University of British Columbia, Vancouver

Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: 2024/03/01

Group Number: 59

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Dingwen Wang	29863370	c9d3b	dingwen1@students.cs.ubc.ca
Dexin Qian	92508183	z0g8h	dqian01@students.cs.ubc.ca
Manushree Singhania	74285081	s1q1o	singhaniamanushree@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

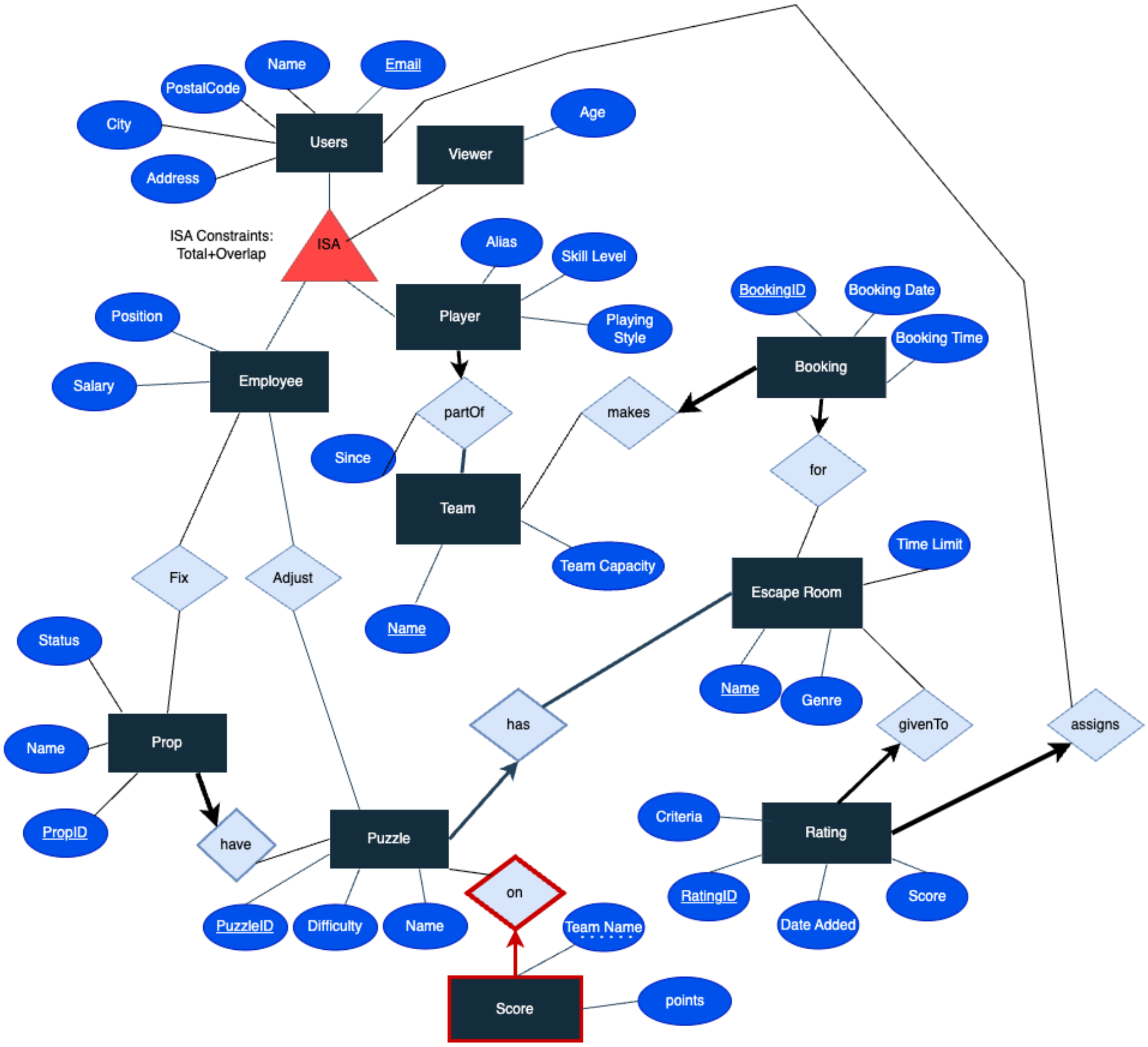
University of British Columbia, Vancouver

Department of Computer Science

Project Summary

The application provides a platform for passionate escape room fans to engage with their favorite teams by getting information such as their performance data. Additionally, it provides escape operators a means to analyze statistics for the rooms such as user ratings that would help them in taking decisions regarding managing and developing their establishments further to build a more compelling attraction.

ER Diagram



Changes to the ER Diagram

- Changed the entity name from “User” to “Users” as “User” is defined as reserved name by Oracle

- Added the attribute Email to Users to ensure that the superclass and subclasses in the ISA hierarchy have the same primary key based on the feedback on Milestone 1. Email will be used as a point of contact for all users and since every user needs to have a unique email, it can be used as a primary key for the entity.

- Removed the attribute Email from Viewer as it has been made an attribute of the superclass.

- Removed the attribute Phone Number from Employee as Email will be stored as the contact information for all the users.

- Added the attributes City, Address, Postal Code to Users so that there would be functional dependencies other than those identified by the primary and candidate keys.

- Added the attribute Salary for Employee so that there would be functional dependencies other than those identified by the primary and candidate keys.

- Changed the attribute names “Booking #”, “Puzzle #”, “Rating #”, “Prop #” to “BookingID”, “PuzzleID”, “RatingID” and “PropID” respectively to ensure that the attribute name is more descriptive, thereby making it easier to understand.

Schema derived from the ER diagram

- Team(Name: Varchar, TeamCapacity: Integer)
    - PK: Name
    - TeamCapacity Default value is 10
- EscapeRoom(Name: Varchar, Genre: Varchar, TimeLimit: Integer)
    - PK: Name
    - TimeLimit Default value is 60
- BookingMakesFor(BookingID: Integer, BookingDate: Date, BookingTime: Time, **Tname**: Varchar, **Ename:** Varchar)
    - PK: BookingID
    - CK: {BookingDate, BookingTime, TName, Ename}
    - FK: Tname references Team(Name), EName references EscapeRoom(Name)
    - Tname is Not Null (participation constraint)
    - Ename is Not Null (participation constraint)
- RatingGivenToAssigns(RatingID: Integer, DateAdded: Date, Score: Integer, Criteria: Varchar, **Name:** Varchar, **Email**: Varchar)

  - PK: RatingID
  - CK: {Email, Name, DateAdded, Score, Criteria}
  - FK: Name references EscapeRoom, Email references Users
  - Name is Not Null (participation constraint)
  - Email is Not Null (participation constraint)
  - Score is Not Null, Default Value is 5
  - Criteria is Not Null
- PuzzleHas(PuzzleID: Integer, Difficulty: Integer, Pname: Varchar, **Ename:** Varchar)
  - PK: PuzzleID
  - CK: {Pname, Ename}
  - FK: Ename references EscapeRoom(Name)
  - Ename is Not Null (participation constraint)
  - Pname is Not Null
- ScoreOnPuzzle(TeamName: Varchar, Points: Integer, **PuzzleID:** Integer)
  - PK: {TeamName, PuzzleID}
  - FK: PuzzleID references PuzzleHas
- PropHave(PropID: Integer, Name: Varchar, Status: Varchar, **PuzzleID:** Integer)
  - PK: PropID
  - FK: PuzzleID references PuzzleHas
  - PuzzleID is Not Null (participation constraint)
- Users(Name: Varchar, Email: Varchar, Address: Varchar, City: Varchar, PostalCode: Varchar)
  - PK: Email
  - CK: {Address, PostalCode}
  - Address is Not Null
  - City is Not Null
  - PostalCode is Not Null
- Employee(**Email**: Varchar, Position: Varchar, Salary: Integer)
  - PK: Email
  - FK: Email references Users
- Viewer(**Email**: Varchar, Age: Integer)
  - PK: Email
  - FK: Email references Users
- PlayerPartOf(**Email**: Varchar, Alias: Varchar, SkillLevel: Integer, PlayingStyle: Varchar, **Name**: Varchar, Since: Date)
  - PK: Email
  - CK: Alias
  - FK: Email references Users, Name references Team
  - Alias is Not Null and Unique
  - SkillLevel Default value is 1
  - Name is Not Null (participation constraint)
- Fix(**Email**: Varchar, **PropID**: Integer)

  - PK: Email, PropID
  - FK: Email references Users, PropID references PropHave
- Adjust(**Email**: Varchar, **PuzzleID**: Integer)
  - PK: Email, PuzzleID
  - FK: Email references Users, PuzzleID references PuzzleHas

Functional Dependencies

(FD not identified by a PK or CK is in brown)

- Team(Name: Varchar, TeamCapacity: Integer)
  - Name -> TeamCapacity
- EscapeRoom(Name: Varchar, Genre: Varchar, TimeLimit: Integer)
  - Name -> Genre, TimeLimit
- BookingMakesFor(BookingID: Integer, BookingDate: Date, BookingTime: Time, **Tname**: Varchar, **Ename:** Varchar)
  - BookingID -> BookingDate, BookingTime, Tname, Ename
  - BookingDate, BookingTime, Tname, Ename -> BookingID
- RatingGivenToAssigns(RatingID: Integer, DateAdded: Date, Score: Integer, Criteria: Varchar, **Name:** Varchar, **Email**: Varchar)
  - RatingID -> DateAdded, Score, Criteria, Name, Email
  - DateAdded, Score, Criteria, Name, Email -> RatingID
- PuzzleHas(PuzzleID: Integer, Difficulty: Integer, Pname: Varchar, **Ename:** Varchar)
  - PuzzleID -> Difficulty, Pname, Ename
  - Pname, Ename -> PuzzleID, Difficulty
  - Pname -> Difficulty
- ScoreOnPuzzle(TeamName: Varchar, Points: Integer, **PuzzleID:** Integer)
  - Teamname, PuzzleID -> Points
- PropHave(PropID: Integer, Name: Varchar, Status: Varchar, **PuzzleID:** Integer)
  - PropID -> Name, Status, PuzzleID
- Users(Name: Varchar, Email: Varchar, Address: Varchar, City: Varchar, PostalCode: Varchar)
  - Email -> Name, Address, City, PostalCode
  - Address, PostalCode -> Name, Email, City
  - PostalCode -> City
- Employee(**Email**: Varchar, Position: Varchar, Salary: Integer)
  - Email -> Position, Salary
  - Position -> Salary
- Viewer(**Email**: Varchar, Age: Integer)
  - Email -> Age
- PlayerPartOf(**Email**: Varchar, Alias: Varchar, SkillLevel: Integer, PlayingStyle: Varchar, **Name**: Varchar, Since: Date)
  - Email -> Alias, SkillLevel, PlayingStyle, Name, Since

  - Alias -> Email, SkillLevel, PlayingStyle, Name, Since
- Fix(**Email**: Varchar, **PropID**: Integer)
  - No non-trivial FD
- Adjust(**Email**: Varchar, **PuzzleID**: Integer)
  - No non-trivial FD

Normalization

(Normalization was done in BCNF)

- Team(Name: Varchar, TeamCapacity: Integer)
  - Only FD identified by a PK or CK, so in BCNF
- EscapeRoom(Name: Varchar, Genre: Varchar, TimeLimit: Integer)
  - Only FD identified by a PK or CK, so in BCNF
- BookingMakesFor(BookingID: Integer, BookingDate: Date, BookingTime: Time, **Tname**: Varchar, **Ename:** Varchar)
  - Only FD identified by a PK or CK, so in BCNF
- RatingGivenToAssigns(RatingID: Integer, DateAdded: Date, Score: Integer, Criteria: Varchar, **Name:** Varchar, **Email**: Varchar)
  - Only FD identified by a PK or CK, so in BCNF
- PuzzleHas(PuzzleID: Integer, Difficulty: Integer, Pname: Varchar, **Ename:** Varchar)
  - Pname^{+} = {Pname, Diffculty}
  - Pname is not a superkey, so not in BCNF, needs decompose
  - Decompse using Pname -> Difficulty, we get
  - R1(Pname, **Ename**, PuzzleID), R2(Pname, Difficulty)
  - R1 will be PuzzleHas, R2 will be PuzzleDifficulty
- ScoreOnPuzzle(TeamName: Varchar, Points: Integer, **PuzzleID:** Integer)
  - Only FD identified by a PK or CK, so in BCNF
- PropHave(PropID: Integer, Name: Varchar, Status: Varchar, **PuzzleID:** Integer)
  - Only FD identified by a PK or CK, so in BCNF
- Users(Name: Varchar, Email: Varchar, Address: Varchar, City: Varchar, PostalCode: Varchar)
  - PostalCode^{+} = {PostalCode, City}
  - PostalCode is not a superkey, so not in BCNF, needs decompose
  - Decompose using PostalCode -> City, we get
  - R1(Name, Email, Address, PostalCode), R2(PostalCode, City)
  - R1 will be Users, R2 will be PostalCity
- Employee(**Email**: Varchar, Position: Varchar, Salary: Integer)
  - Position^{+} = {Position, Salary}
  - Position is not a superkey, so not in BCNF, needs decompose
  - Decompose using Position -> Salary, we get
  - R1(**Email**, Position), R2(Position, Salary)
  - R1 will be Employee, R2 will be PositionSalary

- Viewer(**Email**: Varchar, Age: Integer)
    - Only FD identified by a PK or CK, so in BCNF
- PlayerPartOf(**Email**: Varchar, Alias: Varchar, SkillLevel: Integer, PlayingStyle: Varchar, **Name**: Varchar, Since: Date)
    - Only FD identified by a PK or CK, so in BCNF
- Fix(**Email**: Varchar, **PropID**: Integer)
    - No non-trial FD, so in BCNF
- Adjust(**Email**: Varchar, **PuzzleID**: Integer)
    - No non-trial FD, so in BCNF

Final Tables

- Team(Name: Varchar, TeamCapacity: Integer)
    - PK: Name
    - TeamCapacity Default value is 10
- EscapeRoom(Name: Varchar, Genre: Varchar, TimeLimit: Integer)
    - PK: Name
    - TimeLimit Default value is 60
- BookingMakesFor(BookingID: Integer, BookingDate: Date, BookingTime: Time, **Tname**: Varchar, **Ename:** Varchar)
    - PK: BookingID
    - CK: {BookingDate, BookingTime, TName, Ename}
    - FK: Tname references Team(Name), EName references EscapeRoom(Name)
    - Tname is Not Null (participation constraint)
    - Ename is Not Null (participation constraint)
- RatingGivenToAssigns(RatingID: Integer, DateAdded: Date, Score: Integer, Criteria: Varchar, **Name:** Varchar, **Email**: Varchar)
    - PK: RatingID
    - CK: {Email, Name, DateAdded, Score, Criteria}
    - FK: Name references EscapeRoom, Email references Users
    - Name is Not Null (participation constraint)
    - Email is Not Null (participation constraint)
    - Score is Not Null, Default Value is 5
    - Criteria is Not Null
- PuzzleHas(Pname: Varchar, **Ename:** Varchar, PuzzleID: Integer)
    - PK: PuzzleID
    - CK: {Pname, Ename}
    - FK: Ename references EscapeRoom(Name)
    - Ename is Not Null (participation constraint)
- PuzzleDifficulty(Pname: Varchar, Difficulty: Integer)
    - PK: Pname
- ScoreOnPuzzle(TeamName: Varchar, Points: Integer, **PuzzleID:** Integer)

  - PK: {TeamName, PuzzleID}
  - FK: PuzzleID references PuzzleHas
- PropHave(PropID: Integer, Name: Varchar, Status: Varchar, **PuzzleID:** Integer)
  - PK: PropID
  - FK: PuzzleID references PuzzleHas
  - PuzzleID is Not Null (participation constraint)
- Users(Name: Varchar, Email: Varchar, Address: Varchar, PostalCode: Varchar)
  - PK: Email
  - CK: {Address, PostalCode}
  - Address is Not Null
  - PostalCode is Not Null
- PostalCity(PostalCode: Varchar, City: Varchar)
  - PK: PostalCode
  - City is Not Null
- Employee(**Email:** Varchar, Position: Varchar)
  - PK: Email
  - FK: Email references Users
- PositionSalary(Position: Varchar, Salary: Integer)
  - PK: Position
- Viewer(**Email**: Varchar, Age: Integer)
  - PK: Email
  - FK: Email references Users
- PlayerPartOf(**Email**: Varchar, Alias: Varchar, SkillLevel: Integer, PlayingStyle: Varchar, **Name**: Varchar, Since: Date)
  - PK: Email
  - CK: Alias
  - FK: Email references Users, Name references Team
  - Alias is Not Null and Unique
  - SkillLevel Default value is 1
  - Name is Not Null (participation constraint)
- Fix(**Email**: Varchar, **PropID**: Integer)
  - PK: Email, PropID
  - FK: Email references Users, PropID references PropHave
- Adjust(**Email**: Varchar, **PuzzleID**: Integer)
  - PK: Email, PuzzleID
  - FK: Email references Users, PuzzleID references PuzzleHas

SQL DDL Statements to create tables

```
CREATE TABLE Team (
   Name varchar,
   TeamCapacity integer DEFAULT 10,
   PRIMARY KEY(Name)
);

CREATE TABLE EscapeRoom (
   Name varchar,
   Genre varchar,
   TimeLimit integer DEFAULT 60,
   PRIMARY KEY(Name)
);

CREATE TABLE BookingMakesFor (
   BookingID integer,
   BookingDate date,
   BookingTime time,
   Tname varchar NOT NULL,
   Ename varchar NOT NULL,
   PRIMARY KEY(BookingID),
   FOREIGN KEY (Tname) references Team(Name),
   FOREIGN KEY (EName) references EscapeRoom(Name),
   UNIQUE(BookingDate, BookingTime, Tname, Ename)
);

CREATE TABLE RatingGivenToAssigns(
   RatingID integer,
   DateAdded date,
   Score integer NOT NULL DEFAULT 5,
   Criteria varchar NOT NULL,
   Name varchar NOT NULL,
   Email Varchar NOT NULL,
   PRIMARY KEY(rating),
   FOREIGN KEY(Name) references EscapeRoom(Name),
   FOREIGN KEY(Email) references Users,
   UNIQUE(Email, Name, DateAdded, Score, Criteria)
);

CREATE TABLE PuzzleHas(
   PuzzleID integer,
   Difficulty integer,
   Pname varchar,
   Ename varchar NOT NULL,
   PRIMARY KEY(PuzzleID),
   FOREIGN KEY(Ename) REFERENCES EscapeRoom(Name),
   UNIQUE(PName, EName)
);

CREATE TABLE ScoreOnPuzzle(
   TeamName varchar,
   Points integer,
   PuzzleID integer,
   PRIMARY KEY(TeamName, PuzzleID),
   FOREIGN KEY(PuzzleID) REFERENCES PuzzleHas
);

CREATE TABLE PuzzleDifficulty(
   Pname varchar,
   Difficulty integer,
   PRIMARY KEY(Pname)
);

CREATE TABLE PropHave(
   PropID integer,
   Name varchar,
   Status varchar,
   PuzzleID integer NOT NULL,
   PRIMARY KEY(PropID),
   FOREIGN KEY(PuzzleID) REFERENCES PuzzleHas(PuzzleID)
);

CREATE TABLE Users(
   Name varchar,
   Email varchar,
   Address varchar NOT NULL,
   PostalCode varchar NOT NULL,
   PRIMARY KEY(Email),
   UNIQUE(Address, PostalCode)
);

CREATE TABLE PostalCity(
   PostalCode varchar,
   City varchar NOT NULL,
   PRIMARY KEY(PostalCode)
);

CREATE TABLE Employee(
   Email varchar,
   Position varchar,
   PRIMARY KEY(Email),
   FOREIGN KEY(Email) REFERENCES Users(Email)
);

CREATE TABLE PositionSalary(
   Position varchar,
   Salary integer,
   PRIMARY KEY(Position)
);

CREATE TABLE Viewer(
   Email varchar,
   Age integer,
   PRIMARY KEY(Email)
);

CREATE TABLE PlayerPartOf(
   Email varchar,
   Alias varchar NOT NULL UNIQUE,
   SkillLevel integer DEFAULT 1,
   PlayingStyle varchar,
   Name varchar NOT NULL,
   Since date,
   PRIMARY KEY (Email),
   FOREIGN KEY (Email) REFERENCES Users(Email)
   FOREIGN KEY (Name) REFERENCES Team(Name)
);

CREATE TABLE Fix(
   Email varchar,
   PropID integer,
   PRIMARY (Email, PropID),
   FOREIGN KEY (Email) REFERENCES Users(Email),
   FOREIGN KEY (PropID) REFERENCES PropHave(PropID)
);

CREATE TABLE Adjust(
   Email varchar,
   PuzzleID integer,
   PRIMARY(Email, PuzzleID),
   FOREIGN KEY (Email) REFERENCES Users(Email),
   FOREIGN KEY (PuzzleID) REFERENCES PuzzleHas(PuzzleID)
);
```

INSERT Statements

Team:

```
INSERT INTO Team VALUES ("SEN", 6);
INSERT INTO Team VALUES ("The Rebels", 5);
INSERT INTO Team VALUES ("T369", 5);
INSERT INTO Team VALUES ("Dragon Slayers", 7);
INSERT INTO Team VALUES ("Let us cook", 4);
```

Escape room:

```
INSERT INTO EscapeRoom VALUES ("Mary's Wonderland", "Classic", 60)
INSERT INTO EscapeRoom VALUES ("The giggling", "Horor", 75)
INSERT INTO EscapeRoom VALUES ("Fifteen reasons Vincent lied", "Role-playing", 75)
INSERT INTO EscapeRoom VALUES ("Escape Arkaham", "Horor", 60)
INSERT INTO EscapeRoom VALUES ("Leviathan", "Classic", 120)
```

BookingMakesFor:

```
INSERT INTO BookingMakesFor VALUES ("292872", "2023-11-08", "14:30:00", "SEN", "Mary's Wonderland")
INSERT INTO BookingMakesFor VALUES ("234597", "2023-12-31", "18:30:00", "The Rebels", "The giggling")
INSERT INTO BookingMakesFor VALUES ("292872", "2023-11-08", "14:30:00", "T369", "Fifteen reasons Vincent lied")
INSERT INTO BookingMakesFor VALUES ("292872", "2023-11-08", "14:30:00", "Dragon Slayers", "Escape Arkaham")
INSERT INTO BookingMakesFor VALUES ("292872", "2023-11-08", "14:30:00", "Let us cook", "Leviathan")
```

RatingGivenToAssigns:

```
INSERT INTO RatingGivenToAssigns VALUES ("37283", "2023-11-21", "5",  "Mary's Wonderland", "johndoe@gmail.com")
INSERT INTO RatingGivenToAssigns VALUES ("39283", "2023-10-17", "4", "Mary's Wonderland", "amychen@gmail.com")
INSERT INTO RatingGivenToAssigns VALUES ("48293", "2023-12-13", "3", "Fifteen reasons Vincent lied", "avasimpson@gmail.com")
INSERT INTO RatingGivenToAssigns VALUES ("47193", "2023-08-09", "5", "Fifteen reasons Vincent lied", "johnwick@outlook.com")
INSERT INTO RatingGivenToAssigns VALUES ("39284", "2024-01-08", "4", "Leviathan",
"davidhahn@gmail.com")
```

PuzzleHas:

```
INSERT INTO PuzzleHas VALUES ("Room lock", "Mary's Wonderland", "12")
INSERT INTO PuzzleHas VALUES ("Shelf lock", "Fifteen reasons Vincent lied", "21")
INSERT INTO PuzzleHas VALUES ("Shelf lock", "Fifteen reasons Vincent lied", "23")
INSERT INTO PuzzleHas VALUES ("Statue rotation", "Leviathan", "9")
INSERT INTO PuzzleHas VALUES ("Room lock", "Leviathan", "1")
```

ScoreOnPuzzle:

```
INSERT INTO ScoreOnPuzzle VALUES ("Let us cook", 63, "12")
INSERT INTO ScoreOnPuzzle VALUES ("T369", 73, "1")
INSERT INTO ScoreOnPuzzle VALUES ("T369", 50, "24")
INSERT INTO ScoreOnPuzzle VALUES ("The rabbits", 0, "13")
INSERT INTO ScoreOnPuzzle VALUES ("The Rebels", 89, "6")
```

PuzzleDifficulty:

```
INSERT INTO PuzzleDifficulty VALUES ("Room lock", 4)
INSERT INTO PuzzleDifficulty VALUES ("Crystal ball puzzle", 3)
INSERT INTO PuzzleDifficulty VALUES ("Statue rotation", 2)
INSERT INTO PuzzleDifficulty VALUES ("Room lock", 5)
INSERT INTO PuzzleDifficulty VALUES ("Shelf lock", 2)
```

PuzzleHas:

```
INSERT INTO PuzzleHas VALUES ("34", "Simple Lock", "21")
INSERT INTO PuzzleHas VALUES ("21", "Simple Lock", "12")
INSERT INTO PuzzleHas VALUES ("12", "Simple Lock", "23")
INSERT INTO PuzzleHas VALUES ("6", "Statue", "9")
INSERT INTO PuzzleHas VALUES ("7", "Electronic Password Lock", "1")
```

Users:

```
INSERT INTO _Users_ VALUES ("John Doe", "4090 Wrangler Rd", "T1X 0K2")
INSERT INTO _Users_ VALUES ("Mary Jane", "1000 Bridgeport Rd", "V6V A03")
INSERT INTO _Users_ VALUES ("Chaoyang Zhu", "435 Cambie St", "F8S 4G3")
INSERT INTO _Users_ VALUES ("Mark Jang", "21 Brentwood Blvd", "V6S 3B8")
INSERT INTO _Users_ VALUES ("Muhammad Ali", "78 Hanson St", "M4C 1A1")
```

Postal City:

```
INSERT INTO PostalCity VALUES ("V6V 0A3", "Richmond")
INSERT INTO PostalCity VALUES ("M4C 1A1", "Toronto")
INSERT INTO PostalCity VALUES ("M3H 5J8", "Toronto")
INSERT INTO PostalCity VALUES ("T1X 0K2", "Calgary")
INSERT INTO PostalCity VALUES ("T5A 0A1", "Edmonton")
```

Employee:

```
INSERT INTO Employee VALUES ("johndoe@gmail.com", "Front-desk")
INSERT INTO Employee VALUES ("amychen@gmail.com", "Front-desk")
INSERT INTO Employee VALUES ("avasimpson@outlook.com", "Web develop and maintenance")
INSERT INTO Employee VALUES ("johnwick@outlook.com", "Prop manager")
INSERT INTO Employee VALUES ("davidhahn@gmail.com", "Puzzle design and refinements")
```

PositionSalary:

```
INSERT INTO PositionSalary VALUES ("Front-desk", "35")
INSERT INTO PositionSalary VALUES ("Prop manager", "35")
INSERT INTO PositionSalary VALUES ("Web develop and maintenance", "45")
INSERT INTO PositionSalary VALUES ("Puzzle design and refinements", "45")
INSERT INTO PositionSalary VALUES ("Sanitation", "35")
```

Viewer:

```
INSERT INTO Viewer VALUES ("johndoe@gmail.com", "18")
INSERT INTO Viewer VALUES ("amychen@gmail.com", "23")
INSERT INTO Viewer VALUES ("avasimpson@outlook.com", "")
INSERT INTO Viewer VALUES ("johnwick@outlook.com", "")
INSERT INTO Viewer VALUES ("davidhahn@gmail.com", "45")
```

PlayerPartOf:

```
INSERT INTO PlayerPartOf VALUES ("ssdfe3@gmail.com", "SSD", 2, "Puzzle solver", "", "2022-09-03")
INSERT INTO PlayerPartOf VALUES ("fikz2334@gmail.com", "fikz", "Item finder", "Fred", "2019-07-13")
INSERT INTO PlayerPartOf VALUES ("som12577@outlook.com", "som", "Puzzle solver", "Sam Rayman", "2017-08-14")
INSERT INTO PlayerPartOf VALUES ("lanakiran343@outlook.com", "NaNa", "Alana Kiran", "Item finder", "2020-06-07")
INSERT INTO PlayerPartOf VALUES ("sophie_love233@gmail.com", "Sophine", "Puzzle solver", "2021-04-05")
```

Fix:

```
INSERT INTO Fix VALUES ("johnwick@outlook.com", "12")
INSERT INTO Fix VALUES ("johnwick@outlook.com", "34")
INSERT INTO Fix VALUES ("johnwick@outlook.com", "17")
INSERT INTO Fix VALUES ("AnnaRiver@outlook.com", "8")
INSERT INTO Fix VALUES ("AnnaRiver@outlook.com", "9")
```

Adjust:

```
INSERT INTO Adjust VALUES ("johnwick@outlook.com", "1")
INSERT INTO Adjust VALUES ("johnwick@outlook.com", "3")
INSERT INTO Adjust VALUES ("johnwick@outlook.com", "12")
INSERT INTO Adjust VALUES ("AnnaRiver@outlook.com", "23")
INSERT INTO Adjust VALUES ("AnnaRiver@outlook.com", "19")
```