University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: July 24 2025

Group Number: 29

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Nancy Ma	94979507	yma60	nancyma12321@gmail.com
Sakura Sekiguchi	13028428	s5g9e	sakuras4649@gmail.com
Jennifer Park	27748714	jpark81	jpark3apple@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

2. Summary of the project

The domain of our project is in entertainment, specifically one-to-one turn-based games. The database can model the following aspects of the game: the scoring and rewarding system, players' statistics, skills in categories, and avatars. For example, similar games to Street Fighter and Slay the Spire could use the database to help implement their turns and track players' statistics. In each turn, players can select skills from three main categories: attack, defence and agility, and obtain points according to the properties of the categories. For instance, the player who picked attack would lose more points from their health if the other player picked defence, and the damage would depend on the attribute 'DamagePoints' of the skill entity; the player who picked agility would lose more points if the other player picked attack, and agility beats defence. It is similar to rock, paper, scissors, where one move beats the other. The attribute 'EffectPoints' of the skill entity will add points to players' health and is dependent on 'DamagePoints'. The database would keep track of the categories of the skills, players' information and the results of each round to update players' statistics.

3. ER diagram

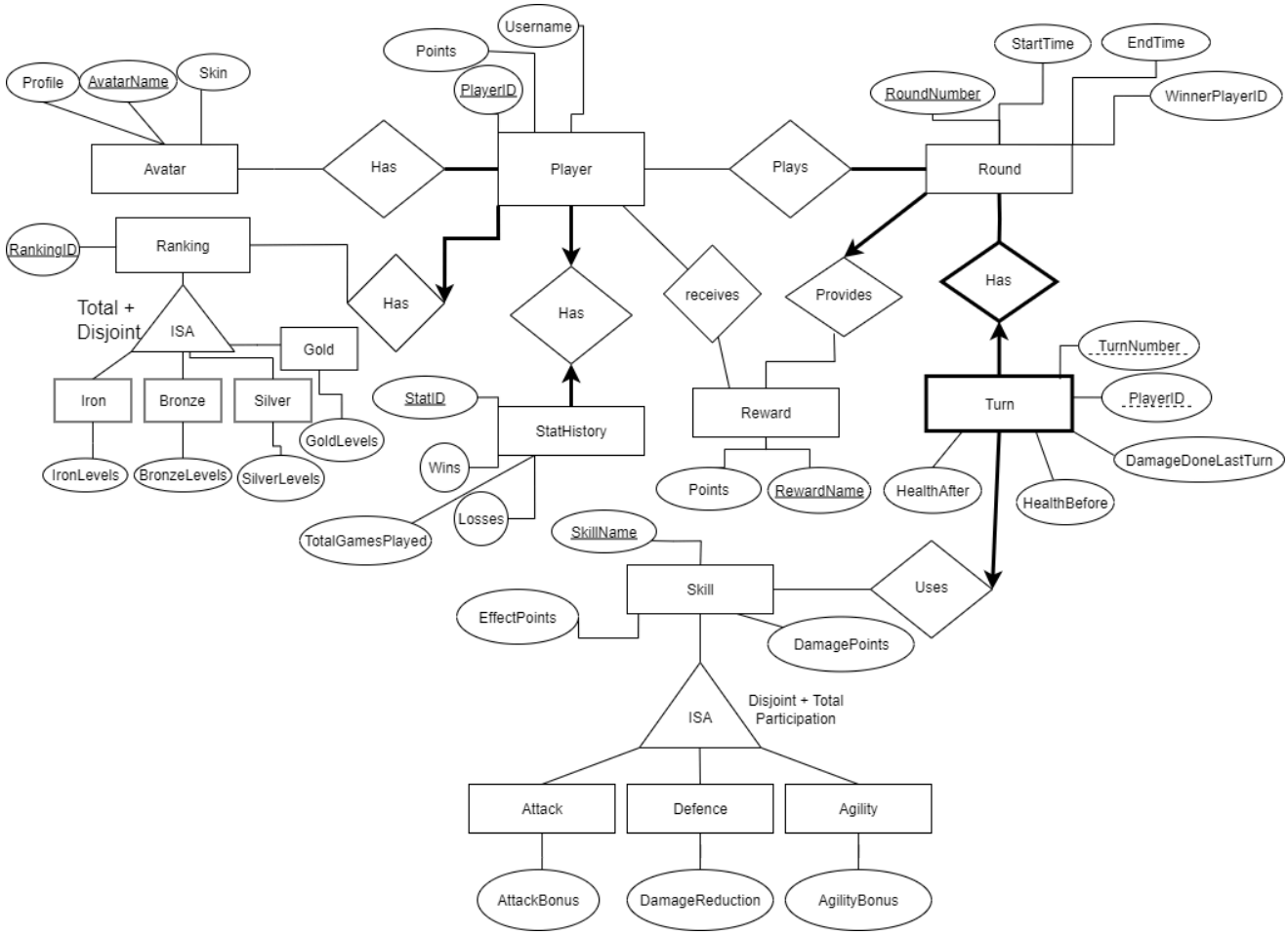
Changes Made:
1. We changed the PlayerID attributes for Ranking and StatHistory entities to RankingID and StatID instead because having the same name in different relations can be confusing.

2. We added the DamagePoints attribute to Skill because we want to have a meaningful functional dependency other than those only involving candidate and primary keys. e.g. DamagePoints -> EffectPoints.

3. We moved the TotalGamesPlayed attribute from Player to StatHistory because we think the attributes (Wins and Losses) for StatHistory go along with TotalGamesPlayed.

4. We changed the Name attributes to AvatarName for Avatar and RewardName for Reward because having the same name in different relations can be confusing.

6. We added the attribute DamageDoneLastTurn to Turn because it can give us a meaningful functional dependency other than those only involving candidate and primary keys, e.g. HealthBefore, DamageDoneLastTurn -> HealthAfter.



4.

Note: we underlined primary keys, bolded **foreign keys,** listed out candidate keys and other constraints under each table with indentation. Subclasses in ISA are in red.

- Avatar (Name: VARCHAR(50), Skin: VARCHAR(50), Profile: VARCHAR(50))
    - Candidate keys: Name, Skin, Profile

- Player_Has_Avatar (**AvatarName**: VARCHAR(50), **PlayerID**: CHAR(10))
    - ASSERTION: a player has at least one avatar
-
- Plays (**RoundNumber**: INTEGER, **PlayerID**: CHAR(10))
    - ASSERTION: a round has at least one player
-
- Round (RoundNumber: INTEGER, StartTime: TIME, EndTime: TIME, WinnerPlayerID: CHAR(10))
-
- Receives (**RewardName**: VARCHAR(50), **PlayerID**: CHAR(10))

- Turn_Has (TurnNumber: INTEGER, PlayerID: CHAR(10), **RoundNumber**: INTEGER, HealthAfter: INTEGER, HealthBefore: INTEGER, DamageDoneLastTurn: INTEGER, **SkillName**: VARCHAR(50))
    - SkillName: NOT NULL and UNIQUE

- Skill (SkillName: VARCHAR(50), EffectPoints: INTEGER, DamagePoints: INTEGER)

- Attack (**SkillName**: VARCHAR(50), AttackBonus: INTEGER)
-
- Defence (**SkillName**: VARCHAR(50), DamageReduction: INTEGER)
-
- Agility (**SkillName**: VARCHAR(50), AgilityBonus: INTEGER)

- Reward (RewardName: VARCHAR(50), Points: INTEGER)

- Round_Provides (RoundNumber: INTEGER, **RewardName**: VARCHAR(50))
    - RewardName: NOT NULL
-
- Player_Has: (PlayerID: CHAR(10), Points: INTEGER, Username: VARCHAR(20), **RankingID: CHAR(10),** StatID:CHAR(10), Wins: INTEGER, Losses: INTEGER, TotalGamesPlayed: INTEGER)
    - Candidate keys: PlayerID, Username, RankingID, StatID
    - StatID:UNIQUE and NOT NULL,
    - RankingID: NOT NULL
-
- Ranking: (RankingID:CHAR(10))
-
- Iron: (**RankingID:CHAR(10)**, IronLevels: VARCHAR(10))
-
- Bronze (**RankingID:CHAR(10)**, BronzeLevels: VARCHAR(10))
-
- Silver: (**RankingID:CHAR(10)**, SilverLevels: VARCHAR(10))
-
- Gold: (**RankingID:CHAR(10)**, GoldLevels: VARCHAR(10))

5. Functional Dependencies for each relation

Avatar (Name: VARCHAR(50), Skin: VARCHAR(50), Profile: VARCHAR(50))
Skin -> Profile, Name
Profile -> Skin, Name
Name -> Profile, Skin

Round (RoundNumber: INTEGER, StartTime: TIME, EndTime: TIME, WinnerPlayerID: CHAR(10))

RoundNumber -> StartTime, EndTime, WinnerPlayerID

Turn_Has (TurnNumber: INTEGER, PlayerID: CHAR(10), **RoundNumber**: INTEGER, HealthAfter: INTEGER, HealthBefore: INTEGER, DamageDoneLastTurn: INTEGER, **SkillName**: VARCHAR(50))
RoundNumber, TurnNumber, PlayerID -> HealthAfter, HealthBefore, DamageDoneLastTurn, SkillName
HealthBefore, DamageDoneLastTurn -> HealthAfter (VIOLATES BCNF)

Round_Provides (RoundNumber: INTEGER, **RewardName**: VARCHAR(50))
RoundNumber -> RewardName

Reward (RewardName: VARCHAR(50), Points: INTEGER)
RewardName -> Points

Player_Has: (PlayerID: CHAR(10), Points: INTEGER, Username: VARCHAR(20), **RankingID: CHAR(10),** StatID:CHAR(10), Wins: INTEGER, Losses: INTEGER, TotalGamesPlayed: INTEGER)
Wins, Losses -> TotalGamesPlayed (VIOLATES BCNF)
PlayerID -> Points, Username, RankingID, StatID, Wins, Losses, TotalGamesPlayed
RankingID -> PlayerID, Points, Username, StatID, Wins, Losses, TotalGamesPlayed
Username -> PlayerID, Points, RankingID, StatID, Wins, Losses, TotalGamesPlayed
StatID -> PlayerID, Points, Username, RankingID, Wins, Losses, TotalGamesPlayed

Skill (SkillName: VARCHAR(50), EffectPoints: INTEGER, DamagePoints: INTEGER)
SkillName -> EffectPoints, DamagePoints
DamagePoints -> EffectPoints (VIOLATES BCNF)

Attack (**SkillName**: VARCHAR(50), AttackBonus: INTEGER)
SkillName -> AttackBonus

Defence (**SkillName**: VARCHAR(50), DamageReduction: INTEGER)
SkillName -> DamageReduction

Agility (**SkillName**: VARCHAR(50), AgilityBonus: INTEGER)
SkillName -> AgilityBonus

Iron: (**RankingID:CHAR(10)**, IronLevels: VARCHAR(10))
RankingID -> IronLevels

Bronze (**RankingID:CHAR(10)**, BronzeLevels: VARCHAR(10))
RankingID -> BronzeLevels

Silver: (**RankingID:CHAR(10),** SilverLevels: VARCHAR(10))
RankingID -> SilverLevels

Gold: (**RankingID:CHAR(10),** GoldLevels: VARCHAR(10))
RankingID -> GoldLevels

Note: the following relations' functional dependencies are omitted because they are trivial.

- Player_Has_Avatar (**AvatarName**: VARCHAR(50), **PlayerID**: CHAR(10))
- Plays (**RoundNumber**: INTEGER, **PlayerID**: CHAR(10))
- Receives (**RewardName**: VARCHAR(50), **PlayerID**: CHAR(10))

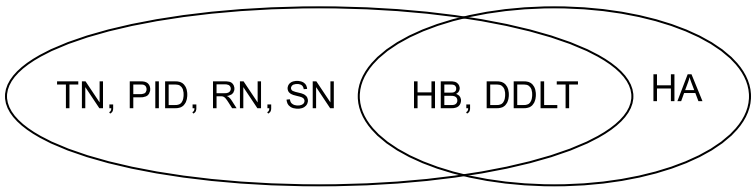
6. Normalization

a) Turn_Has (TurnNumber: INTEGER, PlayerID: CHAR(10), **RoundNumber**: INTEGER, HealthAfter: INTEGER, HealthBefore: INTEGER, DamageDoneLastTurn: INTEGER, **SkillName**: VARCHAR(50))
SkillName: NOT NULL and UNIQUE

RoundNumber, TurnNumber, PlayerID -> HealthAfter, HealthBefore, DamageDoneLastTurn, SkillName
HealthBefore, DamageDoneLastTurn -> HealthAfter (VIOLATES BCNF)

Note: RN: RoundNumber, TN: TurnNumber, PID: PlayerId, HA: HealthAfter, HB: HealthBefore, DDLT: DamageDoneLastTurn, SN: SkillName

R(TN, PID, RN, HA, HB, DDLT, SN)
Use HB, DDLT->HA to decompose



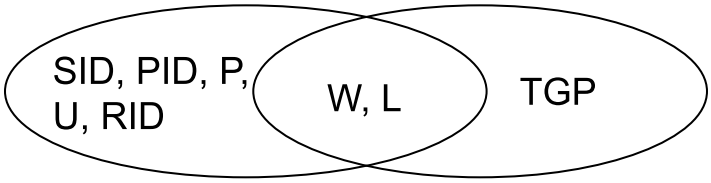
Turn_Has_R1(PlayerID: CHAR(10), **RoundNumber**: INTEGER, TurnNumber: INTEGER, DamageDoneLastTurn: INTEGER, HealthBefore: INTEGER, **SkillName**: VARCHAR(50))
Turn_Has_R2(DamageDoneLastTurn: INTEGER, HealthBefore: INTEGER, HealthAfter: INTEGER)

b) Player_Has: (PlayerID: CHAR(10), Points: INTEGER, Username: VARCHAR(20), **RankingID: CHAR(10),** StatID:CHAR(10), Wins: INTEGER, Losses: INTEGER, TotalGamesPlayed: INTEGER)
Wins, Losses -> TotalGamesPlayed (VIOLATES BCNF)
PlayerID -> Points, Username, RankingID, StatID, Wins, Losses, TotalGamesPlayed
RankingID -> PlayerID, Points, Username, StatID, Wins, Losses, TotalGamesPlayed

Username -> PlayerID, Points, RankingID, StatID, Wins, Losses, TotalGamesPlayed
StatID -> PlayerID, Points, Username, RankingID, Wins, Losses, TotalGamesPlayed

NOTE: SID: statID, W: Wins, L: Losses, TGP: TotalGamesPlayed, PID: PlayerID, P: Points, U: Username, RID: RankingID, SID: StatID

R(PID, P, U, RID, SID, W, L, TGP)
Use W, L->TGP to decompose

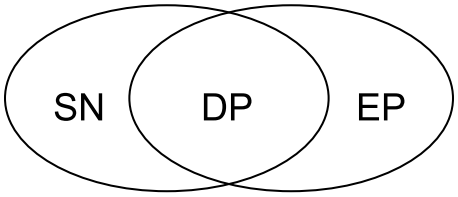


Player_Has_R1(PlayerID: CHAR(10), Points: INTEGER, Username: VARCHAR(20), **RankingID: CHAR(10),** StatID:CHAR(10), Wins: INTEGER, Losses: INTEGER)
Candidate keys: PlayerID, Username, RankingID, StatID
StatID:UNIQUE and NOT NULL,
RankingID: NOT NULL
Player_Has_R2(Wins: INTEGER, Losses: INTEGER, TotalGamesPlayed: INTEGER)

c) Skill (SkillName: VARCHAR(50), EffectPoints: INTEGER, DamagePoints: INTEGER)
SkillName -> EffectPoints, DamagePoints
DamagePoints -> EffectPoints (VIOLATES BCNF)

NOTE: SN: SkillName, DP: DamagePoints, EP: EffectPoints

R(SN, EP, DP)
Use DP->EP to decompose



Skill_R1(SkillName: VARCHAR(50), DamagePoints: INTEGER)
Skill_R2(DamagePoints: INTEGER, EffectPoints: INTEGER)

d) Other tables that don't violate BCNF:
Avatar (Name: VARCHAR(50), Skin: VARCHAR(50), Profile: VARCHAR(50))
Player_Has_Avatar (**AvatarName**: VARCHAR(50), **PlayerID**: CHAR(10))
Round (RoundNumber: INTEGER, StartTime: TIME, EndTime: TIME, WinnerPlayerID: CHAR(10))
Round_Provides (RoundNumber: INTEGER, **RewardName**: VARCHAR(50))

Reward (RewardName: VARCHAR(50), Points: INTEGER)
Player_Has: (PlayerID: CHAR(10), Points: INTEGER, Username: VARCHAR(20), **RankingID: CHAR(10),** StatID:CHAR(10), Wins: INTEGER, Losses: INTEGER, TotalGamesPlayed: INTEGER)
Attack (**SkillName**: VARCHAR(50), AttackBonus: INTEGER)
Defence (**SkillName**: VARCHAR(50), DamageReduction: INTEGER)
Agility (**SkillName**: VARCHAR(50), AgilityBonus: INTEGER)
Iron: (**RankingID:CHAR(10)**, IronLevels: VARCHAR(10))
Bronze (**RankingID:CHAR(10)**, BronzeLevels: VARCHAR(10))
Silver: (**RankingID:CHAR(10)**, SilverLevels: VARCHAR(10))
Gold: (**RankingID:CHAR(10)**, GoldLevels: VARCHAR(10))
Plays (**RoundNumber**: INTEGER, **PlayerID**: CHAR(10))
Receives (**RewardName**: VARCHAR(50), **PlayerID**: CHAR(10))

7. SQL DDL

- Avatar (Name: VARCHAR(50), Skin: VARCHAR(50), Profile: VARCHAR(50))
    - Candidate keys: Name, Skin, Profile

```
CREATE TABLE Avatar (
        Name VARCHAR(50) PRIMARY KEY,
        Skin VARCHAR(50),
        Profile VARCHAR(50));
```

- Player_Has_Avatar (**AvatarName**: VARCHAR(50), **PlayerID**: CHAR(10))
    - ASSERTION: a player has at least one avatar

```
CREATE TABLE Player_Has_Avatar(
        AvatarName VARCHAR(50) NOT NULL,
        PlayerID CHAR(10),
        PRIMARY KEY (AvatarName, PlayerID),
        FOREIGN KEY (AvatarName) REFERENCES Avatar,
        FOREIGN KEY (PlayerID) REFERENCES Player_Has_R1);
```

- Plays (**RoundNumber**: INTEGER, **PlayerID**: CHAR(10))
    - ASSERTION: a round has at least one player

```
CREATE TABLE Plays(
        RoundNumber INTEGER,
        PlayerID CHAR(10),
        PRIMARY KEY (RoundNumber, PlayerID),
```

FOREIGN KEY (RoundNumber) REFERENCES Round,
FOREIGN KEY (PlayerID) REFERENCES Player_Has_R1);

- Round (RoundNumber: INTEGER, StartTime: TIME, EndTime: TIME, WinnerPlayerID: CHAR(10))

```
CREATE TABLE Round(
    RoundNumber INTEGER PRIMARY KEY,
    StartTime TIME,
    EndTime TIME,
    WinnerPlayerID CHAR(10))
```

- Receives (**RewardName**: VARCHAR(50), **PlayerID**: CHAR(10))

```
CREATE TABLE Provides (
    RewardName VARCHAR(50),
    PlayerID CHAR(10),
    PRIMARY KEY(RewardName, PlayerID),
    FOREIGN KEY(RewardName) REFERENCES Reward,
    FOREIGN KEY(PlayerID) REFERENCES Player_Has_R1);
```

Turn_Has_R1(PlayerID: CHAR(10), **RoundNumber**: INTEGER, TurnNumber: INTEGER, DamageDoneLastTurn: INTEGER, HealthBefore: INTEGER, **SkillName**: VARCHAR(50))

- SkillName: NOT NULL and UNIQUE

Turn_Has_R2(DamageDoneLastTurn: INTEGER, HealthBefore: INTEGER, HealthAfter: INTEGER)

```
CREATE TABLE Turn_Has_R1 (
    PlayerID CHAR(10),
    RoundNumber INTEGER,
    TurnNumber INTEGER,
    DamageDoneLastTurn INTEGER,
    HealthBefore INTEGER,
    SkillName VARCHAR(50) NOT NULL UNIQUE,
    PRIMARY KEY(PlayerID, RoundNumber, TurnNumber),
    FOREIGN KEY(RoundNumber) REFERENCES Round,
    FOREIGN KEY(SkillName) REFERENCES Skill_R1);

CREATE TABLE Turn_Has_R2 (
    HealthBefore INTEGER,
    DamageDoneLastTurn INTEGER,
    HealthAfter INTEGER,
```

PRIMARY KEY(HealthBefore, DamageDoneLastTurn));

Skill_R1(SkillName: VARCHAR(50), DamagePoints: INTEGER)
Skill_R2(DamagePoints: INTEGER, EffectPoints: INTEGER)

```
CREATE TABLE Skill_R1 (
        SkillName VARCHAR(50) PRIMARY KEY,
        DamagePoints INTEGER);

CREATE TABLE Skill_R2 (
        DamagePoints INTEGER PRIMARY KEY,
        EffectPoints INTEGER);
```

- Attack (**SkillName**: VARCHAR(50), AttackBonus: INTEGER)

```
CREATE TABLE Attack (
        SkillName VARCHAR(50) PRIMARY KEY,
        AttackBonus INTEGER,
        FOREIGN KEY(SkillName) REFERENCES Skill_R1);
```

- Defence (**SkillName**: VARCHAR(50), DamageReduction: INTEGER)

```
CREATE TABLE Defence (
        SkillName VARCHAR(50) PRIMARY KEY,
        DamageReduction INTEGER,
        FOREIGN KEY(SkillName) REFERENCES Skill_R1);
```

- Agility (**SkillName**: VARCHAR(50), AgilityBonus: INTEGER)

```
CREATE TABLE Agility (
        SkillName VARCHAR(50) PRIMARY KEY,
        AgilityBonus INTEGER,
        FOREIGN KEY(SkillName) REFERENCES Skill_R1);
```

- Reward (RewardName: VARCHAR(50), Points: INTEGER)

```
CREATE TABLE Reward (
        RewardName VARCHAR(50) PRIMARY KEY,
        Points INTEGER);
```

- Round_Provides (RoundNumber: INTEGER, **RewardName**: VARCHAR(50))
    - RewardName: NOT NULL

```
CREATE TABLE Round_Provides (
        RoundNumber INTEGER PRIMARY KEY,
        RewardName VARCHAR(50) NOT NULL,
        FOREIGN KEY(RewardName) REFERENCES Reward
);
```

Player_Has_R1(PlayerID: CHAR(10), Points: INTEGER, Username: VARCHAR(20), **RankingID: CHAR(10),** StatID:CHAR(10), Wins: INTEGER, Losses: INTEGER)

- Candidate keys: PlayerID, Username, RankingID, StatID
- StatID:UNIQUE and NOT NULL,
- RankingID: NOT NULL

Player_Has_R2(Wins: INTEGER, Losses: INTEGER, TotalGamesPlayed: INTEGER)

```
CREATE TABLE Player_Has_R1 (
        PlayerID CHAR(10) PRIMARY KEY,
        Points INTEGER,
        Username VARCHAR(20),
        RankingID CHAR(10) NOT NULL,
        StatID CHAR(10) NOT NULL,
        Wins INTEGER,
        Losses INTEGER,
        UNIQUE (StatID),
        FOREIGN KEY (RankingID) REFERENCES Ranking
);

CREATE TABLE Player_Has_R2 (
        Wins INTEGER,
        Losses INTEGER,
        TotalGamesPlayed INTEGER,
        PRIMARY KEY (Wins, Losses)
);
```

- Ranking: (RankingID:CHAR(10))
-
- Iron: (**RankingID:CHAR(10)**, IronLevels: VARCHAR(10))
-
- Bronze (**RankingID:CHAR(10)**, BronzeLevels: VARCHAR(10))
-
- Silver: (**RankingID:CHAR(10)**, SilverLevels: VARCHAR(10))
-
- Gold: (**RankingID:CHAR(10)**, GoldLevels: VARCHAR(10))

```
CREATE TABLE Ranking (
        RankingID CHAR(10) PRIMARY KEY
);

CREATE TABLE Iron(
        RankingID CHAR(10) PRIMARY KEY,
        IronLevels: VARCHAR(10),
        FOREIGN KEY(RankingID) REFERENCES Ranking
);

CREATE TABLE Bronze (
        RankingID CHAR(10) PRIMARY KEY,
        BronzeLevels: VARCHAR(10),
        FOREIGN KEY(RankingID) REFERENCES Ranking
);

CREATE TABLE Silver (
        RankingID CHAR(10) PRIMARY KEY,
        SilverLevels: VARCHAR(10),
        FOREIGN KEY(RankingID) REFERENCES Ranking
);

CREATE TABLE Gold (
        RankingID CHAR(10) PRIMARY KEY,
        GoldLevels: VARCHAR(10),
        FOREIGN KEY(RankingID) REFERENCES Ranking
);
```

8. INSERT statements

- Avatar (Name: VARCHAR(50), Skin: VARCHAR(50), Profile: VARCHAR(50))
    - Candidate keys: Name, Skin, Profile

```
INSERT INTO Avatar VALUES('Sammie', 'Shy School Girl',
'Age: 16
Traits:
-Collects pressed flowers in her diary
-Hides small candies in her pencil pouch to share with close friends');

INSERT INTO Avatar VALUES('Damien', 'Bodyguard',
'Age: 29
Traits:
-Drinks black coffee like it's water
```

-Unfriendly and does not smile, but will protect the person he's sworn to protect with his life on the line');

INSERT INTO Avatar VALUES('Jerry', 'Ghost',
'Age: UNKNOWN
Traits:
-LOVES harmless pranks');

INSERT INTO Avatar VALUES('Felix', 'Spy',
'Age: 36
Traits:
-Photographic Memory
-Has a room filled with pictures of people');

INSERT INTO Avatar VALUES ('Aeris', 'Fairy',
'Age: 1000
Traits:
-Loves bubble baths
-Scared of thunderstorms
-Collects human trinkets');

- Player_Has_Avatar (**AvatarName**: VARCHAR(50), **PlayerID**: CHAR(10))
    - ASSERTION: a player has at least one avatar

INSERT INTO Player_Has_Avatar VALUES ('Sammie', 'A000000001');
INSERT INTO Player_Has_Avatar VALUES ('Damien', 'A000000002');
INSERT INTO Player_Has_Avatar VALUES ('Jerry', 'A000000003');
INSERT INTO Player_Has_Avatar VALUES (Felix, 'A000000004');
INSERT INTO Player_Has_Avatar VALUES (Aeris, 'A000000005');

- Plays (**RoundNumber**: INTEGER, **PlayerID**: CHAR(10))
    - ASSERTION: a round has at least one player

INSERT INTO Plays VALUES (1, 'A000000001');
INSERT INTO Plays VALUES (2, 'A000000002');
INSERT INTO Plays VALUES (3, 'A000000003');
INSERT INTO Plays VALUES (4, 'A000000004');
INSERT INTO Plays VALUES (5, 'A000000005');

- Round (RoundNumber: INTEGER, StartTime: TIME, EndTime: TIME, WinnerPlayerID: CHAR(10))

INSERT INTO Round VALUES (1, '14:30:00', '15:00:00', 'A000000003');

INSERT INTO Round VALUES (2, ‘2:30:00’, ‘6:00:00’, ‘A000000004’);
INSERT INTO Round VALUES (3, ‘16:30:00’, ‘20:00:00’, ‘A000000001’);
INSERT INTO Round VALUES (4, ‘11:30:00’, ‘12:00:00’, ‘A000000002’);
INSERT INTO Round VALUES (5, ‘10:30:00’, ‘13:00:00’, ‘A000000005’);

- Receives (**RewardName**: VARCHAR(50), **PlayerID**: CHAR(10))

INSERT INTO Receives VALUES(‘Level Up’,‘A000000001’);
INSERT INTO Receives VALUES(‘Rookie Reward’,‘A000000002’);
INSERT INTO Receives VALUES(‘Veteran Bonus’,‘A000000003’);
INSERT INTO Receives VALUES(‘100th round Milestone’,‘A000000004’);
INSERT INTO Receives VALUES(‘Daily Streak Bonus’,‘A000000005’);

Turn_Has_R1(PlayerID: CHAR(10), **RoundNumber**: INTEGER, TurnNumber: INTEGER, DamageDoneLastTurn: INTEGER, HealthBefore: INTEGER, **SkillName**: VARCHAR(50))

- SkillName: NOT NULL and UNIQUE

Turn_Has_R2(DamageDoneLastTurn: INTEGER, HealthBefore: INTEGER, HealthAfter: INTEGER)

INSERT INTO Turn_Has_R1 VALUES(‘A000000001’, 1, 1, 50, 100, ‘Punch’);
INSERT INTO Turn_Has_R1 VALUES(‘A000000002’, 2, 2, 60, 140, ‘Kick’);
INSERT INTO Turn_Has_R1 VALUES(‘A000000003’, 3, 3, 70, 130, ‘Guard’);
INSERT INTO Turn_Has_R1 VALUES(‘A000000004’, 4, 4, 80, 200, ‘Rapid Shift’);
INSERT INTO Turn_Has_R1 VALUES(‘A000000005’, 5, 5, 90, 160, ‘Sidestep’);

INSERT INTO Turn_Has_R2 VALUES(50, 100, 50);
INSERT INTO Turn_Has_R2 VALUES(80, 140, 60);
INSERT INTO Turn_Has_R2 VALUES(60, 130, 70);
INSERT INTO Turn_Has_R2 VALUES(120, 200, 80);
INSERT INTO Turn_Has_R2 VALUES(70, 160, 90);

- Skill_R1(SkillName: VARCHAR(50), DamagePoints: INTEGER)
- Skill_R2(DamagePoints: INTEGER, EffectPoints: INTEGER)

INSERT INTO Skill_R1 VALUES(‘Punch’, 10);
INSERT INTO Skill_R1 VALUES(‘Kick’, 20);
INSERT INTO Skill_R1 VALUES(‘Guard’, 30);
INSERT INTO Skill_R1 VALUES(Shield, 40);
INSERT INTO Skill_R1 VALUES(‘Rapid Shift’,50);
INSERT INTO Skill_R1 kills VALUES(‘Sidestep’, 40);

INSERT INTO Skill_R2 VALUES(10, 100);
INSERT INTO Skill_R2 VALUES(20, 90);

INSERT INTO Skill_R2 VALUES(30, 80);
INSERT INTO Skill_R2 VALUES(40, 70);
INSERT INTO Skill_R2 VALUES(50, 60);
INSERT INTO Skill_R2 kills VALUES(40, 50);

- Attack (**SkillName**: VARCHAR(50), AttackBonus: INTEGER)

INSERT INTO Attack VALUES('Punch', 40);
INSERT INTO Attack VALUES(Punch, 50);
INSERT INTO Attack VALUES(Kick, 60);
INSERT INTO Attack VALUES('Kick', 70);
INSERT INTO Attack VALUES('Kick', 80);

- Defence (**SkillName**: VARCHAR(50), DamageReduction: INTEGER)

INSERT INTO Defence VALUES('Guard', 40);
INSERT INTO Defence VALUES('Guard', 50);
INSERT INTO Defence VALUES('Shield', 60);
INSERT INTO Defence VALUES('Shield', 70);
INSERT INTO Defence VALUES('Shield', 80);

- Agility (**SkillName**: VARCHAR(50), AgilityBonus: INTEGER)

INSERT INTO Agility VALUES('Rapid Shift', 40);
INSERT INTO Agility VALUES('Rapid Shift', 50);
INSERT INTO Agility VALUES('Sidestep', 60);
INSERT INTO Agility VALUES('Sidestep', 70);
INSERT INTO Agility VALUES('Sidestep', 80);

- Reward (RewardName: VARCHAR(50), Points: INTEGER)

INSERT INTO Reward Reward('Level Up', 50);
INSERT INTO Reward Reward('Rookie Reward', 60);
INSERT INTO Reward Reward('Veteran Bonus', 70);
INSERT INTO Reward Reward('100th round Milestone', 80);
INSERT INTO Reward Reward('Daily Streak Bonus', 90);

- Round_Provides (RoundNumber: INTEGER, **RewardName**: VARCHAR(50))
    - RewardName: NOT NULL

INSERT INTO Round_Provides VALUES(1, 'Level Up');
INSERT INTO Round_Provides VALUES(2, 'Rookie Reward');
INSERT INTO Round_Provides VALUES(3, 'Veteran Bonus');
INSERT INTO Round_Provides VALUES(4, '100th round Milestone');

INSERT INTO Round_Provides VALUES(5, ‘Daily Streak Bonus’);

Player_Has_R1(PlayerID: CHAR(10), Points: INTEGER, Username: VARCHAR(20), **RankingID: CHAR(10),** StatID:CHAR(10), Wins: INTEGER, Losses: INTEGER)

- Candidate keys: PlayerID, Username, RankingID, StatID
- StatID:UNIQUE and NOT NULL,
- RankingID: NOT NULL

Player_Has_R2(Wins: INTEGER, Losses: INTEGER, TotalGamesPlayed: INTEGER)

INSERT INTO Player_Has_R1 VALUES (‘A000000001’, 200, 'AceHunter', 'RB00000001', 'S000000001', 10, 5, 15);

INSERT INTO Player_Has_R1 VALUES (‘A000000002’, 130, 'ShadowFury’, 'RI00000001', 'S000000002', 6, 9, 15);

INSERT INTO Player_Has_R1 VALUES (‘A000000003’, 50, 'NovaStar', 'RI00000002', 'S000000003', 3, 2, 5);

INSERT INTO Player_Has_R1 VALUES (‘A000000004’, 300, 'PixelCrush', 'RG00000001', 'S000000004', 16, 5, 21);

INSERT INTO Player_Has_R1 VALUES (‘A000000005’, 180, 'DragonX', 'RB0000003', 'S000000005', 8, 5, 13);

INSERT INTO Player_Has_R1 VALUES (‘A000000006’, 120, 'BlitzQueen', 'RI00000003', 'S000000006', 8, 8, 16);

INSERT INTO Player_Has_R1 VALUES (‘A000000007’, 90, 'SilentScope', 'RI00000004', 'S000000007', 2, 5, 7);

INSERT INTO Player_Has_R1 VALUES (‘A000000008’, 200, 'SkyBreaker', 'RS00000002', 'S000000007', 11, 5, 16);

INSERT INTO Player_Has_R2 VALUES (10, 5, 15);

INSERT INTO Player_Has_R2 VALUES (6, 9, 15);

INSERT INTO Player_Has_R2 VALUES (3, 2, 5);

INSERT INTO Player_Has_R2 VALUES (16, 5, 21);

```
INSERT INTO Player_Has_R2 VALUES (8, 5, 13);

INSERT INTO Player_Has_R2 VALUES (8, 8, 16);

INSERT INTO Player_Has_R2 VALUES (2, 5, 7);

INSERT INTO Player_Has_R2 VALUES (11, 5, 16);
```

- Ranking: (RankingID:CHAR(10))

```
INSERT INTO Ranking VALUES('RI00000001');
INSERT INTO Ranking VALUES('RI00000002');
INSERT INTO Ranking VALUES('RI00000003');
INSERT INTO Ranking VALUES('RI00000004');
INSERT INTO Ranking VALUES('RI00000005');

INSERT INTO Ranking VALUES('RB00000001');
INSERT INTO Ranking VALUES('RB00000002');
INSERT INTO Ranking VALUES('RB00000003');
INSERT INTO Ranking VALUES('RB00000004');
INSERT INTO Ranking VALUES('RB00000005');

INSERT INTO Ranking VALUES('RS00000001');
INSERT INTO Ranking VALUES('RS00000002');
INSERT INTO Ranking VALUES('RS00000003');
INSERT INTO Ranking VALUES('RS00000004');
INSERT INTO Ranking VALUES('RS00000005');

INSERT INTO Ranking VALUES('RG00000001');
INSERT INTO Ranking VALUES('RG00000002');
INSERT INTO Ranking VALUES('RG00000003');
INSERT INTO Ranking VALUES('RG00000004');
INSERT INTO Ranking VALUES('RG00000005');
```

- Iron: (**RankingID:CHAR(10)**, IronLevels: VARCHAR(10))

- Bronze (**RankingID:CHAR(10)**, BronzeLevels: VARCHAR(10))
-
- Silver: (**RankingID:CHAR(10)**, SilverLevels: VARCHAR(10))
-
- Gold: (**RankingID:CHAR(10)**, GoldLevels: VARCHAR(10))

```
INSERT INTO Iron VALUES ('RI00000001', 'Iron I');
INSERT INTO Iron VALUES ('RI00000002', 'Iron I');
INSERT INTO Iron VALUES ('RI00000003', 'Iron III');
INSERT INTO Iron VALUES ('RI00000004', 'Iron IV');
INSERT INTO Iron VALUES ('RI00000005', 'Iron II');

INSERT INTO Bronze VALUES ('RB00000001', 'Bronze I');
INSERT INTO Bronze VALUES ('RB00000002', 'Bronze II');
INSERT INTO Bronze VALUES ('RB00000003', 'Bronze III');
INSERT INTO Bronze VALUES ('RB00000004', 'Bronze I');
INSERT INTO Bronze VALUES ('RB00000005', 'Bronze IV');

INSERT INTO Silver VALUES ('RS00000001’, Silver III');
INSERT INTO Silver VALUES ('RS00000002', 'Silver I');
INSERT INTO Silver VALUES ('RS00000003', 'Silver II');
INSERT INTO Silver VALUES ('RS00000004', 'Silver I');
INSERT INTO Silver VALUES ('RS00000005', 'Silver V');

INSERT INTO Gold VALUES ('RG00000001', 'Gold I');
INSERT INTO Gold VALUES ('RG00000002', 'Gold II');
INSERT INTO Gold VALUES ('RG00000003', 'Gold III');
INSERT INTO Gold VALUES ('RG00000004', 'Gold II');
INSERT INTO Gold VALUES ('RG00000005', 'Gold III');
```