AchievementUnlocked
steam_name
achievement_name
date

Player
name
num_battle_pts
skin_gender
skin_color
skin_face_type
skin_hair_type
steam_name
hrs_played

Trade
id
posted_by
date_posted
time_posted

TradeItem
trade_id
item_name
num_items
bid_price
buyout_price
steam_price
currency

RareItem
name
drop_chance
category
description

Achievement
name
description

SkinFaceLibrary
type
description

SkinHairLibrary
type
description

Roster
player_name*
game_id
num_kills**
battle_pts_earned**
player_rank**

*None if the player chose not to display his/her name in public games

**Roster tracks a specific player's stats for a specific game

RareItemCategory
name
description

ChickenDinner
player_name
game_id

Game
id
date
start_time
end_time
category
perspective
region
num_players

GameCategory
name
description

Ranking*
player_name
rank
game_category
game_perspective
region

*A player can be listed in one or multiple rankings, depending on if he/she plays in multiple game categories (Solo, Duo, Squad, 1-Man Squad), multiple game perspectives (FPP, TPP), or multiple regions (NA, EU, AS)

GamePerspective
type
description

DESCRIPTION

We propose a database for an online shooter game called PlayerUnknown's Battlegrounds (PUBG) that works similarly to World of Warcraft Armory, where every player's PUBG character is publicly viewable (as long as they have played at least one game). The database is simply called **PUBG Armory.** The features to be displayed on each player's profile are:

- a 3D skin (skin hair and skin face types are separate entities in the ERD since each of the types comes with a description);
- videos of the best games, i.e. the games that the player has chosen to make public for PUBG Armory. The videos can be viewed from any player's perspective as long as that player is in the roster for the game (where players can also choose to hide their names from any of the videos that they did not share themselves);
- an auction where the player can place trades for rare apparel items, keys, or crates (Trade, Item, ItemCategory);
- the player's Steam name, hours played and achievements in PUBG (both tied to the Steam account);
- the player's rank in each of the game categories, perspectives, and regions.

All the assumptions are included in the ERD. One general assumption is that for the cardinalities of GameCategory, GamePerspective, SkinFaceLibrary, SkinHairLibrary, and Achievement, they are all one-to-many (vs. zero-to-many) on the outer end because there are so many players in PUBG with so many games played.

AchievementUnlocked

PK,FK	steam_name
PK,FK	achievement_name
	date

Achievement

PK	name
	description

Player

PK	name
	num_battle_pts
	skin_gender
	skin_color
	skin_hair_type
	skin_face_type
	steam_name
	hrs_played

Trade

PK	id
FK	posted_by
	date_posted
	time_posted

TradeItem

PK,FK	trade_id
PK,FK	item_name
	num_items
	bid_price
	buyout_price
	steam_price
	currency

Ra...

PK	
FK	

RareIte...

PK	

Roster

PK,FK	player_name
PK,FK	game_id
	num_kills
	battle_pts_earned
	player_rank

SkinFaceLibrary

PK	type
	description

SkinFaceChange

PK,FK	player_name
PK	date_changed
FK	skin_face_type

SkinHairLibrary

PK	type
	description

SkinHairChange

PK,FK	player_name
PK	date_changed
FK	skin_hair_type

ChickenDinner

PK,FK	player_name
PK,FK	game_id

Game

PK	id
	date
	start_time
	end_time
FK	category
FK	perspective
	region
	num_players

GameCategory

PK	name
	description

Ranking

PK,FK	player_name
	rank
PK,FK	game_category
PK,FK	game_perspective
PK	region

GamePerspective

PK	type
	description

RareItem	
PK	name
	drop_chance
FK	category
	description

RareItemCategory	
PK	name
	description

```
# Query 1: "What's the ratio of each game category over total games played in each of the regions?"

select G.region, ifnull(solo_count, 0) / (ifnull(solo_count, 0)+ifnull(duo_count, 0)+ifnull(squad_count, 0)) as solo_game_ratio, ifnull(duo_count, 0) / (ifnull(solo_count,
0)+ifnull(duo_count, 0)+ifnull(squad_count, 0)) as duo_game_ratio, ifnull(squad_count, 0) / (ifnull(solo_count, 0)+ifnull(duo_count, 0)+ifnull(squad_count, 0)) as
 squad_game_ratio
from Game G
join (select region, count(*) solo_count
from Game
where category like 'Solo'
group by region) RegionSoloCount
    on G.region = RegionSoloCount.region
left join (select region, count(*) duo_count
from Game
where category like 'Duo'
group by region) RegionDuoCount
    on G.region = RegionDuoCount.region
left join (select region, count(*) squad_count
from Game
where category like 'Squad'
group by region) RegionSquadCount
    on G.region = RegionSquadCount.region
group by G.region;

# Real World Value: The results of this query can be potentially sold by us to PUBG's developers to help them optimize the servers (in terms of traffic) for each of the game
 categories in each of the regions.
```

region	solo_game_ratio	duo_game_ratio	squad_game_ratio
Asia	0.2857	0.4286	0.2857
Europe	0.5000	0.0000	0.5000
North America	0.5556	0.2222	0.2222

```sql
# Query 2: "What's the average number of Battle Points earned per hour played?"

select sum(num_battle_pts) / sum(hrs_played) avg_num_bps_per_hr_played
from Player;

# Real World Value: The results of this query can be potentially sold by us to PUBG's developers to help them price the in-game items that can be purchased via BPs more
  accurately.
```

avg_num_bps_per_hr_played
15.9392

```
# Query 3: "What's the most trending item on the PUBG Armory auction?"

select item_name as most_trending_trade_item
from (select item_name, sum(num_items) num_appearances
from TradeItem
group by item_name) ItemCount
order by num_appearances DESC
limit 1;

# Real World Value: The results of this query can be potentially sold by us to Steam to help them determine more accurately the supply for each of the PUBG tradable
 items on third-party marketplaces (like ours) - and thus to price those items more accurately on their Steam marketplace.
```

most_trending_trade_item	
Leather Boots (Brown)	

```sql
# Query 4: "What's the average bid price for each of the items on the PUBG Armory auction?"

select item_name, avg(bid_price/num_items) avg_bid_price, currency
from TradeItem
group by item_name;

# Real World Value: The results of this query can be potentially sold by us to Steam to help them determine more accurately the demand for each of the PUBG tradable
 items on third-party marketplaces (like ours) - and thus to price those items more accurately on their Steam marketplaces.
```

item_name	avg_bid_price	currency
Aviator Sunglasses	1.71	$
Biker Crate	1.66	$
Biker Key	4.63	$
Checkered Jacket	1.24	$
Cloth Mask (Leopard)	1.47	$
Desperado Key	1.59	$
Horn-rimmed Glasses (Black)	2.09	$
Horn-rimmed Glasses (Brown)	2	$
Leather Boots (Brown)	1.91	$
Leather Hoodie (Black)	1.64	$
Long Leather Boots (Brown)	9.82	$
Long-sleeved Leather Shirt	2.65	$
Militia Crate	2.63	$
Punk Knuckle Gloves (Red)	9.17	$
Sleeveless Turtleneck Top (Gray)	2.49	$
Striped Shirt (Gray)	4.91	$
Training Pants (Light Blue)	2	$
Triumph Crate	2.18	$
Triumph Key	2.88	$
Wanderer Crate	1.72	$
Wanderer Key	1.55	$
Wide Pants (Red)	7.47	$

```
# Query 5: "Which is the least famous skin face type amongst the PUBG players in PUBG Armory DB?"

select description as least_famous_face_type
from SkinFaceLibrary
join (select skin_face_type, count(*) skin_face_count
from Player P
group by skin_face_type) SkinFaceCount
    on SkinFaceCount.skin_face_type = SkinFaceLibrary.type
order by skin_face_count ASC
limit 1;


# Real World Value: The results of this query can help PUBG's developers optimize their character customization options (e.g. replace the least famous face type with
 something new) the next time they release a patch.
```

least_famous_face_type

Round Face

```
# Query 6: "Which is the least famous skin hair type amongst the PUBG players in PUBG Armory DB?"

select description as least_famous_hair_type
from SkinHairLibrary
join (select skin_hair_type, count(*) skin_hair_count
from Player P
group by skin_hair_type) SkinHairCount
    on SkinHairCount.skin_hair_type = SkinHairLibrary.type
order by skin_hair_count ASC
limit 1;

# Real World Value: The results of this query can help PUBG's developers optimize their character customization options (e.g. replace the least famous hair type with
 something new) the next time they release a patch.
```

least_famous_hair_type
Undercut