Super Smash Brothers 4

A Database by James Holden

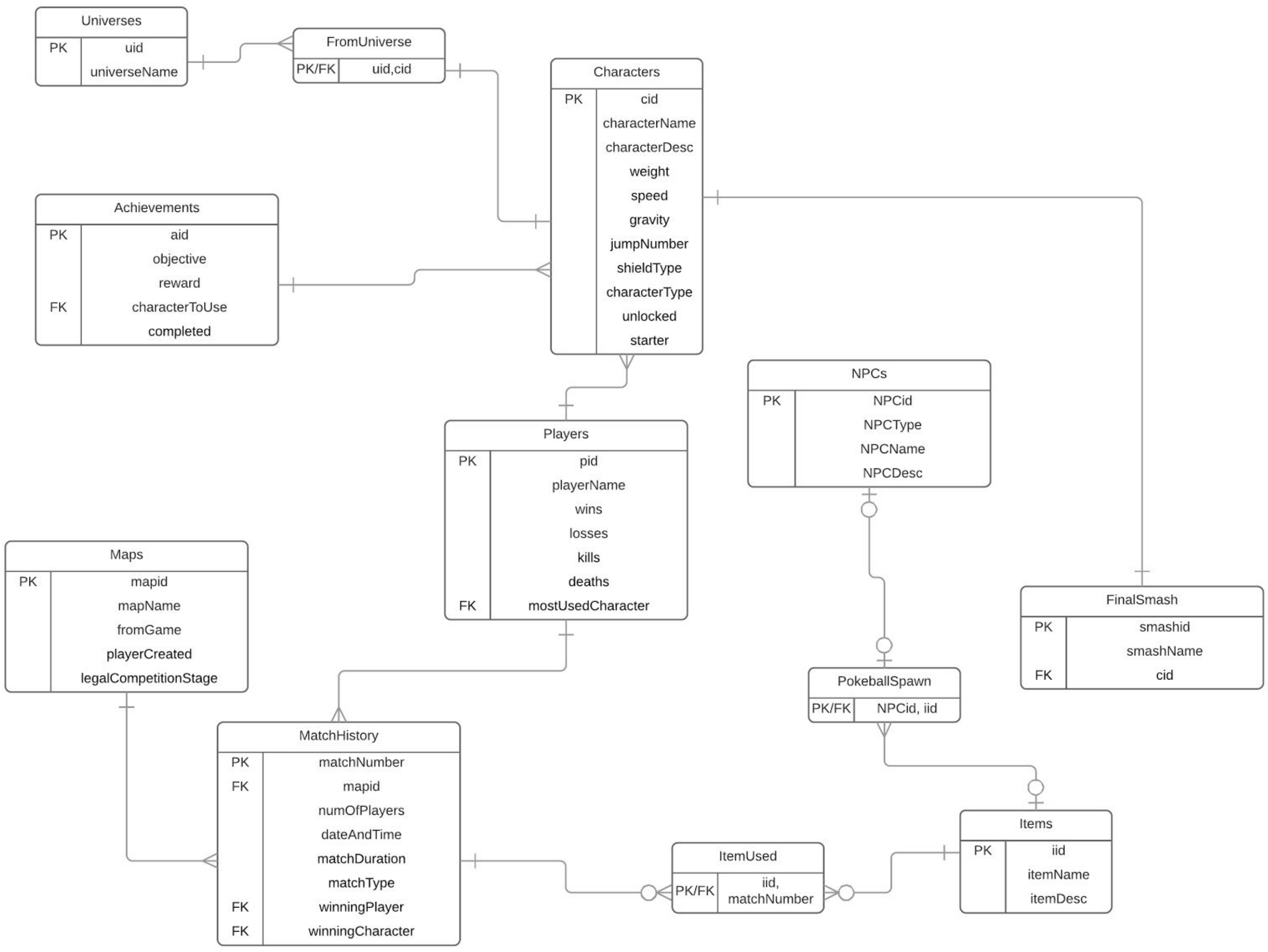
Executive Summary

The purpose of this project was to create a database for the video game “Super Smash Brothers 4” in 3NF. It is much easier to visualize the relationships between all of the data in the game this way.

Another purpose of this project was to attempt to create a database that updates itself through gaining information from other tables in the relational database.

Overall this project helped me to much better understand the relationships between objects in a game that I’ve played for years without understanding the relational database aspects of.

ERD



Universes
PK uid
universeName
FromUniverse
PK/FK uid,cid
Characters
PK cid
characterName
characterDesc
weight
speed
gravity
jumpNumber
shieldType
characterType
unlocked
starter
Achievements
PK aid
objective
reward
FK characterToUse
completed
NPCs
PK NPCid
NPCType
NPCName
NPCDesc
Players
PK pid
playerName
wins
losses
kills
deaths
FK mostUsedCharacter
Maps
PK mapid
mapName
fromGame
playerCreated
legalCompetitionStage
FinalSmash
PK smashid
smashName
FK cid
PokeballSpawn
PK/FK NPCid, iid
MatchHistory
PK matchNumber
FK mapid
numOfPlayers
dateAndTime
matchDuration
matchType
FK winningPlayer
FK winningCharacter
ItemUsed
PK/FK iid, matchNumber
Items
PK iid
itemName
itemDesc

Tables

Create Statements

```
create table Characters(
        cid                         serial,
        characterName               text not null,
        characterDesc               text not null,
        weight                      int not null,
        speed                       decimal not null,
        gravity                     decimal not null,
        jumpNumber                  int DEFAULT 2  not null,
        shieldType                  text DEFAULT 'Bubble' not null,
        characterType               text not null,
        unlocked                    bool not null DEFAULT true,
        starter                     bool DEFAULT true not null,
        primary key (cid)


);


create table Universes(
        uid                 serial,
        universeName        text not null,
        primary key (Uid)
);


create table FromUniverse(
        uid                 int references Universes(uid),
        cid                 serial references characters(cid),
        primary key (Uid, cid)
);
```

```
create table Maps(
        mapid                           char(3) unique not null,
        mapName                         text not null,
        mapDesc                         text not null,
        fromGame                        text not null,
        playerCreated                   bool not null DEFAULT false,
        legalCompetitionStage           bool not null DEFAULT false,
        primary key (mapid)

);



create table Players(
        pid                             char(4) unique not null,
        playerName                      text not null,
        losses                          int not null DEFAULT 0,
        Kills                           int not null DEFAULT 0,
        deaths                          int not null DEFAULT 0,
        mostUsedCharacter               int not null references Characters(cid),
        wins                            int DEFAULT 0,
        primary key (pid)
);


create table Moves(
        moveid                  char(2) unique not null,
        moveName                text not null,
        damage                  text not null,
        primary key (moveid)

);

create table Achievements(
        aid                     char(2) unique not null,
        objective               text not null,
        reward                  text not null,
        characterToUse          int references Characters(cid),
        completed               bool not null DEFAULT false,
        primary key (aid)
);
```

```
create table MatchHistory(
        matchNumber                     int not null,
        mapid                           char(3) not null references Maps(mapid),
        numOfPlayers int not null

                                        check (numOfPlayers > 1 and numOfPlayers <= 8),

        dateAndTime                     timestamp not null,
        matchDuration                   time not null

                                         check(matchDuration > '00:00:00'),

        matchType                       text not null

                                        check (matchType = 'Stock' OR matchType = 'Time'),

        winningPlayer                   char(4)         not null references Players(pid),
        winningCharacter                int not null references Characters(cid),
        primary key(matchNumber)
);

create table NPCs(
        NPCid                   int not null,
        NPCType                 text not null,
        NPCName                 text not null,
        NPCDesc                 text not null,
        primary key (NPCid)

);

create table Items(
        iid                     int unique not null,
        itemName                text not null,
        itemDesc                text not null,
        primary key (iid)
);
```

```
create table PokeballSpawn(
        iid             int not null references Items(iid),
        NPCid           int not null references NPCs(NPCid),
        primary key (iid, NPCid)
);




        create table FinalSmash(
        smashid                 serial,
        cid                     serial references Characters(cid),
        smashName               text not null,
        primary key(smashid)

);

create table itemUsed(
        matchnumber                     int references MatchHistory(matchNumber),
        iid                             int references Items(iid),
        Primary Key (matchnumber, iid)
);
```

Every table has a Primary key that is unique to each row. Many of the tables have foreign keys as well, you can see this through the references made in the create table statements.

Many of the tables also have default values. An example of this is the achievements are all defaulted to have completed = false. The reason for this is because you don’t start off the game with any of the achievements already completed. They wouldn’t be very impressive achievements if that were the case.

There are also several Check constraints. One example of this is the Match History table. A match only has two types, Stock(which is lives) or Time. The check constraint makes sure it is one of these two. Anything else would be wrong.

Functional Dependencies

Characters Table:

cid → characterName, characterDesc, weight, speed, gravity, jumpNumber, shieldType, characterType, unlocked, starter

Players Table:

pid → playerName, wins, losses, kills, deaths, mostUsedCharacter

NPCs Table:

NPCid → NPCType, NPCName, NPCDesc

Items Table:

iid → itemName, itemDesc

Match History Table:

matchNumber → mapid, numOfPlayers, dateAndTime, matchDuration, matchType, winningPlayer, winningCharacter

Maps Table:

mapid → mapName, fromGame, playerCreated, legalCompetitionStage

FinalSmash Table:

smashid → smashName, cid

Achievements Table:

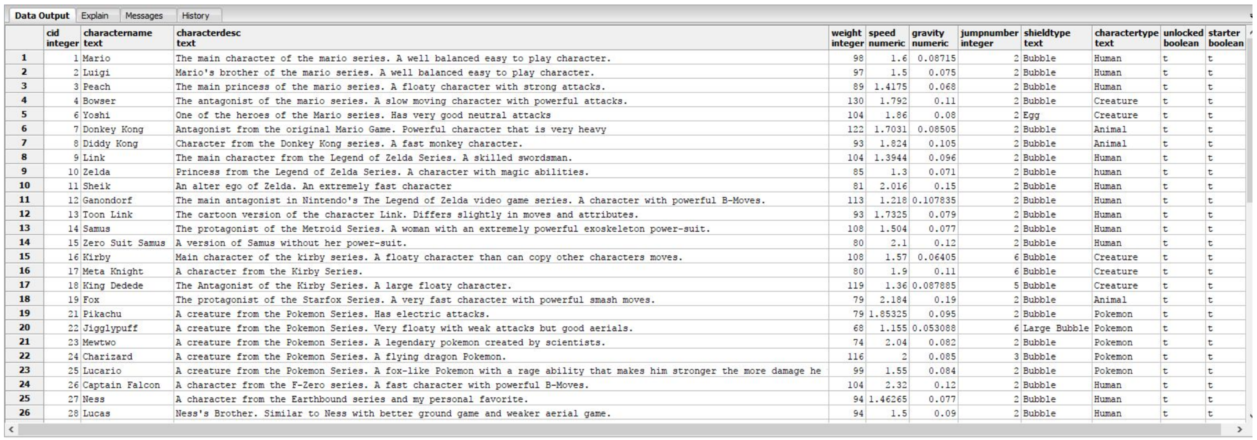
aid → objective, reward, characterToUse, completed

Universes Table:

uid → universeName

SELECT* Queries:

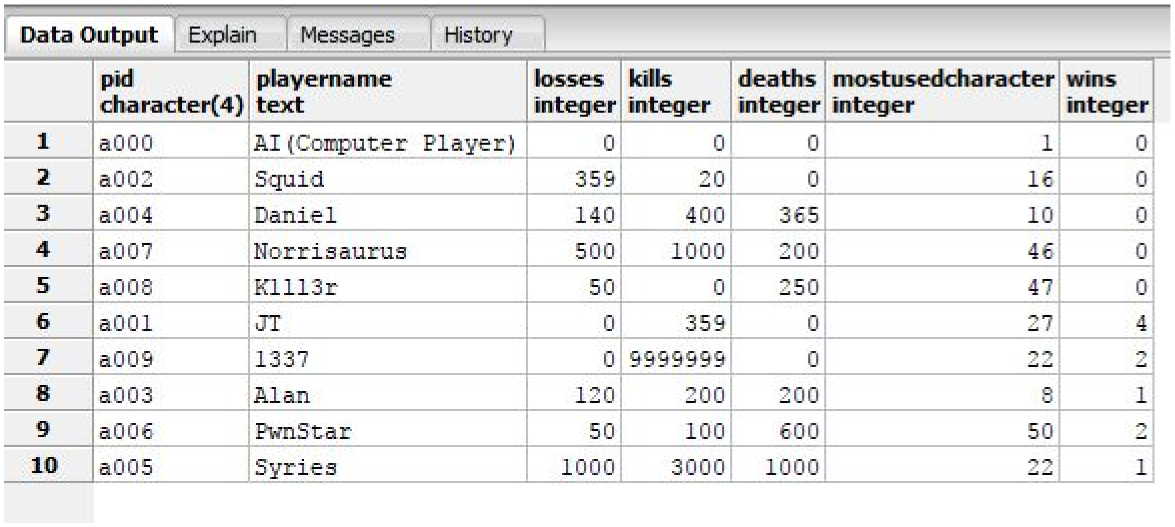
SELECT* FROM Characters;



Data Output | Explain | Messages | History

	cid integer	charactername text	characterdesc text	weight integer	speed numeric	gravity numeric	jumpnumber integer	shieldtype text	charactertype text	unlocked boolean	starter boolean
1	1	Mario	The main character of the mario series. A well balanced easy to play character.	98	1.6	0.08715	2	Bubble	Human	t	t
2	2	Luigi	Mario's brother of the mario series. A well balanced easy to play character.	97	1.5	0.075	2	Bubble	Human	t	t
3	3	Peach	The main princess of the mario series. A floaty character with strong attacks.	89	1.4175	0.068	2	Bubble	Human	t	t
4	4	Bowser	The antagonist of the mario series. A slow moving character with powerful attacks.	130	1.792	0.11	2	Bubble	Creature	t	t
5	6	Yoshi	One of the heroes of the Mario series. Has very good neutral attacks	104	1.86	0.08	2	Egg	Creature	t	t
6	7	Donkey Kong	Antagonist from the original Mario Game. Powerful character that is very heavy	122	1.7031	0.08505	2	Bubble	Animal	t	t
7	8	Diddy Kong	Character from the Donkey Kong series. A fast monkey character.	93	1.824	0.105	2	Bubble	Animal	t	t
8	9	Link	The main character from the Legend of Zelda Series. A skilled swordsman.	104	1.3944	0.096	2	Bubble	Human	t	t
9	10	Zelda	Princess from the Legend of Zelda Series. A character with magic abilities.	85	1.3	0.071	2	Bubble	human	t	t
10	11	Sheik	An alter ego of Zelda. An extremely fast character	81	2.016	0.15	2	Bubble	Human	t	t
11	12	Ganondorf	The main antagonist in Nintendo's The Legend of Zelda video game series. A character with powerful B-Moves.	113	1.218	0.107835	2	Bubble	Human	t	t
12	13	Toon Link	The cartoon version of the character Link. Differs slightly in moves and attributes.	93	1.7325	0.079	2	Bubble	Human	t	t
13	14	Samus	The protagonist of the Metroid Series. A woman with an extremely powerful exoskeleton power-suit.	108	1.504	0.077	2	Bubble	Human	t	t
14	15	Zero Suit Samus	A version of Samus without her power-suit.	80	2.1	0.12	2	Bubble	Human	t	t
15	16	Kirby	Main character of the kirby series. A floaty character than can copy other characters moves.	108	1.57	0.06405	6	Bubble	Creature	t	t
16	17	Meta Knight	A character from the Kirby Series.	80	1.9	0.11	6	Bubble	Creature	t	t
17	18	King Dedede	The Antagonist of the Kirby Series. A large floaty character.	119	1.36	0.087885	5	Bubble	Creature	t	t
18	19	Fox	The protagonist of the Starfox Series. A very fast character with powerful smash moves.	79	2.184	0.19	2	Bubble	Animal	t	t
19	21	Pikachu	A creature from the Pokemon Series. Has electric attacks.	79	1.85325	0.095	2	Bubble	Pokemon	t	t
20	22	Jigglypuff	A creature from the Pokemon Series. Very floaty with weak attacks but good aerials.	68	1.155	0.053088	6	Large Bubble	Pokemon	t	t
21	23	Mewtwo	A creature from the Pokemon Series. A legendary pokemon created by scientists.	74	2.04	0.082	2	Bubble	Pokemon	t	t
22	24	Charizard	A creature from the Pokemon Series. A flying dragon Pokemon.	116	2	0.085	3	Bubble	Pokemon	t	t
23	25	Lucario	A creature from the Pokemon Series. A fox-like Pokemon with a rage ability that makes him stronger the more damage he	99	1.55	0.084	2	Bubble	Pokemon	t	t
24	26	Captain Falcon	A character from the F-Zero series. A fast character with powerful B-Moves.	104	2.32	0.12	2	Bubble	Human	t	t
25	27	Ness	A character from the Earthbound series and my personal favorite.	94	1.46265	0.077	2	Bubble	Human	t	t
26	28	Lucas	Ness's Brother. Similar to Ness with better ground game and weaker aerial game.	94	1.5	0.09	2	Bubble	Human	t	t

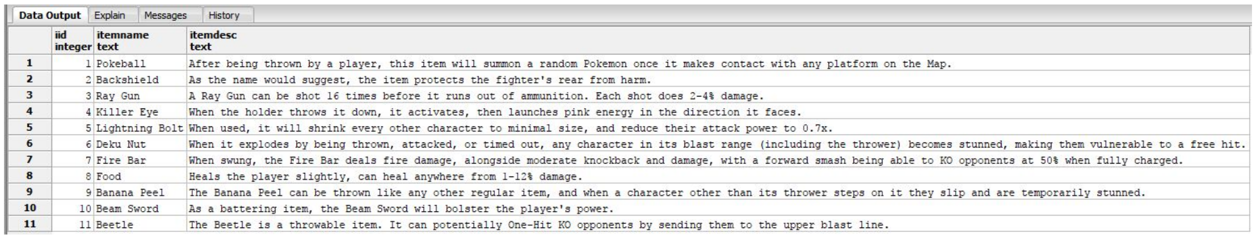
SELECT* FROM Players;



Data Output | Explain | Messages | History

	pid character(4)	playername text	losses integer	kills integer	deaths integer	mostusedcharacter integer	wins integer
1	a000	AI(Computer Player)	0	0	0	1	0
2	a002	Squid	359	20	0	16	0
3	a004	Daniel	140	400	365	10	0
4	a007	Norrisaurus	500	1000	200	46	0
5	a008	K1113r	50	0	250	47	0
6	a001	JT	0	359	0	27	4
7	a009	1337	0	9999999	0	22	2
8	a003	Alan	120	200	200	8	1
9	a006	PwnStar	50	100	600	50	2
10	a005	Syries	1000	3000	1000	22	1

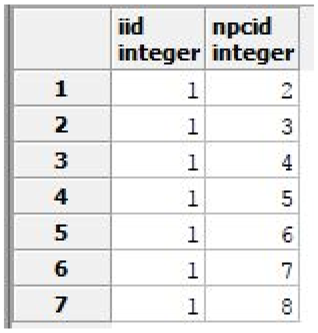
SELECT* FROM Items;



Data Output | Explain | Messages | History

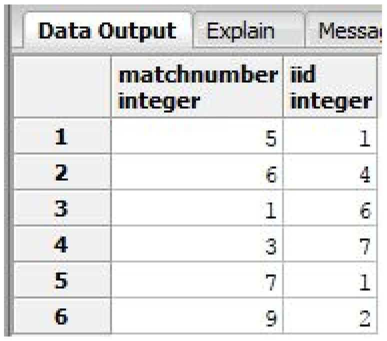
	iid integer	itemname text	itemdesc text
1	1	Pokeball	After being thrown by a player, this item will summon a random Pokemon once it makes contact with any platform on the Map.
2	2	Backshield	As the name would suggest, the item protects the fighter's rear from harm.
3	3	Ray Gun	A Ray Gun can be shot 16 times before it runs out of ammunition. Each shot does 2-4% damage.
4	4	Killer Eye	When the holder throws it down, it activates, then launches pink energy in the direction it faces.
5	5	Lightning Bolt	When used, it will shrink every other character to minimal size, and reduce their attack power to 0.7x.
6	6	Deku Nut	When it explodes by being thrown, attacked, or timed out, any character in its blast range (including the thrower) becomes stunned, making them vulnerable to a free hit.
7	7	Fire Bar	When swung, the Fire Bar deals fire damage, alongside moderate knockback and damage, with a forward smash being able to KO opponents at 50% when fully charged.
8	8	Food	Heals the player slightly, can heal anywhere from 1-12% damage.
9	9	Banana Peel	The Banana Peel can be thrown like any other regular item, and when a character other than its thrower steps on it they slip and are temporarily stunned.
10	10	Beam Sword	As a battering item, the Beam Sword will bolster the player's power.
11	11	Beetle	The Beetle is a throwable item. It can potentially One-Hit KO opponents by sending them to the upper blast line.

SELECT* FROM PokeballSpawn;



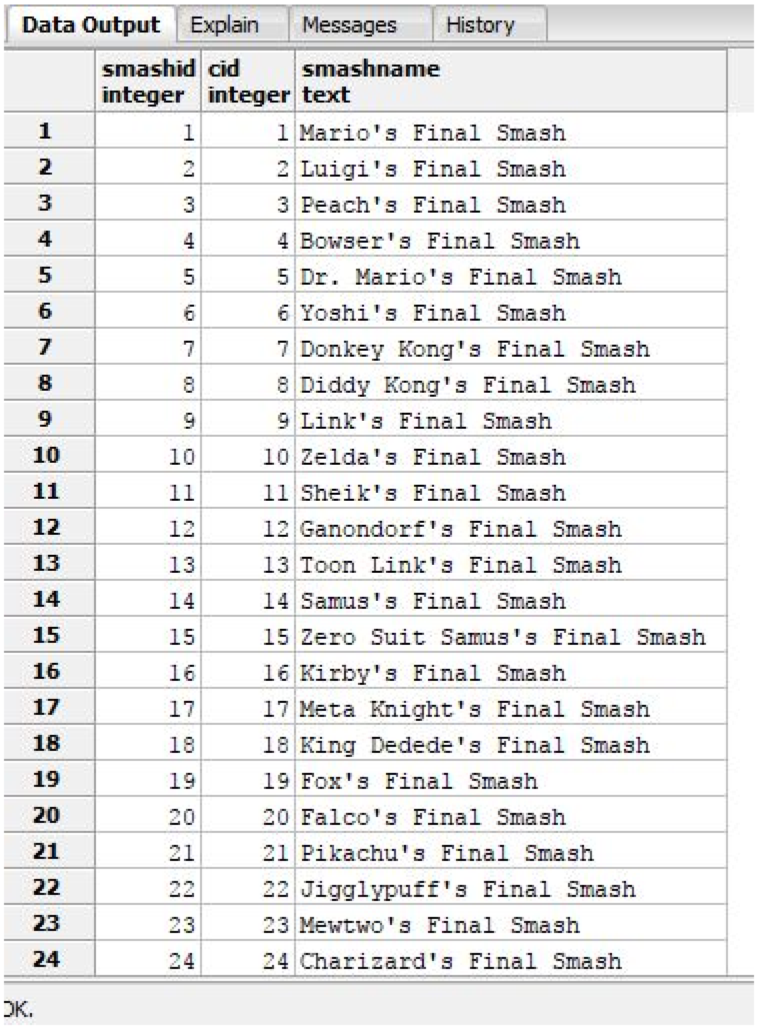
	iid integer	npcid integer
1	1	2
2	1	3
3	1	4
4	1	5
5	1	6
6	1	7
7	1	8

SELECT* FROM ItemUsed;



	matchnumber integer	iid integer
1	5	1
2	6	4
3	1	6
4	3	7
5	7	1
6	9	2

SELECT* FROM FinalSmash;

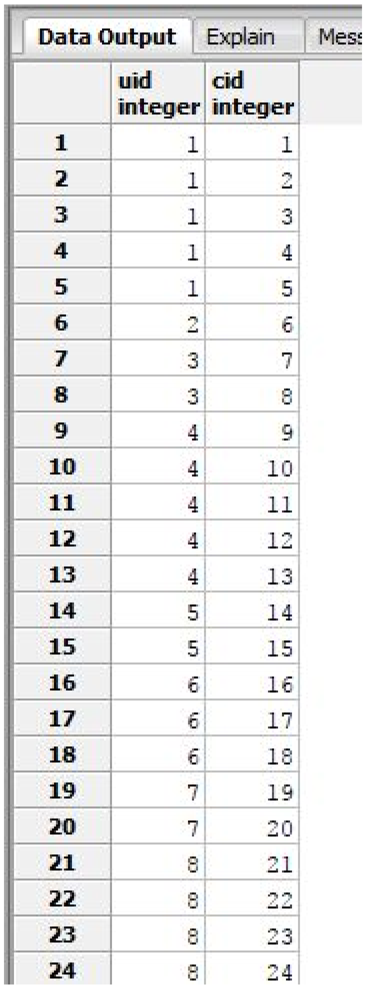


Data Output | Explain | Messages | History

	smashid integer	cid integer	smashname text
1	1	1	Mario's Final Smash
2	2	2	Luigi's Final Smash
3	3	3	Peach's Final Smash
4	4	4	Bowser's Final Smash
5	5	5	Dr. Mario's Final Smash
6	6	6	Yoshi's Final Smash
7	7	7	Donkey Kong's Final Smash
8	8	8	Diddy Kong's Final Smash
9	9	9	Link's Final Smash
10	10	10	Zelda's Final Smash
11	11	11	Sheik's Final Smash
12	12	12	Ganondorf's Final Smash
13	13	13	Toon Link's Final Smash
14	14	14	Samus's Final Smash
15	15	15	Zero Suit Samus's Final Smash
16	16	16	Kirby's Final Smash
17	17	17	Meta Knight's Final Smash
18	18	18	King Dedede's Final Smash
19	19	19	Fox's Final Smash
20	20	20	Falco's Final Smash
21	21	21	Pikachu's Final Smash
22	22	22	Jigglypuff's Final Smash
23	23	23	Mewtwo's Final Smash
24	24	24	Charizard's Final Smash

OK.

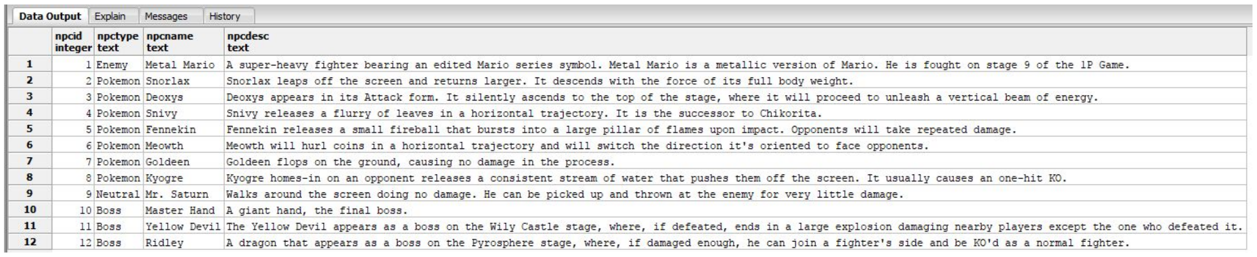
SELECT* FROM FromUniverse;



Data Output | Explain | Mess

	uid integer	cid integer
1	1	1
2	1	2
3	1	3
4	1	4
5	1	5
6	2	6
7	3	7
8	3	8
9	4	9
10	4	10
11	4	11
12	4	12
13	4	13
14	5	14
15	5	15
16	6	16
17	6	17
18	6	18
19	7	19
20	7	20
21	8	21
22	8	22
23	8	23
24	8	24

SELECT* FROM NPCs;



Data Output | Explain | Messages | History

	npcid integer	npctype text	npcname text	npcdesc text
1	1	Enemy	Metal Mario	A super-heavy fighter bearing an edited Mario series symbol. Metal Mario is a metallic version of Mario. He is fought on stage 9 of the 1P Game.
2	2	Pokemon	Snorlax	Snorlax leaps off the screen and returns larger. It descends with the force of its full body weight.
3	3	Pokemon	Deoxys	Deoxys appears in its Attack form. It silently ascends to the top of the stage, where it will proceed to unleash a vertical beam of energy.
4	4	Pokemon	Snivy	Snivy releases a flurry of leaves in a horizontal trajectory. It is the successor to Chikorita.
5	5	Pokemon	Fennekin	Fennekin releases a small fireball that bursts into a large pillar of flames upon impact. Opponents will take repeated damage.
6	6	Pokemon	Meowth	Meowth will hurl coins in a horizontal trajectory and will switch the direction it's oriented to face opponents.
7	7	Pokemon	Goldeen	Goldeen flops on the ground, causing no damage in the process.
8	8	Pokemon	Kyogre	Kyogre homes-in on an opponent releases a consistent stream of water that pushes them off the screen. It usually causes an one-hit KO.
9	9	Neutral	Mr. Saturn	Walks around the screen doing no damage. He can be picked up and thrown at the enemy for very little damage.
10	10	Boss	Master Hand	A giant hand, the final boss.
11	11	Boss	Yellow Devil	The Yellow Devil appears as a boss on the Wily Castle stage, where, if defeated, ends in a large explosion damaging nearby players except the one who defeated it.
12	12	Boss	Ridley	A dragon that appears as a boss on the Pyrosphere stage, where, if damaged enough, he can join a fighter's side and be KO'd as a normal fighter.

Views:

In SQL a view is essentially a virtual table. It is not created through a create table statement and insert functions but rather through queries. This can prove to be incredibly useful when working with relational databases especially because one can create views within views to as many views as needed.

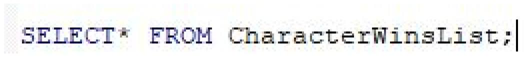
An example of the View I made in my database:

--View of which characters won how many times--

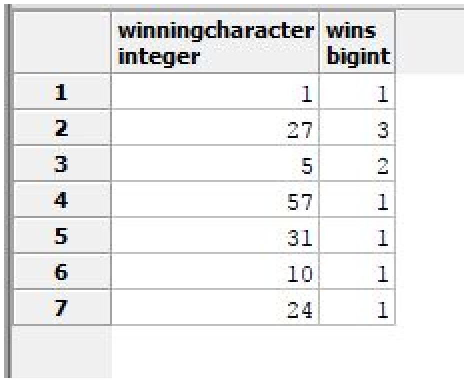
```
CREATE VIEW CharacterWinsList AS

                SELECT winningCharacter, COUNT(winningCharacter) AS wins
                FROM matchHistory
                GROUP BY winningCharacter;
```

Now although I never created a table or column to keep track of which characters won the most, I was able to create a virtual table to do so. The virtual table is called CharacterWinsList. It selects each row in the MatchHistory table and counts which characters have how many wins.

If I run a query on my view such as



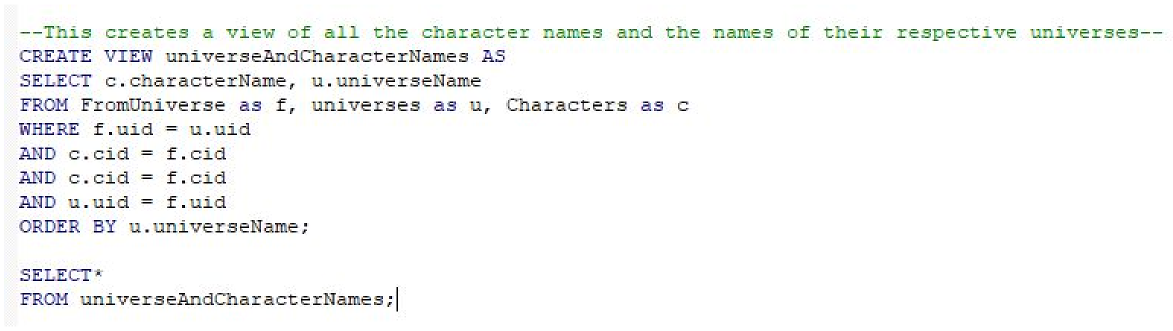
```
SELECT* FROM CharacterWinsList;
```



	winningcharacter integer	wins bigint
1	1	1
2	27	3
3	5	2
4	57	1
5	31	1
6	10	1
7	24	1

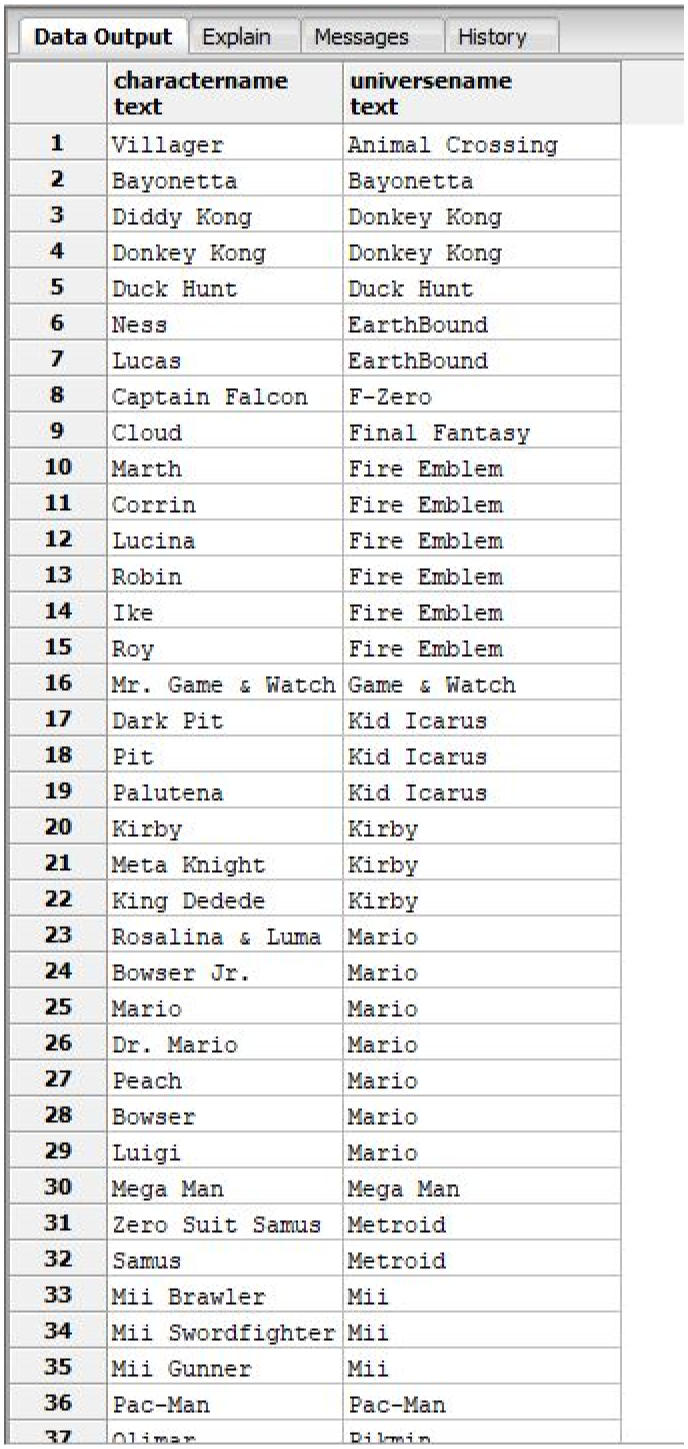
It treats it as a table and now I have access to a virtual table of the characterids and how many wins each character has. In this case even if two different players win on separate occasions with the same character, it counts as two wins for the character. This is because we are measuring the character's strength with this query.

This next view is a bit more complicated because it combines information from 3 tables.



```
--This creates a view of all the character names and the names of their respective universes--
CREATE VIEW universeAndCharacterNames AS
SELECT c.characterName, u.universeName
FROM FromUniverse as f, universes as u, Characters as c
WHERE f.uid = u.uid
AND c.cid = f.cid
AND c.cid = f.cid
AND u.uid = f.uid
ORDER BY u.universeName;

SELECT*
FROM universeAndCharacterNames;
```

This view creates a virtual table via a query in order to see the names of what characters are from what universe names. This is the result.



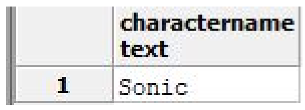
Data Output | Explain | Messages | History

	charactername text	universename text
1	Villager	Animal Crossing
2	Bayonetta	Bayonetta
3	Diddy Kong	Donkey Kong
4	Donkey Kong	Donkey Kong
5	Duck Hunt	Duck Hunt
6	Ness	EarthBound
7	Lucas	EarthBound
8	Captain Falcon	F-Zero
9	Cloud	Final Fantasy
10	Marth	Fire Emblem
11	Corrin	Fire Emblem
12	Lucina	Fire Emblem
13	Robin	Fire Emblem
14	Ike	Fire Emblem
15	Roy	Fire Emblem
16	Mr. Game & Watch	Game & Watch
17	Dark Pit	Kid Icarus
18	Pit	Kid Icarus
19	Palutena	Kid Icarus
20	Kirby	Kirby
21	Meta Knight	Kirby
22	King Dedede	Kirby
23	Rosalina & Luma	Mario
24	Bowser Jr.	Mario
25	Mario	Mario
26	Dr. Mario	Mario
27	Peach	Mario
28	Bowser	Mario
29	Luigi	Mario
30	Mega Man	Mega Man
31	Zero Suit Samus	Metroid
32	Samus	Metroid
33	Mii Brawler	Mii
34	Mii Swordfighter	Mii
35	Mii Gunner	Mii
36	Pac-Man	Pac-Man
37	Olimar	Pikmin

Example Queries

This is a query to get the fastest character.

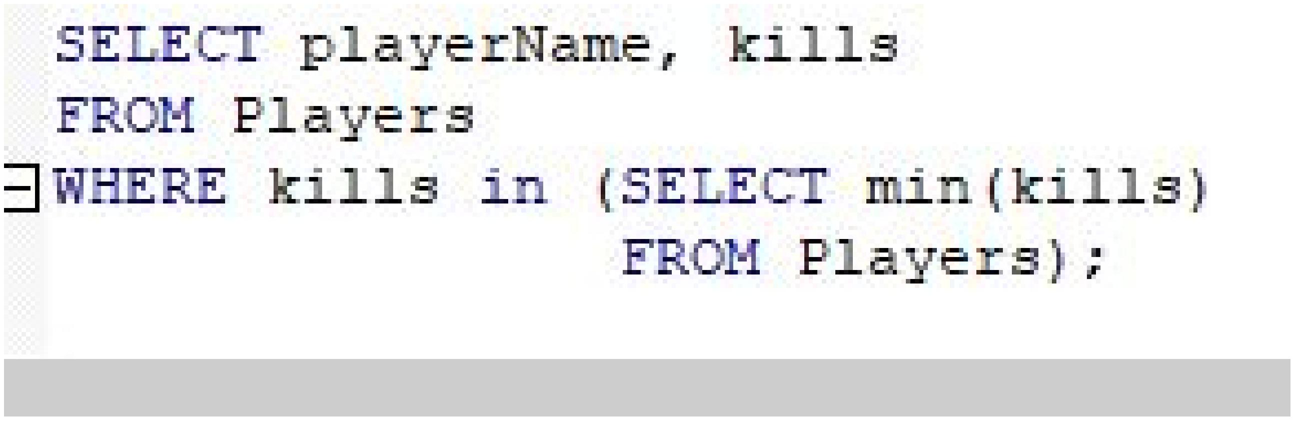
```
SELECT characterName

FROM characters

WHERE speed in     (SELECT max(speed)

                    FROM Characters);
```



	charactername text
1	Sonic

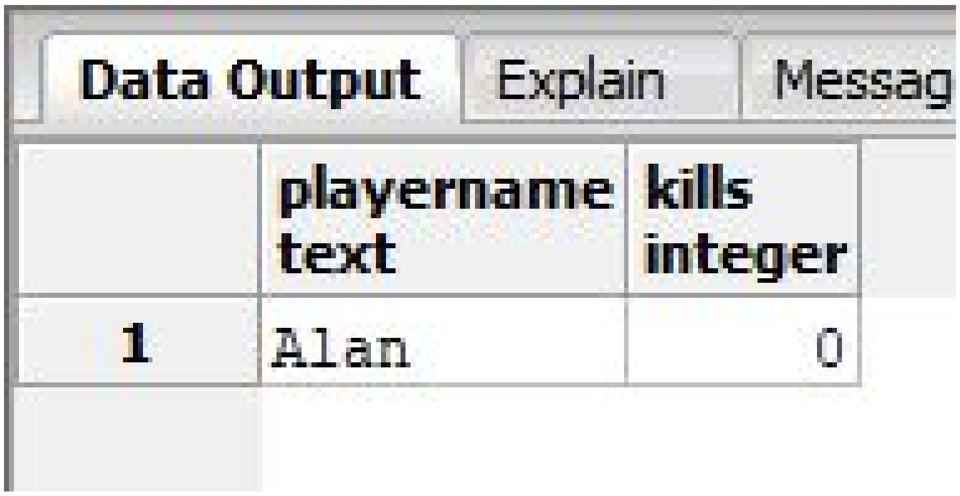
Obviously Sonic the Hedgehog was the result for the fastest speed query. This same query structure could be used to find the slowest speed using min(speed) or really the max or min of any attribute.

For Example I can figure out which player got the least kills and with how many kills with this query.



```
SELECT playerName, kills
FROM Players
WHERE kills in (SELECT min(kills)
                FROM Players);
```

This query returns:

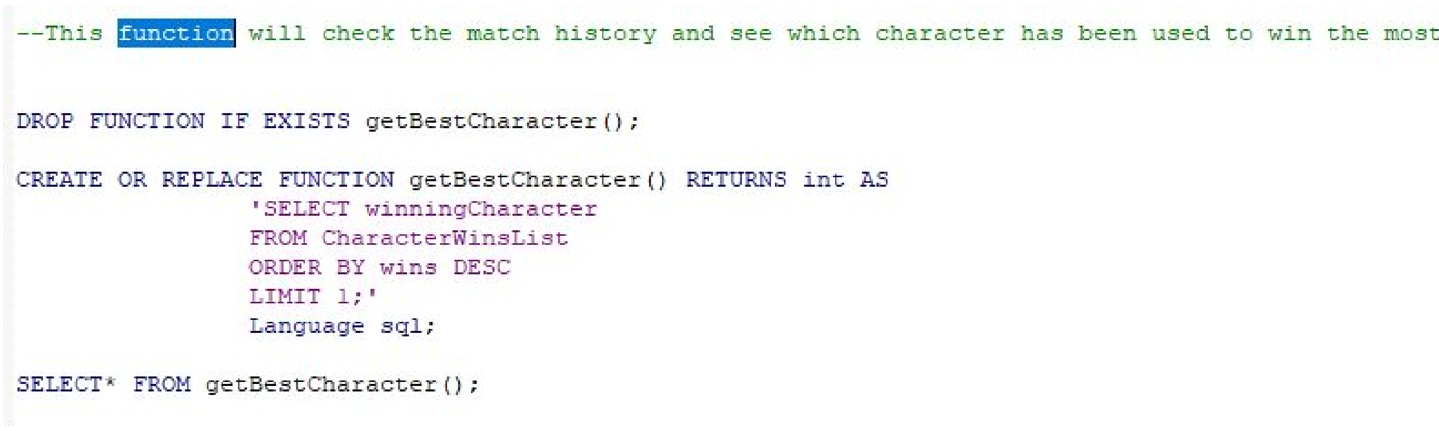


	playername text	kills integer
1	Alan	0

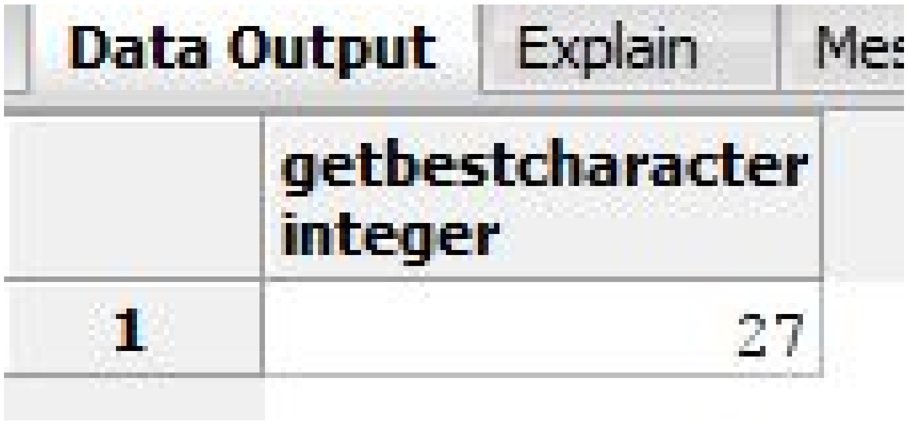
Oh… Well that’s a shame.

Procedures

One Function I created was to find out who the best Character is. This Function goes through the MatchHistory table to see which character has the most wins.



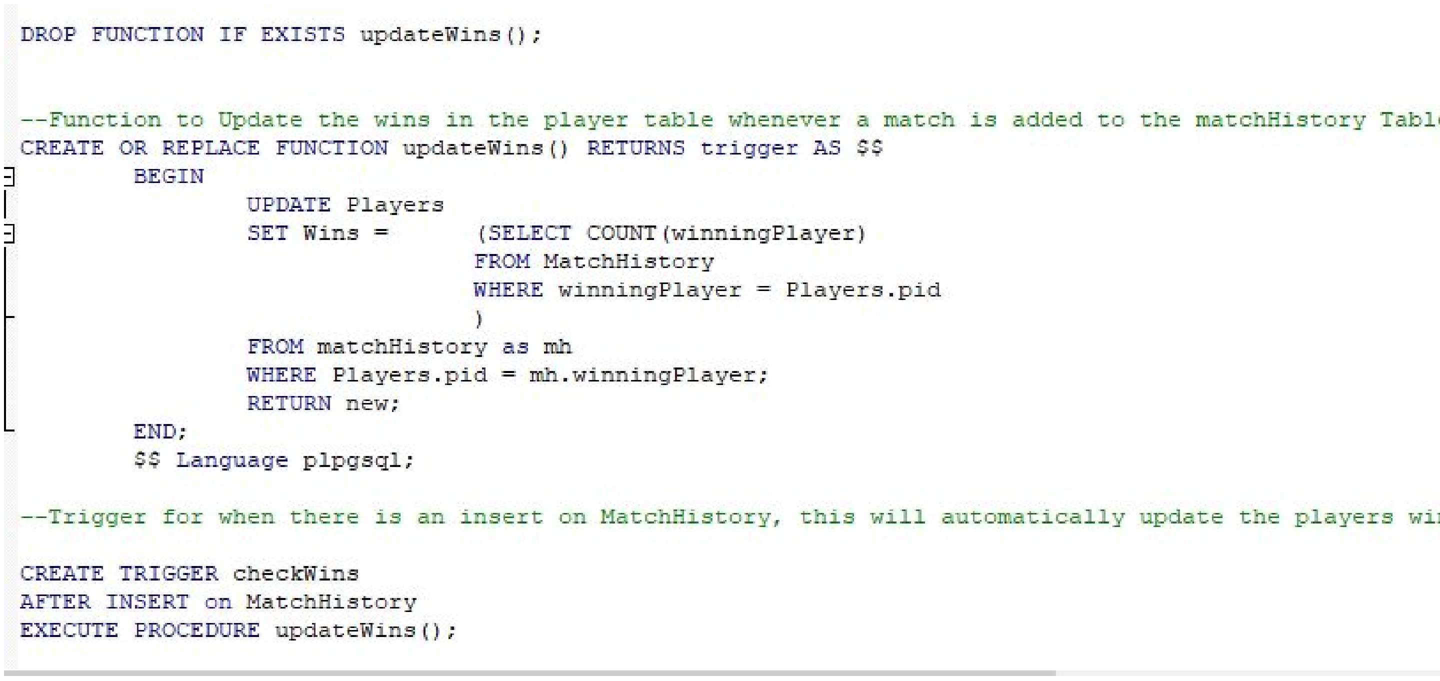
```
--This function will check the match history and see which character has been used to win the most

DROP FUNCTION IF EXISTS getBestCharacter();

CREATE OR REPLACE FUNCTION getBestCharacter() RETURNS int AS
                'SELECT winningCharacter
                FROM CharacterWinsList
                ORDER BY wins DESC
                LIMIT 1;'
                Language sql;

SELECT* FROM getBestCharacter();
```



	getbestcharacter integer
1	27

This returns 27 which is the CharacterID of Ness. Most of the games have been won with Ness so that is why the character shows up.

This next part is a trigger and function.



```
DROP FUNCTION IF EXISTS updateWins();


--Function to Update the wins in the player table whenever a match is added to the matchHistory Tabl
CREATE OR REPLACE FUNCTION updateWins() RETURNS trigger AS $$
        BEGIN
                UPDATE Players
                SET Wins =      (SELECT COUNT(winningPlayer)
                                FROM MatchHistory
                                WHERE winningPlayer = Players.pid
                                )
                FROM matchHistory as mh
                WHERE Players.pid = mh.winningPlayer;
                RETURN new;
        END;
        $$ Language plpgsql;

--Trigger for when there is an insert on MatchHistory, this will automatically update the players wi

CREATE TRIGGER checkWins
AFTER INSERT on MatchHistory
EXECUTE PROCEDURE updateWins();
```

What is essentially happening is whenever a new row is inserted into MatchHistory, there must have been a winner to the match. This function updates the players table to add one win to whoever just won on the MatchHistory Table.

“CheckWins” is a trigger used to cause the UpdateWins() function to execute.

This is very useful because I don’t have to insert new data for when a match happens. This function takes MatchHistory.winningPlayer, finds the PlayerID in the players table, and adds one to players.wins.

Implementation Notes and Problems

The database turned out pretty much exactly how I wanted it to. I do regret populating the tables so much however. I populated the characters table with all 58 characters from the game, and their real attributes according to the wiki. I should have just populated the table with 15 or so and only put the important attributes, not weight and speed and such.

Future Enhancements

I would definitely like to see if I could use an API to populate the tables in the future. I was looking into the wiki API and I might have been able to write a function that would populate the characters table with all their real stats if I had more time. I could do this for almost every table in the database if I figured it out and I would be able to populate hundreds of hours of work in just seconds.