

-- Create a players table to store player information.

CREATE TABLE players (

player\_id SERIAL PRIMARY KEY,

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL, -- Assuming each player's email is unique.

register\_date TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP

);

-- Create a levels table to store game level details.

CREATE TABLE levels (

level\_id SERIAL PRIMARY KEY,

level\_name VARCHAR(255) NOT NULL,

difficulty VARCHAR(50) -- This is just an example; you can have more columns as per your requirements.

);

-- Create a gaming\_progress table to store each player's progress in different levels.

CREATE TABLE gaming\_progress (

progress\_id SERIAL PRIMARY KEY,

player\_id INT REFERENCES players(player\_id), -- Foreign key to players table

level\_id INT REFERENCES levels(level\_id), -- Foreign key to levels table

score INT NOT NULL DEFAULT 0, -- Store player's score for this level

completion\_date TIMESTAMP,

UNIQUE(player\_id, level\_id) -- Each player should have only one score entry per level.

);

-- Create a leaderboard view for quick access to top players by score for each level.

CREATE VIEW leaderboard AS

SELECT

p.first\_name,

p.last\_name,

l.level\_name,

gp.score

FROM

players p

JOIN

gaming\_progress gp ON p.player\_id = gp.player\_id

JOIN

levels l ON l.level\_id = gp.level\_id

ORDER BY

gp.score DESC, -- Order primarily by score

gp.completion\_date ASC -- If scores are tied, the player who completed earlier is ranked higher.

LIMIT 10; -- Top 10 scores; adjust as required.







