Entities:

- **User:** Stores our site's user data. We assume each user will only link to one Riot account.
- **Player:** Represents players from Valorant matches. Cannot be a User because not every Valorant player is a user of our site.
- **Pro_Player:** A pro player is still a Player, but has more attributes that a normal player won't have on our site (Social media links, Twitch, Pro team, profile picture, etc). We assume that a pro player does not switch teams throughout their career.
- **Game:** Stores information about the game and acts as a way to relate other entities. Date will be a unix timestamp that sorts the games by when it was played.
- **Player_Stats:** Holds a player's stats for each game that they play. Each row will contain the team_side (not representing the Pro Player's team), represented by whether or not the team started on the attacking side or defending side (0 or 1 respectively).
- **Maps:** Holds the average win rates for each map and has a map_name to relate the Blitz.gg data with the Riot API and VLR.gg data. We assume that there are no collisions between map names and we store it as a string.
- **Weapons:** Each row contains the weapon data for each map and rank, which are the other two primary keys to form a superkey. It is related back to Maps and therefore the other datasets using the map_name.
- **Agents:** This contains data about how each Valorant agent performs in a particular map and rank.

One-to-one relationships:
User -> Player (a user can only have one Valorant account linked)

Pro_Player -> Player (a pro player is still a player)

One-to-many relationships:

Player -> Player_Stats (each row in Player_Stats represents a game that the player played in)

Game -> Player_Stats (for every game, there will be 10 row in Player_Stats for the individual players)

Many-to-many relationships:

Maps -> Weapons, Agents, Games (many different weapons, agents, games played on a couple different maps)

Agents-> Player_Stats (for every row, one agent was played in one game, but we have a lot of games for the same agent)

Functional Dependencies

Table: Weapons

Map, rank, weapon_name -> body_shot_p, leg_shot_p, head_shot_p, dmg_per_round, kill_per_match

- Each attribute is atomic
- primary key is combination of map, rank, weapon_name and all other attributes depend on this entire combination
- No attributes that depend on non-key attributes, all depend on primary key

Table: Maps

Map_name -> win_rate_attacks

- Each attribute is atomic
- primary key is Map_name
- No attributes that depend on non-key attributes, all depend on primary key

Table: Agents

agent_name, map_name, rank -> kills, deaths, assists, win_rate, pick_rate, first_blood, num_matches, q_usage, w_usage, e_usage, r_usage

- Each attribute is atomic
- primary key is combination of agent_name, map_name, rank and all other attributes depend on this entire combination
- No attributes that depend on non-key attributes, all depend on primary key

Table: Game

Game_id -> team_id_start_t, team_id_start_ct, map, date

- Each attribute is atomic
- primary key is game_id
- No attributes that depend on non-key attributes, all depend on primary key

Table: Individual Stats

player_id, game_id, team_id -> rank, agent, kills, deaths, assists, kill_assist_trade_survive, combat_score, dmg_per_round, head_shot_ratio, first_blood, first_death

- Each attribute is atomic
- The primary key is the combination of player_id, game_id, team_id and all other attributes depending on this entire combination. Team_id represents starting side (attack/ defend) during that game, not a specific team_name like Faze or Cloud
- No attributes that depend on non-key attributes, all depend on primary key

Table: Player

Player_id -> riot_id, current_rank

- Each attribute is atomic
- primary key is player_id
- No attributes that depend on non-key attributes, all depend on primary key

Table: Pro Player

player_id -> twitch, team

- Each attribute is atomic
- primary key is player_id
- No attributes that depend on non-key attributes, all depend on primary key

User

User_id -> riot_api_credentials, player_id, pro_lookalike

- Each attribute is atomic
- primary key is User_id
- No attributes that depend on non-key attributes, all depend on primary key

Relational Schema

Player_Stats(player_id: int [PK], game_id: int [PK], team_id: int [PK], rank: varchar(20), agent: varchar(20), kills: int, deaths: int, assists: int, kill_assist_trade_survive: int, combat_score: int, dmg_per_round: int, head_shot_ratio: int, first_blood: int, first_death: int)

Player(player_id: int [PK], riot_id: varchar(20), current_rank: int)

Pro_Player(player_id: int [PK], riot_id: int, current_rank: varchar(20))

User(user_id: int [PK], player_id: int, riot_api_credentials: varchar(256), pro_lookalike: varchar(20))

Game(game_id: int [PK], team_id_start_t: int, team_id_start_ct: int, map: varchar(20), date: varchar(20))

Weapons(map: varchar(20) [PK], rank: varchar(20) [PK], weapon_name: varchar(20), body_shot_p: float, leg_shot_p: float, head_shot_p: float, dmg_per_round: int, kill_per_match: float)

Agents(agent_name: varchar(20) [PK] , map_name: varchar(20) [PK], rank: varchar(20) [PK], kills: float, deaths: float, assists: float, win_rate: float, pick_rate: float, first_blood: int, num_matches: int, q_usage: float, w_usage: float, e_usage: float, r_usage: float)

Maps(map_name: varchar(20), win_rate_attacks: float)