



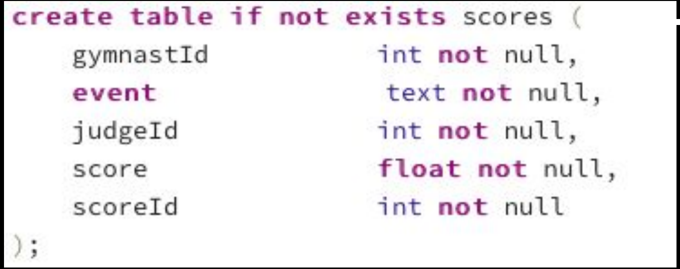
addressid integer	street text	zip text	city text	state text	country text
13	91 Fulton St	07005	Boonton	NJ	USA
16	44 Worth St	10013	New York	NY	USA
12	212 Oakly Ave	10282	New York	NY	USA

Functional Dependencies

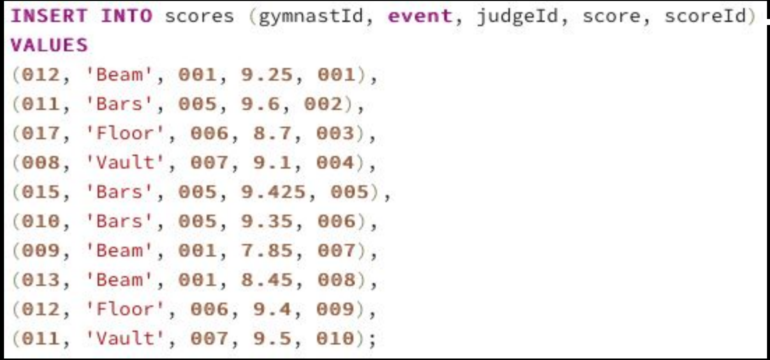
addressId→ street, zip, city, state, country

This table is for the address information for each team. I have included a country column just for future references although each gym must be found in the United States in order to compete in this competition. City and State can also be implied by zip code but I have chosen to ignore this fact for now to make the tables more user friendly. Zip code has been entered into the system as text just incase a zip has a dash in it- we want to prevent the system from accidentally treating that as a subtraction of numbers.

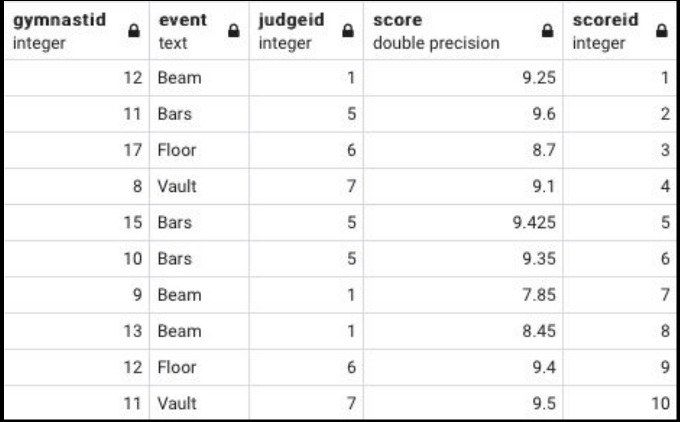
Scores Table



```
create table if not exists scores (
    gymnastId                int not null,
    event                     text not null,
    judgeId                  int not null,
    score                    float not null,
    scoreId                  int not null
);
```



```
INSERT INTO scores (gymnastId, event, judgeId, score, scoreId)
VALUES
(012, 'Beam', 001, 9.25, 001),
(011, 'Bars', 005, 9.6, 002),
(017, 'Floor', 006, 8.7, 003),
(008, 'Vault', 007, 9.1, 004),
(015, 'Bars', 005, 9.425, 005),
(010, 'Bars', 005, 9.35, 006),
(009, 'Beam', 001, 7.85, 007),
(013, 'Beam', 001, 8.45, 008),
(012, 'Floor', 006, 9.4, 009),
(011, 'Vault', 007, 9.5, 010);
```



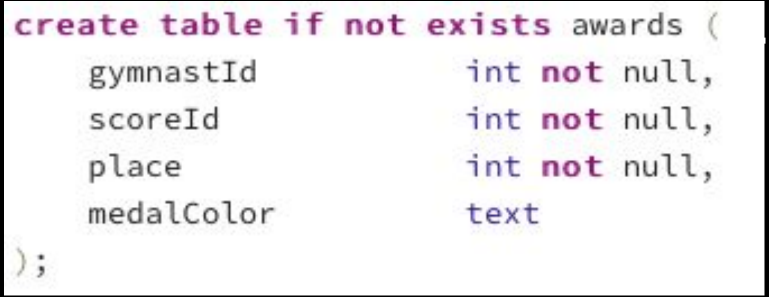
gymnastid integer	event text	judgeid integer	score double precision	scoreid integer
12	Beam	1	9.25	1
11	Bars	5	9.6	2
17	Floor	6	8.7	3
8	Vault	7	9.1	4
15	Bars	5	9.425	5
10	Bars	5	9.35	6
9	Beam	1	7.85	7
13	Beam	1	8.45	8
12	Floor	6	9.4	9
11	Vault	7	9.5	10

Functional Dependencies

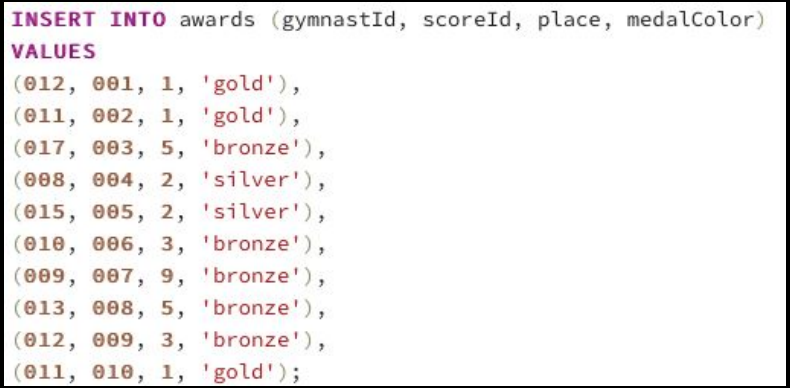
gymnastId, judgeId → score
judgeId → event
score→ scoreId

In theory this table will have the scores of every gymnast that competed on every single event. Gymnasts do not have to compete on every event but by splitting up this table from the gymnasts table as a table of all the collective data it is easier to query for the placings table.

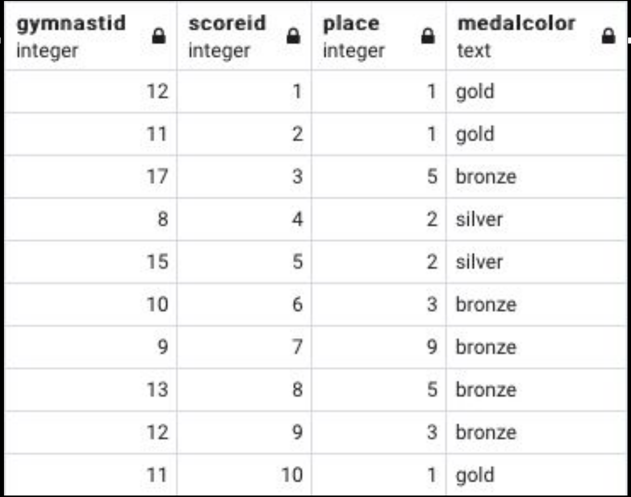
Awards Table



```
create table if not exists awards (
    gymnastId           int not null,
    scoreId             int not null,
    place               int not null,
    medalColor          text
);
```



```
INSERT INTO awards (gymnastId, scoreId, place, medalColor)
VALUES
(012, 001, 1, 'gold'),
(011, 002, 1, 'gold'),
(017, 003, 5, 'bronze'),
(008, 004, 2, 'silver'),
(015, 005, 2, 'silver'),
(010, 006, 3, 'bronze'),
(009, 007, 9, 'bronze'),
(013, 008, 5, 'bronze'),
(012, 009, 3, 'bronze'),
(011, 010, 1, 'gold');
```



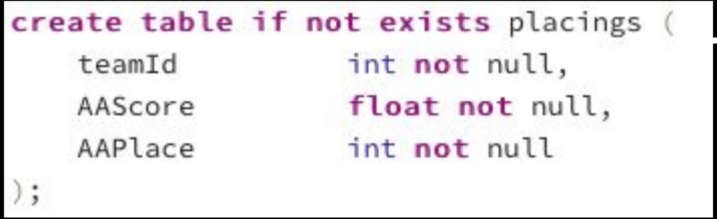
gymnastid integer	scoreid integer	place integer	medalcolor text
12	1	1	gold
11	2	1	gold
17	3	5	bronze
8	4	2	silver
15	5	2	silver
10	6	3	bronze
9	7	9	bronze
13	8	5	bronze
12	9	3	bronze
11	10	1	gold

Functional Dependencies

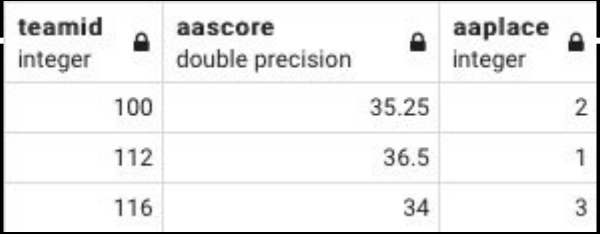
gymnastId, scoreId→ place
place→ medalColor

Please not that not every competition is required to give out a medal but they are required to have the place of the gymnast per every event listed somewhere. In this competition every gymnast will be receiving some form of a medal since this was a qualifier but this is not always the case.

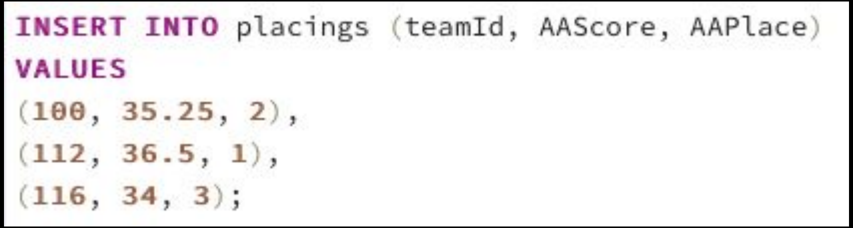
Placings Table



```
create table if not exists placings (
    teamId          int not null,
    AAScore         float not null,
    AAPlace         int not null
);
```



teamid integer	aascore double precision	aaplace integer
100	35.25	2
112	36.5	1
116	34	3



```
INSERT INTO placings (teamId, AAScore, AAPlace)
VALUES
(100, 35.25, 2),
(112, 36.5, 1),
(116, 34, 3);
```

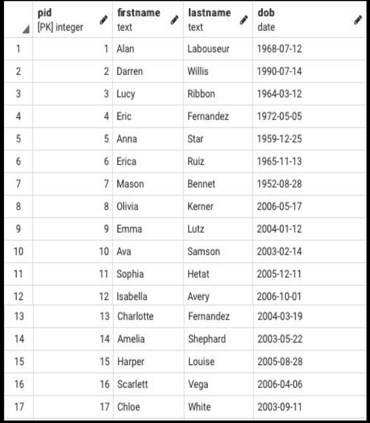
Functional Dependencies

teamId ⟶ AAScore, AAPlace

It is important to note that every competition finds the All Around team score differently. In this competition due to the lack of competitors from some teams, they averaged out each score of each gymnast on the team on every event.

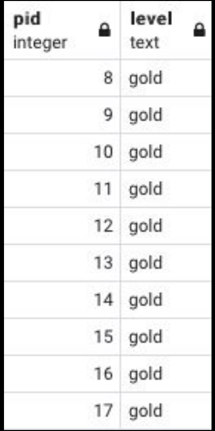
Database Snapshot

People



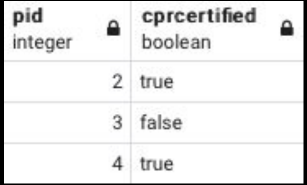
	pid [PK] integer	firstname text	lastname text	dob date
1	1	Alan	Labouseur	1968-07-12
2	2	Darren	Willis	1990-07-14
3	3	Lucy	Ribbon	1964-03-12
4	4	Eric	Fernandez	1972-05-05
5	5	Anna	Star	1959-12-25
6	6	Erica	Ruiz	1965-11-13
7	7	Mason	Bennet	1952-08-28
8	8	Olivia	Kerner	2006-05-17
9	9	Emma	Lutz	2004-01-12
10	10	Ava	Samson	2003-02-14
11	11	Sophia	Hetat	2005-12-11
12	12	Isabella	Avery	2006-10-01
13	13	Charlotte	Fernandez	2004-03-19
14	14	Amelia	Shephard	2003-05-22
15	15	Harper	Louise	2005-08-28
16	16	Scarlett	Vega	2006-04-06
17	17	Chloe	White	2003-09-11

Gymnasts



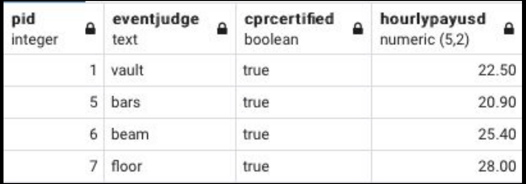
pid integer	level text
8	gold
9	gold
10	gold
11	gold
12	gold
13	gold
14	gold
15	gold
16	gold
17	gold

Coaches



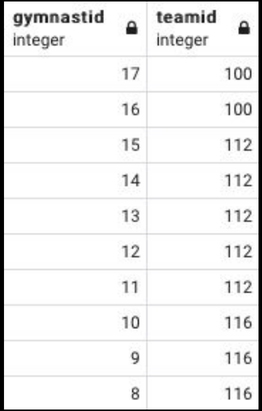
pid integer	cprcertified boolean
2	true
3	false
4	true

Judges



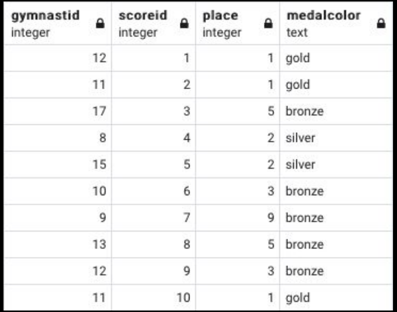
pid integer	eventjudge text	cprcertified boolean	hourlypayusd numeric (5,2)
1	vault	true	22.50
5	bars	true	20.90
6	beam	true	25.40
7	floor	true	28.00

Team



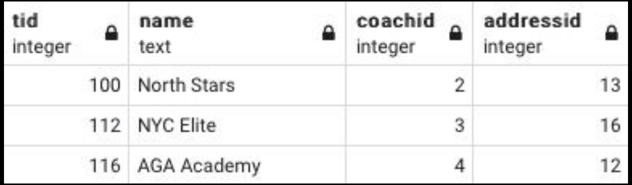
gymnastid integer	teamid integer
17	100
16	100
15	112
14	112
13	112
12	112
11	112
10	116
9	116
8	116

Awards



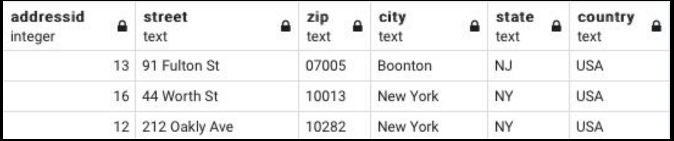
gymnastid integer	scoreid integer	place integer	medalcolor text
12	1	1	gold
11	2	1	gold
17	3	5	bronze
8	4	2	silver
15	5	2	silver
10	6	3	bronze
9	7	9	bronze
13	8	5	bronze
12	9	3	bronze
11	10	1	gold

Teams



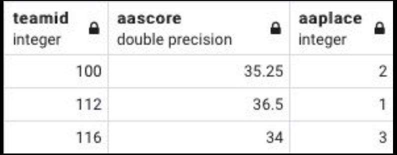
tid integer	name text	coachid integer	addressid integer
100	North Stars	2	13
112	NYC Elite	3	16
116	AGA Academy	4	12

Address



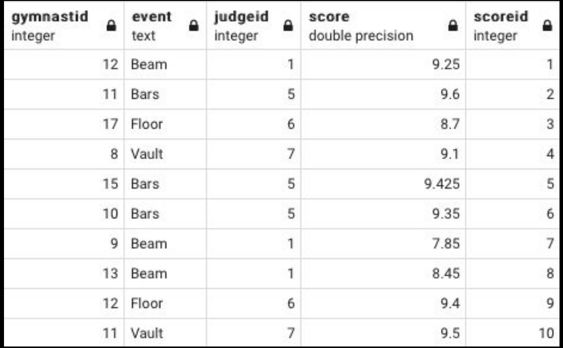
addressid integer	street text	zip text	city text	state text	country text
13	91 Fulton St	07005	Boonton	NJ	USA
16	44 Worth St	10013	New York	NY	USA
12	212 Oakly Ave	10282	New York	NY	USA

Placings



teamid integer	aascore double precision	aaplace integer
100	35.25	2
112	36.5	1
116	34	3

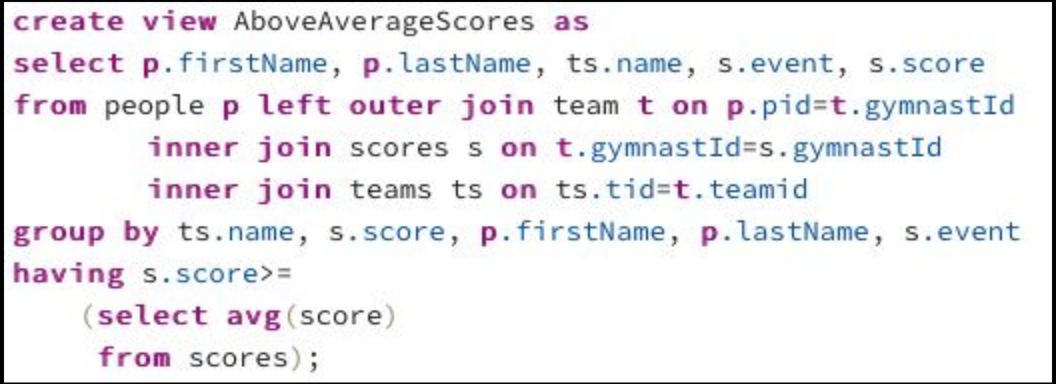
Scores



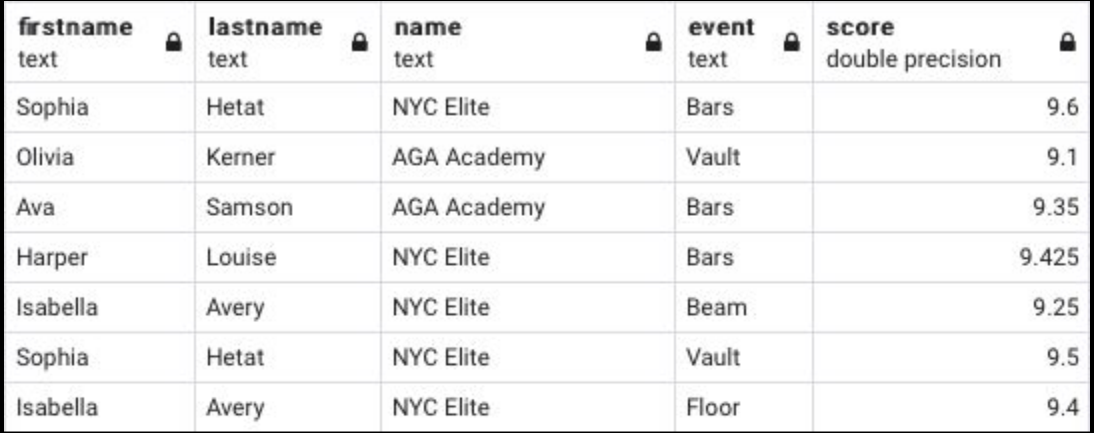
gymnastid integer	event text	judgeid integer	score double precision	scoreid integer
12	Beam	1	9.25	1
11	Bars	5	9.6	2
17	Floor	6	8.7	3
8	Vault	7	9.1	4
15	Bars	5	9.425	5
10	Bars	5	9.35	6
9	Beam	1	7.85	7
13	Beam	1	8.45	8
12	Floor	6	9.4	9
11	Vault	7	9.5	10

USA GYMNASTICS.

Views: What gym's athletes score above the average scores?

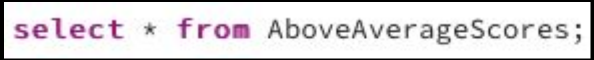


```
create view AboveAverageScores as
select p.firstName, p.lastName, ts.name, s.event, s.score
from people p left outer join team t on p.pid=t.gymnastId
        inner join scores s on t.gymnastId=s.gymnastId
        inner join teams ts on ts.tid=t.teamid
group by ts.name, s.score, p.firstName, p.lastName, s.event
having s.score>=
    (select avg(score)
     from scores);
```



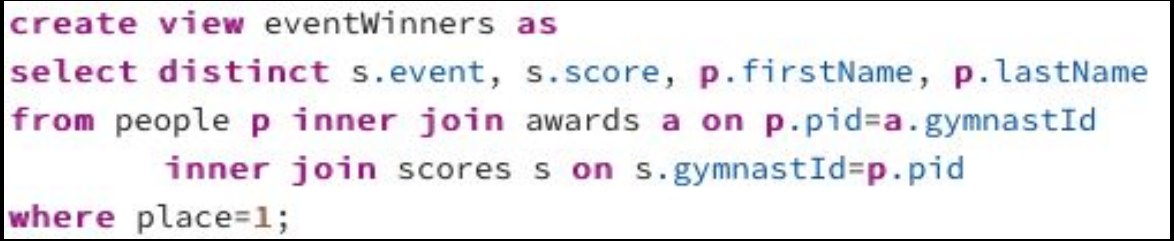
firstname text	lastname text	name text	event text	score double precision
Sophia	Hetat	NYC Elite	Bars	9.6
Olivia	Kerner	AGA Academy	Vault	9.1
Ava	Samson	AGA Academy	Bars	9.35
Harper	Louise	NYC Elite	Bars	9.425
Isabella	Avery	NYC Elite	Beam	9.25
Sophia	Hetat	NYC Elite	Vault	9.5
Isabella	Avery	NYC Elite	Floor	9.4

This query was created in order to help coaches see where their team is doing in the competition and how they did overall. To find what gymnast is scoring above the average scores based off of the inputted judge scores from the competition, and to find what gym they are from, simply use this query which is now been created as a view to add simplicity for the users.



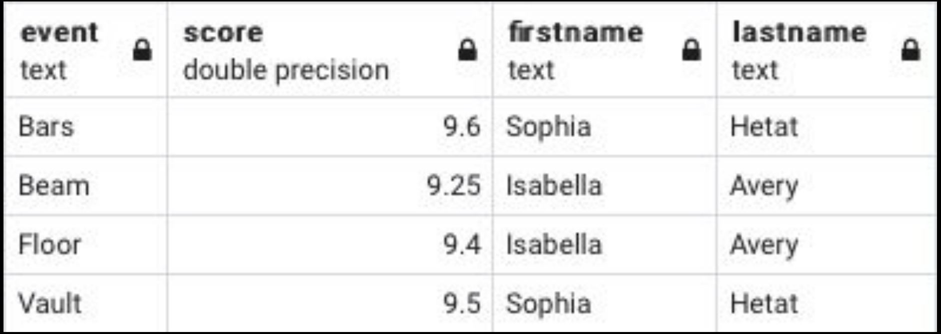
```
select * from AboveAverageScores;
```

Views: Who won what event?

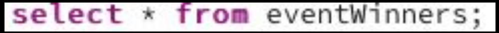


```sql
create view eventWinners as
select distinct s.event, s.score, p.firstName, p.lastName
from people p inner join awards a on p.pid=a.gymnastId
        inner join scores s on s.gymnastId=p.pid
where place=1;
```

This query was created to display the event winners, their score and their first and last name. This can be helpful for special recognitions for event score records or for overall placings and medalings.



event text	score double precision	firstname text	lastname text
Bars	9.6	Sophia	Hetat
Beam	9.25	Isabella	Avery
Floor	9.4	Isabella	Avery
Vault	9.5	Sophia	Hetat



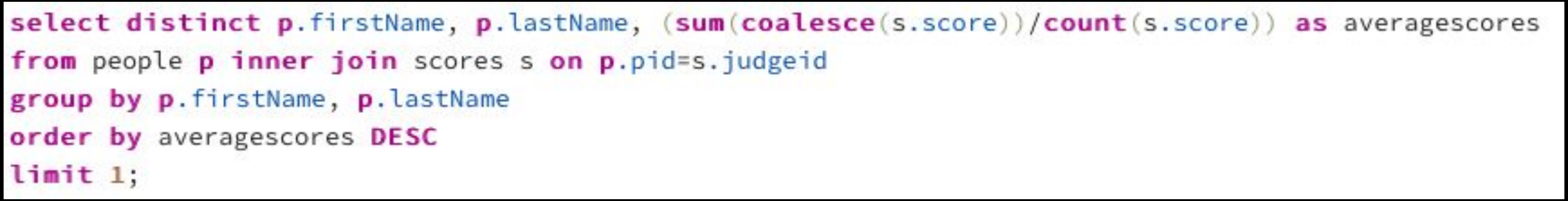
```sql
select * from eventWinners;
```

USA GYMNASTICS

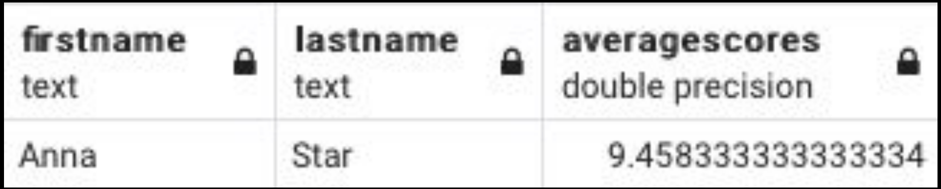
USAG Regionals

Common Reports

Common Reports: What judge gives out the highest scores?



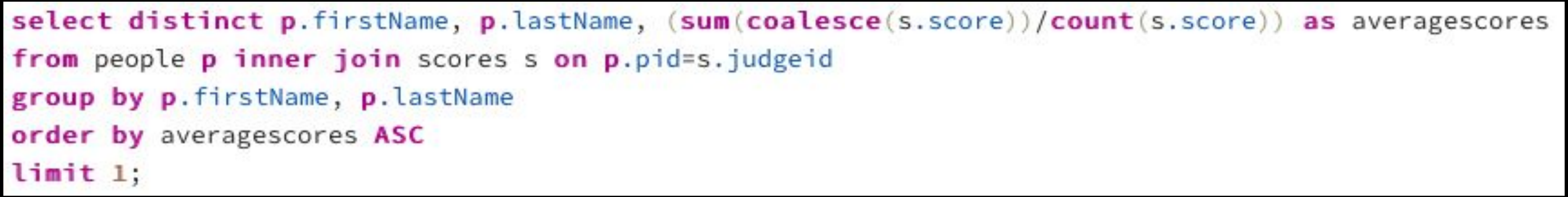
```
select distinct p.firstName, p.lastName, (sum(coalesce(s.score))/count(s.score)) as averagescores
from people p inner join scores s on p.pid=s.judgeid
group by p.firstName, p.lastName
order by averagescores DESC
limit 1;
```

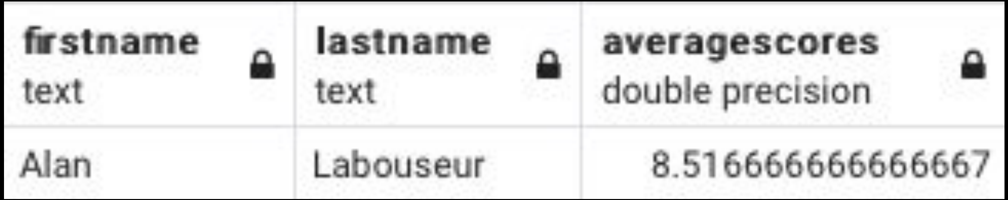


firstname text	lastname text	averagescores double precision
Anna	Star	9.45833333333334

This query was created to help the USAG administration to ensure all judges are properly adhering to the guidelines for judging and make sure all judges are following the same standards. If a judge is consistently giving extremely high scores then there may be an error in the judge education which may invalidate gymnast’s scores in the future. The goal is to keep the judging equivalent.

Common Reports: What judge gives out the lowest scores?

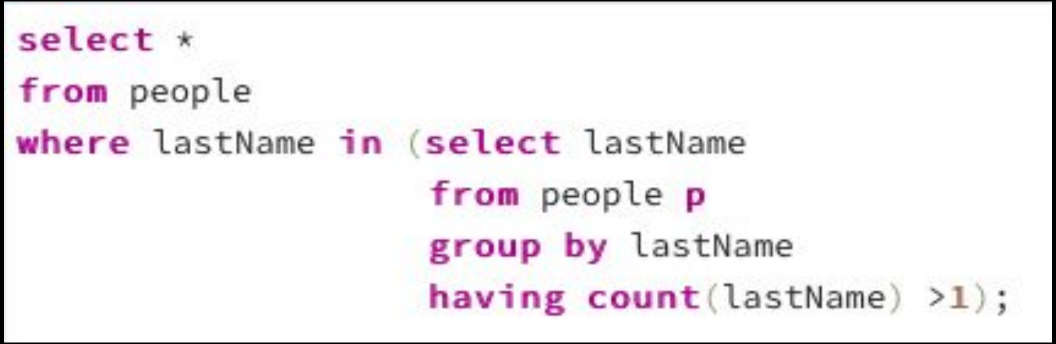

```
select distinct p.firstName, p.lastName, (sum(coalesce(s.score))/count(s.score)) as averagescores
from people p inner join scores s on p.pid=s.judgeid
group by p.firstName, p.lastName
order by averagescores ASC
limit 1;
```



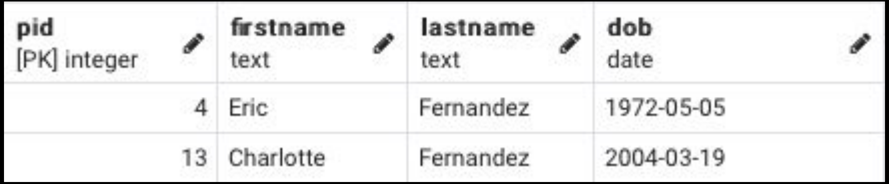
firstname text	lastname text	averagescores double precision
Alan	Labouseur	8.51666666666667

This query was created to fulfill the same purpose as the last- to help the USAG administration to ensure all judges are properly adhering to the guidelines for judging and make sure all judges are following the same standards. If a judge is consistently giving extremely low scores then there may be an error in the judge education which may invalidate gymnast's scores in the future. The goal is to keep the judging equivalent.

Common Reports: Is anybody related?



```
select *
from people
where lastName in (select lastName
                   from people p
                   group by lastName
                   having count(lastName) >1);
```



pid [PK] integer	firstname text	lastname text	dob date
4	Eric	Fernandez	1972-05-05
13	Charlotte	Fernandez	2004-03-19

This query will find if any gymnasts, coaches or more importantly judges are related to any competitors in the competition. If two gymnasts are related that is not a problem- nor is it if a gymnast and coach are related but if a judge has any family relations to any gymnasts or coaches then this needs to be evaluated in order to prevent potential bias. From the sample results we can make the assumption that Eric Fernandez and Charlotte Fernandez are related but we cannot be sure unless we check the larger USAG database to verify this claim.

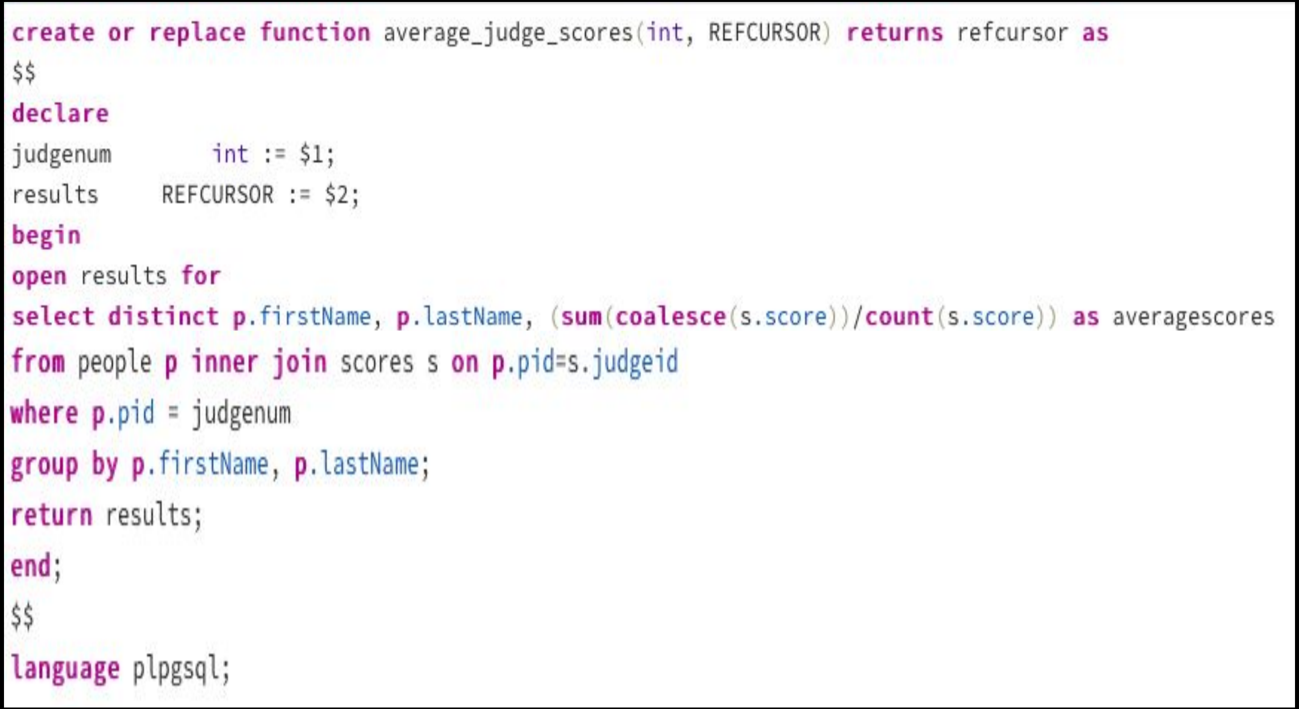
USAG Regionals

USA GYMNASTICS

Stored Procedures

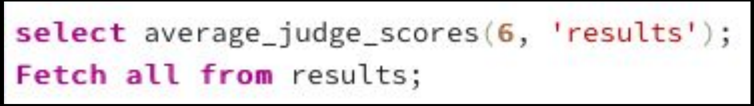
Stored Procedure: Find Judge Average Score

USA GYMNASTICS

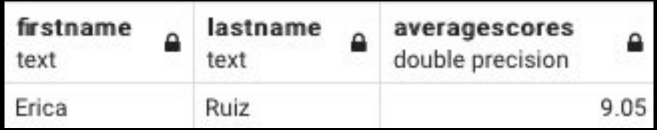


```
create or replace function average_judge_scores(int, REFCURSOR) returns refcursor as
$$
declare
judgenum        int := $1;
results     REFCURSOR := $2;
begin
open results for
select distinct p.firstName, p.lastName, (sum(coalesce(s.score))/count(s.score)) as averagescores
from people p inner join scores s on p.pid=s.judgeid
where p.pid = judgenum
group by p.firstName, p.lastName;
return results;
end;
$$
language plpgsql;
```

The average_judge_scores stored procedure can be used to look up the average of all the scores given out by each judge at the competition by passing through the judge's pid.

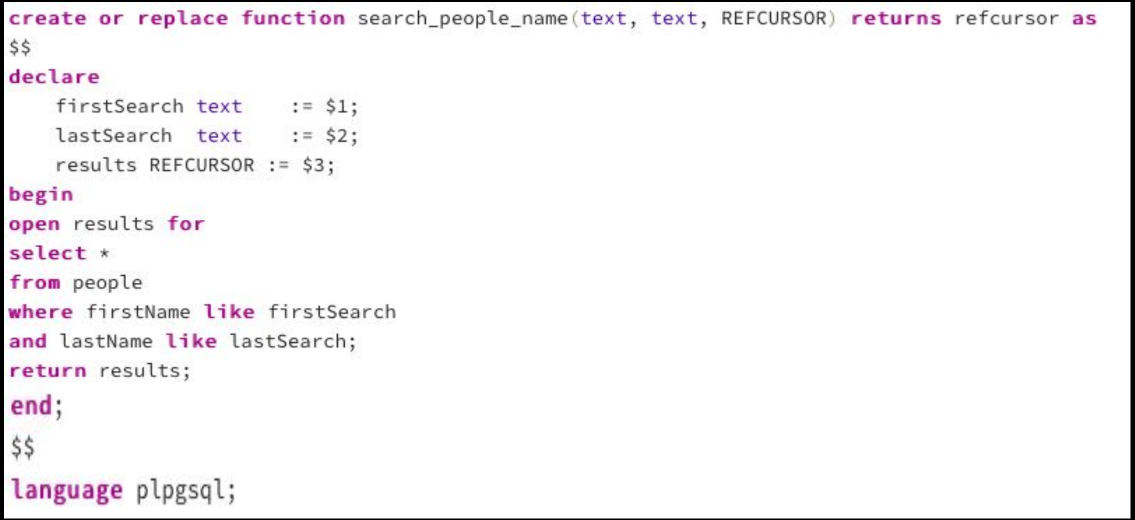


```
select average_judge_scores(6, 'results');
Fetch all from results;
```



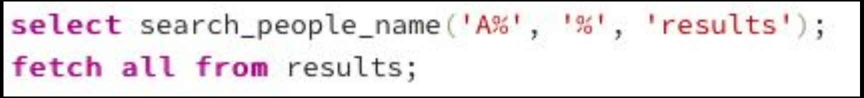
firstname text	lastname text	averagescores double precision
Erica	Ruiz	9.05

Stored Procedure: Find Name

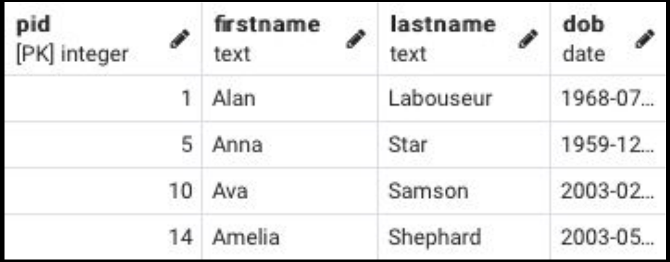


```
create or replace function search_people_name(text, text, REFCURSOR) returns refcursor as
$$
declare
    firstSearch text    := $1;
    lastSearch  text    := $2;
    results REFCURSOR := $3;
begin
open results for
select *
from people
where firstName like firstSearch
and lastName like lastSearch;
return results;
end;
$$
language plpgsql;
```

The find_people_name stored procedure can be used to look up the the name of a gymnast, judge or coach in the system if you may only know their first or last or just a few letter from each part of their name. This especially comes in handy when engraving trophies and you want to ensure all names are spelled correctly.



```
select search_people_name('A%', '%', 'results');
fetch all from results;
```



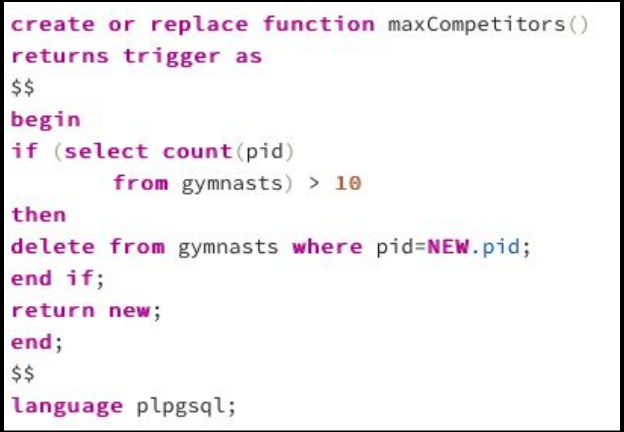
pid [PK] integer	firstname text	lastname text	dob date
1	Alan	Labouseur	1968-07...
5	Anna	Star	1959-12...
10	Ava	Samson	2003-02...
14	Amelia	Shephard	2003-05...

USAG Regionals

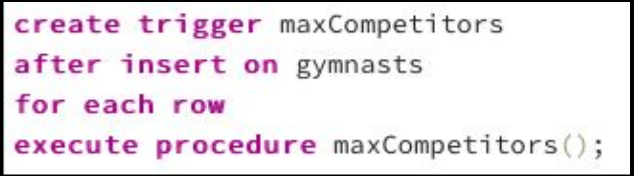
USA GYMNASTICS

Trigger

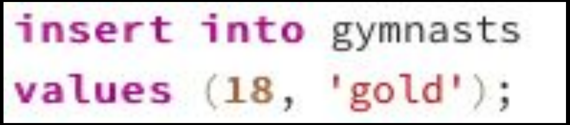
Trigger: Limit to number of competitors



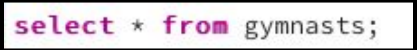
```
create or replace function maxCompetitors()
returns trigger as
$$
begin
if (select count(pid)
       from gymnasts) > 10
then
delete from gymnasts where pid=NEW.pid;
end if;
return new;
end;
$$
language plpgsql;
```



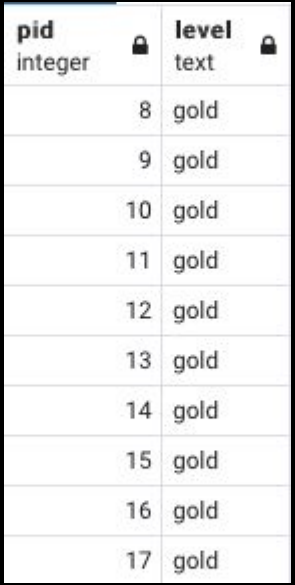
```
create trigger maxCompetitors
after insert on gymnasts
for each row
execute procedure maxCompetitors();
```



```
insert into gymnasts
values (18, 'gold');
```



```
select * from gymnasts;
```



pid integer	level text
8	gold
9	gold
10	gold
11	gold
12	gold
13	gold
14	gold
15	gold
16	gold
17	gold

This trigger has been completed to limit the number of competitors in this competition. In this sample dataset I put a limit on the number of competitors to 10 since we are dealing with limited data- as we can see we are unable to add any more athletes to the gymnasts table because this database has already reached capacity.

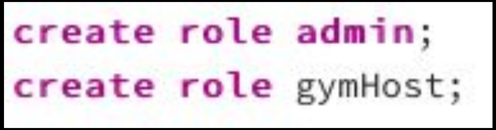
USAG Regionals

USA GYMNASTICS

Security

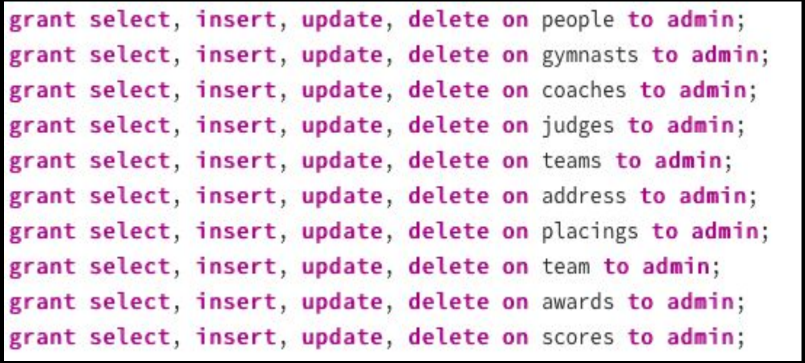
Security-

There should only be two primary users of this database: the USAG headquarters administrators and the gym/company hosting the competition.



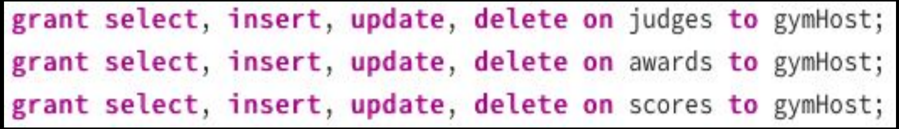
```
create role admin;
create role gymHost;
```

ADMIN: has access to the entirety of the database since all information must be reported immediately back to USAG headquarters and their database to ensure validity of competition.



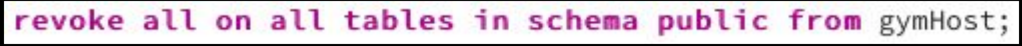
```
grant select, insert, update, delete on people to admin;
grant select, insert, update, delete on gymnasts to admin;
grant select, insert, update, delete on coaches to admin;
grant select, insert, update, delete on judges to admin;
grant select, insert, update, delete on teams to admin;
grant select, insert, update, delete on address to admin;
grant select, insert, update, delete on placings to admin;
grant select, insert, update, delete on team to admin;
grant select, insert, update, delete on awards to admin;
grant select, insert, update, delete on scores to admin;
```

GYMHOST: only has access to judges, awards and scores. The gym has access to judges since they negotiate pay and awards and scores are also done through the competition since each piece of data is constructed on a case by case basis of the competition.



```
grant select, insert, update, delete on judges to gymHost;
grant select, insert, update, delete on awards to gymHost;
grant select, insert, update, delete on scores to gymHost;
```

If there is a scandal between the gymHost that highered the judges and the USAG guidelines, immediately the admin will:



```
revoke all on all tables in schema public from gymHost;
```

USAG Regionals

Implementations, Known Problems and Future Enhancements

Implementation Notes

It is important to note that the USAG administration system has an artificial key created for all gymnasts that are entered into the system but I choose to give each person their own artificial key for this competition because it is a possibility that coaches or judges may have mix ups in the USAG database system because we cannot be sure if they previously competed in competitions or have different badge numbers that allow them to access the floor as judges or coaches.

It is also important to note that this is for women's gymnastics competitions. The database for a men's meet would be slightly different due to there being a total of 6 different events and the scoring is slightly different. Either was this database can be easily implemented to adapt to a men's competition if necessary- please do not hesitate to reach out if need be.

Known Problems

- It is important to note that due to newer covid guidelines in order to maintain social distancing, there will likely be a trigger limit on the people table.
- In the future there will likely be a column in the people table with a boolean data type asking if person has received the covid vaccine- if not they may be put to the bottom of the people list and possibly be excluded from the competition or required to wear a mask based off of the ratio of vaccinated to unvaccinated.
- It is more than reasonable to assume that in the USAG database they have an artificial key for all of the gymnasts but I have created a separate one just for this database's purposes in case they do not. If needed, the people pid (or gymnast ID) can easily be changed to what is already entered in the USAG database to avoid confusion.
- I realize having a table called "team" and "teams" may be very confusing- since one table was a composite it just made sense in my head.

Future Enhancements

- In the future it may be important to note that if a competition is larger there will be different times as a result of covid capacity so the awards table for team AA awards may run in different databases. Sometimes competitions will have different age groups compete at different times so a gym can appear in multiple sessions.
- This can be implemented and advanced to work for a worlds gymnastics national competition which will have gymnasts from all over the world.
- Implementing an award that is money instead of a medal may be interesting- especially if it is based off of the number of speculators at the meet.
- Find a way to ensure that the judges have a valid ID and have been verified in the USAG judging system and that they have completed all of the most recent trainings.

USAG Regionals

USA GYMNASTICS

Thank you for your time.