ANLT-242 RELATIONAL DATABASES

COURSE PROJECT:

BIG TEETH REALITY TV SHOW DATABASE

by

TEAM STU

|  |  |  |
| --- | --- | --- |
| Matthew Swogger | Ali Taheri | Arda Ugur |
| 989295197 | 989294669 | 989289609 |
| m\_swogger@u.pacific.edu | s\_taheritari@u.pacific.edu | a\_ugur1@pacific.edu |
|  |  |  |
|  |  |  |
| School of Engineering and Computer Science, Ms. Sci. Data Science | | |
| University of the Pacific | | |
|  |  |  |
|  | December 2017 |  |

*This page intentionally left blank.*

Table of Contents

[Business Case 4](#_Toc500701898)

[Contestant Details 4](#_Toc500701899)

[Episodes and Events 6](#_Toc500701900)

[Voting 6](#_Toc500701901)

[Required Business Solution 8](#_Toc500701902)

[Proposed Business Solutions 9](#_Toc500701903)

[Normalized Lost of Tables 9](#_Toc500701904)

[Database Entity Relationship Diagram 10](#_Toc500701905)

[Database Tables 11](#_Toc500701906)

[Sample Data 15](#_Toc500701907)

[Requested Database Queries 16](#_Toc500701908)

[Technical Notes on Database Design 24](#_Toc500701909)

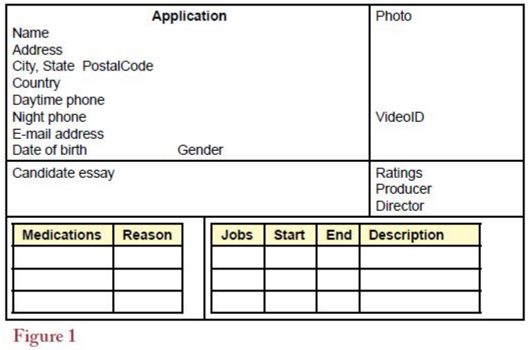
[Database Team: Team STU 24](#_Toc500701910)

[Database Tools 25](#_Toc500701911)

# Business Case

A new reality show is entering production. A cross between Animal Planet and Fear Factor, it focuses on conflicts between humans and animals with big teeth (crocodiles, tigers, lions, and so on). Contestants never know if they will be chased by the animals or if they will be eating them. The insurance costs will be huge, but the producers think there is a big audience. Producers always think that, but at least they are willing to pay money to find out. The next step is to find people crazy enough to sign the waivers and participate in the show.

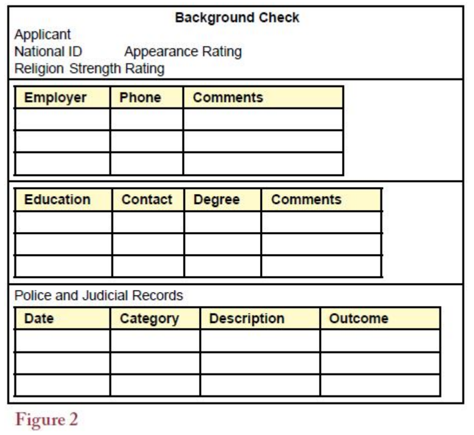
## Contestant Details



**Figure-1** shows the basic contestant application form. The producers are adding a twist and recruiting worldwide. The goal is to build more suspense by adding communication problems among contestants and perhaps to foster some nationalistic audience participation. Contestants are asked to submit a photo, which is scanned and stored in the database. They are also encouraged to submit a short video interview. These interviews are only used for evaluating the final tier of candidates and portions might be used during a broadcast. The lead producer and the director evaluate each applicant and provide a quick rating (1-5). The highest rated candidates get a second look and become a finalist in the selection process.

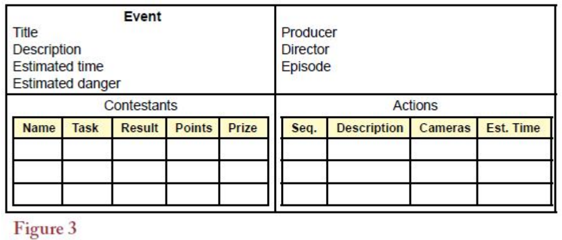
All applicants undergo a background check. Several reality-based shows have experienced problems when contestants were revealed to have unsavory pasts— including being arrested for violent crimes. Although the producers want risk-taking contestants, they also want to avoid embarrassing public disclosures. **Figure-2** shows the basic background data that investigators obtain on the finalist contestants. It is not foolproof, but by making a few phone calls, the investigators get a reasonable idea of the candidate’s background. National ID and Religion would be text fields. Appearance Rating and Strength Rating are numeric from 1-10. The producers and directors then select the contestants for an episode.

**Note:** The Employer list below does not have to link directly to the Job list in **Figure-1.**



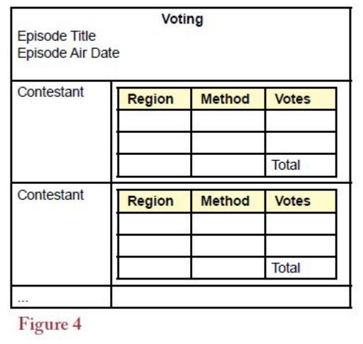
## Episodes and Events

Actual episodes consist of one or more events. Generally, there is only time for one or two events per episode, but the producers are thinking about the possibility of an hour- long special. **Figure-3** shows the two main aspects of the event: the setup from the perspective of the director, and the results from the perspective of the contestants. The director focuses on the sequence of actions, where the cameras will be located, and the estimated time of each section. For each event, a contestant is given one primary task. Sometimes they work in teams, so two or three people might be given the same task. The task result might just be a completion, or it might include a time for races. Either way, contestants are awarded points (sometimes negative points) for their role in the task (points range from -10 to +10). Occasionally, small prizes are awarded for completing a task, but contestants are really competing for the overall prizes in the event. The danger level will be numeric from 1-10.



## Voting

When the episode is aired, the audience is asked to participate by voting for favored contestants. The vote totals are used to determine which contestants to bring back for future episodes, and to give prizes for the overall season leader. Each vote has the episode title, date, contestant, region, and method associated with it. The executive producers are trying to entice advertisers by supporting several means of voting. Consequently, they want to track the actual method used to cast a vote. Common methods include telephone, cell phone/text messaging, e-mail, and a website. These should be available in a dropdown on a form which means it is not a simple text field. They also need to track audience participation by geographic region. On the report shown in **Figure-4**, the regional level is global and also supports a dropdown list of predefined areas (you can use continents, but it could be broken down to finer grained level).



# Required Business Solution

1. Create a normalized list of tables for each of the above forms.
2. Create an ERD which shows these tables and fields.
3. Define the tables in a DBMS along with the necessary relationships and integrity constraints.
4. Enter sample data to populate the tables to test your design.
5. Create queries to answer the following questions:
   1. Calculate the average producer and director ratings for a specific event's team members (you pick the event title).
   2. Determine which actions will take the longest. Rank all actions from longest to shortest.
   3. List each contestant's total votes by region and method for a specific episode (you pick the episode title). Rank this from highest to lowest.
   4. Identify which contestants have not participated in any events.
   5. What is the highest estimated danger level for any event?
   6. Show a relational expression for any one query

**Note:** Insert constraints on the fields specified in above that have limits on them.

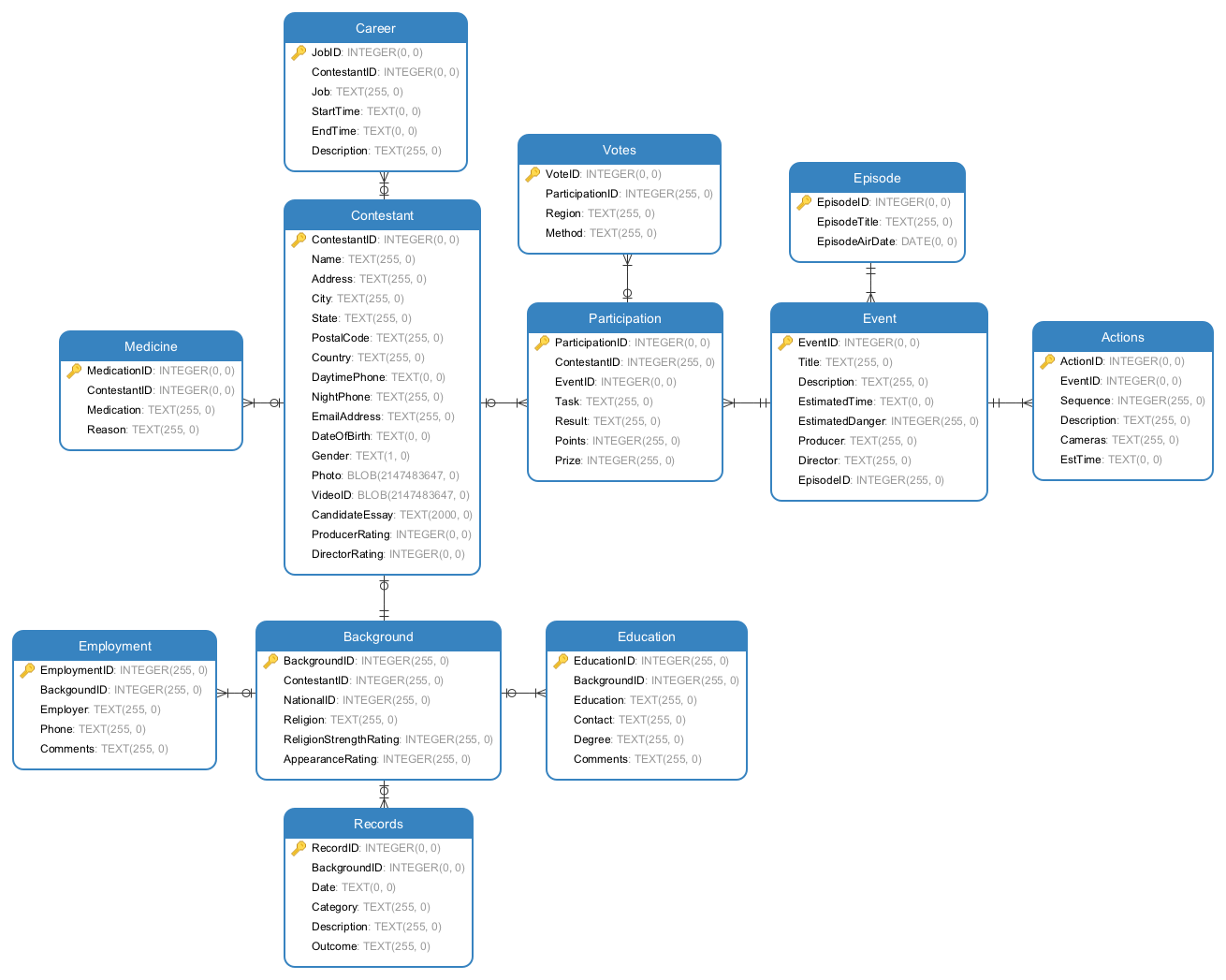
# Proposed Business Solutions

## Normalized List of Tables

The following list shows the database tables attributed to each information form collected from the applicants of **Big Theeth Reality TV** show.

|  |  |  |
| --- | --- | --- |
| **Information Form** |  | **Corresponding Database Table(s)** |
|  |  |  |
| Basic Contestant Application Form |  | Contestant |
|  |  | Medicine |
|  |  | Career |
|  |  |  |
| Background Check Form |  | Background |
|  |  | Employment |
|  |  | Education |
|  |  | Records |
|  |  |  |
| Episodes and Events Form |  | Event |
|  |  | Participation |
|  |  | Actions |
|  |  | Episode |
|  |  |  |
| Voting |  | Votes |

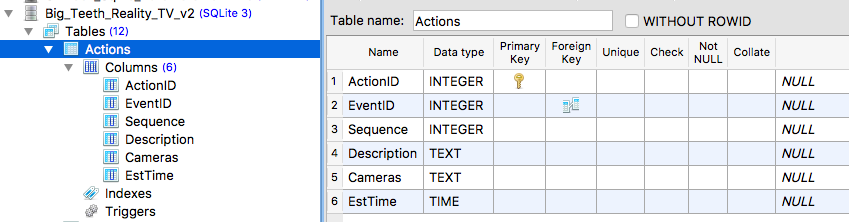
## Database Entity Relationship Diagram



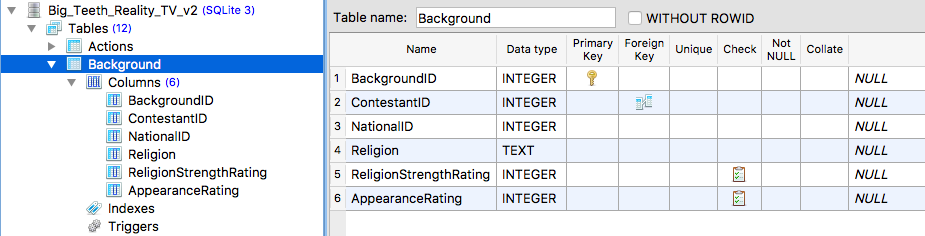
## Database Tables

Below are the screenshots of database tables defined in SQLiteStudio database management software for Big Teeth Reality TV show database.

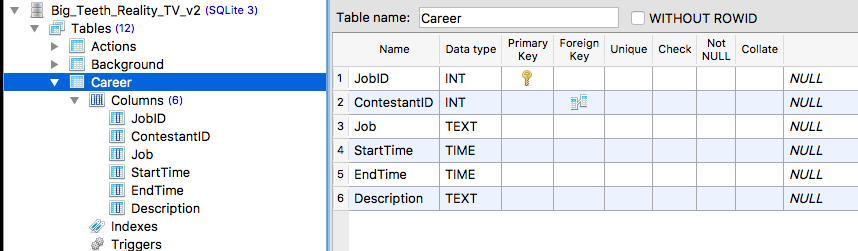
**Actions Table**



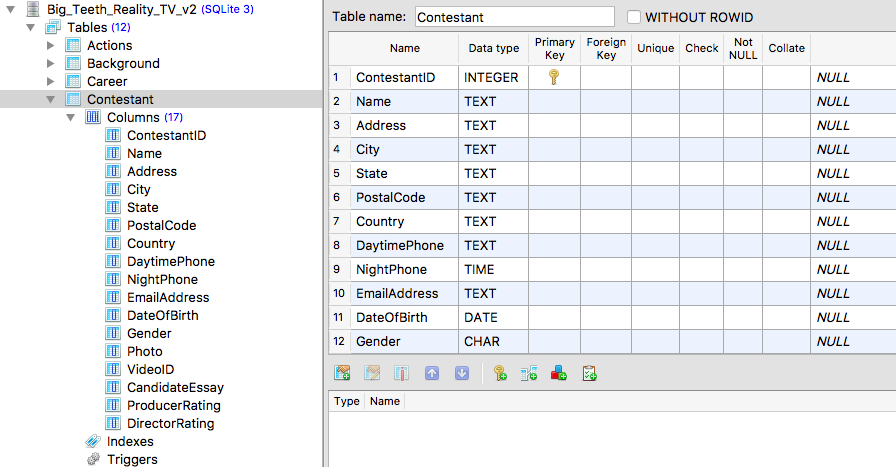
**Background Table**



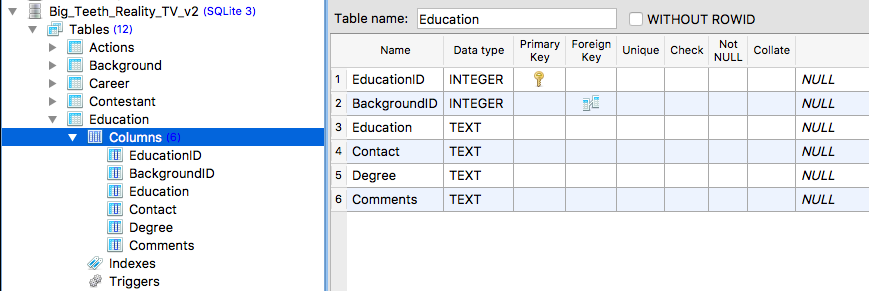
**Career Table**

****

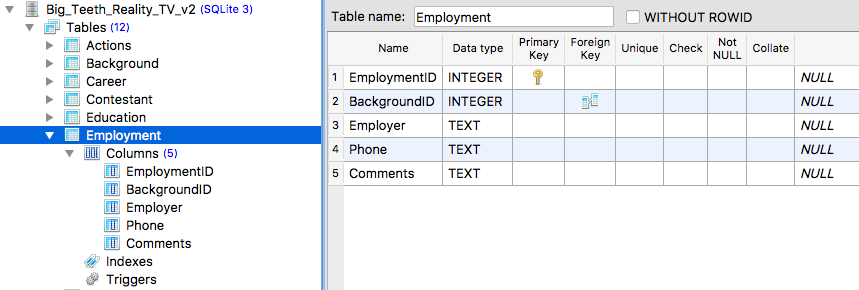
**Contestant Table**

****

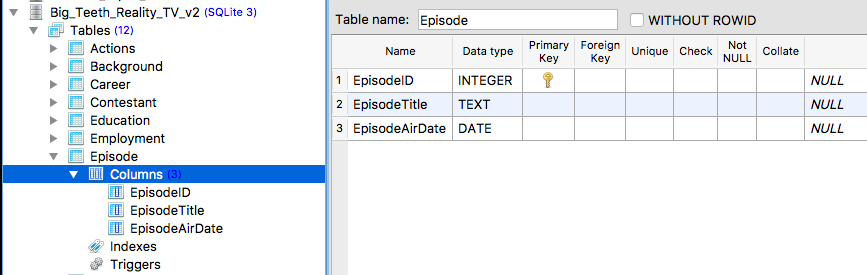
**Education Table**

****

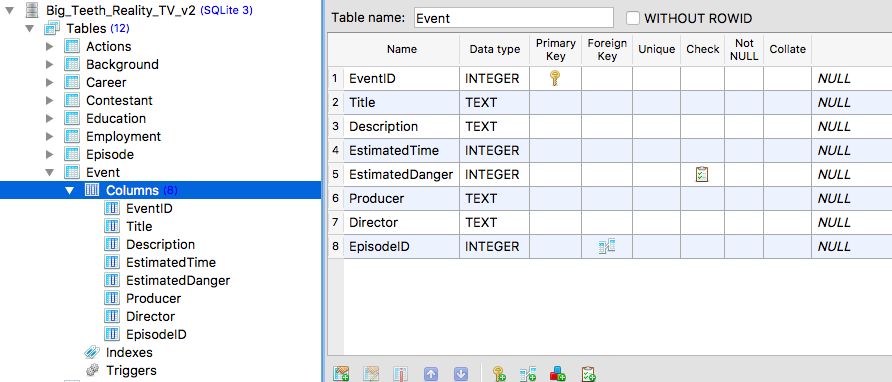
**Employement Table**

****

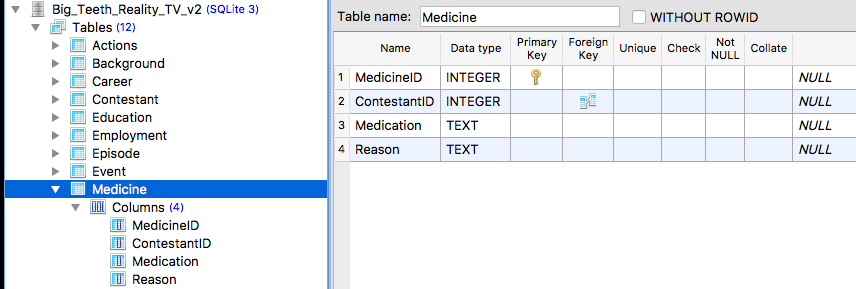
**Episode Table**

****

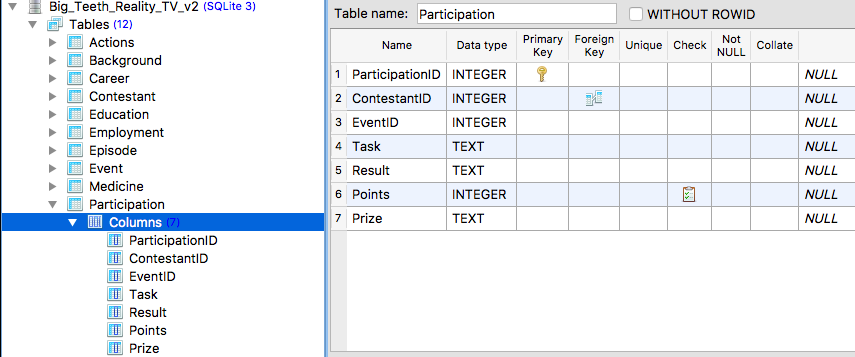
**Event Table**

****

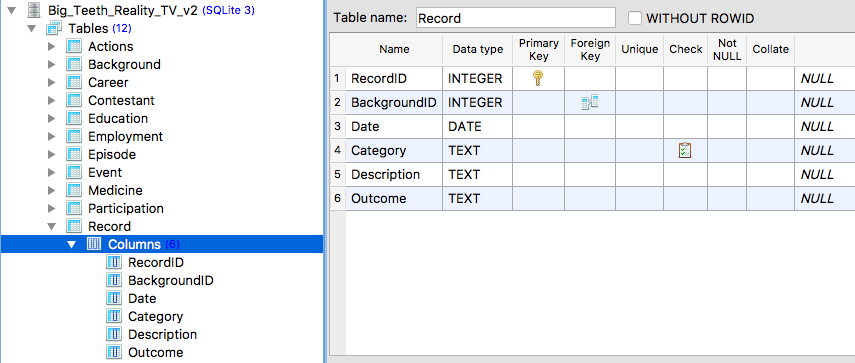
**Medicine Table**

****

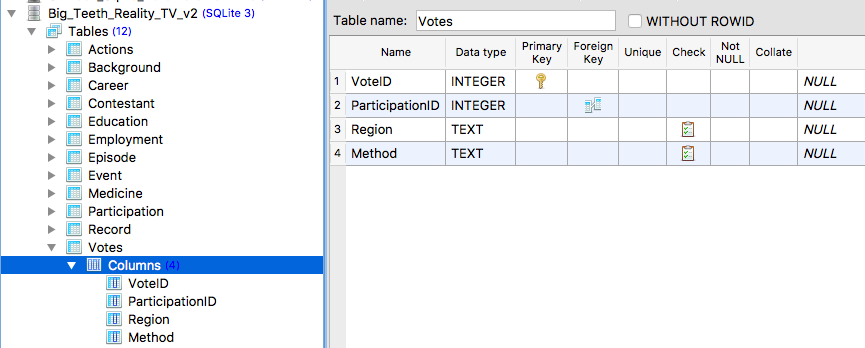
**Participation Table**

****

**Record Table**

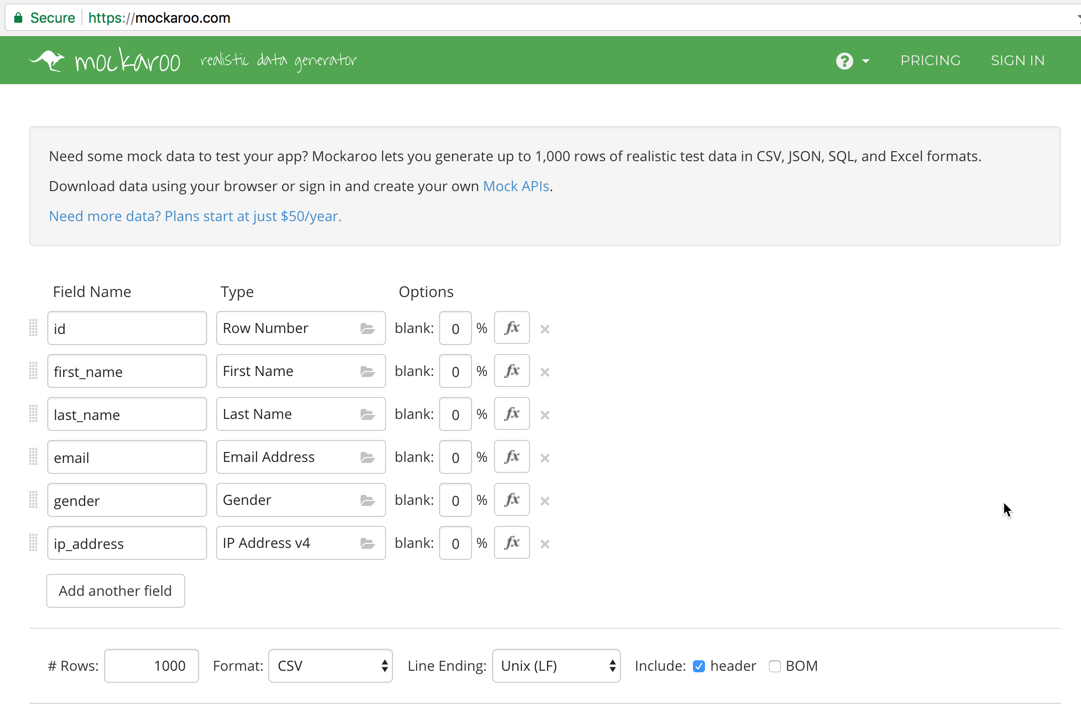
****

**Votes Table**

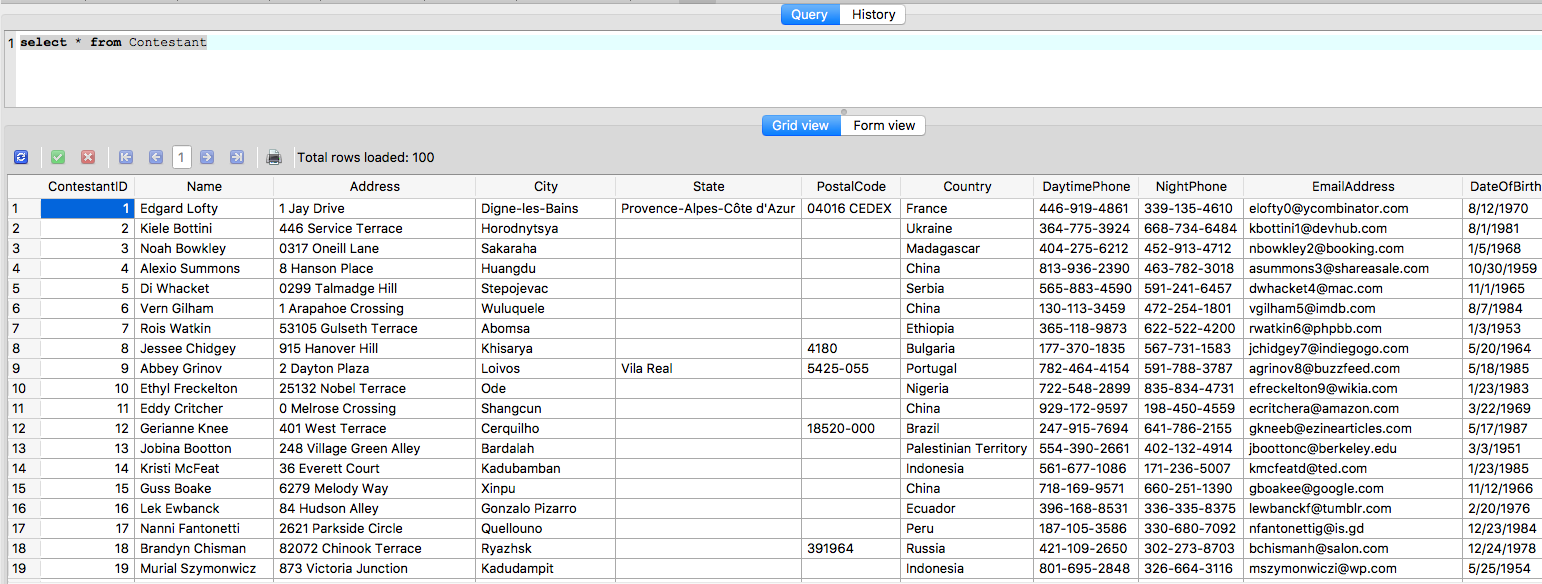
****

## Sample Data

Sample data for the proposed database solution is generated using mockaroo.com, an online, realistic test data generation application. The test data generated is provided as .csv files along with the submitted proposed database design specifications document.



A sample generic search query is shown in below screenshot as working proof of concept.



## Requested Database Queries

**The average producer and director ratings for a specific event's team members**

SQL Query:

SELECT AVG(ProducerRating), AVG(DirectorRating)

FROM Contestant

INNER JOIN Participation ON Contestant.ContestantID=Participation.ContestantID

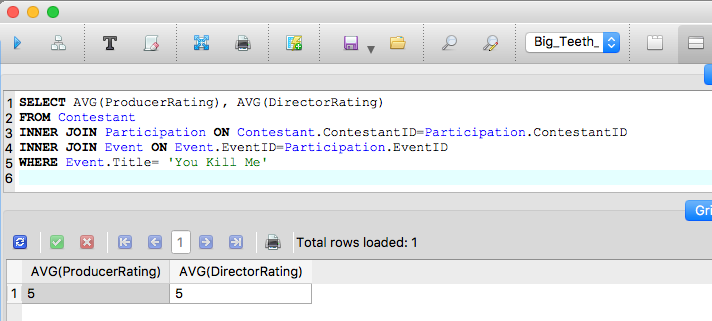
INNER JOIN Event ON Event.EventID=Participation.EventID

WHERE Event.Title= 'You Kill Me'

Relational Algebra:

ℑAVG(ProducerRating), AVG(DirectorRating) (Event.Title= 'You Kill Me' (Contestantjoinop < ContestantID= ContestantID> (Participation <EventID=EventID> Event)))

Query Output on SQLiteStudio:



**Determine which actions will take the longest. Rank all actions from longest to shortest.**

SQL Query:

SELECT EventID,Title,Sequence, EstTime

FROM Actions

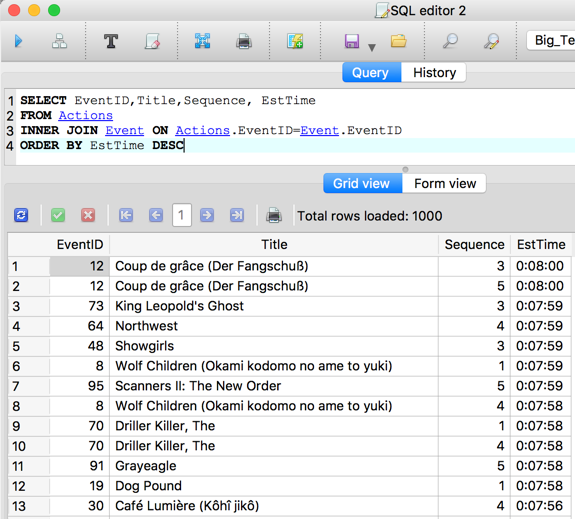
INNER JOIN Event ON Actions.EventID=Event.EventID

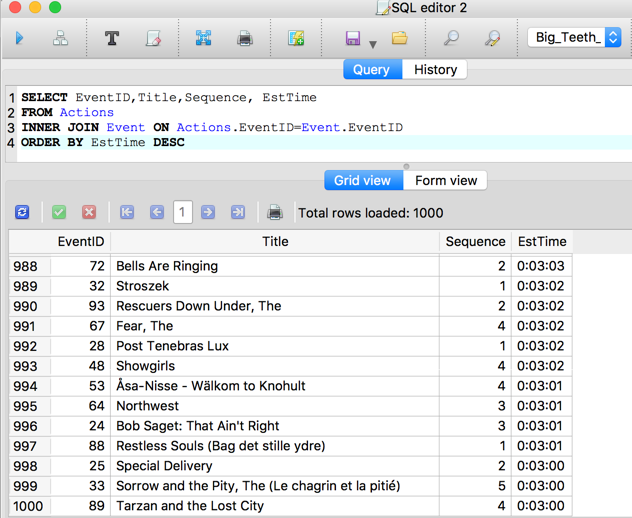
ORDER BY EstTime DESC

Relational Algebra:

 EventID,Title,Sequence, EstTime (Actionsjoinop <EventID=EventID> Event <EventID=EventID>)

Query Output on SQLiteStudio:





**List each contestant's total votes by region and method for a specific episode and rank this from highest to lowest.**

SQL Query:

SELECT Region,Method,Count(\*) AS TotalVotes

FROM Episode

INNER JOIN Event ON Episode.EpisodeID=Event.EpisodeID

INNER JOIN Participation ON Event.EventID=Participation.EventID

INNER JOIN Votes ON Participation.ParticipationID=Votes.ParticipationID

INNER JOIN Contestant ON Contestant.ContestantID=Participation.ContestantID

WHERE EpisodeTitle= 'Dirty Harry'

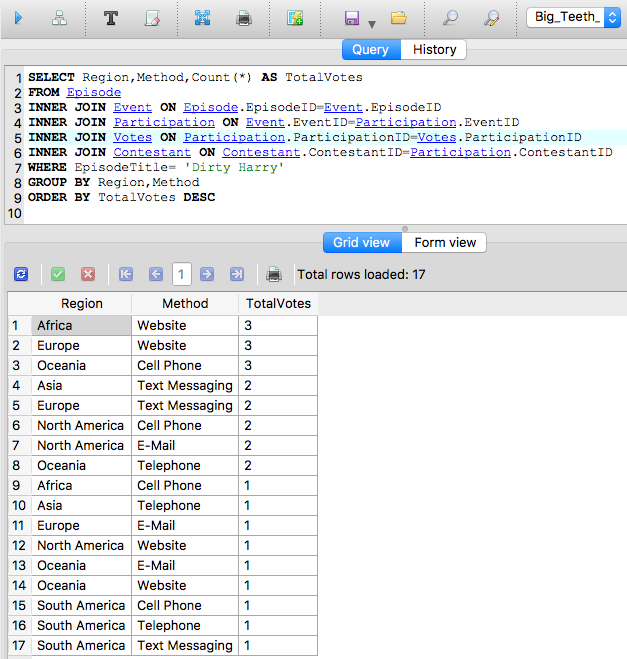
GROUP BY Region,Method

ORDER BY TotalVotes DESC

Relational Algebra:

Region , Method ℑ Count(\*) AS TotalVotes (EpisodeTitle= 'Dirty Harry' (Episodejoinop < EpisodeID= EpisodeID> (Eventjoinop <EventID=EventID> (Participationjoinop <PartcipationID= PartcipationID > (Votesjoinop < ContestantID= ContestantID > Contestant)))))

Query Output on SQLiteStudio:



**Identify which contestants have not participated in any events.**

SQL Query:

SELECT Name

FROM Contestant

EXCEPT

SELECT Name

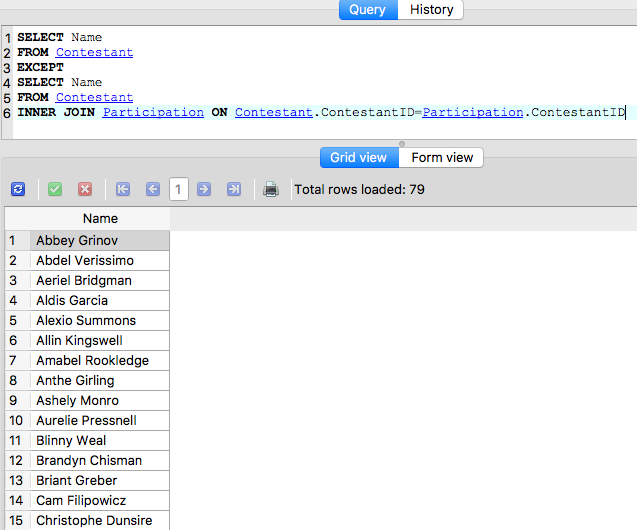
FROM Contestant

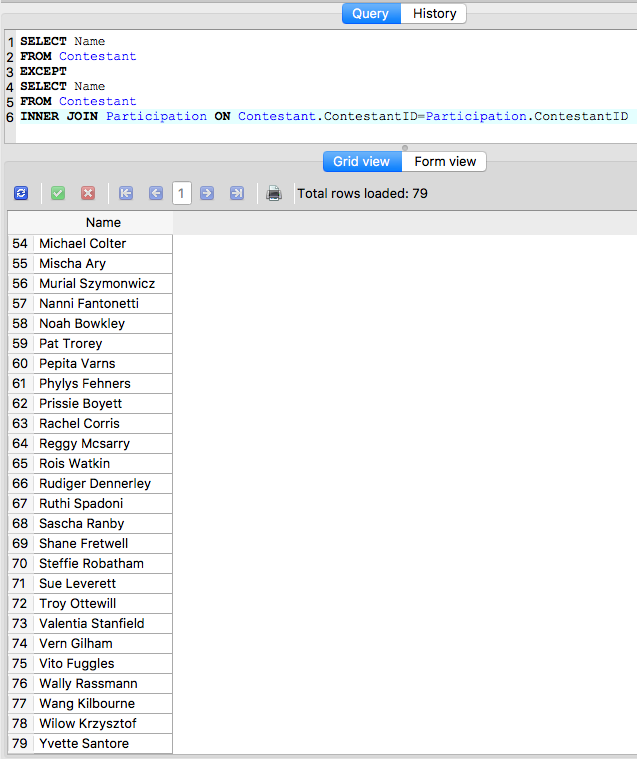
INNER JOIN Participation ON Contestant.ContestantID=Participation.ContestantID

Relational Algebra:

 Name (Contestant) ⎯  Name (Contestantjoinop < ContestantID= ContestantID > Participation)

Query Output on SQLiteStudio:





**What is the highest estimated danger level for any event?**

SQL Query:

SELECT EventID,Title, EstimatedDanger

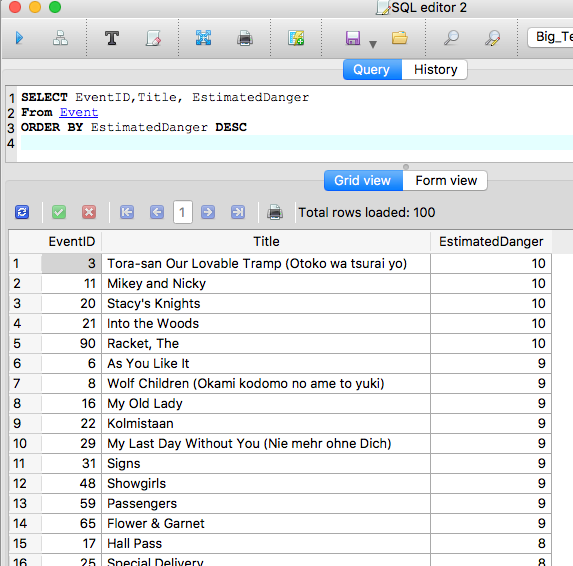
From Event

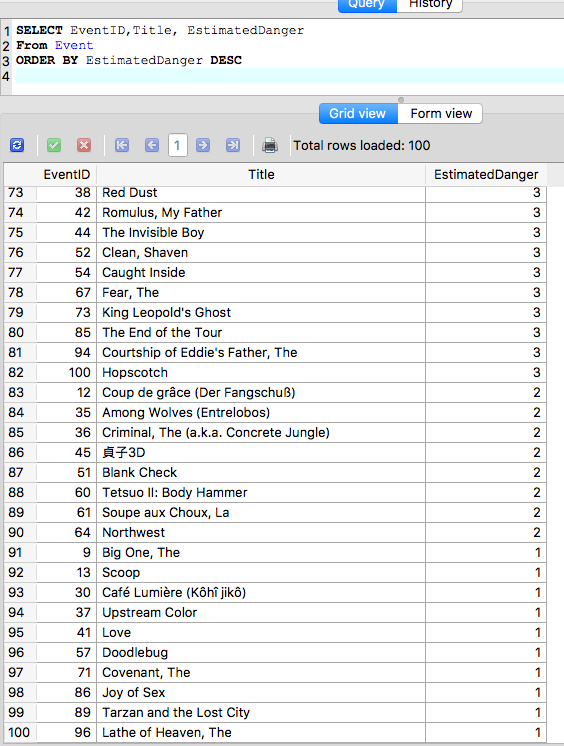
ORDER BY EstimatedDanger DESC

Relational Algebra:

 EventID, Title, EsttimatedDanger (Event)

Query Output on SQLiteStudio:





**Show a relational expression for any one query**

Relational expressions are provided for each query above.

# Technical Notes on Database Design

In this section, we are going to provide a summary information regarding the underlying reasoning of Big Teeth Reality TV Show database.

* Entity Relationship Diagram designed with business end users’ easy understanding of the database structure.
* While it could be possible to have less number of tables in the database, we decided to segregate the information in separate tables for easy readability and retrieval of information with simpler SQL statements. For example, our proposed database structure could have Background table embedded in Contestant table and Episode table could be embedded in Votes table. However, we decided to implement our present entity relationship solution, mainly to let business end users with little or no database/SQL experience to get up to speed and productive in shorter time.
* For large databases in order of gigabytes, our present database structure could have encounter some performance issues, however, for the given and projected size of Big Teeth Reality TV Show database, we decided to trade off optimized storage concern with more readable and understandable database with simpler queries.
* We know that every episode has some events and every event has some contestants with different participationID values. Required business solution asks for vote contestants in episode but in our ERD, people vote contestants in event. It means that our voting system is more accurate than problem definition.

# Database Team: Team STU

This proposed database solution came to life with perfectly balanced and coordinated efforts of Mr. Swogger, Mr. Taheri, and Mr. Ugur. While everyone equally worked on developing the presented database solution, special highlights for the team members could be summarized as follows:

* Ali Taheri: Finding alternative and easier tools and coming up with relational algebraic expressions for required database queries.
* Matt Swogger: Matt’s current software development experience was instrumental in deciding on the final design decisions of the presented database structure and ERD.
* Arda Ugur: Arda’s extensive technical writing experience made it possible to document the team’s technical work and presenting a readable business proposal document. Arda also has experience with SQL and Oracle DBS which made him an important member of the team for verifying the proposed solutions and illustrated database queries.

# Database Tools

**Database Vendor:**

SQLite: https://www.sqlite.org/

**Relational Database Management Software:**

SQLiteStudio: <https://sqlitestudio.pl/index.rvt>

**Database Data Modeling:**

Navicat Data Modeler: <https://www.navicat.com/>

**Test Data Generation:**

Mockaroo: <https://mockaroo.com/>