DRAGON BALL Z

FINAL PROJECT

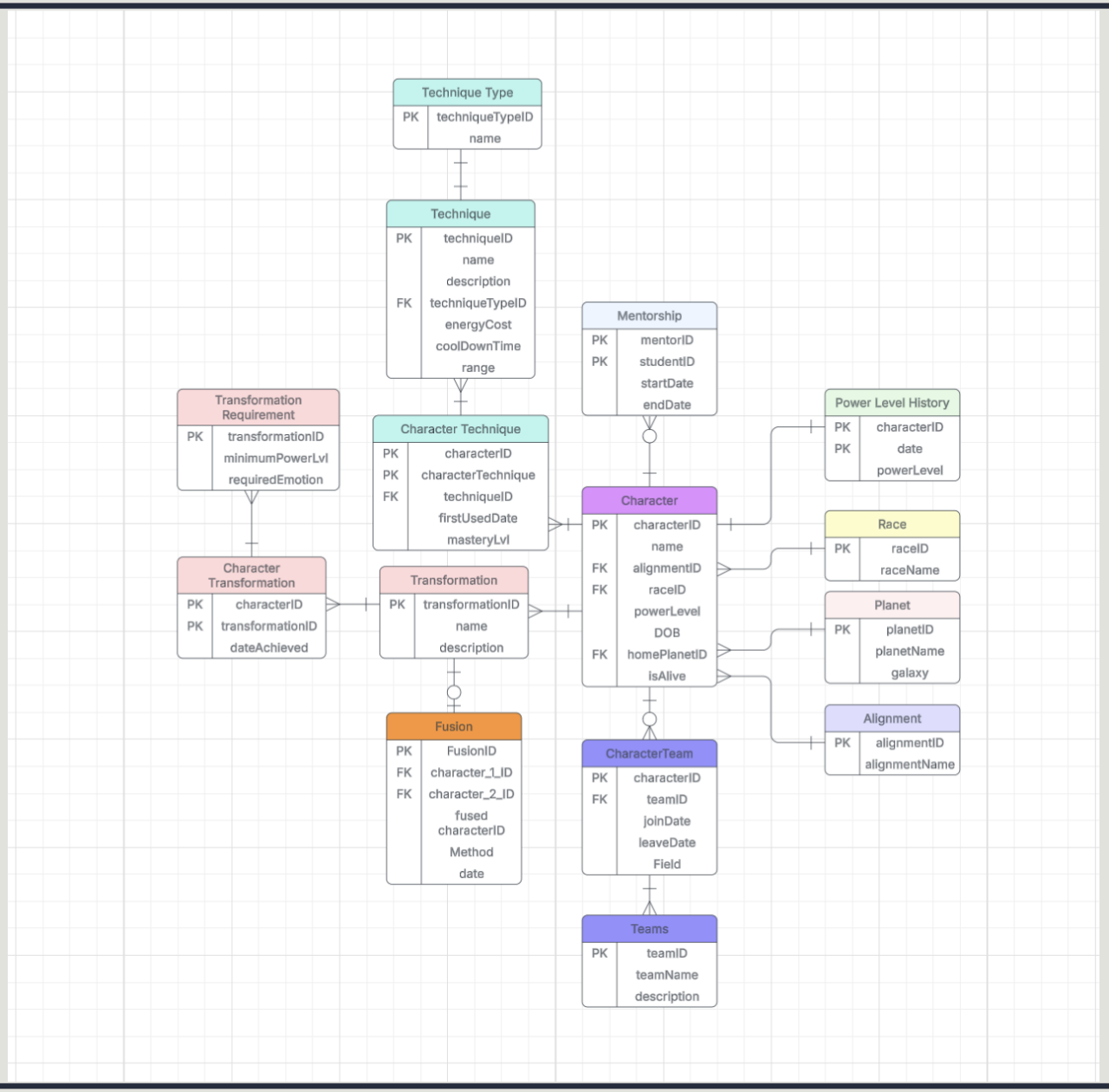
SHEA DECARO

SUMMARY

The Show “Dragon Ball Z” is a complex animated show for all ages that consists of a vast group of heroes and villains. Each character possesses unique powers, some of which were not apparent until after specialized training or specific events critical to their plot. Part of the appeal of this show is the constant building of power and rising of energy within the main cast as they encounter obstacles and antagonists looking to take over or destroy the world.

Hence, the Dragon Ball Z database organizes the unique qualities of these characters into various tables, and highlights changes over time that can be referenced throughout the show.

ER Diagram


Technique Type
PK techniqueTypeID
name
Technique
PK techniqueID
name
description
FK techniqueTypeID
energyCost
coolDownTime
range
Mentorship
PK mentorID
PK studentID
startDate
endDate
Transformation Requirement
PK transformationID
minimumPowerLvl
requiredEmotion
Character Technique
PK characterID
PK characterTechnique
FK techniqueID
firstUsedDate
masteryLvl
Power Level History
PK characterID
PK date
powerLevel
Character
PK characterID
name
FK alignmentID
FK raceID
powerLevel
DOB
FK homePlanetID
isAlive
Race
PK raceID
raceName
Character Transformation
PK characterID
PK transformationID
dateAchieved
Transformation
PK transformationID
name
description
Planet
PK planetID
planetName
galaxy
Alignment
PK alignmentID
alignmentName
Fusion
PK FusionID
FK character_1_ID
FK character_2_ID
fused characterID
Method
date
CharacterTeam
PK characterID
FK teamID
joinDate
leaveDate
Field
Teams
PK teamID
teamName
description

Character table - Keeps track of all characters in the show.

```
CREATE TABLE Character (
    character_id SERIAL PRIMARY KEY,
    name TEXT NOT NULL,
    race_id INT REFERENCES
Race(race_id),
    home_planet_id INT REFERENCES
Planet(planet_id),
    alignment_id INT REFERENCES
Alignment(alignment_id),
    date_of_birth DATE,
    is_alive BOOLEAN DEFAULT TRUE
);
```

DEPENDENCIES –

- `character_id` → `name`, `race_id`, `home_planet_id`, `alignment_id`, `date_of_birth`, `is_alive`
    - The `character_id` uniquely determines all other attributes of a character.

SAMPLE DATA-

	character_id [PK] integer	name text	race_id integer	home_planet_id integer	alignment_id integer	date_of_birth date	is_alive boolean
1	1	Goku	1	1	1	1984-05-09	true
2	2	Vegeta	1	1	2	1984-04-02	true
3	3	Piccolo	2	2	1	1984-05-09	true
4	4	Gohan	1	3	1	1989-05-05	true
5	5	Frieza	[null]	[null]	2	[null]	false
6	6	Cell	4	[null]	2	[null]	false
7	7	Majin Buu	5	[null]	2	[null]	false
8	8	Krillin	3	3	1	[null]	true
9	9	Bulma	3	3	1	1985-08-18	true
10	10	Mr. Satan	3	3	1	[null]	true
11	11	Alan Labouseur	3	3	1	1912-06-23	false

Race table - Lists all possible Races in the show.

```
CREATE TABLE Race (
    race_id SERIAL PRIMARY KEY,
    race_name TEXT NOT NULL UNIQUE
);
```

DEPENDENCIES –

- `race_id` → `race_name`
  - The `race_id` uniquely determines the `race_name`.

SAMPLE DATA-

	race_id [PK] integer	race_name text
1	1	Saiyan
2	2	Namekian
3	3	Human
4	4	Android
5	5	Majin

Planet table - The names of planets within the show.

```
CREATE TABLE Planet (
    planet_id SERIAL PRIMARY KEY,
    name TEXT NOT NULL UNIQUE,
    galaxy TEXT
);
```

DEPENDENCIES –

- `planet_id` → `name`, `galaxy`
  - The `planet_id` uniquely determines the `name` and `galaxy`.

SAMPLE DATA-

	planet_id [PK] integer	name text	galaxy text
1	1	Vegeta	Unknown
2	2	Namek	Unknown
3	3	Earth	Milky Way
4	4	Other World	[null]
5	5	Cell Games Arena	Milky Way

Alignment table - alignment distinguishes the stance of a character.

```
CREATE TABLE Alignment (
    alignment_id SERIAL PRIMARY KEY,
    name TEXT NOT NULL UNIQUE
);
```

DEPENDENCIES –

- `alignment_id` → `name`
  - The `alignment_id` uniquely determines the `name`.

SAMPLE DATA-

	alignment_id [PK] integer	name text
1	1	Good
2	2	Evil
3	3	Neutral

Power Level History Table - shows fluctuation of a characters power level over time.

```
CREATE TABLE PowerLevelHistory (
    character_id INT REFERENCES Character(character_id),
    date DATE,
    power_level INT CHECK (power_level >= 0),
    PRIMARY KEY (character_id, date)
);
```

DEPENDENCIES –

- `character_id, date → power_level`
  - The combination of `character_id` and `date` uniquely determines the `power_level`.
  - This is a composite key.

SAMPLE DATA-

	character_id [PK] integer	date [PK] date	power_level bigint
1	1	1984-01-01	416
2	1	1989-01-01	150000000
3	1	1995-01-01	3000000000
4	2	1984-01-01	18000
5	2	1990-01-01	120000000
6	4	1989-01-01	1000
7	4	1993-01-01	800000000

Mentorship table - some characters require special training from an expert to develop their skills, this table documents what characters taught/were taught and when.

```
CREATE TABLE Mentorship (
    mentor_id INT REFERENCES Character(character_id),
    student_id INT REFERENCES Character(character_id),
    start_date DATE,
    end_date DATE,
    PRIMARY KEY (mentor_id, student_id)
);
```

DEPENDENCIES –

- `mentor_id, student_id → start_date, end_date`
    - The combination of `mentor_id` and `student_id` uniquely determines the `start_date` and `end_date`.
    - This is a composite key.

SAMPLE DATA-

	mentor_id [PK] integer	student_id [PK] integer	start_date date	end_date date
1	3	1	1984-01-01	1989-01-01
2	1	4	1990-01-01	1995-01-01
3	1	8	1984-01-01	1985-01-01

Character Team table - the teams a character is a part of and when they joined or left.

```
CREATE TABLE CharacterTeam (
    character_id INT REFERENCES Character(character_id),
    team_id INT REFERENCES Team(team_id),
    join_date DATE,
    leave_date DATE,
    PRIMARY KEY (character_id, team_id)
);
```

DEPENDENCIES –

- `character_id, team_id → join_date, leave_date`
    - The combination of `character_id` and `team_id` uniquely determines the `join_date` and `leave_date`.
    - This is a composite key.

SAMPLE DATA-

	character_id [PK] integer	team_id [PK] integer	join_date date	leave_date date
1	1	1	1986-01-01	[null]
2	2	1	1987-01-01	[null]
3	3	1	1984-01-01	[null]
4	4	1	1989-01-01	[null]
5	8	1	1984-01-01	[null]
6	9	1	1986-01-01	[null]

Team table - lists of possible teams a character could be on.

```
CREATE TABLE Team (
    team_id SERIAL PRIMARY KEY,
    name TEXT NOT NULL UNIQUE,
    description TEXT
);
```

DEPENDENCIES –

- `team_id` → `name`, `description`
  - The `team_id` uniquely determines the `name` and `description`.

SAMPLE DATA-

	team_id [PK] integer	name text	description text
1	1	Z Fighters	The main group of heroes protecting Earth.
2	2	Ginyu Force	Friezas elite fighting squad.
3	3	Universe 7 Team	Team for the Tournament of Power.

Transformation table - list of possible transformations a character could have.

```
CREATE TABLE Transformation (
    transformation_id SERIAL PRIMARY KEY,
    name TEXT NOT NULL,
    description TEXT
);
```

DEPENDENCIES –

- `transformation_id` → `name`, `description`
    - The `transformation_id` uniquely determines the `name` and `description`.

SAMPLE DATA-

	transformation_id [PK] integer	name text	description text
1	1	Super Saiyan	The first and most iconic Saiyan transformation.
2	2	Super Saiyan 2	An advanced version of Super Saiyan.
3	3	Super Saiyan 3	A further advanced and powerful form.
4	4	Super Saiyan God	A divine transformation.
5	5	Super Saiyan Blue	A combination of Super Saiyan and Super Saiyan Go...

Character Transformation table - documents when a specific character transformed and what their transformation was.

```
CREATE TABLE CharacterTransformation (
    character_id INT REFERENCES Character(character_id),
    transformation_id INT REFERENCES Transformation(transformation_id),
    date_achieved DATE,
    PRIMARY KEY (character_id, transformation_id)
);
```

DEPENDENCIES –

- `character_id, transformation_id → date_achieved`
    - The combination of `character_id` and `transformation_id` uniquely determines the `date_achieved`.
    - This is a composite key.

SAMPLE DATA-

	character_id [PK] integer	transformation_id [PK] integer	date_achieved date
1	1	1	1989-01-01
2	2	1	1990-01-01
3	1	2	1992-01-01
4	4	2	1993-01-01
5	1	3	1995-01-01
6	1	4	2015-01-01
7	1	5	2015-02-01
8	2	5	2015-02-01
9	2	4	2015-02-01

Transformation Requirement table - some transformations require certain triggers in order to enable them; this table shows the minimum power level and required emotion to trigger a transformation

```
CREATE TABLE TransformationRequirement (
    transformation_id INT PRIMARY KEY REFERENCES Transformation(transformation_id),
    min_power_level INT CHECK (min_power_level >= 0),
    required_emotion TEXT
);
```

DEPENDENCIES –

- `transformation_id` → `min_power_level`, `required_emotion`
    - The `transformation_id` uniquely determines the `min_power_level` and `required_emotion`.

SAMPLE DATA-

	transformation_id [PK] integer	min_power_level bigint	required_emotion text
1	1	50000	Rage
2	2	1000000	Intense Training
3	3	4000000000	Intense Training
4	4	10000000000	Divine Ki
5	5	100000000000	Divine Ki and Super Saiyan

Fusion table - Lists various fusions between characters and the dates in which they happened.

```
CREATE TABLE Fusion (
    fusion_id SERIAL PRIMARY KEY,
    character_1_id INT REFERENCES Character(character_id),
    character_2_id INT REFERENCES Character(character_id),
    fused_character_id INT REFERENCES
Character(character_id),
    method TEXT,
    date DATE
);
```

DEPENDENCIES –

- `fusion_id` → `character_1_id`, `character_2_id`, `fused_character_id`, `method`, `date`
    - The `fusion_id` uniquely determines all other attributes of a fusion event.

SAMPLE DATA-

	fusion_id [PK] integer	character_1_id integer	character_2_id integer	fused_character_id integer	method text	date date
1	3	1	2	[null]	Potara Earrings	1995-01-01
2	1	1	2	12	Fusion Dance	1995-01-01
3	2	1	2	12	Fusion Dance	1995-01-01

Character Technique table - the direct link between a character and the specific technique they use

```
CREATE TABLE CharacterTechnique (
    character_id INT REFERENCES Character(character_id),
    technique_id INT REFERENCES Technique(technique_id),
    first_used_date DATE,
    mastery_level TEXT CHECK (mastery_level IN ('beginner', 'intermediate', 'mastered')),
    PRIMARY KEY (character_id, technique_id)
);
```

DEPENDENCIES –

- `character_id, technique_id → first_used_date, mastery_level`
    - The combination of `character_id` and `technique_id` uniquely determines the `first_used_date` and `mastery_level`.
    - This is a composite key.

SAMPLE DATA-

	character_id [PK] integer	technique_id [PK] integer	first_used_date date	mastery_level text
1	1	1000111	1986-01-01	mastered
2	2	1000112	1986-01-01	mastered
3	3	1000113	1986-01-01	mastered
4	1	1000115	1987-01-01	mastered
5	1	1000117	1987-04-01	mastered
6	4	1000111	1990-01-01	intermediate
7	4	1000114	1990-01-01	mastered
8	1	1000116	1992-05-01	mastered
9	1	1000118	1995-01-01	mastered
10	2	1000111	1995-01-01	mastered

Technique table - names of the moves within the techniques

```
CREATE TABLE Technique (
    technique_id SERIAL PRIMARY KEY,
    name TEXT NOT NULL UNIQUE,
    description TEXT,
    technique_type_id INT REFERENCES
TechniqueType(technique_type_id),
    energy_cost INT CHECK (energy_cost >= 0),
    cooldown_time INT CHECK (cooldown_time >= 0),
    range TEXT CHECK (range IN ('short', 'mid', 'long'))
);
```

DEPENDENCIES –

- `technique_id` → `name`, `description`, `technique_type_id`, `energy_cost`, `cooldown_time`, `range`
    - The `technique_id` uniquely determines all other attributes of a technique.

SAMPLE DATA-

	technique_id [PK] integer	name text	description text	technique_type_id integer	energy_cost integer	cooldown_time integer	range text
1	1000111	Kamehameha	A powerful energy beam attack.	1	50	2	long
2	1000112	Galick Gun	Vegetas signature energy wave.	1	45	3	long
3	1000113	Special Beam Cannon	A drilling energy beam attack.	1	60	5	mid
4	1000114	Masenko	A rapid fire energy beam attack.	1	30	1	mid
5	1000115	Spirit Bomb	An energy sphere attack.	1	100	10	long
6	1000116	Dragon Fist	A powerful melee attack.	2	40	4	short
7	1000117	Instant Transmission	The ability to teleport.	3	20	0	short
8	1000118	Fusion Dance	A dance to fuse two characters.	3	0	30	short

Technique Type table - the specific names of individual *types* of techniques

```
CREATE TABLE TechniqueType (
    technique_type_id SERIAL PRIMARY KEY,
    name TEXT NOT NULL UNIQUE
);
```

DEPENDENCIES –

- `technique_type_id` → `name`
    - The `technique_type_id` uniquely determines the `name`.

SAMPLE DATA-

	technique_type_id [PK] integer	name text
1	1	Energy Projection
2	2	Melee Attack
3	3	Support
4	4	Transformation

VIEWS

VIEW 1

Saiyan transformation is an extremely important piece of what makes Dragon Ball Z so popular. With every transformation the individual characters experience a boost of power. Here we can see what character transformed, which form they took and the date of transformation.

```
CREATE VIEW SaiyanTransformations AS
SELECT
    c.name AS saiyan_name,
    t.name AS transformation_name,
    ct.date_achieved AS transformation_date
FROM
    Character c
JOIN
    CharacterTransformation ct ON c.character_id =
ct.character_id
JOIN
    Transformation t ON ct.transformation_id =
t.transformation_id
WHERE
    c.race_id = (SELECT race_id FROM Race WHERE
race_name = 'Saiyan');
```

	saiyan_name text	transformation_name text	transformation_date date
1	Goku	Super Saiyan	1989-01-01
2	Vegeta	Super Saiyan	1990-01-01
3	Goku	Super Saiyan 2	1992-01-01
4	Gohan	Super Saiyan 2	1993-01-01
5	Goku	Super Saiyan 3	1995-01-01
6	Goku	Super Saiyan God	2015-01-01
7	Goku	Super Saiyan Blue	2015-02-01
8	Vegeta	Super Saiyan Blue	2015-02-01
9	Vegeta	Super Saiyan God	2015-02-01

VIEW 2

Another unique trait of these characters are the techniques that they are able to master. As we can see here, only Vegeta is seen being able to do Galick Gun, and only Piccolo can do Special Beam Cannon.

```
CREATE VIEW TechniqueUsage AS
SELECT
    c.name AS character_name,
    t.name AS technique_name
FROM
    Character c
JOIN
    CharacterTechnique ct ON c.character_id = ct.character_id
JOIN
    Technique t ON ct.technique_id = t.technique_id;
```

	character_name text	technique_name text
1	Goku	Kamehameha
2	Vegeta	Galick Gun
3	Piccolo	Special Beam Cannon
4	Goku	Spirit Bomb
5	Goku	Instant Transmission
6	Gohan	Kamehameha
7	Gohan	Masenko
8	Goku	Dragon Fist
9	Goku	Fusion Dance
10	Vegeta	Kamehameha

VIEW 3

The main characters in this show are typically seen as the most powerful, especially Goku as seen represented in the character power level table. Each character varies in their power level, and oftentimes the power level is detected by others that have special equipment or have honed in their skills; the more powerful you are, the more easily detected you are by others (unless you know how to hide it).

```
CREATE VIEW CharacterPowerLevels AS
SELECT
    c.name AS character_name,
    plh.power_level AS current_power_level
FROM
    Character c
JOIN
    PowerLevelHistory plh ON c.character_id = plh.character_id
WHERE
    plh.date = (SELECT MAX(date) FROM PowerLevelHistory WHERE
character_id = c.character_id);
```

	character_name text	current_power_level bigint
1	Goku	3000000000
2	Vegeta	120000000
3	Gohan	800000000

REPORTS

REPORT 1

The main characters in this show are typically seen as the most powerful, especially Goku as seen represented in the character power level table. Each character varies in their power level, and oftentimes the power level is detected by others that have special equipment or have honed in their skills; the more powerful you are, the more easily detected you are by others (unless you know how to hide it).

```
SELECT
    c.name AS character_name,
    t.name AS technique_name,
    tt.name AS technique_type,
    ct.mastery_level
FROM
    Character c
JOIN
    CharacterTechnique ct ON c.character_id = ct.character_id
JOIN
    Technique t ON ct.technique_id = t.technique_id
JOIN
    TechniqueType tt ON t.technique_type_id =
tt.technique_type_id
ORDER BY
    c.name, t.name;
```

	character_name text	technique_name text	technique_type text	mastery_level text
1	Gohan	Kamehameha	Energy Projection	intermediate
2	Gohan	Masenko	Energy Projection	mastered
3	Goku	Dragon Fist	Melee Attack	mastered
4	Goku	Fusion Dance	Support	mastered
5	Goku	Instant Transmission	Support	mastered
6	Goku	Kamehameha	Energy Projection	mastered
7	Goku	Spirit Bomb	Energy Projection	mastered
8	Piccolo	Special Beam Cannon	Energy Projection	mastered
9	Vegeta	Galick Gun	Energy Projection	mastered
10	Vegeta	Kamehameha	Energy Projection	mastered

REPORT 2

The main characters in this show are typically seen as the most powerful, especially Goku as seen represented in the character power level table. Each character varies in their power level, and oftentimes the power level is detected by others that have special equipment or have honed in their skills; the more powerful you are, the more easily detected you are by others (unless you know how to hide it).

```
SELECT
    c1.name AS mentor_name,
    c2.name AS student_name,
    m.start_date,
    m.end_date
FROM
    Mentorship m
JOIN
    Character c1 ON m.mentor_id = c1.character_id
JOIN
    Character c2 ON m.student_id = c2.character_id
ORDER BY
    mentor_name, student_name;
```

	mentor_name text	student_name text	start_date date	end_date date
1	Goku	Gohan	1990-01-01	1995-01-01
2	Goku	Krillin	1984-01-01	1985-01-01
3	Piccolo	Goku	1984-01-01	1989-01-01

REPORT 3

The main characters in this show are typically seen as the most powerful, especially Goku as seen represented in the character power level table. Each character varies in their power level, and oftentimes the power level is detected by others that have special equipment or have honed in their skills; the more powerful you are, the more easily detected you are by others (unless you know how to hide it).

```
SELECT
    r.race_name,
    AVG(plh.power_level) AS average_power_level
FROM
    Character c
JOIN
    PowerLevelHistory plh ON c.character_id = plh.character_id
JOIN
    Race r ON c.race_id = r.race_id
WHERE plh.date = (SELECT MAX(date) FROM PowerLevelHistory
WHERE character_id = c.character_id)
GROUP BY
    r.race_name
ORDER BY
    AVG(plh.power_level) DESC;
```

	race_name text	average_power_level numeric
1	Saiyan	1306666666.66666667
2	Namekian	408.0000000000000000
3	Human	100.0000000000000000

STORED PROCEDURES

STORED PROCEDURE

```
CREATE OR REPLACE FUNCTION GetTransformationsByCharacter (
    p_character_name TEXT
)
RETURNS TABLE (transformation_name TEXT, date_achieved DATE)
LANGUAGE plpgsql
AS $$
BEGIN
  RETURN QUERY
    SELECT
        t.name,
        ct.date_achieved
    FROM
        Character c
    JOIN
        CharacterTransformation ct ON c.character_id = ct.character_id
    JOIN
        Transformation t ON ct.transformation_id = t.transformation_id
    WHERE
        c.name = p_character_name
    ORDER BY
        ct.date_achieved;
END;
$$;
```

Returns every transformation of a specific character

	transformation_name text	date_achieved date
1	Super Saiyan	1989-01-01
2	Super Saiyan 2	1992-01-01
3	Super Saiyan 3	1995-01-01
4	Super Saiyan God	2015-01-01
5	Super Saiyan Blue	2015-02-01

```
—calling on the procedure—
SELECT *
FROM GetTransformationsByCharacter('Goku');
```

TRIGGERS

TRIGGER 1

```
CREATE OR REPLACE FUNCTION prevent_future_birthdate()
    RETURNS TRIGGER
    LANGUAGE plpgsql
AS $$
BEGIN
    IF NEW.date_of_birth > CURRENT_DATE THEN
        RAISE EXCEPTION 'Date of birth cannot be in the future';
    END IF;
    RETURN NEW;
END;
$$;

CREATE TRIGGER check_future_birthdate
    BEFORE INSERT OR UPDATE ON Character
    FOR EACH ROW
    EXECUTE PROCEDURE prevent_future_birthdate();
```

Triggers like these save us from headaches in the future, as one wrong input could completely mess up the data we've done so far. Without this trigger we have a higher chance of our database crashing or experiencing problems later on.

```
ERROR:  Date of birth cannot be in the future
CONTEXT:  PL/pgSQL function prevent_future_birthdate() line 4 at RAISE
```

—testing trigger—

```
INSERT INTO Character (name, race_id, home_planet_id, alignment_id, date_of_birth)
VALUES ('TestCharacter2', 1, 1, 1, '2026-01-01');  -- Replace with valid race_id, etc.
```

TRIGGER 2

```
CREATE OR REPLACE FUNCTION set_default_is_alive()
    RETURNS TRIGGER
    LANGUAGE plpgsql
AS $$
BEGIN
    IF NEW.is_alive IS NULL THEN
        NEW.is_alive := TRUE;
    END IF;
    RETURN NEW;
END;
$$;

CREATE TRIGGER default_character_alive
    BEFORE INSERT ON Character
    FOR EACH ROW
    EXECUTE PROCEDURE set_default_is_alive();
```

This trigger is Nice because it saves the hassle of inputing the extra information on the character, though it does need improving on. What if the character is actually dead right now? A lot of times they die but are brought back.

	name text	is_alive boolean
1	TestDefaultAlive	true

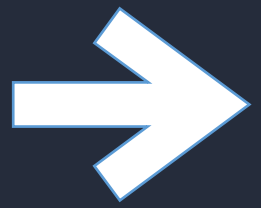
```
—testing trigger—
INSERT INTO Character (name, race_id, home_planet_id, alignment_id, date_of_birth)
VALUES ('TestDefaultAlive', 1, 1, 1, '2000-01-01');

SELECT name, is_alive FROM Character WHERE name = 'TestDefaultAlive';
```

SECURITY

SECURITY

```
Create role fighter;
GRANT SELECT ON Character TO fighter;
GRANT SELECT ON Race TO fighter;
GRANT SELECT ON Planet TO fighter;
GRANT SELECT ON Alignment TO fighter;
GRANT SELECT ON Technique TO fighter;
GRANT SELECT ON TechniqueType TO fighter;
GRANT SELECT ON CharacterTechnique TO fighter;
GRANT SELECT ON PowerLevelHistory TO fighter;
GRANT SELECT ON Transformation TO fighter;
GRANT SELECT ON CharacterTransformation TO fighter;
GRANT SELECT ON Team TO fighter;
GRANT SELECT ON CharacterTeam TO fighter;
```



In order to improve on their current technique, or to look for potential opponents, fighters would likely access any specific information regarding technique, power level and general character info.

```
Create role researcher;
GRANT SELECT ON ALL TABLES IN SCHEMA public TO researcher;
GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA public TO researcher;
```

Researchers, on the other hand, should have access to all possible information in order to have a full picture of what they're studying.

```
REVOKE INSERT ON PowerLevelHistory FROM fighter;
```

Maybe having too much information could be considered dirty…if the other guy can't prepare for what he's up against then neither should the fighter accessing the system!

-KNOWN PROBLEMS-

While the program is functioning in most all individual pieces, there could be added information to make The entire picture clearer in the Dragon Ball Z Universe. Not only this, but fussy syntax errors need to be Resolved with certain tables. There could also be future problems with the default trigger if a character isn't Actually alive but it isn't inputted into the table that they are deceased.

-FUTURE ENHANCEMENTS-

- A table for battles that indicates who went against who as well as wins/losses
- Advanced triggers such as being able to pull up the character with the highest power level at any given time
- Documenting relationships such as father/mother/son/daughter
- Having a column for the amount of episodes a character appeared in


DRAGON BALL Z