# CPSC 304 Project Cover Page

Milestone #: 2

Date: Oct 15th 2023

Group Number: 28

| Name | Student Number | CS Alias (Userid) | Preferred Email Address |
|---|---|---|---|
| Ram Jayakumar | 15967981 | ramj21 | ramjayakumar21@gmail.com |
| Amin Fahiminia | 13006549 | afahimi | afahimi@student.ubc.ca |
| Mercury Mcindoe | 85594505 | merc0606 | mercurymcindoe@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.

1. **A completed cover page (template on Canvas)**

2. **A brief (~2-3 sentences) summary of your project. Many of your TAs are managing multiple projects so this will help them remember details about your project.**

The application is set in the realm of fantasy role-playing games (RPGs), where both players and non-playable characters (NPCs) coexist and interact in a dynamically evolving game universe. Within this realm, characters traverse through diverse landscapes, engage in social exchanges, align themselves with factions, embark on quests, and oversee inventories filled with a wide array of items.

3. **The ER diagram you are basing your item #3 (below) on.**

Below is our improved ER diagram from the milestone 1 submission, with the specified changes:

1. The attribute “Interaction ID” was changed to “Interaction Time”.

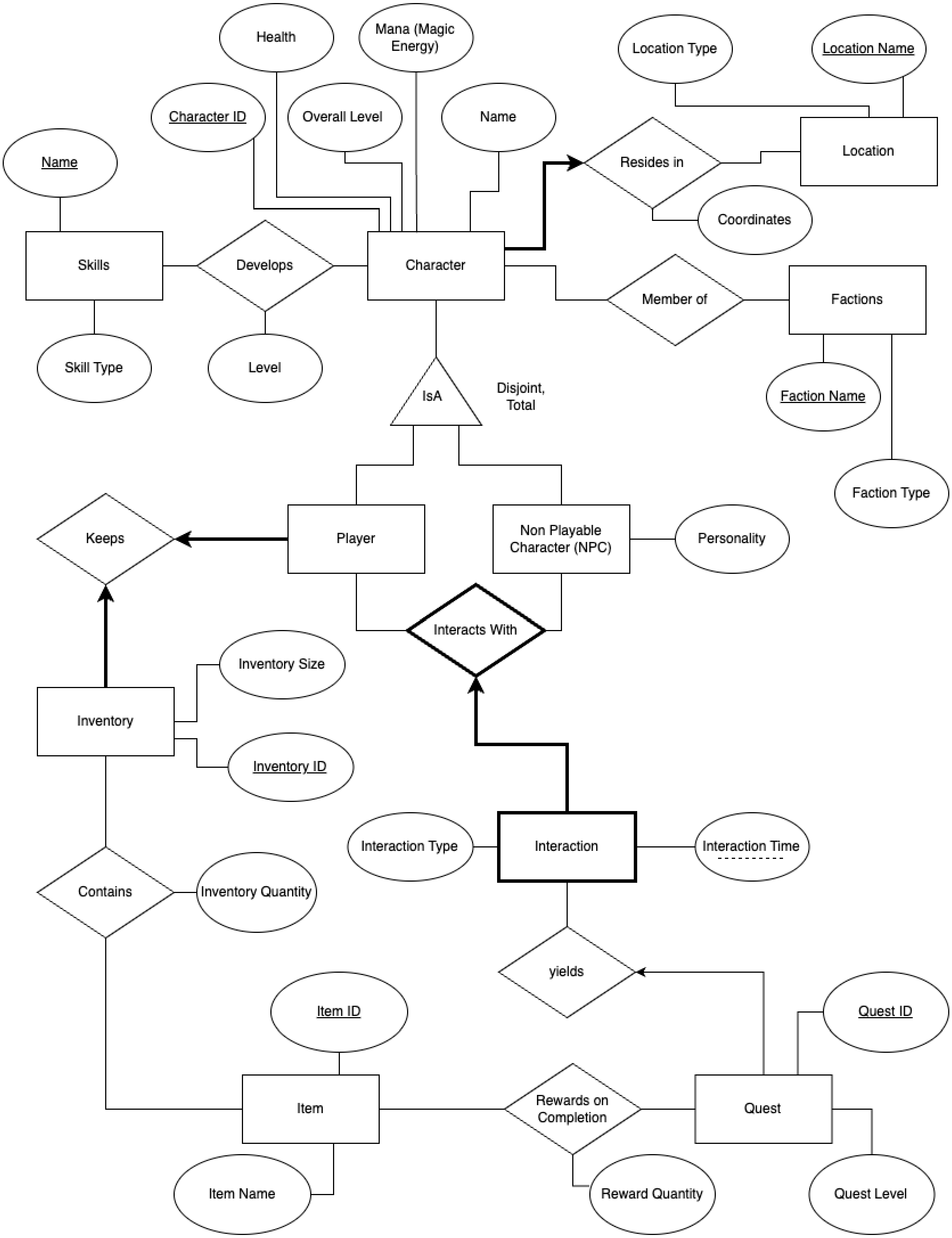
This change was made by TA Request for better reasoning on “Interaction” being a weak entity. Before, “Interaction ID” made the use of foreign keys “PlayableCharacter.CharacterID" and “NonPlayableCharacter.CharacterID” redundant as each interaction had a unique “Interaction ID” to be identified by. With this change, interactions are identified by the time they occured, and because multiple can occur at the same time, the two previously mentioned Character IDs are required to identify a specific interaction.

2. The “Yields” relationship from Interactions to Quests has been changed from “one to one” to “one to many”.

This change was made to reflect the idea that an interaction can lead to multiple quests, but each quest can only come a single interaction. Before, the same quest could come from different interactions.

3. The “Quest” entity gained the attribute “Quest Level”, the “Skill” entity gained the attribute “Skill Type”, and the “Non Playable Character” entity gained the attribute “Personality”

These attributes were added to give a better explanation for the existence based on TA request. Before, these entities solely consisted of their primary key.


Health
Mana (Magic Energy)
Location Type
Location Name
Character ID
Overall Level
Name
Resides in
Location
Name
Coordinates
Skills
Develops
Character
Member of
Factions
Skill Type
Level
Faction Name
IsA
Disjoint, Total
Faction Type
Keeps
Player
Non Playable Character (NPC)
Personality
Interacts With
Inventory Size
Inventory
Inventory ID
Interaction Type
Interaction
Interaction Time
Contains
Inventory Quantity
yields
Item ID
Quest ID
Item
Rewards on Completion
Quest
Item Name
Reward Quantity
Quest Level

4. **The schema derived from your ER diagram (above).**

Underlined attributes are Primary Keys
**Bolded attributes are Foreign Keys**

**Character**(Character ID: int, **Location Name**: string *not null*, Health: int, Overall Level: int, Mana: int, Name: string, Coordinates: string)

- Candidate Keys: {Character ID}
- Primary Key: Character ID
- Foreign Keys: (Location Name) references Location
- Location Name is NOT NULL

**Location**(Location Name: string, Location Type: string)

- Candidate Keys: {Location Name}
- Primary Key: Location Name
- Foreign Keys:

**MemberOf**(**Character ID**: int, **Faction Name**: string)

- Candidate Keys: {Character ID, Faction Name}
- Primary Key: Character ID, Faction Name
- Foreign Keys:
    - (Character ID) references Character
    - (Faction Name) references Factions

**Factions**(Faction Name: string, Faction Type: string)

- Candidate Keys: {Faction Name}
- Primary Key: Faction Name
- Foreign Keys:

**Skill**(Skill Name: string, Skill Type: string)

- Candidate Keys: {Skill Name}
- Primary Key: Skill Name
- Foreign Keys:

**Develops**(**Skill Name:** string, **Character ID**: int, Level: int)

- Candidate Keys: {Skill Name, Character ID}
- Primary Key: Skill Name, Character ID
- Foreign Keys:
    - (Skill Name) references Skill

    - (Character ID) references Character

**Player**(**Character ID**: int, **Inventory ID**: int *unique*)

- Candidate Keys: {Character ID}, {Inventory ID}
- Primary Key: Character ID
- Foreign Keys:
    - (Inventory ID) references Inventory
    - (Character ID) references Character
- Inventory ID is UNIQUE

**NonPlayableCharacter**(**Character ID**: int, Personality: string)

- Candidate Keys: {Character ID}
- Primary Key: Character ID
- Foreign Keys:
    - (Character ID) references Character

**Inventory**(Inventory ID: int, **Player.CharacterID**: int, Inventory Size: int)

- Candidate Keys: {Inventory ID}, {Player.CharacterID}
- Primary Key: Inventory ID
- Foreign Keys: (Player.CharacterID) from Player
- Player.CharacterID is UNIQUE and NOT NULL

**Contains**(**Inventory ID**: int, **Item ID:** int, Inventory Quantity: int)

- Candidate Keys: {Inventory ID, Item ID}
- Primary Key: Inventory ID, Item ID
- Foreign Keys:
    - (Inventory ID) references Inventory
    - (Item ID) references Item

**Item**(Item ID: int, Item Name: string)

- Candidate Keys: {Item ID}
- Primary Key: Item ID
- Foreign Keys:

**RewardsOnCompletion**(**Item ID**: int, **Quest ID**: int**,** Reward Quantity: int)

- Candidate Keys: {Item ID, Quest ID}
- Primary Key: Item ID
- Foreign Keys:
    - (Item ID) references Item

    - (Quest ID) references YieldsQuest

**Interactions**(Player.Character ID: int**,** NonPlayableCharacter.Character ID: int Interaction Time: int *not null*, Interaction Type: string)

- Candidate Keys: {Player.Character ID, NonPlayableCharacter.Character ID, Interaction Time}
- Primary Key: Player.Character ID, NonPlayableCharacter.Character ID, Interaction Time
- Foreign Keys:
    - (Player.Character ID) references Player
    - (NonPlayableCharacter.Character ID) references NonPlayableCharacter
- Interaction Time is a weak entity key

**YieldsQuest**(QuestID: int, **Player.CharacterId**: int**,** **NonPlayableCharacter.CharacterId**: int, **Interaction Time**: int, Quest Level: int)

- Candidate Keys: {QuestID}
- Primary Key: QuestID
- Foreign Keys:
    - (Player.Character ID, NonPlayableCharacter.Character ID, Interaction Time) references Interactions
- Foreign keys can be null as quests may not be given by NPC (eg. found on town job board)

**5. Functional Dependencies (FDs) a. Identify the functional dependencies in your relations, including the ones involving all candidate keys (including the primary key). PKs and CKs are considered functional dependencies and should be included in the list of FDs.**

**Character**

Character ID → Health, Overall Level, Mana, Name, Location Name, Coordinates
Coordinates → Location Name

**Location**

Location Name → Location Type

**MemberOf**

Character ID → Faction Name

**Factions**

Faction Name → Faction Type

**Skill**
Skill Name → Skill Type

**Develops**
Skill Name, Character ID → Level

**Player**
Character ID → Inventory ID
Inventory ID → Character ID

**NonPlayableCharacter**
Character ID → Personality

**Inventory**
Inventory ID → Inventory Size, Player.CharacterID
Player.CharacterID → InventoryID

**Contains**
Inventory ID, Item ID → Inventory Quantity

**Item**
Item ID → Item Name

**RewardsOnCompletion**
Quest ID, Item ID → Reward Quantity
Reward Quantity → Item ID

**Interactions**
Player.CharacterID, NonPlayableCharacter.CharacterID, Interaction Time → Interaction Type

**YieldsQuest**
Quest ID → Player.CharacterID, NonPlayableCharacter.CharacterID, Interaction Time, Quest Level

**6. Normalization a. Normalize each of your tables to be in 3NF or BCNF. Give the list of tables, their primary keys, their candidate keys, and their foreign keys after**

**normalization. You should show the steps taken for the decomposition. Should there be errors, and no work is shown, no partial credit can be awarded without steps shown. The format should be the same as Step 3, with tables listed similar to Table1(attr1:domain1, attr2:domain2, ...). ALL Tables must be listed, not only the ones post-normalization.**

We will be normalizing our relations to BCNF. The list of the new relations can be found at the end of this question, after attempting normalization on all relations.

**Character**

1. Character ID → Health, Overall Level, Mana, Name, Location Name, Coordinates
2. Coordinates → Location Name

FD 1 is in BCNF, which means it is in BCNF, as {Character ID +} = {Character ID, Health, Overall Level, Mana, Name, Location Name, Coordinates}
FD 2 is NOT in BCNF, as Coordinates is not a super key {Coordinates +} = {Coordinates, Location Name}

1. **Find the minimal key of Character**

We know the minimal key of Character by knowing it's primary key: Character ID

2. **Decompose non-BCNF functional dependencies**

Coordinates → Location Name

| LEFT CIRCLE | MIDDLE CIRCLE | RIGHT CIRCLE |
|---|---|---|
| Character ID, Health Overall Level, Mana, Name | Coordinates | Location Name |

There are no more functional dependencies that are not in BCNF to decompose.
The new relations are:

**CharacterInfo**(Character ID: int, **Coordinates**: string, Health: int, Overall Level: int, Mana, Name: int)
**CoordinateLocations**(Coordinates: string, **Location Name:** string)

A more detailed list of relations is found at the end of this question.

**Location**

1. Location Name → Location Type

FD 1 has Location Name as a super key, and is therefore in BCNF. All FD’s are in BCNF, and the relation stays unchanged.

**Factions**

1. Faction Name → Faction Type

FD 1 has Faction Name as a super key, and is therefore in BCNF. All FD’s are in BCNF, and the relation stays unchanged.

**MemberOf**

1. Character ID → Faction Name

FD 1 has Character ID as a super key, and is therefore in BCNF. All FD’s are in BCNF, and the relation stays unchanged.

**Skill**

1. Skill Name → Skill Type

FD 1 has Skill Name as a super key, and is therefore in BCNF. All FD’s are in BCNF, and the relation stays unchanged.

**Develops**

1. Skill Name, Character ID → Level

FD 1 has Skill Name, Character ID as a super key, and is therefore in BCNF. All FD’s are in BCNF, and the relation stays unchanged.

**Player**

1. Character ID → Inventory ID
2. Inventory ID → Character ID

FD 1 has Character ID as a super key, and is therefore in BCNF. FD 2 has Inventory ID as a superkey, and is also in BCNF. We also know that this is true as all relations with just two attributes are in BCNF. All FD’s are in BCNF, and the relation stays unchanged.

**NonPlayableCharacter**

1. Character ID → Personality

FD 1 has Character ID as a super key, and is therefore in BCNF. All FD’s are in BCNF, and the relation stays unchanged.

**Inventory**

1. Inventory ID → Inventory Size, Player.CharacterID
2. Player.CharacterID → InventoryID

FD 1 has Inventory ID as a super key, and is therefore in BCNF. Also, FD 2 only has two attributes, hence it is also in BCNF.
Therefore, All FD’s are in BCNF, and the relation stays unchanged.

**Contains**

1. Inventory ID, Item ID → Inventory Quantity

FD 1 has Inventory ID, Item ID as a super key, and is therefore in BCNF. All FD’s are in BCNF, and the relation stays unchanged.

**Item**

1. Item ID → Item Name

FD 1 has Item ID as a super key, and is therefore in BCNF. All FD’s are in BCNF, and the relation stays unchanged.

**RewardsOnCompletion**

1. Quest ID,  Item ID → Reward Quantity
2. Reward Quantity →  Item ID

FD 1 has Quest ID, Item ID as a super key, and is therefore in BCNF. FD 2 however is not in BCNF, as Item ID, Reward Quantity is not a super key ({Reward Quantity +} = {Reward Quantity, Item ID}).

1. **Find the minimal key of RewardsOnCompletion**

We know the minimal key of Character by knowing it’s primary key: Quest ID, Item ID

2. **Decompose non-BCNF functional dependencies**

Reward Quantity → Item ID

| LEFT CIRCLE | MIDDLE CIRCLE | RIGHT CIRCLE |
|---|---|---|
| Quest ID | Reward Quantity | Item ID |

There are no more functional dependencies that are not in BCNF to decompose.
The new relations are:

**QuestRewards**(**Quest ID**: int, **Reward Quantity**: int *unique*)
**RewardItems**(Reward Quantity: int, **Item ID**: int)

**Interactions**

1. Player.CharacterID, NonPlayableCharacter.CharacterID, Interaction Time → Interaction Type

FD 1 has Player.CharacterID, NonPlayableCharacter.CharacterID, and Interaction Time as a super key, and is therefore in BCNF. All FD's are in BCNF, and the relation stays unchanged.

**YieldsQuest**

1. Quest ID → Player.CharacterID, NonPlayableCharacter.CharacterID, Interaction Time, Quest Level

FD 1 has Quest ID as a super key, and is therefore in BCNF. All FD's are in BCNF, and the relation stays unchanged.

**– New Relations after Normalization –**

**CharacterInfo(**Character ID: int, **Coordinates**: string, Health: int, Overall Level: int, Mana, Name: int)

- Candidate Keys: {Character ID}
- Primary Key: Character ID
- Foreign Keys: (Coordinates) references CoordinateLocations

**CoordinateLocations**(Coordinates: string, **Location Name:** string *not null*)

- Candidate Keys: {Coordinates}
- Primary Key: Coordinates
- Foreign Keys: (Location Name) references Location
- Location Name is NOT NULL

**Location**(Location Name: string, Location Type: string)

- Candidate Keys: {Location Name}
- Primary Key: Location Name
- Foreign Keys:

**MemberOf**(**Character ID**: int, **Faction Name**: string)

- Candidate Keys: {Character ID, Faction Name}
- Primary Key: Character ID, Faction Name
- Foreign Keys:
    - (Character ID) references Character
    - (Faction Name) references Factions

**Factions**(Faction Name: string, Faction Type: string)

- Candidate Keys: {Faction Name}
- Primary Key: Faction Name
- Foreign Keys:

**Skill**(Skill Name: string, Skill Type: string)

- Candidate Keys: {Skill Name}
- Primary Key: Skill Name
- Foreign Keys:

**Develops**(**Skill Name:** string, **Character ID**: int, Level: int)

- Candidate Keys: {Skill Name, Character ID}
- Primary Key: Skill Name, Character ID
- Foreign Keys:
    - (Skill Name) references Skill
    - (Character ID) references Character

**Player**(Character ID: int, **Inventory ID**: int *unique not null*)

- Candidate Keys: {Character ID}, {Inventory ID}
- Primary Key: Character ID
- Foreign Keys:
    - (Inventory ID) references Inventory
- Inventory ID is UNIQUE
- Inventory ID is NOT NULL

**NonPlayableCharacter**(Character ID: int, Personality: string)

- Candidate Keys: {Character ID}
- Primary Key: Character ID
- Foreign Keys:

**Inventory**(Inventory ID: int, **Player.CharacterID**: int *not null*, Inventory Size: int)

- Candidate Keys: {Inventory ID}, {Player.CharacterID}
- Primary Key: Inventory ID
- Foreign Keys: Player.CharacterID
- Player.CharacterID is UNIQUE and NOT NULL

**Contains**(**Inventory ID**: int, **Item ID**: int, Inventory Quantity: int)

- Candidate Keys: {Inventory ID, Item ID}
- Primary Key: Inventory ID, Item ID
- Foreign Keys:
    - (Inventory ID) references Inventory
    - (Item ID) references Item

**Item**(Item ID: int, Item Name: string)

- Candidate Keys: {Item ID}
- Primary Key: Item ID
- Foreign Keys:

**QuestRewards**(**Quest ID**: int, **Reward Quantity**: int)

- Candidate Keys: {Quest ID}
- Primary Key: Quest ID
- Foreign Keys:
    - (Reward Quantity) references RewardItems
    - (Quest ID) references YieldsQuest

**RewardItems**(Reward Quantity: int, **Item ID**: int)

- Candidate Keys: {Reward Quantity}
- Primary Key: Reward Quantity
- Foreign Keys:
    - (Item ID) references Item

**Interactions**(Player.Character ID: int**,** NonPlayableCharacter.Character ID: int Interaction Time: int *not null*, Interaction Type: string)

- Candidate Keys: {Player.Character ID, NonPlayableCharacter.Character ID, Interaction Time}

- Primary Key: Player.Character ID, NonPlayableCharacter.Character ID, Interaction Time
- Foreign Keys:
    - (Player.Character ID) references Player
    - (NonPlayableCharacter.Character ID) references NonPlayableCharacter
- Interaction Time is a weak entity key

**YieldsQuest**(QuestID: int, **Player.CharacterId**: int**,** **NonPlayableCharacter.CharacterId**: int, **Interaction Time**: int, Quest Level: int)

- Candidate Keys: {QuestID}
- Primary Key: QuestID
- Foreign Keys:
    - (Player.Character ID, NonPlayableCharacter.Character ID, Interaction Time) references Interactions
- Foreign keys can be null as quests may not be given by NPC (eg. found on town job board)

**7. The SQL DDL statements are required to create all the tables from item #6.**

NOTE: We will be using PostgreSQL syntax as our DBMS for our SQL statements.

```
CREATE TABLE Location (
   LocationName VARCHAR(255) PRIMARY KEY,
   LocationType VARCHAR(255)
);

CREATE TABLE CoordinateLocations (
   Coordinates VARCHAR(255) PRIMARY KEY,
   LocationName VARCHAR(255) NOT NULL,
   FOREIGN KEY (LocationName) REFERENCES Location(LocationName)
      ON UPDATE CASCADE
      ON DELETE CASCADE
);

CREATE TABLE CharacterInfo (
   CharacterID INT PRIMARY KEY,
   Coordinates VARCHAR(255),
   Health INT,
   OverallLevel INT,
    Mana INT,
    CharacterName VARCHAR(255),
    FOREIGN KEY (Coordinates) REFERENCES
CoordinateLocations(Coordinates)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE Factions (
    FactionName VARCHAR(255) PRIMARY KEY,
    FactionType VARCHAR(255)
);

CREATE TABLE MemberOf (
    CharacterID INT,
    FactionName VARCHAR(255),
    PRIMARY KEY (CharacterID, FactionName),
    FOREIGN KEY (CharacterID) REFERENCES CharacterInfo(CharacterID)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (FactionName) REFERENCES Factions(FactionName)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE Skill (
    SkillName VARCHAR(255) PRIMARY KEY,
    SkillType VARCHAR(255)
);

CREATE TABLE Develops (
    SkillName VARCHAR(255),
    CharacterID INT,
    Level INT,
    PRIMARY KEY (SkillName, CharacterID),
    FOREIGN KEY (SkillName) REFERENCES Skill(SkillName)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (CharacterID) REFERENCES CharacterInfo(CharacterID)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE Player (
    CharacterID INT PRIMARY KEY,
    InventoryID INT UNIQUE,
    FOREIGN KEY (CharacterID) REFERENCES CharacterInfo(CharacterID)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE Inventory (
    InventoryID INT PRIMARY KEY,
    PlayerCharacterID INT UNIQUE NOT NULL,
    InventorySize INT,
    FOREIGN KEY (PlayerCharacterID) REFERENCES Player(CharacterID)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE NonPlayableCharacter (
    CharacterID INT PRIMARY KEY,
    Personality VARCHAR(255),
    FOREIGN KEY (CharacterID) REFERENCES CharacterInfo(CharacterID)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE Item (
    ItemID INT PRIMARY KEY,
    ItemName VARCHAR(255)
);

CREATE TABLE Contains (
    InventoryID INT,
    ItemID INT,
    InventoryQuantity INT,
    PRIMARY KEY (InventoryID, ItemID),
    FOREIGN KEY (InventoryID) REFERENCES Inventory(InventoryID)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (ItemID) REFERENCES Item(ItemID)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE Interactions (
    PlayerCharacterID INT,
    NonPlayableCharacterCharacterID INT,
    InteractionTime INT NOT NULL,
    InteractionType VARCHAR(255),
    PRIMARY KEY (PlayerCharacterID, NonPlayableCharacterCharacterID,
InteractionTime),
    FOREIGN KEY (PlayerCharacterID) REFERENCES Player(CharacterID)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (NonPlayableCharacterCharacterID) REFERENCES
NonPlayableCharacter(CharacterID)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE YieldsQuest(
    QuestID INT PRIMARY KEY,
    PlayerCharacterID INT,
    NonPlayableCharacterID INT,
    InteractionTime INT,
    QuestLevel INT,
    FOREIGN KEY (PlayerCharacterID, NonPlayableCharacterID,
InteractionTime)
        REFERENCES Interactions(PlayerCharacterID,
NonPlayableCharacterCharacterID, InteractionTime)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE RewardItems (
    RewardQuantity INT PRIMARY KEY,
    ItemID INT,
    FOREIGN KEY (ItemID) REFERENCES Item(ItemID)
```

```
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE QuestRewards (
    QuestID INT PRIMARY KEY,
    RewardQuantity INT,
    FOREIGN KEY (QuestID) REFERENCES YieldsQuest(QuestID)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (RewardQuantity) REFERENCES
RewardItems(RewardQuantity)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

**8. INSERT statements to populate each table with at least 5 tuples. You will likely want to have more than 5 tuples so that you can have meaningful queries later**

```
INSERT INTO Location (LocationName, LocationType)
VALUES
  ('Floating City', 'Sky'),
  ('The Great Forest', 'Forest'),
  ('The End', 'Endgame'),
  ('Atlantis', 'Sea'),
  ('Verudon', 'City');


INSERT INTO CoordinateLocations (Coordinates, LocationName)
VALUES
  ('10-20-103', 'Floating City'),
  ('15-25-20', 'The Great Forest'),
  ('5-15-20', 'The Great Forest'),
  ('5-15-22', 'The Great Forest'),
  ('20-30-40', 'The End'),
  ('22-36-40', 'The End');


INSERT INTO CharacterInfo (CharacterID, Health, OverallLevel, Mana,
CharacterName, Coordinates)
VALUES
  (1382, 100, 28, 38, 'RemmyRoblox', '10-20-103'),
  (3312, 90, 4, 40, 'Mercury1989', '15-25-20'),
  (5869, 80, 3, 60, 'AminTheAdmin', '5-15-20'),
  (1129, 70, 2, 70, 'SteveGamer', '20-30-40'),
  (1269, 60, 1, 80, 'xX_JoeShmoe_Xx', '22-36-40'),
  (2001, 1120, 28, 38, 'Average Man', '5-15-20'),
  (2002, 90, 4, 40, 'Witch', '5-15-20'),
  (2003, 82, 3, 60, 'Bug', '5-15-22'),
  (2004, 70, 2, 70, 'Demon', '20-30-40'),
  (2005, 50, 1, 80, 'Demon Dark', '22-36-40');


INSERT INTO Factions (FactionName, FactionType)
VALUES
  ('Skyguard', 'Holy'),
  ('Forestwatch', 'Neutral'),
  ('Endkeepers', 'Dark'),
  ('Seaguard', 'Neutral'),
  ('Citywatch', 'Lawful');

INSERT INTO MemberOf (CharacterID, FactionName)
VALUES
  (1382, 'Skyguard'),
  (3312, 'Forestwatch'),
  (5869, 'Forestwatch'),
  (1129, 'Endkeepers'),
  (1269, 'Endkeepers');

INSERT INTO Skill (SkillName, SkillType)
VALUES
  ('Sky Slash', 'Attack'),
  ('Forest Heal', 'Heal'),
  ('End Shield', 'Defense'),
  ('Sea Wave', 'Attack'),
  ('City Guard', 'Defense');

INSERT INTO Develops (SkillName, CharacterID, Level)
VALUES
  ('Sky Slash', 1382, 10),
  ('Forest Heal', 3312, 5),
  ('Forest Heal', 5869, 3),
  ('End Shield', 1129, 8),
  ('City Guard', 1269, 2);

INSERT INTO Player (CharacterID, InventoryID)
VALUES
  (1382, 1001),
  (3312, 1002),
  (5869, 1003),
  (1129, 1004),
  (1269, 1005);

INSERT INTO NonPlayableCharacter (CharacterID, Personality)
VALUES
  (2001, 'Aggressive'),
  (2002, 'Helpful'),
  (2003, 'Mysterious'),
  (2004, 'Talkative'),
  (2005, 'Silent');

INSERT INTO Inventory (InventoryID, PlayerCharacterID, InventorySize)
VALUES
  (1001, 1382, 20),
  (1002, 3312, 15),
  (1003, 5869, 18),
  (1004, 1129, 25),
  (1005, 1269, 10);

INSERT INTO Item (ItemID, ItemName)
VALUES
  (1001, 'Sword of Light'),
  (1002, 'Dark Shield'),
  (1003, 'Health Potion'),
  (1004, 'Bow of Eternity'),
  (1005, 'Ring of Strength');

INSERT INTO Contains (InventoryID, ItemID, InventoryQuantity)
VALUES
  (1001, 1001, 1),
  (1002, 1002, 1),
  (1003, 1003, 5),
  (1004, 1004, 2),
  (1005, 1005, 3);

INSERT INTO Interactions (PlayerCharacterID, NonPlayableCharacterCharacterID,
InteractionTime, InteractionType)
VALUES
  (1382, 2001, 1634280600, 'Conversation'),
  (3312, 2002, 1634366700, 'Combat'),
  (5869, 2003, 1634452800, 'Trade'),
  (1129, 2004, 1634538600, 'Trade'),
  (1269, 2005, 1634624700, 'Conversation');

INSERT INTO YieldsQuest(QuestID, PlayerCharacterID, NonPlayableCharacterID,
InteractionTime, QuestLevel)
VALUES
  (101, 1382, 2001, 1634280600, 68),
  (112, 3312, 2002, 1634366700, 47),
  (312, 5869, 2003, 1634452800, 18),
  (12, 1129, 2004, 1634538600, 120),
  (99, 1269, 2005, 1634624700, 38);

INSERT INTO RewardItems (RewardQuantity, ItemID)
VALUES
  (1, 1001),
  (2, 1002),
  (5, 1003),
  (3, 1004),
  (4, 1005);

INSERT INTO QuestRewards (QuestID, RewardQuantity)
VALUES
  (101, 1),
  (112, 1),
  (312, 5),
  (12, 1),
  (99, 3);
```