South Park Database

Database Design Project

Mike McGinnis

14

Table of Contents:

Executive Summary

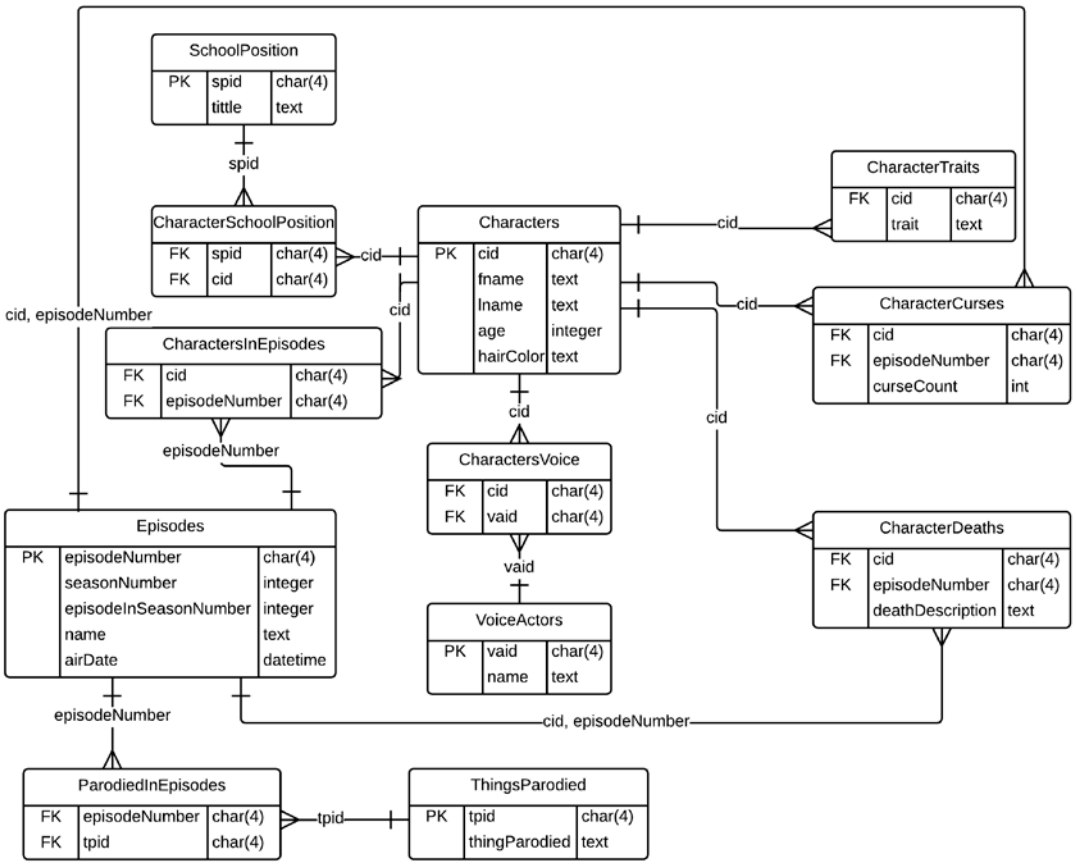
Overview

South Park is a TV show that airs on Comedy Central. It has been on the air since 1997 and follows the adventures of four ten year old boys in the fictional town of South Park, Colorado. The show's creators are Trey Parker, and Matt Stone. The two of them voice the majority of the male characters on the show. The show tackles current issues by parodying them in way that is both funny and thought provoking. There is no issue the show won't tackle from religion, to politics. Keeping track of character information, what has already been parodied, and number of times Kenny has been killed (running gag), can lead to the creation of funnier episodes and ensures you don't accidently parody something twice unintentionally.

Objectives

The purpose of this paper is to provide both fun and useful information about the show South Park. Since the boys are only ten, it is funny how much they curse, so there is a table to keep track of curses per character per episode. There is also a running gag that the character Kenny dies a lot so we'll keep track of what episodes he dies in. To improve the show we will keep track of everything that has been parodied so far, this way if someone comes up with an idea for an episode they can quickly check to make sure they haven't parodied it already.

Entity Relationship Diagram



Tables

Characters

Purpose: This table is used to store information about the characters who appear in the show such as their name, age, and hair color.

Create Statement

```
CREATE TABLE characters (
    cid           char(4) unique,
    fname         text,
    lname         text,
    age           integer,
    hairColor     text,
    primary key(cid)
);
```

Functional Dependencies

cid → fname, lname, age, hairColor

Sample Data

CID	FName	LName	Age	HairColor
C001	Stan	Marsh	10	Black
C002	Kyle	Broflovski	10	Red
C003	Kenny	McCormick	10	Blond
C004	Eric	Cartman	10	Brown
C005	Herbert	Garrison	41	Gray
C006	Randy	Marsh	45	Black
C007	Butters	Stotch	10	Blond
C008	Token	Black	10	Black
C009	Jerome(Chef)	McElroy	38	Black

VoiceActors

Purpose: This table is used to store information about the people who provide the voices for the characters. It stores their name and their ID as a voice actor.

Create Statement

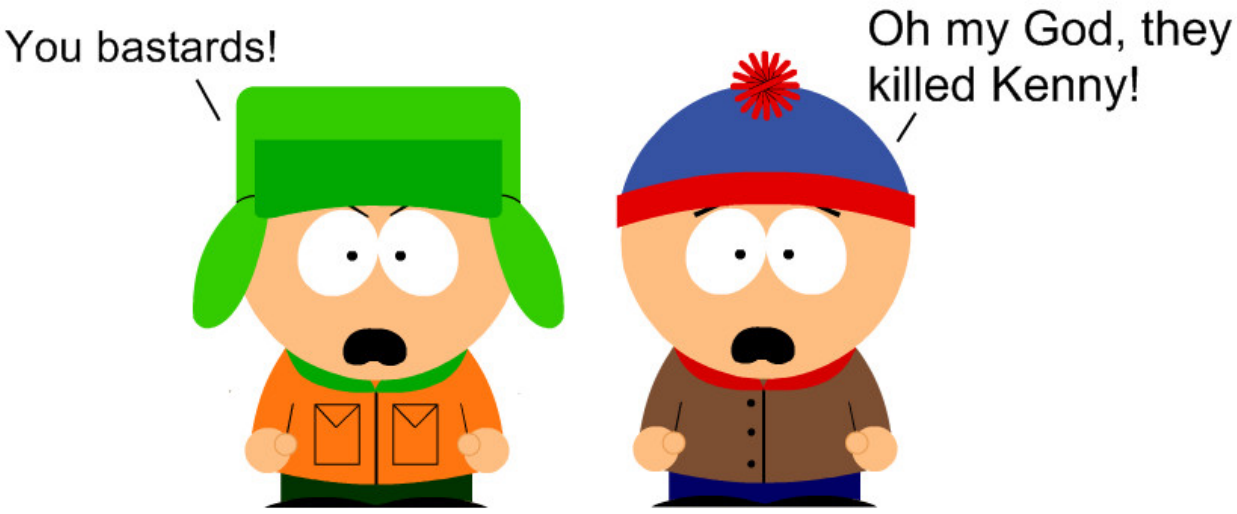
```
CREATE TABLE voiceActors (
    vaid        char(4) unique,
    name        text,
    primary key(vaid)
);
```

Functional Dependencies

vaid → name

Sample Data

VAID	Name
V001	Trey Parker
V002	Matt Stone
V003	Adrien Beard
V004	Issac Hayes



CharactersVoice

Purpose: This table is used to store information about which characters the voice actors provide their voices for

Create Statement

```
CREATE TABLE charactersVoice (
    cid         char(4) unique,
    vaid        char(4)
);
```

Functional Dependencies

cid → vaid

Sample Data

CID	VAID
C001	V001
C002	V002
C003	V002
C004	V001
C005	V001
C006	V001
C007	V002
C008	V003
C009	V004

CharacterTraits

Purpose: This table is used to store information about what defining character trait each character has

Create Statement

```
CREATE TABLE characterTraits (
    cid         char(4) unique,
    trait       text
);
```

Functional Dependencies

cid → trait

Sample Data

CID	Character Trait
C001	Leader
C002	Jew
C003	Poor
C004	Fat Ass
C005	Gay
C006	Lorde
C007	Constantly Grounded
C008	Only Black Kid
C009	Chef

Episodes

Purpose: This table is used to store information the episodes the show has aired. It keeps track of the episode number, which season it was it, what episode number it was in the season, the name of the episode and the original air date.

Create Statement

```
CREATE TABLE episodes (
    episode#            char(4),
    season#             integer,
    episodeInSeason#    integer,
    name                text,
    airDate             dateTime,
    primary key(episode#)
);
```

Functional Dependencies

episode# → season#, episodeInSeason#, name, airdate

Sample Data

Episode#	Season#	episodeInSeason#	Name	airDate
66	5	1	It Hits the Fan	06-20-2001
137	9	12	Trapped in the Closet	11-16-2005
161	11	8	Le Petit Tourette	10-03-2007
215	15	6	City Sushi	06-01-2011

ThingsParodied

Purpose: This table is used to store the things that have been parodied in South Park. South Park has parodied everything from celebrities, religions, politics, and mental disorders and we need a place to store all of them.

Create Statement

```
CREATE TABLE thingsParodied (
    tpid                char(4),
    thingParodied       text,
    primary key(tpid)
);
```

Functional Dependencies

tpid → thingParodied

Sample Data

Tpid	ThingParodied
P001	Chicago Hope
P002	Tom Cruise
P003	Scientology
P004	R. Kelly
P005	Tourettes
P006	Asian Accents
P007	Split Personality Disorder
P008	Horror Movies
P009	Psycho

ParodiedInEpisodes

Purpose: This table is used to store information about what was parodied in each episode.

Create Statement

```
CREATE TABLE parodiedInEpisodes (
    episode#            char(4),
    tpid                char(4)
);
```

Functional Dependencies

tpid→ Episode#

Sample Data

Tpid	Episode#
P001	66
P002	137
P003	137
P004	137
P005	161
P006	215
P007	215
P008	215
P009	215

CharactersInEpisodes

Purpose: This table is used to store what characters appeared in each episode.

Create Statement

```
CREATE TABLE charactersInEpisodes (
    cid             char(4),
    episode#        char(4)
);
```

Functional Dependencies

cid, episode# →

Sample Data

Cid	Episode#
C001	66
C002	66
C003	66
C004	66
C005	66
C007	66
C001	137
C002	137
C003	137
C004	137
C006	137
C007	137
C008	137
C001	161
C002	161
C003	161
C004	161
C005	161
C007	161
C001	215
C002	215
C003	215
C004	215
C007	215
C008	215

CharacterCurses

Purpose: This table is used to store the number of times each character curses per episode

Create Statement

```
CREATE TABLE characterCurses (
    cid         char(4),
    episode#    char(4),
    curseCount  integer
);
```

Functional Dependencies

cid, episode# → curseCount

Sample Data

Cid	Episode#	CurseCount
C001	66	18
C002	66	15
C003	66	14
C004	66	25
C005	66	4
C006	66	5
C007	66	6
C009	66	10
C001	137	4
C002	137	5
C003	137	2
C004	137	4
C006	137	3
C001	161	7
C002	161	6
C003	161	2
C004	161	94
C001	215	3
C002	215	6
C003	215	2
C004	215	5
C005	215	1
C007	215	8

CharacterDeaths

Purpose: This table is used to store information about what was parodied in each episode.

Create Statement

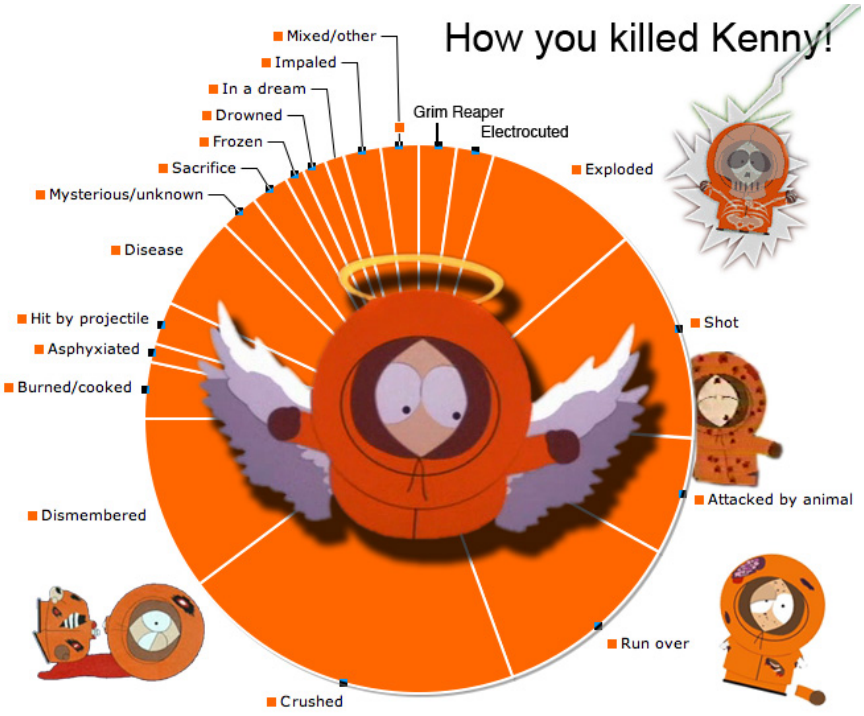
```
CREATE TABLE characterDeaths (
    cid                 char(4),
    episode#            char(4),
    deathDescription    text
);
```

Functional Dependencies

cid, episode# → description

Sample Data

Cid	Episode#	Description
C003	66	Vomited his intestines out after getting the plague



SchoolPosition

Purpose: This table is used to store what positions are available in the school

Create Statement

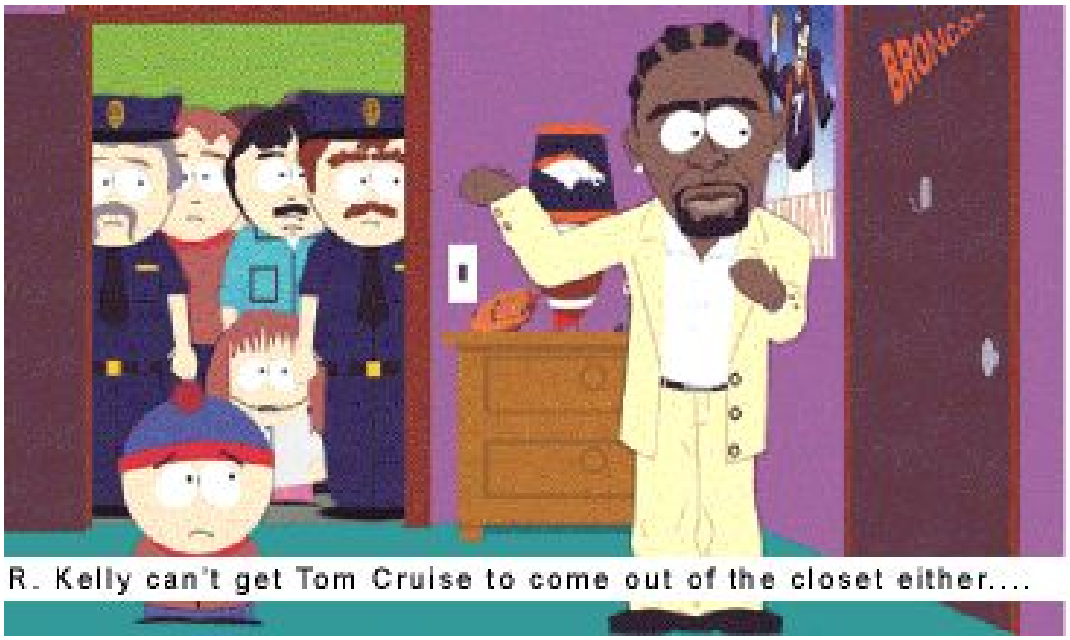
```
CREATE TABLE schoolPosition (
    spid            char(4),
    tittle          text,
    primary key(spid)
);
```

Functional Dependencies

spid → tittle

Sample Data

Spid	Tittle
S001	3rd Grade Teacher
S002	4th Grade Teacher
S003	Chef



CharacterSchoolPosition

Purpose: This table is used to store what position in the school a character holds

Create Statement

```
CREATE TABLE characterSchoolPosition (
    spid            char(4),
    cid             char(4)
);
```

Functional Dependencies

spid → cid

Sample Data

Spid	Cid
S001	C005
S002	C005
S003	C009

VIEWS

Purpose: To be a mini episode guide displaying basic information such as the episode number, the name of the episode, and the air date

Code

```
CREATE VIEW mini_channelGuide AS
select e.episodeNumber, e.name, e.airDate
from episodes e
order by e.episodeNumber
```

Purpose: To be a more comprehensive episode guide that also includes what characters appeared in the episode

Code

```
CREATE VIEW channelGuide AS
select e.episodeNumber, e.name as episodeName, e.airDate, c.fname as firstName, c.lname as lastName
from episodes e, characters c, charactersInEpisodes cie
where c.cid = cie.cid
and cie.episodeNumber = e.episodeNumber
```

Sample Output

	episodenumber character(4)	episodename text	firstname text	lastname text
1	66	It Hits the Fan	Stan	Marsh
2	66	It Hits the Fan	Kyle	Broflovs
3	66	It Hits the Fan	Kenny	McCormic
4	66	It Hits the Fan	Eric	Cartman
5	66	It Hits the Fan	Herbert	Garrison
6	66	It Hits the Fan	Butters	Stotch
7	137	Trapped in the Closet	Stan	Marsh
8	137	Trapped in the Closet	Kyle	Broflovs
9	137	Trapped in the Closet	Kenny	McCormic
10	137	Trapped in the Closet	Eric	Cartman
11	137	Trapped in the Closet	Randy	Marsh
12	137	Trapped in the Closet	Butters	Stotch
13	137	Trapped in the Closet	Token	Black
14	161	Le Petit Tourette	Stan	Marsh
15	161	Le Petit Tourette	Kyle	Broflovs
16	161	Le Petit Tourette	Kenny	McCormic
17	161	Le Petit Tourette	Eric	Cartman
18	161	Le Petit Tourette	Herbert	Garrison
19	161	Le Petit Tourette	Butters	Stotch
20	215	City Sushi	Stan	Marsh
21	215	City Sushi	Kyle	Broflovs

Purpose: To be an episode guide that includes what the episode parodied

Code

```
CREATE VIEW episode_Parodies AS
select e.episodeNumber, e.name as episodeName, e.airDate,tp.thingParodied
from episodes e, parodiedInEpisodes pie, thingsParodied tp
where e.episodeNumber = pie.episodeNumber
and pie.tpid = tp.tpid
```

Sample Output

	episodenumber character(4)	name text	airdate date	thingparodied text
1	66	It Hits the Fan	2001-06-20	Chicago Hope
2	137	Trapped in the Closet	2005-11-16	Tom Cruise
3	137	Trapped in the Closet	2005-11-16	Scientology
4	137	Trapped in the Closet	2005-11-16	R. Kelly
5	161	Le Petit Tourette	2007-10-03	Tourettes
6	215	City Sushi	2011-06-01	Asian Accents
7	215	City Sushi	2011-06-01	Split Personali
8	215	City Sushi	2011-06-01	Horror Movies
9	215	City Sushi	2011-06-01	Psycho

Purpose: To display information about the characters including their name, age, hair color, and character trait

Code

```
CREATE VIEW characterInfo AS
select c.fname as firstName, c.lname as lastName, c.hairColor, c.age, ct.trait as characterTrait
from characters c, characterTraits ct
where c.cid = ct.cid
```

Sample Output

	firstname text	lastname text	haircolor text	age integer	charactertrait text
1	Stan	Marsh	Black	10	Leader
2	Kyle	Broflovski	Red	10	Jew
3	Kenny	McCormick	Blond	10	Poor
4	Eric	Cartman	Brown	10	Fat Ass
5	Herbert	Garrison	Gray	41	Gay
6	Randy	Marsh	Black	45	Lorde
7	Butters	Stotch	Blond	10	Constantly Grounded
8	Token	Black	Black	10	Only Black Kid
9	Jerome (Chef)	McElroy	black	38	Chef

Reports

Purpose: Display the number of times Kenny has been killed

Code

```
select count(*) as kennyDeathCount
from characterDeaths
where cid = 'c003'
```

Purpose: To show the current total of curses each of the character has said on the show

Code

```
select c.fname as firstName, c.lname as lastName, sum(curseCount) as curseTotal
from characters c, characterCurses cc
where c.cid = cc.cid
group by c.fname, c.lname, c.cid
order by c.cid
```

Sample Output137

	firstname text	lastname text	cursetotal bigint
1	Stan	Marsh	32
2	Kyle	Broflovski	32
3	Kenny	McCormick	20
4	Eric	Cartman	128
5	Herbert	Garrison	5
6	Randy	Marsh	8
7	Butters	Stotch	14
8	Jerome (Chef)	McElroy	10

Stored Procedure

Purpose: To see the total curses said for a specific episode

Code

```
CREATE OR REPLACE FUNCTION totalCurses(episode char) returns char as $$
declare result int;
BEGIN
SELECT sum(curseCount) into result
from characterCurses cc
where cc.episodeNumber = episode;
return result;
end
$$ LANGUAGE plpgsql;
```

Example Input

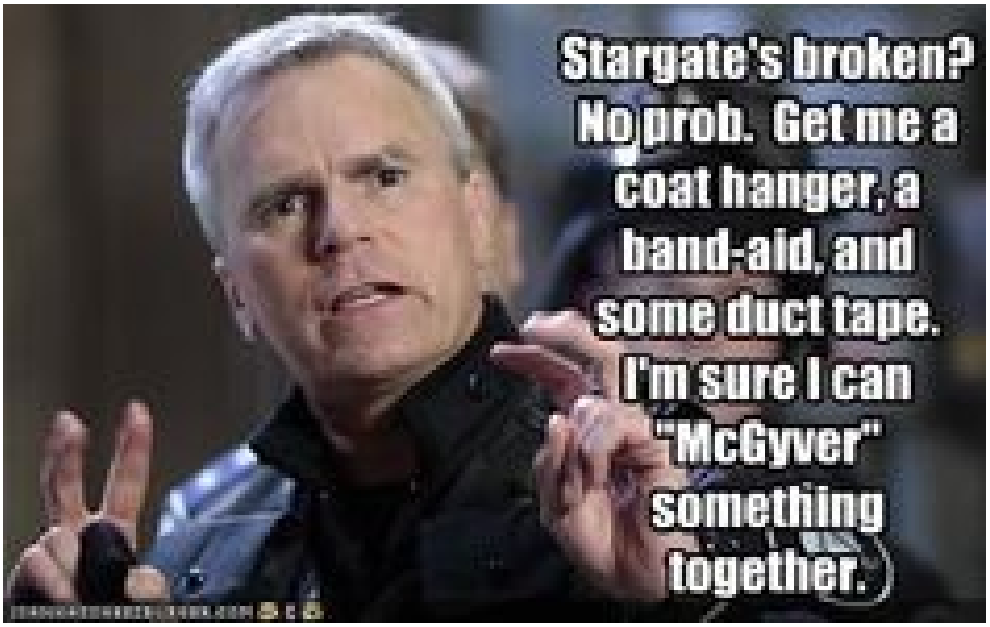
```
select totalCurses('66');
```

Triggers

I honestly could not figure out how to correctly code/implement triggers so I will simply list what triggers I planned on implementing. I will go back and actually implement them if I figure out how.

Purpose: After each new episodes information is added to the database it will run the stored procedure to determine the number of curses said in the new episode.

Purpose: After something starts tending on twitter it will be stored as long as it hasn't already been parodied on the show. After 3 trending things are added it will suggest that they be used in a new episode. (Insert code for this here)



Me trying to figure out how to make triggers work

Security

For this database there are most likely only three different user types

1. The Admin(godlike) who has full access to do as he wish(hopefully he uses his powers for good not evil)

   ```
   CREATE ROLE admin
   GRANT ALL PRIVILEGES
   ON ALL TABLES IN SCHEMA PUBLIC
   To admin
   ```
2. The general user who only has the most basic of privileges
   ```
   CREATE ROLE basicUser
   GRANT SELECT
   ON ALL TABLES IN SCHEMA PUBLIC
   TO basicUser
   ```
3. A user with slightly more privileges than the average user, let's call them superUser. They can select and insert new entries. They can't delete records or change records
   ```
   CREATE ROLE superUser
   GRANT SELECT, INSERT
   ON ALL TABLES IN SCHEMA PUBLIC
   TO superUser
   ```

Implementation Notes/Known Issues/ Future Enhancements

- For the most part the setup and design of the database went fairly well.
- The hardest part was definitely coming up with an idea to do
- The easiest part was gathering the test data
- Inserting it was a decent amount of work just because of how much data there was for each episode
- The hardest part about inserting the data was that I kept making spelling mistakes and getting errors, it was quite frustrating.
- Another hard part about this project was coming up with ideas of what to do for views, and reports.
    - If I had to do this project again I would probably pick a different topic. As much fun as I had doing South Park, it did not lend itself well to coming up with ideas for stored procedures, and triggers.
- I believe I came up with a good number of tables that represents just about all you would want to know about the characters.
- If I had more time and more patience I would have had a table that keeps stores all the social criticisms and commentary that Trey and Matt have put in the show over the years. That table would be so large that I would have had to spend all semester to properly do it.
- ER diagram seems pretty good, the issue was how much info to include, the diagram could have been much larger with many more tables but I feel this is was a good about for a project of this scale.
- It is not as specific as possible because I wasn't sure how specific to make it, can be easily fixed
- The security privileges granted could become a problem will revoke if necessary.
- Stored procedure return <null> if an invalid episode number is entered or no curses were uttered in the episode. Should probably fix to display a more user friendly result.

- For future enhancements I plan to add a social commentary table as well as what episodes provided backlash in real life. If I clean it up enough I could start my very own South Park wiki with this database. Also connect to twitter and take the trending topics and compare it to the list of already parodied topics. If it isn't there it will take it and suggest it as a future episode.