



PROJECT - RELATIONAL DATABASE

PREPARED FOR

Big Teeth Reality TV

PREPARED BY

Team Name: **PINCHOT GROUP;**

- Harmeet Singh (ID# 989470994)
- Purwa Mugdiya (ID# 989468918)
- Meiyi Zhou (ID# 989470002)
- Vamshi Krishna Perabathula (ID #989472817)
- Sai Pridhvi Pinninti (ID #989469046)
- Pavan Kumar Pekala (ID #989468153)
- Aman Singh (ID# 989467252)
-



Introduction

This report is prepared for the Big Teeth Reality TV Database Management Team by Pinchot Advisors to guide and recommend the reality TV show company on how to best manage their database for their upcoming reality TV show.

This report is prepared by students of The University of Pacific enrolled in the first year of the Masters of Data Science program. The individual team members' strengths have been utilized to produce a comprehensive report and database management solution on Valentina.

Reliance & Limitations:

The contents of this report and recommendations made are based on the project descriptions for the Big Teeth Reality TV show and the inferences made by the individual team members of Pinchot Advisors Group.



Normalized List of Tables

In our creation of the normalized tables, we have followed the principles of 1NF (First Normal Form), 2NF (Two Normal Form), 3NF (Three Normal Form) to reduce data redundancy, ensure data accuracy when collecting and updating data, simplifying the overall data management process.

The list of our 17 normalized tables with brief explanations of their existence is given below:

Key :- Primary Key, Foreign Key

1. Applicants

Applicants(Contestant_id, Location_id, FirstName, LastName, Email_Address, Gender, DOB, Personal_Phone, Business_Phone, Street_Address, Postal_Code, Candidate_Essay, Video_Link, Photo, Finalist)

Explanation: The Applicant table consists of the attributes that are unlikely to change during the period of reality TV show (from application stage to final episode). An additional attribute for 'Finalist' was created to flag whether an applicant was finally selected after getting good ratings and a clean background check.

2. Location

Location(Location_id, Country_Code, City, State, Country, Continent)

Explanation: The locations table consists of all the generic information about various regions (city, state, country) that can be pulled from a global table. This way the location of the contestants/applicants will be consistent and the addresses can be verified at time of input.

3. Education

Education(Education_id, Contestant_id, School_Name, Degree, Contact, Comments)

Explanation: Each education record for every contestant would be unique and an individual can have 0 or more education backgrounds. Thus a separate table is created for this.

4. Medications

Medication(Medication_id, Medication_Name)

-

Explanation: A separate table is created for the unique medications. For example, multiple contestants can be using the same medication and thus to avoid duplication in the health_details table (next table), the medication table was created.

5. Health_Details

Health_Details(Contestant_id, Medication_id, Reason)

Explanation: This table stores all the different health records (a combination of the contestant and medication_id can be used to identify the reason for each candidate's reason for using a certain medication).

6. Job_Details

Job_Details(Job_id, Contestant_id, Job_Role, Job_Description, Start_Date, End_Date)

Explanation: This table stores all the records of job experience for each contestant. The job record in this case is the primary key as individuals are expected to have different jobs and different start and end dates. If the form had a defined list of jobs that applicants can select, then a separate table for the Job roles and descriptions could have been created.

7. Employer_History

Employer_History(Employer_id, Contestant_id, Employer_name, Phone, Comments)

Explanation: Similarly, each contestant's employment records will be unique and each contestant can have more than one employer in the past. Thus, the unique employer id is the primary key.

8. Ratings

Ratings(Rating_id, Contestant_id, Rating, Rated_by)

Explanation: Each contestant will be rated by the producer or director and it's not expected that both the producer and director would rate the applicant in the same order or at the same time. Thus a longer table structure than a wider table structure is chosen so that new ratings are appended as and when the producer/director rates the applicants. Each rating_id is unique but an applicant will appear twice to allow for the two separate ratings.

9. Finalist_Background

Finalist_Background(National_id, Contestant_id, Religion, Appearance_rating, Strength_rating)

Explanation: Since every contestant's background check is done, we created a separate table for the basic background check. The National ID in this case is the primary key. However, we do not expect any duplication in the table as each contestant's religion and ratings would only be provided once and won't change.

10. Police_Records

Police_Records([Record_id](#), [National_id](#), Date, Category, Record_Description, Outcome)

Explanation: Each police record is unique and each contestant may have no police records or may have multiple records and thus a separate table for the police records.

11. Episode

Episode([Episode_id](#), Episode_Title, Episode_Air_Date, Director, Producer)

Explanation: Each episode of the reality TV show would be unique. Each episode will be aired on different dates with different producers and directors. There could be multiple or singular events and these are then left out of the episodes tables. This table therefore just key meta information that's unique to each episode.

12. Event

Event([Event_id](#), [Episode_id](#), Title, Event_Description, Estimated_Time_mins, Estimated_Danger)

Explanation: There could be multiple events in an episode or just one. Therefore, the list of all events is stored in a separate Event table.

13. Event_Direction

Event_Direction([Event_id](#), [Sequence_No](#), Sequence_Description, Cameras, Estimated_Time)

Explanation: The director's plan for each event is going to be unique which has multiple layers in the form of sequences. Therefore, the combination of the Event_id and Sequence_No is a composite primary key (because any given event can have multiple sequences).

14. Tasks

Tasks([Task_id](#), [Event_id](#), Task_Name, Task_Prize, Group_Task)

Explanation: The tasks are expected to be unique and there could be multiple tasks in any given event of an episode. Here, we have also added a check to confirm whether the Task is a group task or not.

-

15. Tasks_Contestants

Tasks_Contestants(Task_id, Contestant_id, Task_Result, Points)

Explanation: Each task can have one or more contestants and thus a separate table for this was created to remove this redundancy in the Tasks table above. In this table, each task and contestants specific results and points (from -10 to 10) can be captured.

16. Votes

Votes(Vote_id, Episode_id, Contestant_id, Location_id, Method_id)

Explanation: Each unique vote will capture the episode related details, the contestant the vote is for, the location from which the vote was generated and the method of the vote. Because these tables already exist, we created foreign keys to link to the original tables to avoid duplication of data.

17. Vote_Method

Vote_Method(Method_id, Method_Type)

Explanation: Since there are 4 unique vote method types, and they may change or evolve in the future, we create a separate table to store these constraints.



Relationships Explained

We have based on our understanding of the project problem statement, outlined the various relationships between entities and expressed them below:

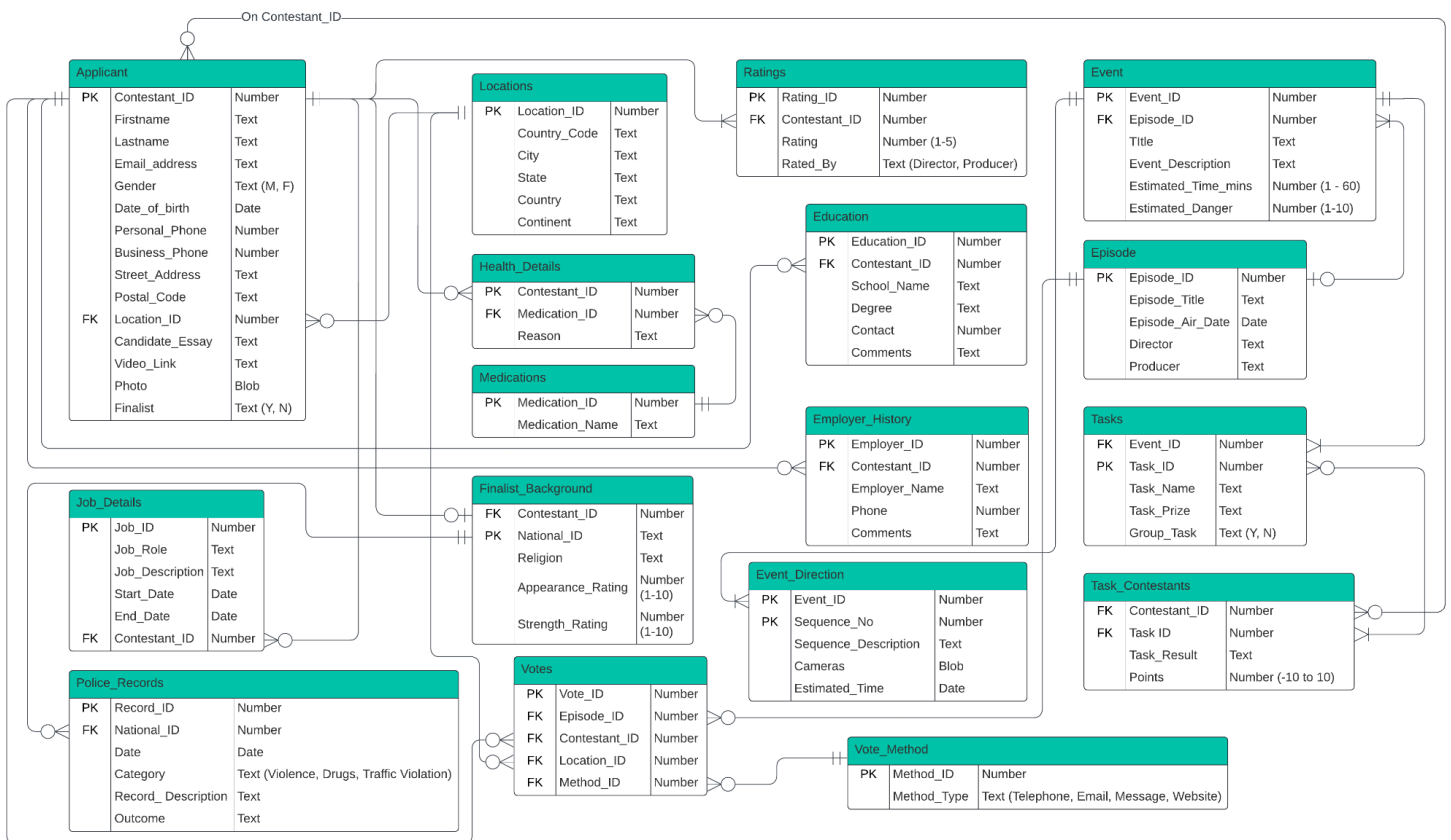
#	Relationship	Keys & Table
1- One and Only One TO Zero or Many (1 : 0,M)		
1.1	Each Contestant can have 0 or many health records, but each unique health record relates to only one given contestant	Key: Contestant_ID Tables: Applicants & Health Details
1.2	Many or One Contestant belong to a specific location, but each location relates to either a contestant or a vote and thus the relationship with contestants is zero or many.	Key: Location_ID Tables: Applicants and Location
1.3	Each Contestant can have 0 or many education records, but each unique education record relates to only one given contestant.	Key: Contestant_id Tables: Applicants and Education
1.4	Each Contestant can have 0 or many Job records, but each unique Job record relates to only one given contestant	Key: Contestant_id Tables: Applicant and Job Details
1.5	Each Health problem can have 0 or many medication records, but each unique medication record relates to only one given Health problem	Key: Medication_id Tables: Health Details, Medication
1.6	Each Contestant can have 0 or many Employer history records, but each unique Employer relates to only one given contestant	Key: Contestant_id Tables: Applicant and Employer History
1.7	Each Finalist Background can have 0 or many Police Records, but each unique Police record relates to only one given Contestant	Key: National_id Tables: Finalist Background and Police Records
1.8	Each Location can have 0 or many Votes but each Vote record relates to at least one Location.	Key: Location_id Tables: Location and Vote

1.9	Each Contestant can have zero or many Votes but each vote belongs to one and only one Contestant	Key: Contestant_id Tables: Applicant and Vote
1.10	Each Vote Method can have 0 or many Votes, but each Vote will have one and only one Vote Method	Key: Method_id Tables: Vote and Vote Method
1.11	Each Episode can have 0 or many Votes, but each Vote belongs to one and only one Episode	Key: Episode_id Tables: Episode and Vote
2 - One and Only One TO One or Many (1 : 1,M)		
2.1	Each Contestant can have one or many Ratings but each Rating belongs to one and only one Contestant	Key: Contestant_id Tables: Applicant and Rating
2.2	Each Event will have at least one or many sequences of events.	Key: Event_id Tables: Event and Event Direction
2.3	Each Event can have 1 or many Tasks, but each task will have one and only one Event.	Key: Event_id Tables: Event and Tasks
3 - Zero or One TO One or Many (0,1 : 1,M)		
3.1	Each Episode can have 1 or many Events but each Event record relates to 0 or 1 Episode	Key: Episode_id Tables: Episode, Event
3.2	Each Task can have one or many contestants, but each contestant belongs to at least one Task.	Key: Task_id Tables: Tasks and Task Contestants
4 - Zero or One TO One and Only One (0,1 : 1)		
4.1	Applicant can have zero or one Finalist background record, but Finalist Background have one and only one specific Applicant record	Key: Contestant_id Tables: Applicant and Finalist Background



Entity Relationship Diagram

Based on the set of normalized tables and the identification of the relationships in the previous sections, the following ERD was created on the LucidChart platform:



Pinchot ERD.png - Click for a clearer and expanded view of the ERD

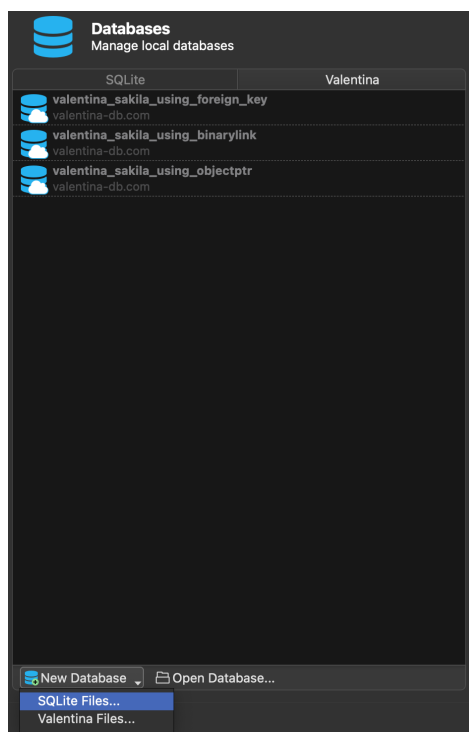
The relationships between the various entities (and keys) is illustrated using the conventional crow's foot notation in the ERD above.



Database & Table Creation

We have used Valentina's local database feature to create our database and tables. Below is screenshots of the process we followed to create the database and tables:

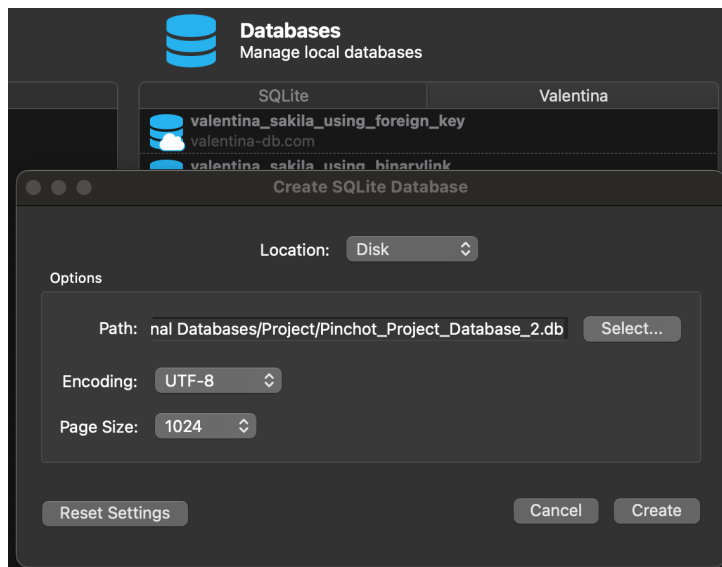
1. Creating a Local Database



- Click on the 'New Database' button to create a local database.
- Click on 'SQLite Files...'

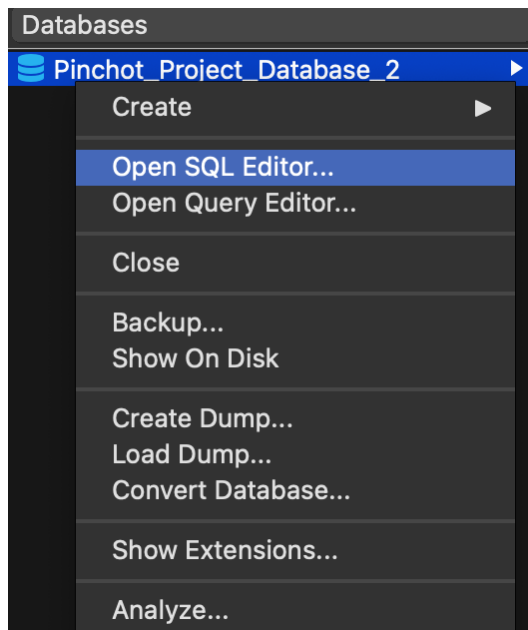
-

2. Define Database path and name



- Specify the file path and database name
- Click 'Create' to complete database creation process

3. Select Database and Create Tables using SQL Editor



- Right click on the created database and select 'Open SQL Editor...'

-

4. Create Queries for each table, attributes, constraints and relationships

```

1 CREATE TABLE `Locations` (
2   `Location_ID` NUMBER(10) UNIQUE,
3   `Country_Code` Text,
4   `City` Text,
5   `State` Text,
6   `Country` Text,
7   `Continent` Text,
8   PRIMARY KEY (`Location_ID`)
9 );
10
11 CREATE TABLE `Applicant` (
12   `Contestant_ID` NUMBER(10) UNIQUE,
13   `Firstname` Text,
14   `Lastname` Text,
15   `Email_address` Text,
16   `Gender` CHAR(1) CHECK (GENDER IN ('M', 'F')) ,
17   `Date_of_birth` DATE,
18   `Personal_Phone` NUMBER,
19   `Business_Phone` NUMBER,
20   `Street_Address` Text,
21   `Postal_Code` Text,
22   `Location_ID` NUMBER(10),
23   `Candidate_Essay` Text,
24   `Video_Link` Text,
25   `Photo` Blob,
26   `Finalist` Text ,
27   PRIMARY KEY (`Contestant_ID`),

```

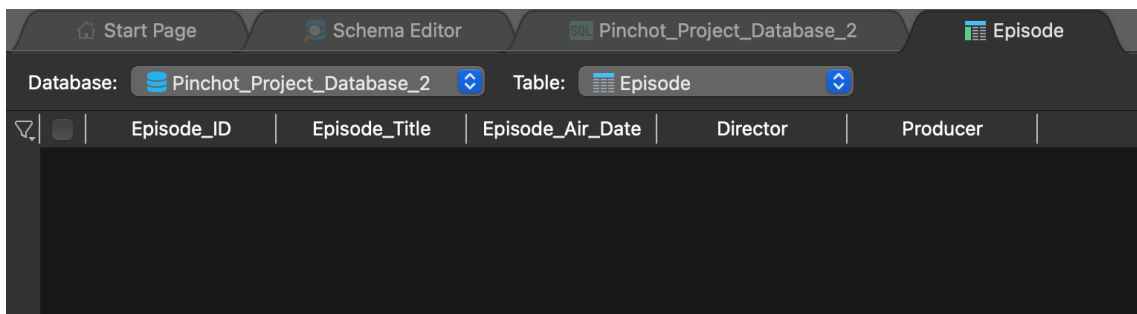
In the above snippet, one can see the table creation queries for the “locations’ and ‘Applicant’ tables. The full SQL text for all table creation is provided in Appendix 1 of the document.

Table Creation Output

Now that we have created the tables via SQL editor query execution, we can view the result of the tables below:

The 17 tables for our normalized set of tables can be seen above.

To further confirm that the tables were created successfully, a snippet of the episode table is given below:

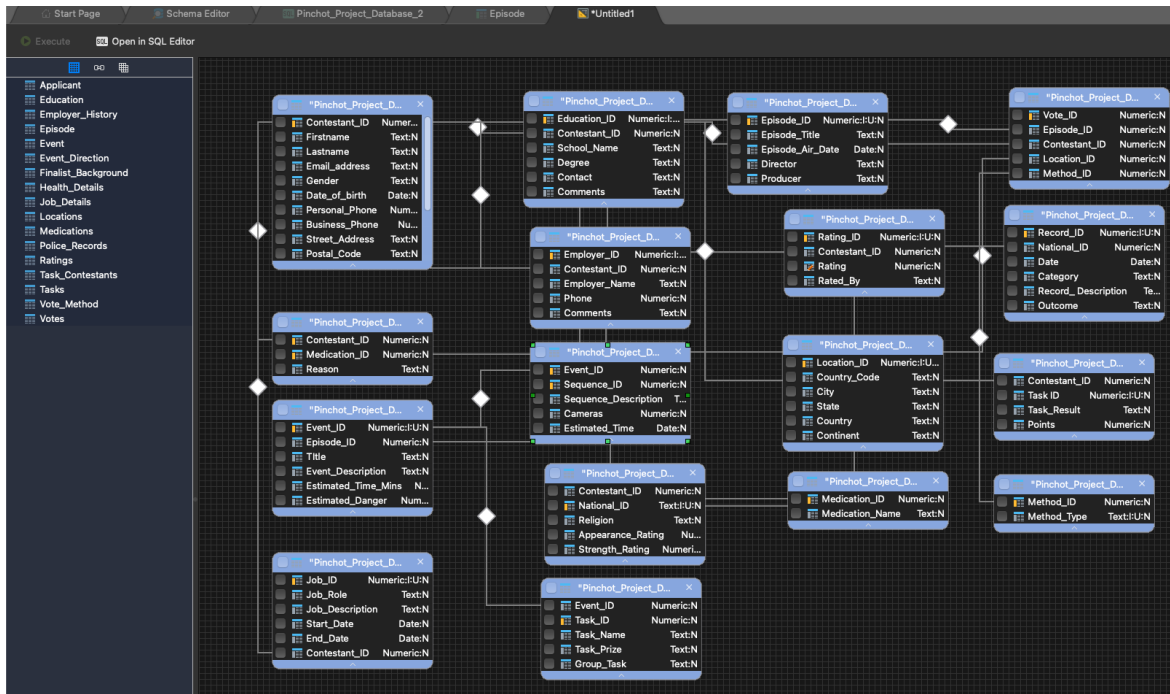


The screenshot shows a database interface with tabs for 'Start Page', 'Schema Editor', 'Pinchot_Project_Database_2', and 'Episode'. The 'Episode' tab is active, displaying the table structure for 'Episode'. The table has columns: Episode_ID, Episode_Title, Episode_Air_Date, Director, and Producer.

Episode_ID	Episode_Title	Episode_Air_Date	Director	Producer
------------	---------------	------------------	----------	----------

It can be confirmed that the various tables were created successfully. A link to our database file uploaded on google drive is provided in Appendix 2 of this document.

Lastly, we can view the full list of tables and their relationships on the Valentina Query Builders Canvas:



Note: The SQL Text for the table creation is available in Appendix 1 of this document.



Constraints Explained

Keeping in mind with the context of the problem statement in the project, we have inferred and employed the following constraints:

#	Description in Project Requirement	Constraint Implemented
1	Application form has a field for Gender	Gender can be "M" for Male and 'F' for Female
2	Applicants are rated by both Director and Producer out of 5	Ratings can be integer between 1 and 5
3	The contestants are selected by producers and directors based on the Appearance and Strength rating,	Ratings can be numeric from 1-10.
4	Contestants are awarded for their role in the task.	The points range from -10 to +10.
5	Estimated time of any Event in Minutes	Between 1 and 60 minutes.
6	The contestants competing for overall prizes in an event have an estimated danger level.	From 1 -10.
7	The common methods used to cast a vote.	Methods include 'Telephone', 'Messages', 'Website', and 'E-Mail'.
8	Whether the Applicant was finalized	'Y' or "N" based on evaluation of candidate
9	Estimated Time of any Sequence	Between 1 and 60 Minutes (Since no maximum time for a sequence is provided)
10	Whether a Task is a Group Task or Not	Value of 'Y' or "N"
11	Category of an existing Police Record for a contestant	Categories include 'Violence', 'Drugs' and 'Traffic Violation',

Note: These constraints can be confirmed to be implemented in the SQL Query Text for the Table Creation in Appendix 1. (We have highlighted these in **green** in the Appendix 1).



Creating Sample Data

We have created for this project some sample data to showcase how Big Teeth Reality TV can harness the power of SQL query building to answer key questions and provide the audience with statistics during the episode airings.

Some screenshots of the tables with sample data is as follows:

Ratings Table:

	Rating_ID	Contestant_ID	Rating	Rated_By
1	1	1	4	Producer
2	2	1	5	Director
3	3	2	5	Producer
4	4	2	3	Director
5	5	3	2	Producer
6	6	3	3	Director
7	7	4	3	Producer
8	8	4	4	Director
9	9	5	4	Producer
10	10	5	4	Director

Episodes Table:

	Episode_ID	Episode_Title	Episode_Air_Date	Director	Producer
1	1	Pilot Episode	2022-01-10	John Director	Jane Producer
2	2	The Adventure Begins	2022-01-17	Alice Director	Bob Producer
3	3	Mystery Unfolds	2022-01-24	Charlie Director	Diana Producer
4	4	Love and Betrayal	2022-01-31	Eva Director	Frank Producer
5	5	In the Spotlight	2022-02-07	Grace Director	George Producer
6	6	The Turning Point	2022-02-14	Holly Director	Henry Producer
7	7	A Twist of Fate	2022-02-21	Ian Director	Ivy Producer
8	8	Epic Showdown	2022-02-28	Jake Director	Jill Producer
9	9	Closure and Redemption	2022-03-07	Kevin Director	Kelly Producer
10	10	Grand Finale	2022-03-14	Liam Director	Laura Producer

Note: The script for creating the sample data on Valentina is included in Appendix 3 of this document.



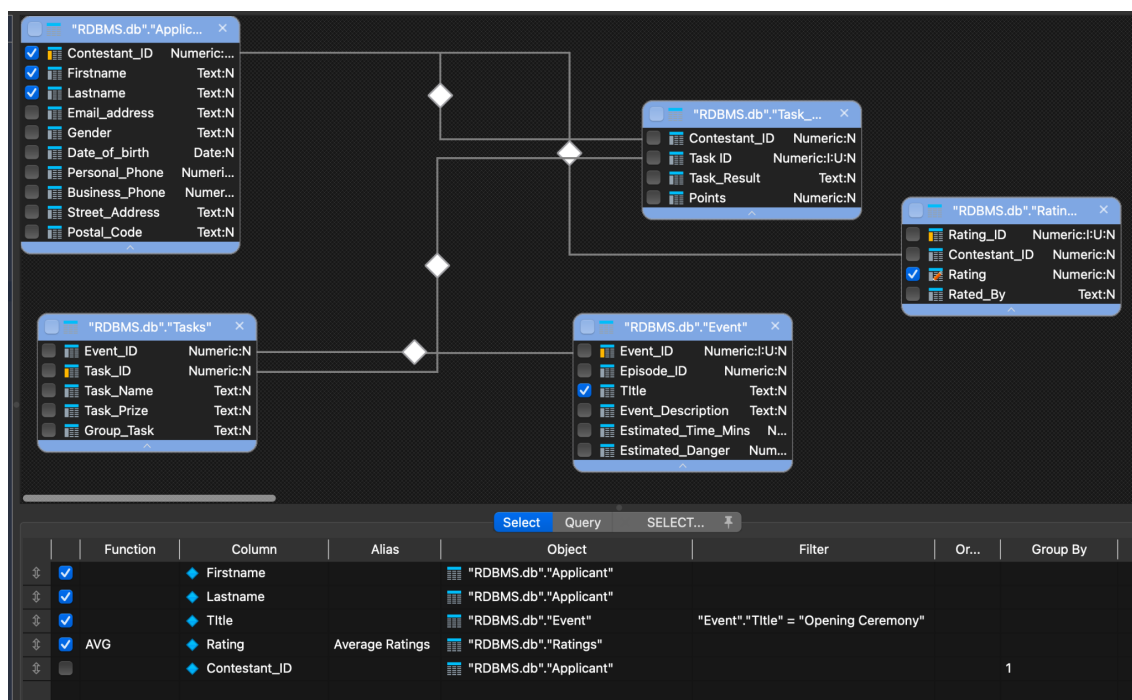
SQL Queries & Results

Given the sample data, we proceeded to run SQL queries to answer the questions in the project description:

Q1: Calculating the average producer and director ratings for a specific event's team members (you pick the event title).

In our database model, the ratings table has a list of ratings for each applicant/contestant and an attribute which relates to whether the rating is made by the Producer or Director. Therefore, we have chosen to average the ratings for contestants in the 'Opening Ceremony' Event by checking which contestants took part in that specific event from the tasks list and then joining with our Ratings table.

Visual Query:



SQL Query:

```
SELECT  "Applicant"."Firstname",

        "Applicant"."Lastname",

        "Event"."Title",

        AVG( "Ratings"."Rating" ) AS "Average Ratings"

FROM    "Applicant"

INNER JOIN "Ratings" ON "Applicant"."Contestant_ID" = "Ratings"."Contestant_ID"

INNER JOIN "Task_Contestants" ON "Task_Contestants"."Contestant_ID" =

"Applicant"."Contestant_ID"

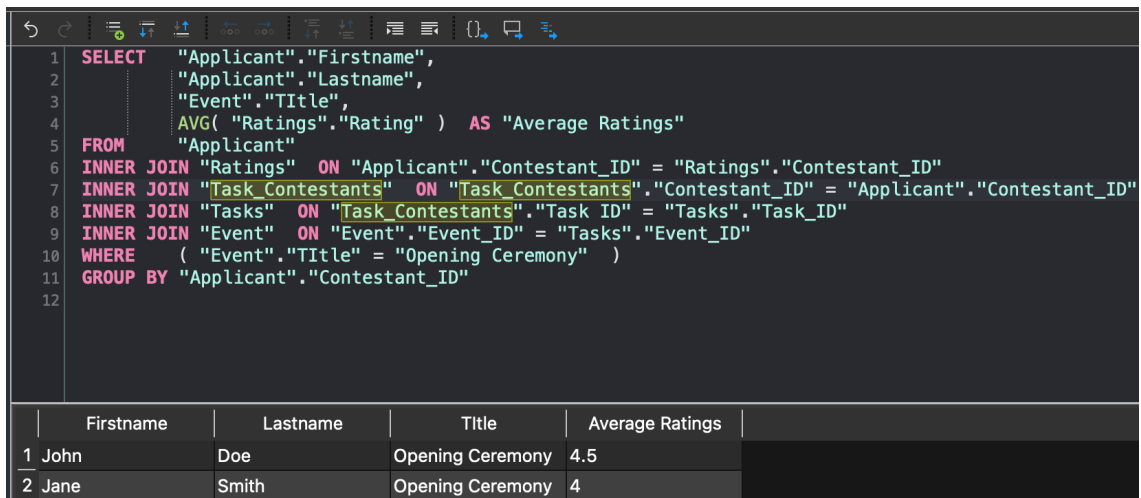
INNER JOIN "Tasks" ON "Task_Contestants"."Task_ID" = "Tasks"."Task_ID"

INNER JOIN "Event" ON "Event"."Event_ID" = "Tasks"."Event_ID"

WHERE   ( "Event"."Title" = "Opening Ceremony" )

GROUP BY "Applicant"."Contestant_ID"
```

Query & Result:



The screenshot shows a SQL query editor with a dark background. The query is displayed in a monospaced font with syntax highlighting. Below the query, the results are shown in a table with a light gray header and two data rows. The table has five columns: Firstname, Lastname, Title, Average Ratings, and an empty column. The first row shows John Doe with an average rating of 4.5, and the second row shows Jane Smith with an average rating of 4.0.

```
1 SELECT  "Applicant"."Firstname",
2          "Applicant"."Lastname",
3          "Event"."Title",
4          AVG( "Ratings"."Rating" ) AS "Average Ratings"
5 FROM    "Applicant"
6 INNER JOIN "Ratings" ON "Applicant"."Contestant_ID" = "Ratings"."Contestant_ID"
7 INNER JOIN "Task_Contestants" ON "Task_Contestants"."Contestant_ID" = "Applicant"."Contestant_ID"
8 INNER JOIN "Tasks" ON "Task_Contestants"."Task_ID" = "Tasks"."Task_ID"
9 INNER JOIN "Event" ON "Event"."Event_ID" = "Tasks"."Event_ID"
10 WHERE   ( "Event"."Title" = "Opening Ceremony" )
11 GROUP BY "Applicant"."Contestant_ID"
12
```

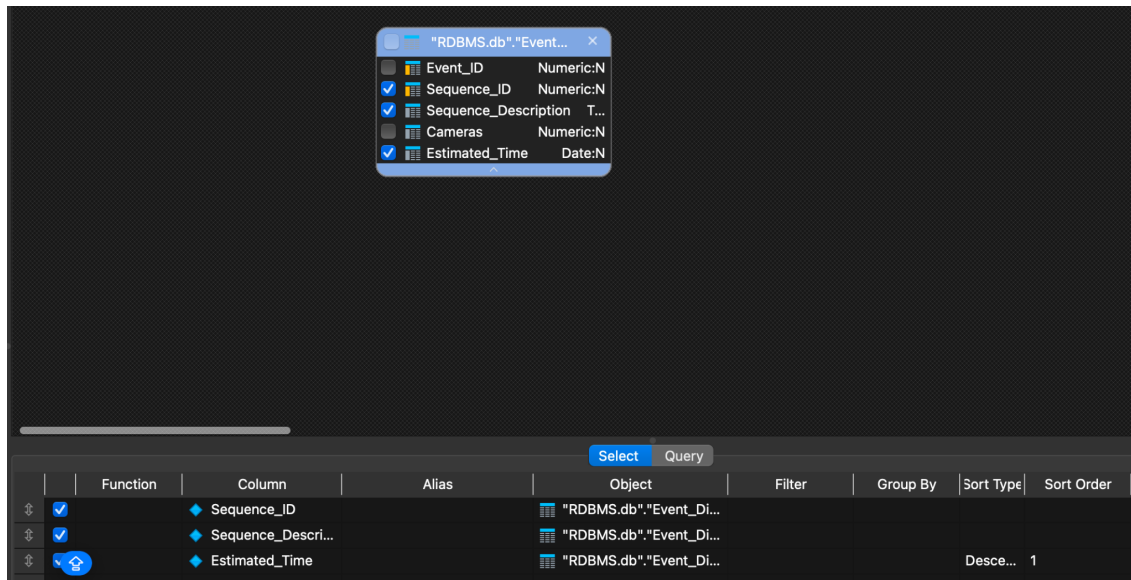
	Firstname	Lastname	Title	Average Ratings	
1	John	Doe	Opening Ceremony	4.5	
2	Jane	Smith	Opening Ceremony	4	

It can be seen above that John Doe who participated in the Opening Ceremony Event had an average rating of 4.5 whereas Jane Smith had an average rating of 4.0.

Q2: Determining which actions will take the longest. Rank all actions from longest to shortest.

In our case, the various actions relate to the various unique sequences. We therefore, use the Event Direction table which has all the necessary information to carry out this query.

Visual Query:



SQL Query:

```
SELECT "Event_Direction"."Sequence_ID",  
       "Event_Direction"."Sequence_Description",  
       "Event_Direction"."Estimated_Time",  
       DENSE_RANK() OVER (ORDER BY "Event_Direction"."Estimated_Time" DESC)  
AS RANK  
FROM "Event_Direction"  
ORDER BY "Event_Direction"."Estimated_Time" DESC
```

Query & Result:

```
1 SELECT "Event_Direction"."Sequence_ID",
2        "Event_Direction"."Sequence_Description",
3        "Event_Direction"."Estimated_Time",
4        DENSE_RANK() OVER (ORDER BY "Event_Direction"."Estimated_Time" DESC) AS RANK
5 FROM "Event_Direction"
6 ORDER BY "Event_Direction"."Estimated_Time" DESC
```

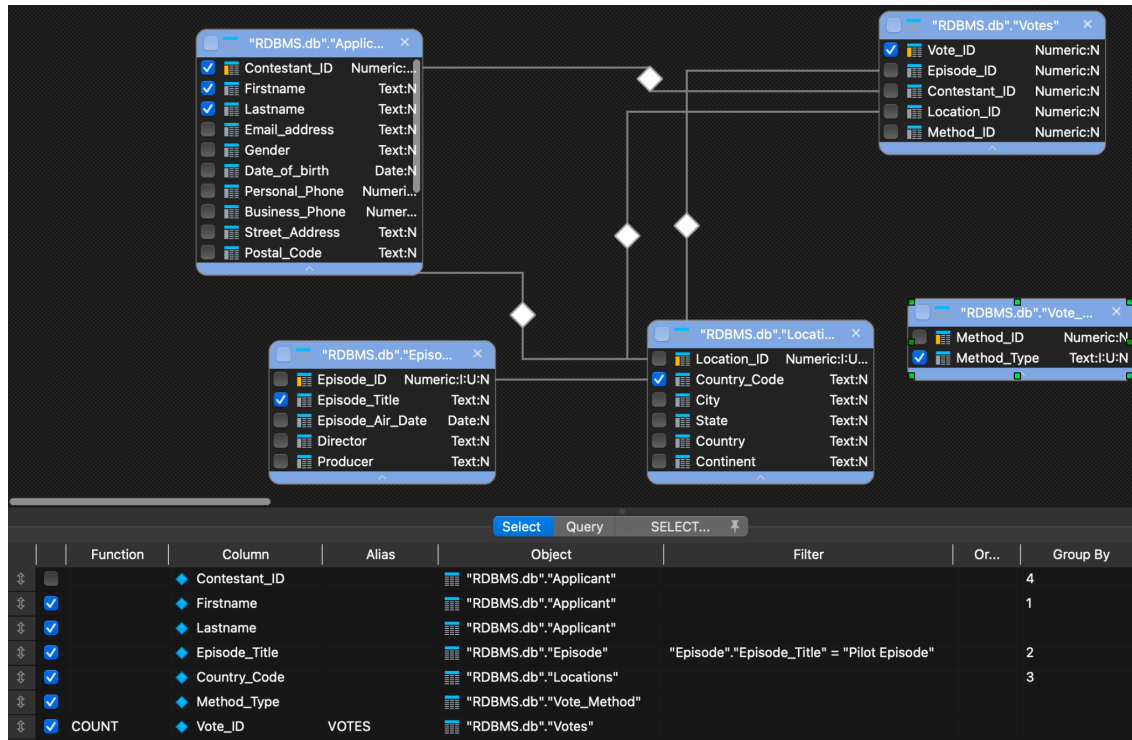
	Sequence_ID	Sequence_Description	Estimated_Time	RANK
1	1	Clue Discovery Shots	40	1
2	2	Plot Unraveling Scene	40	1
3	1	Critical Decision Setup	35	2
4	2	Alliance Challenge Briefing	35	2
5	1	Introduction Shots	30	3
6	2	Obstacle Course Start	30	3
7	3	Puzzle Challenge Setup	30	3
8	1	Plot Twist Scene	15	4
9	2	Character Development Interviews	15	4
10	1	Love Triangle Discussion	10	5

As can be seen from above, the Clue Discovery Shots and Plot Unraveling Scenes are the sequences with longest duration in our generated dataset.

Q3: Listing each contestant's total votes by region and method for a specific episode (you pick the episode title). Rank this from highest to lowest.

In our case, we have selected the 'Pilot Episode' to execute this query. Reliance is made on the information in the Votes, Votes_Method, Location, Episode and Applicant tables.

Visual Query:



SQL Query:

```
SELECT "Applicant"."Firstname",
       "Applicant"."Lastname",
       "Episode"."Episode_Title",
       "Locations"."Country_Code",
       "Vote_Method"."Method_Type",
       COUNT( "Vote_Method"."Method_ID" ) AS "VOTES",
```

```

        DENSE_RANK() OVER (ORDER BY "VOTES" DESC) AS RANK

FROM   "Applicant"

INNER JOIN "Votes" ON "Applicant"."Contestant_ID" = "Votes"."Contestant_ID"

INNER JOIN "Locations" ON "Locations"."Location_ID" = "Applicant"."Location_ID"

INNER JOIN "Episode" ON "Episode"."Episode_ID" = "Votes"."Episode_ID" ,

        "Vote_Method"

WHERE   ( "Episode"."Episode_Title" = "Pilot Episode" )

GROUP BY "Episode"."Episode_Title",

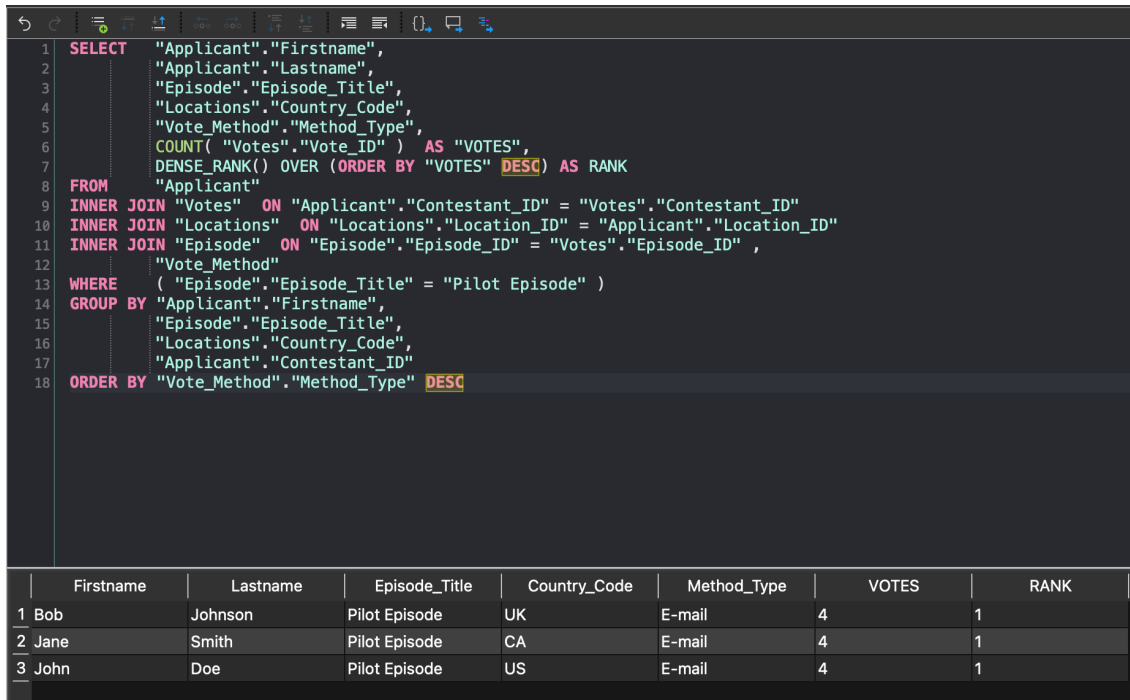
        "Applicant"."Contestant_ID",

        "Locations"."Country_Code"

ORDER BY "Vote_Method"."Method_Type" DESC

```

Query & Result:



The screenshot shows a SQL IDE with a query editor and a results pane. The query is a complex SQL statement using DENSE_RANK() to rank contestants based on votes. The results pane displays a table with 7 columns: Firstname, Lastname, Episode_Title, Country_Code, Method_Type, VOTES, and RANK. The results show three rows, all with 4 votes and a rank of 1.

	Firstname	Lastname	Episode_Title	Country_Code	Method_Type	VOTES	RANK
1	Bob	Johnson	Pilot Episode	UK	E-mail	4	1
2	Jane	Smith	Pilot Episode	CA	E-mail	4	1
3	John	Doe	Pilot Episode	US	E-mail	4	1

Given the limited data we used, all contestants in the Pilot Episode got 4 each and thus ranked highest. However, given a complete database was used, the query would still run and print the required result.

Q4: Identifying which contestants have not participated in any events.

We do this by performing a left join on the Applicants and Tasks_Contestants table. This is because if an individual that was flagged as a finalist does not appear in the Contestant_Tasks table, then that contestant did not participate in any tasks, events or episode.

Visual Query:

	Function	Column	Alias	Object	Filter
<input checked="" type="checkbox"/>		Contestant_ID		"RDBMS.db"."Applicant"	
<input checked="" type="checkbox"/>		Firstname		"RDBMS.db"."Applicant"	
<input checked="" type="checkbox"/>		Lastname		"RDBMS.db"."Applicant"	
<input checked="" type="checkbox"/>		Finalist		"RDBMS.db"."Applicant"	"Applicant"."Finalist" = "Yes"
<input type="checkbox"/>		Task ID		"RDBMS.db"."Task_Contestants"	"Task_Contestants"."Task ID" IS NULL

SQL Query:

```
SELECT "Applicant"."Contestant_ID",  
       "Applicant"."Firstname",  
       "Applicant"."Lastname",  
       -
```

```

"Applicant"."Finalist"

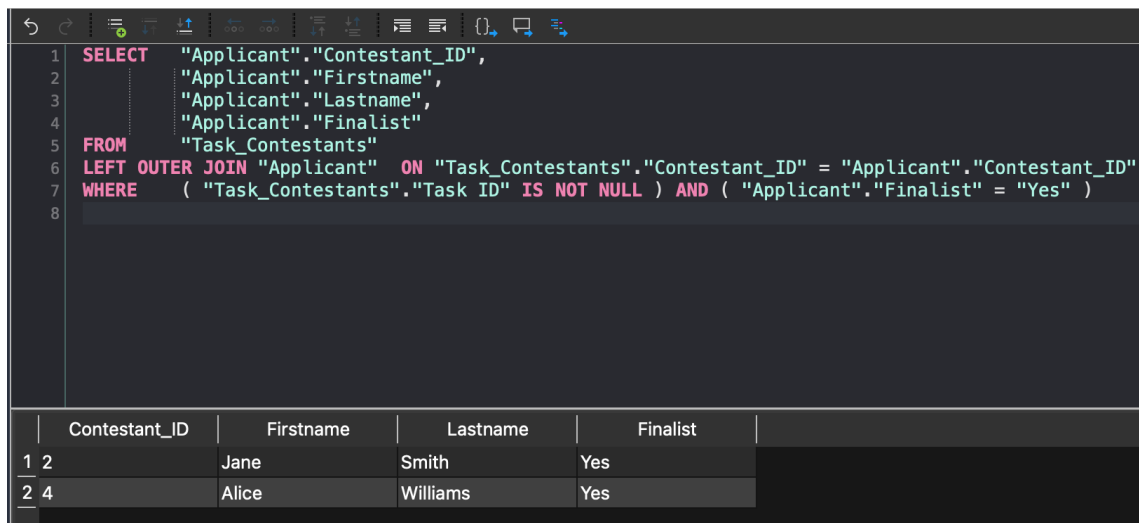
FROM "Task_Contestants"

LEFT OUTER JOIN "Applicant" ON "Task_Contestants"."Contestant_ID" =
"Applicant"."Contestant_ID"

WHERE ( "Task_Contestants"."Task ID" IS NOT NULL ) and ( "Applicant"."Finalist" =
"Yes" )

```

Query & Result:



```

1 SELECT "Applicant"."Contestant_ID",
2        "Applicant"."Firstname",
3        "Applicant"."Lastname",
4        "Applicant"."Finalist"
5 FROM   "Task_Contestants"
6 LEFT OUTER JOIN "Applicant" ON "Task_Contestants"."Contestant_ID" = "Applicant"."Contestant_ID"
7 WHERE  ( "Task_Contestants"."Task ID" IS NOT NULL ) AND ( "Applicant"."Finalist" = "Yes" )
8

```

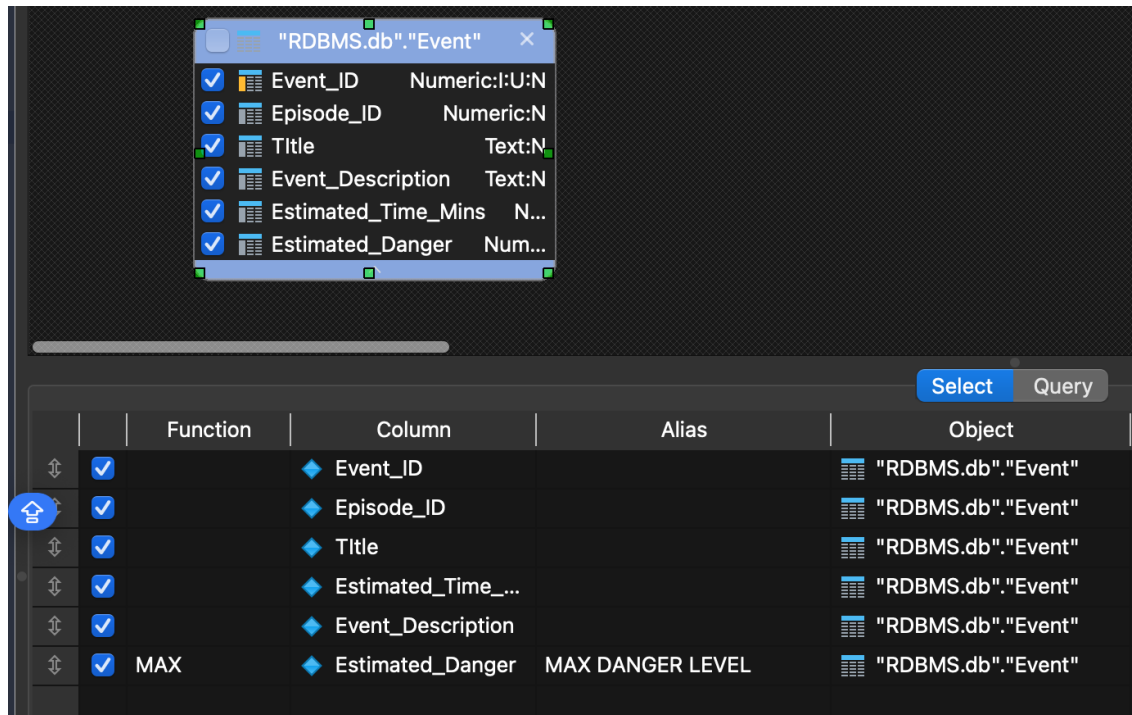
	Contestant_ID	Firstname	Lastname	Finalist
1 2		Jane	Smith	Yes
2 4		Alice	Williams	Yes

In the result above, we see there are 2 applicants that were marked as Finalists but were not assigned to any task. These are Jane Smith and Alice Williams.

Q5: What is the highest estimated danger level for any event?

The estimated danger for any event is available in the Events table. Here we perform a simple Aggregate query on the Estimated_Danger attribute to get the desired result.

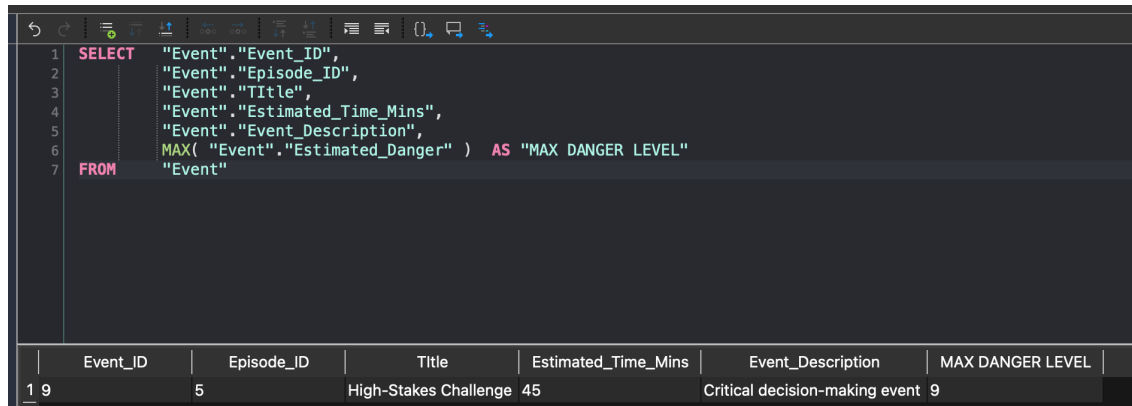
Visual Query:



SQL Query:

```
SELECT "Event"."Event_ID",  
       "Event"."Episode_ID",  
       "Event"."Title",  
       "Event"."Estimated_Time_Mins",  
       "Event"."Event_Description",  
       MAX( "Event"."Estimated_Danger" ) AS "MAX DANGER LEVEL"  
FROM   "Event"
```


Query & Result:



```
1 SELECT "Event"."Event_ID",
2        "Event"."Episode_ID",
3        "Event"."Title",
4        "Event"."Estimated_Time_Mins",
5        "Event"."Event_Description",
6        MAX( "Event"."Estimated_Danger" ) AS "MAX DANGER LEVEL"
7 FROM "Event"
```

	Event_ID	Episode_ID	Title	Estimated_Time_Mins	Event_Description	MAX DANGER LEVEL
1	9	5	High-Stakes Challenge	45	Critical decision-making event	9

The highest danger level in our data is 9 which is for the High Stakes Challenge event in Episode 5 of the reality TV show.

Q6: Show a relational expression for any one query

For showing the relational expression, here we are going through the Q1 SQL query to perform the relational expression on that. In here we are focusing on performing the all necessary join. Now here we are performing the inner joins as series of pairwise join operations:

$R1 = \text{Ratings} \bowtie \text{Ratings.Contestant_ID} = \text{Task_Contestants.Contestant_ID}$ first and then we will apply the aggregation.

Task_Contestants

$R2 = R1 \bowtie R1.Task_ID = \text{Tasks.Task_ID}$ Tasks

$R3 = R2 \bowtie R2.Event_ID = \text{Event.Event_ID}$ Event

2. Projecting the necessary attributes for final result which includes event titles and ratings:

$R4 = \pi_{\text{Event.Title}, \text{Ratings.Rating}} (R3)$

3. Here we are applying the group by operation with the aggregation of average of ratings:

$\text{Result} = \gamma_{\text{Event.Title}; \text{AVG}(\text{Ratings.Rating})} (R4)$



ROLE OF MEMBERS

Name of Member & Student ID	Description of Activities
Harmeet Singh (ID# 9809470994)	<ul style="list-style-type: none">- Reviewed and recommended changes for the Normalized tables drafted by other members- Designed the ERD on LucidChart and Defined the relationship cardinalities using crow's foot notations- Assisted with creating the table generation SQL queries- Assisted with interpreting the questions for the SQL queries- Reviewed and updated the constraints for the data types- Explored how to create local databases on Valentina- Created draft report structure (word and slides)- Helped conceptualize the story telling for our presentation and report- Coordinated team activities (project manager role)
Purwa Mugdiya (ID# 989468918)	<ul style="list-style-type: none">- Created the Normalized list of Tables For Part 1, 2 and 3.- Created Relationships between the tables.- Reviewed and Assisted in ERD and Database creation.
Aman Singh (#ID 989467252)	<ul style="list-style-type: none">- Creation of 3rd part normalization (Voting)- Helped in creating the tables in ERD- Inputs to the PPT- Input to the word document
Vamshi Krishna Perabathula (ID# 989472817)	<ul style="list-style-type: none">- Created Normalization tables with my group mate Purva for part 1,2,3.- Created Relationship and integrity for tables 1,2,3.- Wrote down the SQL queries for given question of 5 and with relational expression with my group mate Prudhvi
Sai Pridhvi Pinninti (# 989469046)	<ul style="list-style-type: none">- Created Tables & Sample Data in SQL- Code for SQL Queries for Q1,Q2,Q3,Q4,Q5.
Meiyi Zhou (ID# 989470002)	<p>In this project, Pavan is my partner to complete Part 3- Voting, we have conversations with him and create the database. Then, I joined every meeting and attended every discussion with my group members. I contributed the normalized list of tables for figure 4, with Pavan. In the preparation for the presentation, I prepared explanations for relationships and spoke the explanation for ERD when we were at the presentation.</p>
Pavan Kumar Pekala (ID# 989468153)	<ul style="list-style-type: none">- Created the constraints table and worked on explanations.- Worked on the voting tables.



REFERENCES

Some of the references that enabled the execution of this project are as follows:

- Video on '[Lucid Chart - Graphically Architecting/Creating SQL Database Tables - 2022](#)'
- Extraction of random strings and characters for our sample data using generative AI (ChatGPT)

APPENDIX 1 - Table Creation

SQL Query

Below is the SQL text for the tables created in Valentina Database for this project, the key constraints inferred from the project description are highlighted in green:

```
CREATE TABLE `Locations` (  
  `Location_ID` Number(10) UNIQUE,  
  `Country_Code` Text,  
  `City` Text,  
  `State` Text,  
  `Country` Text,  
  `Continent` Text,  
  PRIMARY KEY (`Location_ID`)  
);
```

```
CREATE TABLE `Applicant` (  
  `Contestant_ID` Number(10) UNIQUE,  
  `Firstname` Text,  
  `Lastname` Text,  
  `Email_address` Text,  
  `Gender` CHAR(1) CHECK (GENDER IN ('M', 'F'))  
  ,  
  `Date_of_birth` Date,  
  `Personal_Phone` Number,  
  `Business_Phone` Number,  
  `Street_Address` Text,  
  `Postal_Code` Text,
```

```
  `Location_ID` Number(10),  
  `Candidate_Essay` Text,  
  `Video_Link` Text,  
  `Photo` Blob,  
  `Finalist` Text ,  
  PRIMARY KEY (`Contestant_ID`),  
  FOREIGN KEY (`Location_ID`) REFERENCES  
  `Locations`(`Location_ID`)  
);
```

-

```
CREATE TABLE `Job_Details` (
  `Job_ID` Number(10) UNIQUE,
  `Job_Role` Text,
  `Job_Description` Text,
  `Start_Date` Date,
  `End_Date` Date,
  `Contestant_ID` Number,
  PRIMARY KEY (`Job_ID`),
  FOREIGN KEY (`Contestant_ID`) REFERENCES
  `Applicant`(`Contestant_ID`)
);
```

```
CREATE TABLE `Finalist_Background` (
  `Contestant_ID` Number(10),
  `National_ID` VARCHAR(10) UNIQUE,
  `Religion` Text,
  `Appearance_Rating` NUMBER CHECK
  (Appearance_Rating BETWEEN 1 AND 10) ,
  `Strength_Rating` Number CHECK
  (Strength_Rating BETWEEN 1 AND 10),
  PRIMARY KEY (`National_ID`)
);
```

-

```
CREATE TABLE `Police_Records` (
  `Record_ID` Number(10) UNIQUE,
  `National_ID` Number(10),
  `Date` Date,
  `Category` CHAR CHECK (Category IN
  ('Violence', 'Drugs', 'Traffic Violations')) ,
  `Record_Description` Text,
  `Outcome` Text,
  PRIMARY KEY (`Record_ID`),
  FOREIGN KEY (`National_ID`) REFERENCES
  `Finalist_Background`(`National_ID`)
);
```

```
CREATE TABLE `Employer_History` (
  `Employer_ID` Number(10) UNIQUE,
  `Contestant_ID` Number(10),
  `Employer_Name` Text,
  `Phone` Number,
  `Comments` Text,
  PRIMARY KEY (`Employer_ID`),
  FOREIGN KEY (`Contestant_ID`) REFERENCES
  `Applicant`(`Contestant_ID`)
);
```

```
CREATE TABLE `Education` (
  `Education_ID` Number(10) UNIQUE,
  `Contestant_ID` Number(10),
  `School_Name` Text,
  `Degree` Text,
  `Contact` VARCHAR,
  `Comments` Text,
  PRIMARY KEY (`Education_ID`),
  FOREIGN KEY (`Contestant_ID`) REFERENCES
  `Applicant`(`Contestant_ID`)
);
```

```
CREATE TABLE `Episode` (
  `Episode_ID` Number(3) UNIQUE,
  `Episode_Title` Text,
  `Episode_Air_Date` Date,
  `Director` Text,
  `Producer` Text,
  PRIMARY KEY (`Episode_ID`)
);
```

```

CREATE TABLE `Event` (
    `Event_ID` Number(4) UNIQUE,
    `Episode_ID` Number(3),
    `Title` Text,
    `Event_Description` Text,
    `Estimated_Time_Mins` Number CHECK
(Estimated_Time_Mins BETWEEN 1 AND 60),
    `Estimated_Danger` Number CHECK
(Estimated_Danger BETWEEN 1 AND 10),
    PRIMARY KEY (`Event_ID`),
    FOREIGN KEY (`Episode_ID`) REFERENCES
`Episode`(`Episode_ID`)
);

```

```

CREATE TABLE `Task_Contestants` (
    `Contestant_ID` Number(10),
    `Task_ID` Number(5) UNIQUE,
    `Task_Result` Text,
    `Points` Number CHECK (Points BETWEEN -10
AND 10)
);

```

-

```

CREATE TABLE `Tasks` (
    `Event_ID` Number(4),
    `Task_ID` Number(5),
    `Task_Name` Text,
    `Task_Prize` Text,
    `Group_Task` CHAR(1) CHECK (Group_Task IN
('Y', 'N')),
    PRIMARY KEY (`Task_ID`),
    FOREIGN KEY (`Event_ID`) REFERENCES
`Event`(`Event_ID`)
);

```

```

CREATE TABLE `Event_Direction` (
    `Event_ID` Number(4),
    `Sequence_ID` Number(2),
    `Sequence_Description` Text,
    `Cameras` NUMBER,
    `Estimated_Time` Date,
    PRIMARY KEY (`Event_ID`, `Sequence_ID`),
    FOREIGN KEY (`Event_ID`) REFERENCES
`Event`(`Event_ID`)

```

```

);

CREATE TABLE `Ratings` (
    `Rating_ID` Number(10) UNIQUE,
    `Contestant_ID` Number(10),
    `Rating` Number CHECK (Rating BETWEEN 1
and 5) ,
    `Rated_By` CHAR CHECK (Rated_By IN
('Producer', 'Director')) ,
    PRIMARY KEY (`Rating_ID`),
    FOREIGN KEY (`Contestant_ID`) REFERENCES
`Applicant`(`Contestant_ID`)
);

```

```

CREATE TABLE `Vote_Method` (
    `Method_ID` Number(1),
    `Method_Type` CHAR CHECK (Method_Type IN
('Telephone', 'Messages', 'Website', 'E-mail'))
UNIQUE,
    PRIMARY KEY (`Method_ID`)
);

```

```
CREATE TABLE `Medications` (
  `Medication_ID` Number(3),
  `Medication_Name` Text,
  PRIMARY KEY (`Medication_ID`)
);
```

```
CREATE TABLE `Health_Details` (
  `Contestant_ID` Number(10),
  `Medication_ID` Number(3),
  `Reason` Text,
  PRIMARY KEY (`Contestant_ID`,
  `Medication_ID`),
  FOREIGN KEY (`Medication_ID`) REFERENCES
  `Medications`(`Medication_ID`),
  FOREIGN KEY (`Contestant_ID`) REFERENCES
  `Applicant`(`Contestant_ID`)
);
```

```
CREATE TABLE `Votes` (
  `Vote_ID` Number(10),
  `Episode_ID` Number(3),
  `Contestant_ID` Number(10),
  `Location_ID` Number(10),
  `Method_ID` Number(1),
  PRIMARY KEY (`Vote_ID`),
  FOREIGN KEY (`Episode_ID`) REFERENCES
  `Episode`(`Episode_ID`),
  FOREIGN KEY (`Contestant_ID`) REFERENCES
  `Applicant`(`Contestant_ID`),
  FOREIGN KEY (`Location_ID`) REFERENCES
  `Locations`(`Location_ID`)
);
```

APPENDIX 2 - Link to Database File

The link to our created database file for this project is [here](#).

APPENDIX 3 - SQL

Text for Creating Sample Data

```
INSERT INTO Locations (Location_ID, Country_Code, City, State, Country, Continent)
```

```
VALUES
```

```
(1, 'US', 'New York', 'NY', 'United States', 'North America'),
```

```
(2, 'CA', 'Toronto', 'ON', 'Canada', 'North America'),
```

```
(3, 'UK', 'London', '', 'United Kingdom', 'Europe'),
```

```
(4, 'FR', 'Paris', '', 'France', 'Europe'),
```

```
(5, 'DE', 'Berlin', '', 'Germany', 'Europe'),
```

```
(6, 'JP', 'Tokyo', '', 'Japan', 'Asia'),
```

```
(7, 'BR', 'Sao Paulo', 'SP', 'Brazil', 'South America'),
```

```
(8, 'IN', 'Mumbai', 'MH', 'India', 'Asia'),
```

```
(9, 'ZA', 'Cape Town', 'WC', 'South Africa', 'Africa'),
```

```
(10, 'AU', 'Sydney', 'NSW', 'Australia', 'Oceania');
```

```
INSERT INTO `Applicant` (
```

```
`Contestant_ID`,
```

```
`Firstname`,
```

```
`Lastname`,
```

```
`Email_address`,
```

```
`Gender`,
```

```
`Date_of_birth`,
```

```
`Personal_Phone`,
```

```
`Business_Phone`,
```

```
-
```

```

`Street_Address`,

`Postal_Code`,

`Location_ID`,

`Candidate_Essay`,

`Video_Link`,

`Photo`,

`Finalist`

) VALUES

(1, 'John', 'Doe', 'john.doe@example.com', 'M', '1990-01-15', 1234567890, 9876543210, '123 Main St', '12345', 1, 'Essay
text 1', 'https://video-link1.com', NULL, 'No'),

(2, 'Jane', 'Smith', 'jane.smith@example.com', 'F', '1988-05-20', 9876543210, 1234567890, '456 Oak St', '56789', 2, 'Essay
text 2', 'https://video-link2.com', NULL, 'Yes'),

(3, 'Bob', 'Johnson', 'bob.johnson@example.com', 'M', '1995-09-10', 5555555555, 6666666666, '789 Pine St', '67890', 3,
'Essay text 3', 'https://video-link3.com', NULL, 'No'),

(4, 'Alice', 'Williams', 'alice.williams@example.com', 'F', '1987-03-25', 1111111111, 2222222222, '101 Elm St', '54321', 4,
'Essay text 4', 'https://video-link4.com', NULL, 'Yes'),

(5, 'Charlie', 'Brown', 'charlie.brown@example.com', 'M', '1992-11-30', 9999999999, 8888888888, '202 Maple St', '98765',
5, 'Essay text 5', 'https://video-link5.com', NULL, 'No'),

(6, 'Eva', 'Garcia', 'eva.garcia@example.com', 'F', '1998-07-15', 3333333333, 4444444444, '303 Cedar St', '13579', 6, 'Essay
text 6', 'https://video-link6.com', NULL, 'Yes'),

(7, 'David', 'Lee', 'david.lee@example.com', 'M', '1980-04-05', 7777777777, 6666666666, '404 Birch St', '24680', 7, 'Essay
text 7', 'https://video-link7.com', NULL, 'No'),

(8, 'Grace', 'Turner', 'grace.turner@example.com', 'F', '1985-12-20', 2222222222, 1111111111, '505 Walnut St', '97531', 8,
'Essay text 8', 'https://video-link8.com', NULL, 'Yes'),

(9, 'Frank', 'Miller', 'frank.miller@example.com', 'M', '1994-06-10', 8888888888, 3333333333, '606 Oak St', '46895', 9,
'Essay text 9', 'https://video-link9.com', NULL, 'No'),

(10, 'Olivia', 'Davis', 'olivia.davis@example.com', 'F', '1993-02-28', 4444444444, 9999999999, '707 Pine St', '75309', 10,
'Essay text 10', 'https://video-link10.com', NULL, 'Yes');

INSERT INTO `Job_Details` (

`Job_ID`,

`Job_Role`,

`Job_Description`,

`Start_Date`,

`End_Date`,

`Contestant_ID`

-

```

) VALUES

(1, 'Software Engineer', 'Developing web applications', '2022-01-01', '2023-01-01', 1),
(2, 'Marketing Specialist', 'Creating and implementing marketing campaigns', '2021-05-15', '2022-06-30', 2),
(3, 'Data Analyst', 'Analyzing and interpreting data', '2022-03-10', '2023-03-10', 3),
(4, 'Project Manager', 'Leading project teams', '2021-02-20', '2022-04-30', 4),
(5, 'UX/UI Designer', 'Designing user interfaces and experiences', '2022-09-01', '2023-09-01', 5),
(6, 'Human Resources Specialist', 'Managing HR processes', '2021-11-15', '2023-01-15', 6),
(7, 'Financial Analyst', 'Analyzing financial data', '2021-04-05', '2022-05-15', 7),
(8, 'Product Manager', 'Overseeing product development', '2022-07-01', '2023-07-01', 8),
(9, 'IT Support Specialist', 'Providing technical support', '2022-05-20', '2023-05-20', 9),
(10, 'Sales Representative', 'Selling products and services', '2021-08-10', '2022-09-30', 10);

INSERT INTO `Finalist_Background` (

`Contestant_ID`,
`National_ID`,
`Religion`,
`Appearance_Rating`,
`Strength_Rating`

) VALUES

(1, 'ABC123456', 'Christian', 8, 7),
(2, 'XYZ789012', 'Muslim', 9, 8),
(3, '123ABC456', 'Hindu', 7, 9),
(4, '456XYZ789', 'Buddhist', 6, 6),
(5, '789123ABC', 'Jewish', 10, 7),
(6, 'DEF456GHI', 'Sikh', 8, 8),
(7, 'JKL789MNO', 'Atheist', 5, 4),
(8, 'PQR123STU', 'Agnostic', 9, 9),
(9, 'VWX456YZA', 'Bahá'í Faith', 7, 6),
(10, 'BCD789EFG', 'Other', 6, 5);

INSERT INTO `Police_Records` (

`Record_ID`,
-

```

`National_ID`,

`Date`,

`Category`,

`Record_ Description`,

`Outcome`

) VALUES

(1, 'ABC123456', '2022-01-05', 'Violence', 'Assault in public place', 'Arrested'),

(2, 'XYZ789012', '2021-08-20', 'Drugs', 'Possession of controlled substance', 'Community service'),

(3, '123ABC456', '2022-03-15', 'Traffic Violations', 'Speeding ticket', 'Fine paid'),

(4, '456XYZ789', '2021-11-10', 'Violence', 'Domestic dispute', 'Warning issued'),

(5, '789123ABC', '2022-05-02', 'Traffic Violations', 'Running a red light', 'Traffic school attended'),

(6, 'DEF456GHI', '2021-12-28', 'Drugs', 'Drug trafficking', 'Probation'),

(7, 'JKL789MNO', '2022-02-08', 'Violence', 'Bar fight', 'Released without charges'),

(8, 'PQR123STU', '2021-10-17', 'Traffic Violations', 'Driving under the influence', 'License suspended'),

(9, 'VWX456YZA', '2022-04-30', 'Drugs', 'Possession of drug paraphernalia', 'Community service'),

(10, 'BCD789EFG', '2022-07-12', 'Traffic Violations', 'Reckless driving', 'Fine paid');

```

```

INSERT INTO `Employer_History` (

`Employer_ID`,

`Contestant_ID`,

`Employer_Name`,

`Phone`,

`Comments`

) VALUES

(1, 1, 'ABC Company', 1234567890, 'Positive work experience'),

(2, 2, 'XYZ Corporation', 9876543210, 'Excellent communication skills'),

(3, 3, '123 Industries', 5555555555, 'Team player'),

(4, 4, '456 Enterprises', 1111111111, 'Dependable and reliable'),

(5, 5, '789 Ltd.', 9999999999, 'Creative problem solver'),

(6, 6, 'DEF Group', 3333333333, 'Adaptable and quick learner'),

(7, 7, 'JKL Solutions', 7777777777, 'Attention to detail'),

```

-

```
(8, 8, 'PQR Innovations', 222222222, 'Leadership qualities'),
(9, 9, 'VWX Co.', 8888888888, 'Excellent organizational skills'),
(10, 10, 'BCD Tech', 4444444444, 'Effective multitasker');
```

```
INSERT INTO `Education` (
```

```
`Education_ID`,
`Contestant_ID`,
`School_Name`,
`Degree`,
`Contact`,
`Comments`
```

```
) VALUES
```

```
(1, 1, 'ABC University', 'Bachelor of Science in Computer Science', '123-456-7890', 'Graduated with honors'),
(2, 2, 'XYZ College', 'Bachelor of Arts in Marketing', '987-654-3210', 'Recipient of marketing excellence award'),
(3, 3, '123 Institute of Technology', 'Master of Science in Data Science', '555-555-5555', 'Thesis on data analytics'),
(4, 4, '456 Business School', 'Master of Business Administration', '111-111-1111', 'Focus on strategic management'),
(5, 5, '789 Design Academy', 'Bachelor of Fine Arts in Graphic Design', '999-999-9999', 'Portfolio featured in local art exhibition'),
(6, 6, 'DEF Technical Institute', 'Associate Degree in Information Technology', '333-333-3333', 'Certified in network administration'),
(7, 7, 'JKL College of Science', 'Bachelor of Science in Biology', '777-777-7777', 'Research published in scientific journal'),
(8, 8, 'PQR School of Engineering', 'Master of Science in Electrical Engineering', '222-222-2222', 'Received research grant for renewable energy project'),
(9, 9, 'VWX Academy of Business', 'Bachelor of Business Administration', '888-888-8888', 'Internship with Fortune 500 company'),
(10, 10, 'BCD Institute of Technology', 'Bachelor of Science in Computer Engineering', '444-444-4444', 'Developed award-winning software project');
```

```
INSERT INTO `Episode` (
```

```
`Episode_ID`,
`Episode_Title`,
`Episode_Air_Date`,
`Director`,
```

```
-
```

`Producer`

) VALUES

(1, 'Pilot Episode', '2022-01-10', 'John Director', 'Jane Producer'),
(2, 'The Adventure Begins', '2022-01-17', 'Alice Director', 'Bob Producer'),
(3, 'Mystery Unfolds', '2022-01-24', 'Charlie Director', 'Diana Producer'),
(4, 'Love and Betrayal', '2022-01-31', 'Eva Director', 'Frank Producer'),
(5, 'In the Spotlight', '2022-02-07', 'Grace Director', 'George Producer'),
(6, 'The Turning Point', '2022-02-14', 'Holly Director', 'Henry Producer'),
(7, 'A Twist of Fate', '2022-02-21', 'Ian Director', 'Ivy Producer'),
(8, 'Epic Showdown', '2022-02-28', 'Jake Director', 'Jill Producer'),
(9, 'Closure and Redemption', '2022-03-07', 'Kevin Director', 'Kelly Producer'),
(10, 'Grand Finale', '2022-03-14', 'Liam Director', 'Laura Producer');

INSERT INTO `Event` (

`Event_ID`,

`Episode_ID`,

`Title`,

`Event_Description`,

`Estimated_Time_Mins`,

`Estimated_Danger`

) VALUES

(1, 1, 'Opening Ceremony', 'Introduction to the show', 10, 1),
(2, 1, 'First Challenge', 'Physical obstacle course', 30, 5),
(3, 1, 'Team Building Activity', 'Collaborative puzzle solving', 20, 3),
(4, 2, 'Mysterious Encounter', 'Unexpected plot twist', 15, 7),
(5, 2, 'Character Development', 'Emotional and personal stories', 25, 2),
(6, 3, 'Clue Discovery', 'Investigation and clue finding', 40, 6),
(7, 3, 'Plot Unraveling', 'Revealing the central mystery', 35, 8),
(8, 4, 'Love Triangle', 'Romantic subplot intensifies', 20, 4),
(9, 5, 'High-Stakes Challenge', 'Critical decision-making event', 45, 9),
(10, 5, 'Unexpected Alliance', 'Characters join forces', 30, 3);

-

```

INSERT INTO `Task_Contestants` (
    `Contestant_ID`,
    `Task ID`,
    `Task_Result`,
    `Points`
) VALUES
    (1, 1, 'Completed', 5),
    (2, 2, 'Completed with bonus', 10),
    (3, 3, 'Failed', -5),
    (3, 4, 'Completed', 7),
    (4, 5, 'Completed with penalty', 4);

```

```

INSERT INTO `Tasks` (
    `Event_ID`,
    `Task_ID`,
    `Task_Name`,
    `Task_Prize`,
    `Group_Task`
) VALUES
    (1, 1, 'Obstacle Course', 'Bonus Points', 'N'),
    (1, 2, 'Puzzle Challenge', 'Immunity Idol', 'N'),
    (2, 3, 'Mystery Task', 'Extra Screen Time', 'N'),
    (2, 4, 'Emotional Test', 'Character Development Boost', 'N'),
    (3, 5, 'Investigation Challenge', 'Clue Advantage', 'N'),
    (3, 6, 'Revelation Task', 'Plot Twist Opportunity', 'N'),
    (4, 7, 'Love Triangle Test', 'Romantic Subplot Bonus', 'N'),
    (5, 8, 'Critical Decision Task', 'High-Stakes Advantage', 'N'),
    (5, 9, 'Alliance Challenge', 'Teamwork Bonus', 'Y'),
    (5, 10, 'Character Development Task', 'Personal Growth Boost', 'N');

```

-

```

INSERT INTO `Event_Direction` (
    `Event_ID`,
    `Sequence_ID`,
    `Sequence_Description`,
    `Cameras`,
    `Estimated_Time`
) VALUES
    (1, 1, 'Introduction Shots', 3, '30'),
    (1, 2, 'Obstacle Course Start', 5, '30'),
    (1, 3, 'Puzzle Challenge Setup', 2, '30'),
    (2, 1, 'Plot Twist Scene', 4, '15'),
    (2, 2, 'Character Development Interviews', 3, '15'),
    (3, 1, 'Clue Discovery Shots', 3, '40'),
    (3, 2, 'Plot Unraveling Scene', 4, '40'),
    (4, 1, 'Love Triangle Discussion', 2, '10'),
    (5, 1, 'Critical Decision Setup', 4, '35'),
    (5, 2, 'Alliance Challenge Briefing', 3, '35');

```

```

INSERT INTO `Ratings` (
    `Rating_ID`,
    `Contestant_ID`,
    `Rating`,
    `Rated_By`
) VALUES

```

```

    (1, 1, 4, 'Producer'),
    (2, 1, 5, 'Director'),
    (3, 2, 5, 'Producer'),
    (4, 2, 3, 'Director'),
    (5, 3, 2, 'Producer'),
    (6, 3, 3, 'Director'),
    (7, 4, 3, 'Producer'),

```

-


```
(8, 4, 4, 'Director'),  
(9, 5, 4, 'Producer'),  
(10, 5, 4, 'Director');
```

```
INSERT INTO `Vote_Method` (  
    `Method_ID`,  
    `Method_Type`  
) VALUES  
(1, 'Telephone'),  
(2, 'Messages'),  
(3, 'Website'),  
(4, 'E-mail');
```

```
INSERT INTO `Medications` (  
    `Medication_ID`,  
    `Medication_Name`  
) VALUES  
(1, 'Aspirin'),  
(2, 'Ibuprofen'),  
(3, 'Acetaminophen'),  
(4, 'Lisinopril'),  
(5, 'Simvastatin'),  
(6, 'Levothyroxine'),  
(7, 'Metformin'),  
(8, 'Atorvastatin'),  
(9, 'Amlodipine'),  
(10, 'Omeprazole');
```

```
INSERT INTO `Health_Details` (  
    `Contestant_ID`,  
    `Medication_ID`,  
    `Reason`  
) VALUES
```

-

```

(1, 1, 'Headache'),
(1, 2, 'Muscle pain'),
(2, 3, 'Fever'),
(2, 4, 'High blood pressure'),
(3, 5, 'Cholesterol management'),
(3, 6, 'Thyroid disorder'),
(4, 7, 'Diabetes'),
(4, 8, 'Cholesterol management'),
(5, 9, 'High blood pressure'),
(5, 10, 'Acid reflux');
INSERT INTO `Votes` (
    `Vote_ID`,
    `Episode_ID`,
    `Contestant_ID`,
    `Location_ID`,
    `Method_ID`
) VALUES
(1, 1, 1, 1, 1),
(2, 1, 2, 2, 2),
(3, 1, 3, 3, 3),
(4, 2, 4, 4, 4),
(5, 2, 5, 5, 1),
(6, 3, 6, 6, 2),
(7, 3, 7, 7, 3),
(8, 4, 8, 8, 4),
(9, 4, 9, 9, 1),
(10, 5, 10, 10, 2);

```