

# CPSC 304 PROJECT

## COVER PAGE

Milestone #:   2  

Date: 2024-02-27

Group Number:   94  

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Aaditya Desai	41381799	w8g6e	desai.aaditya.d@gmail.com
Aryaan Habib	11833316	z2a9c	habibaryaan@gmail.com
Devin Proothi	68619774	l9q5j	devinpr2021@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

## **Summary**

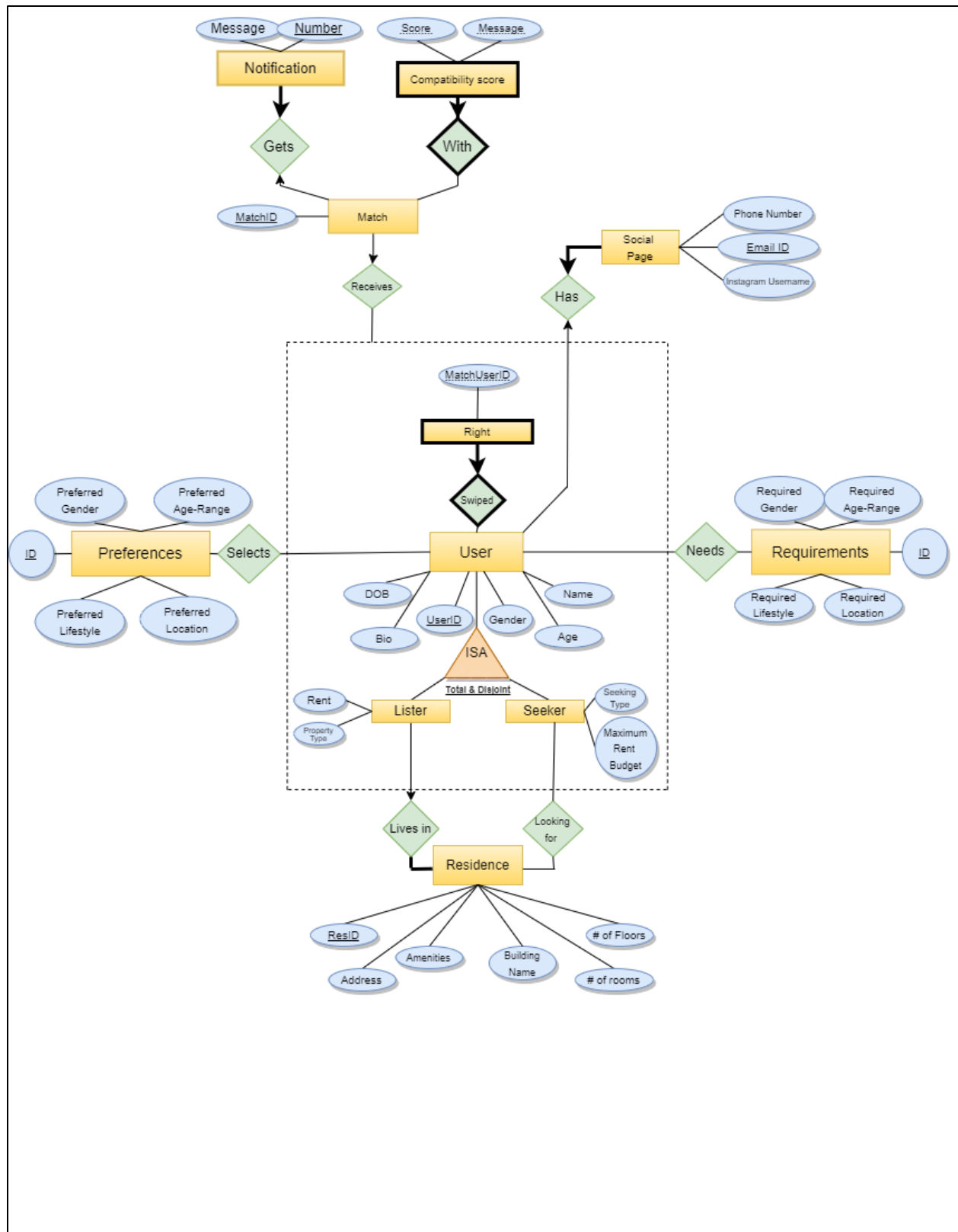
Swipemates is a roommate matching application designed for the students of University of British Columbia (UBC), aiming to simplify the process of finding compatible roommates for shared accommodation on and off campus. Users create detailed profiles, specify preferences and requirements, and swipe through potential matches. The application calculates compatibility scores and facilitates communication between matched users, enhancing the overall housing experience for UBC students.

## **ER DIAGRAM**














In the updated ER diagram, we introduced a new entity called 'Right' to represent the action of a user swiping right on another user. This entity forms a weak relationship with the 'User' entity, denoted by the 'swipe' relationship, indicating that a 'Right' instance depends on a corresponding 'User' instance. Additionally, an aggregation relationship was established between 'User', 'Right', and 'Match', signifying that a 'Match' is composed of 'User' and 'Right' entities. This modification allows us to track the swipes made by a user and facilitates the recording of successful matches.

These changes enhance the ER diagram by capturing the relationship dynamics within the system and enabling effective tracking of user interactions and matches

Added a Total and Disjoint constraint on the ISA, signifying that a user can only be a lister or seeker but not both.



**Relational Schema** (Primary Key: Underline, Foreign Key: **Bold**, Candidate Key: *Italics*)

-  User (UserID: varchar, Name: varchar NOT NULL, Gender: varchar NOT NULL, Age: integer, Bio: text, DOB: Date NOT NULL)
-  **ISA:** Lister\_LivesIn (**ListerID**: varchar, **ResID**: varchar NOT NULL, Rent: integer NOT NULL, Property Type: varchar NOT NULL)  
Seeker (**SeekerID**: varchar, Maximum Rent Budget: integer, Seeking Type: varchar)
-  Residence (ResID: varchar, Address: varchar, Amenities: varchar, Number of Floors: integer, Building Name: varchar, Number of Rooms: integer)
-  LookingFor (**ResID**: varchar, **UserID**: varchar)
-  Preferences (PreferencesID: varchar, Preferred Gender: varchar, Preferred Age-Range: varchar, Preferred Lifestyle: varchar, Preferred Location: varchar)
-  Selects (**UserID**: varchar, **PreferencesID**: varchar)
-  Requirements (RequirementsID: varchar, Required Gender: varchar, Required Age-Range: varchar, Required Lifestyle: varchar, Required Location: varchar)
-  Needs (**UserID**: varchar, **RequirementsID**: varchar)
-  Receives\_Match (MatchID: varchar, **UserID**: varchar NOT NULL, **MatchUserID**: varchar NOT NULL)
-  Gets\_Notification (**MatchID**: varchar, Number: integer NOT NULL, Message: varchar NOT NULL)
-  CompatibilityScore (**MatchID**: varchar, Score: varchar NOT NULL, Message: varchar)
-  SocialPage\_Has (EmailID: varchar, **UserID**: varchar, *Phone Number*: integer, *Instagram Username*: varchar)
-  Swipe\_Right (MatchUserId: varchar, **UserID**: varchar)

## **Functional Dependencies**

- User:
  - UserID (PK) → Name, Gender, Age, Bio, DOB
  - DOB → Age
- Lister\_LivesIn:
  - ListerID (PK) → Name, Gender, Age, Bio, DOB, ResID, Rent
  - DOB → Age
- Seeker:
  - SeekerID (PK) → Name, Gender, Age, Bio, DOB, Maximum Rent Budget, SeekingType
  - DOB → Age
- Residence:
  - ResID (PK) → Address, Amenities, Number of Floors, Building Name, Number of Rooms
  - Address → Building Name
  - Building Name → Number of Floors
- Looking For:
  - FDs in this relation are trivial and therefore not mentioned.
- Preferences:
  - PreferencesID (PK) → Preferred Gender, Preferred Age Range, Preferred Lifestyle, Preferred Location
- Selects:
  - FDs in this relation are trivial and therefore not mentioned.
- Requirements:
  - RequirementsID (PK) → Required Gender, Required Age Range, Required Lifestyle, Required Location
- Needs:
  - FDs in this relation are trivial and therefore not mentioned.
- Receives\_Match:
  - MatchID (PK) → UserID, MatchUserID
- Gets\_Notification:

- Number (PK)  $\rightarrow$  MatchID, Message
- Compatibility Score:
  - Score  $\rightarrow$  Message
- SocialPage\_Has:
  - EmailID (PK)  $\rightarrow$  UserID, Phone Number, Instagram Username
  - Phone Number (CK)  $\rightarrow$  EmailID, UserID, Instagram Username
  - Instagram Username (CK)  $\rightarrow$  EmailID, UserID, Phone Number
- Swipe\_Right:
  - FDs in this relation are trivial and therefore not mentioned.

### **Normalization:**

- **User**

- DOB  $\rightarrow$  Age

$DOB^+ = \{DOB, Age\}$

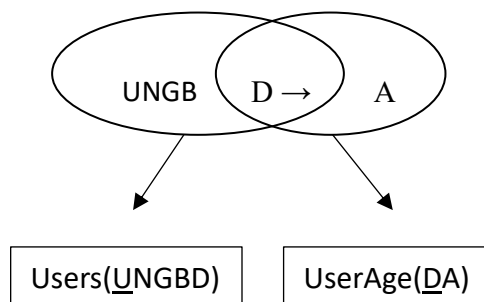
Since DOB is not a superkey, this dependency is not in BCNF.

User has the following attributes: UserID, Name, Gender, Age, Bio, DOB

Let User =  $\{U, N, G, A, B, D\}$

Decomposing Relation User (U, N, G, A, B, D) on  $D \rightarrow A$

then



Both Users & UserAge are in BCNF and all the dependencies are preserved.

Therefore, User is decomposed into 2 new relations, i.e. Users and UserAge.

[Note: Underline (under) : indicates Primary Key]

- **Lister**

- DOB  $\rightarrow$  Age

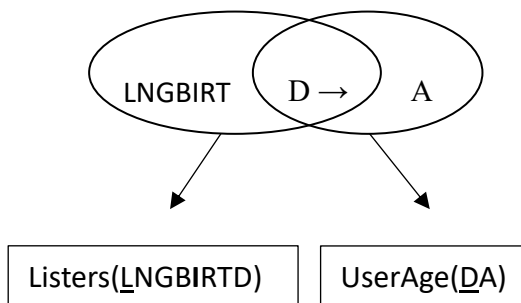
$$DOB^+ = \{DOB, Age\}$$

Since DOB is not a superkey, this dependency is not in BCNF.

Lister has the following attributes: ListerID, Name, Gender, Age, Bio, DOB, ResId, Rent, Property Type.

Let Lister = {L, N, G, A, B, D, I, R, T}

Decomposing Relation Lister (L, N, G, A, B, D, I, R, T) on  $D \rightarrow A$   
then



Both Listers & UserAge are in BCNF and all the dependencies are preserved.

Therefore, Lister is decomposed into 2 new relations, i.e. Listers and UserAge.

[Note: Underline (under): indicates Primary Key, Bold (**bold**) : indicates Foreign Key]

- **Seeker**

- DOB  $\rightarrow$  Age

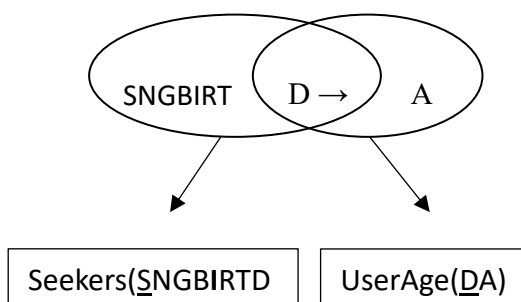
$$DOB^+ = \{DOB, Age\}$$

Since DOB is not a superkey, this dependency is not in BCNF.

Seeker has the following attributes: SeekerID, Name, Gender, Age, Bio, DOB, ResId, Maximum Rent Budget, Seeking Type.

Let Seeker = {S, N, G, A, B, D, I, R, T}

Decomposing Relation Seeker (S, N, G, A, B, D, I, R, T) on  $D \rightarrow A$   
then



Both Seekers & UserAge are in BCNF and all the dependencies are preserved.  
 Therefore, Seeker is decomposed into 2 new relations, i.e. Seekers and UserAge.  
 [Note: Underline (under): indicates Primary Key, Bold (**bold**) : indicates Foreign Key]

- **Residence**

- Building Name  $\rightarrow$  Number of Floors
- Address  $\rightarrow$  Building Name

$\text{Building Name}^+ = \{\text{Building Name}, \text{Number of Floors}\}$

Since Building Name is not a superkey, this dependency is not in BCNF.

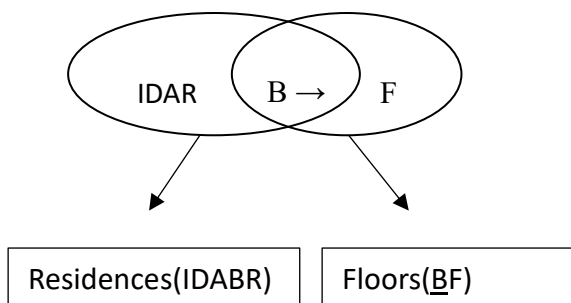
$\text{Address}^+ = \{\text{Address}, \text{Building Name}, \text{Number of Floors}\}$

Since Address is not a superkey, this dependency is not in BCNF.

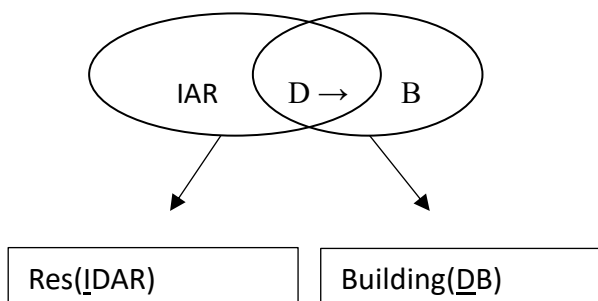
Residence has the following attributes: ResID, Address, Amenities, Number of Floors, Building Name, Number of Rooms.

Let  $\text{Residence} = \{I, D, A, F, B, R\}$

Decomposing Relation Residence (I, D, A, F, B, R) on  $B \rightarrow F$  then



Residences is not in BCNF and can be further decomposed on  $D \rightarrow B$ :





Floors, Res & Building are all in BCNF and all the dependencies are preserved.

Therefore, Residence is decomposed into 3 new relations, i.e. Floors, Res & Building.

[Note: Underline (under): indicates Primary Key]

- Compatibility Score

- $\text{Score} \rightarrow \text{Message}$

$\text{Score}^+ = \{\text{Score}, \text{Message}\}$

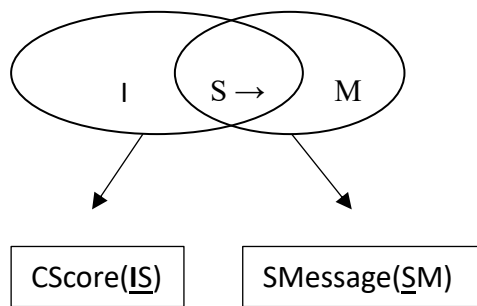
Since Score is not a superkey, this dependency is not in BCNF.

Compatibility Score has the following attributes: MatchID, Score, Message

Let Compatibility Score = {I, S, M}

Decomposing Relation Compatibility Score (I, S, M) on  $S \rightarrow M$

then



Both CScore & SMessage are in BCNF and all the dependencies are preserved.

Therefore, Compatibility Score is decomposed into 2 new relations, i.e. CScore & SMessage.

[Note: Underline (under): indicates Primary Key, Bold (**bold**) : indicates Foreign Key]

## SQL DDL Statements

```
CREATE TABLE UserAge (  
    DOB DATE PRIMARY KEY,  
    Age INT NOT NULL  
);
```

```
CREATE TABLE Users (  
    UserID VARCHAR(16) PRIMARY KEY,  
    Name VARCHAR(255) NOT NULL,  
    Gender VARCHAR(10) NOT NULL,  
    Bio TEXT,  
    DOB DATE NOT NULL,  
    FOREIGN KEY (DOB) REFERENCES UserAge(DOB)  
);
```

```
CREATE TABLE Floors (  
    BuildingName VARCHAR(255) PRIMARY KEY,  
    NumberOfFloors INT NOT NULL  
);
```

```
CREATE TABLE Building (  
    Address VARCHAR(255) PRIMARY KEY,  
    BuildingName VARCHAR(255) NOT NULL,  
    FOREIGN KEY (BuildingName) REFERENCES Floors(BuildingName)  
        ON UPDATE CASCADE  
);
```

```
CREATE TABLE Res (  
    ResID VARCHAR(16) PRIMARY KEY,  
    Address VARCHAR(255) NOT NULL,  
    Amenities TEXT,  
    NumberOfRooms INT,  
    FOREIGN KEY (Address) REFERENCES Building(Address)  
        ON UPDATE CASCADE  
);
```

```
CREATE TABLE Lister (  

```

```
ListerID VARCHAR(16) PRIMARY KEY,  
ResID VARCHAR(16),  
Rent INT,  
FOREIGN KEY (ListerID) REFERENCES Users(UserID)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE,  
FOREIGN KEY (ResID) REFERENCES Res(ResID)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE  
);
```

```
CREATE TABLE Seeker (  
    SeekerID VARCHAR(16) PRIMARY KEY,  
    MaximumRentBudget INT,  
    SeekingType VARCHAR(255),  
    FOREIGN KEY (SeekerID) REFERENCES Users(UserID)  
        ON DELETE CASCADE  
        ON UPDATE CASCADE  
);
```

```
CREATE TABLE LookingFor (  
    ResID VARCHAR(16),  
    UserID VARCHAR(16),  
    PRIMARY KEY (ResID, UserID),  
    FOREIGN KEY (ResID) REFERENCES Res(ResID)  
        ON DELETE CASCADE  
        ON UPDATE CASCADE,  
    FOREIGN KEY (UserID) REFERENCES Users(UserID)  
        ON DELETE CASCADE  
        ON UPDATE CASCADE  
);
```

```
CREATE TABLE Preferences (  
    PreferencesID VARCHAR(16) PRIMARY KEY,  
    PreferredGender VARCHAR(10),  
    PreferredAgeRange VARCHAR(20),  
    PreferredLifestyle VARCHAR(255),  
    PreferredLocation VARCHAR(255)  
);
```

```
CREATE TABLE Selects (  
    UserID VARCHAR(16),  
    PreferencesID VARCHAR(16),  
    PRIMARY KEY (UserID, PreferencesID),  
    FOREIGN KEY (UserID) REFERENCES Users(UserID)  
        ON DELETE CASCADE  
        ON UPDATE CASCADE,  
    FOREIGN KEY (PreferencesID) REFERENCES Preferences(PreferencesID)  
        ON DELETE CASCADE  
        ON UPDATE CASCADE  
);
```

```
CREATE TABLE Requirements (  
    RequirementsID VARCHAR(16) PRIMARY KEY,  
    RequiredGender VARCHAR(10),  
    RequiredAgeRange VARCHAR(20),  
    RequiredLifestyle VARCHAR(255),  
    RequiredLocation VARCHAR(255)  
);
```

```
CREATE TABLE Needs (  
    UserID VARCHAR(16),
```

```
RequirementsID VARCHAR(16),
PRIMARY KEY (UserID, RequirementsID),
FOREIGN KEY (UserID) REFERENCES Users(UserID)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
FOREIGN KEY (RequirementsID) REFERENCES
Requirements(RequirementsID)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

```
CREATE TABLE SocialPageHas (
    EmailID VARCHAR(255) PRIMARY KEY,
    UserID VARCHAR(16) NOT NULL,
    PhoneNumber VARCHAR(15) UNIQUE,
    InstagramUsername VARCHAR(30) UNIQUE,
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

```
CREATE TABLE SwipeRight (
    MatchUserID VARCHAR(16),
    UserID VARCHAR(16),
    PRIMARY KEY (MatchUserID, UserID),
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

```
CREATE TABLE ReceivesMatch (
    MatchID VARCHAR(16),
```

```

        UserID VARCHAR(16) NOT NULL,
        MatchUserID VARCHAR(16) NOT NULL,
        PRIMARY KEY (MatchID),
        FOREIGN KEY (UserID) REFERENCES Users(UserID)
            ON DELETE CASCADE
            ON UPDATE CASCADE,
        FOREIGN KEY (MatchUserID) REFERENCES SwipeRight(MatchUserID)
            ON DELETE CASCADE
            ON UPDATE CASCADE
    );

CREATE TABLE GetsNotification (
    Number INT PRIMARY KEY,
    MatchID VARCHAR(16) NOT NULL,
    Message VARCHAR(255) NOT NULL,
    FOREIGN KEY (MatchID) REFERENCES ReceivesMatch(MatchID)
        ON DELETE CASCADE
);

CREATE TABLE SMessage (
    Score VARCHAR(10) PRIMARY KEY,
    Message TEXT NOT NULL
);

CREATE TABLE CScore (
    MatchID VARCHAR(16),
    Score VARCHAR(10),
    PRIMARY KEY (MatchID, Score),
    FOREIGN KEY (MatchID) REFERENCES ReceivesMatch(MatchID)
        ON DELETE CASCADE,
    FOREIGN KEY (Score) REFERENCES SMessage(Score)
        ON UPDATE CASCADE
);

```

## ON DELETE CASCADE

);

### **Insert Statements**

INSERT INTO UserAge (DOB, Age) VALUES

('2000-12-26', 23),  
('1988-12-12', 36),  
('1990-07-23', 34),  
('2000-05-16', 24),  
('1993-11-30', 31);

INSERT INTO Users (UserID, Name, Gender, Bio, DOB) VALUES

('L123456', 'Devin Proothi', 'Male', 'UBC, Year 3, CS.', '2000-12-26'),  
('S123456', 'Aryaan Habib', 'Male', 'UBC, Year 3, Forestry.', '1988-12-12'),  
('L101010', 'Nabeel Ali', 'Female', 'UBC. Year 3. Psychology.', '1990-07-23'),  
('S101010', 'Kartikeya Chaturvedi', 'Male', 'UBC. Year 4. Sauder', '2000-12-26'),  
('S987654', 'Aaditya Desai', 'Male', 'UBC. Year 3. English.', '1993-11-30'),  
('L765432', 'Sid Jain', 'Male', 'UBC. Year 6. International Relations.', '2000-12-26'),  
('L345678', 'Rishav', 'Male', 'UBC. Year 5. International Relations.', '1993-11-30'),  
('L012120', 'Aneyeant', 'Female', 'UBC. Year 1. International Relations.', '1993-11-30'),  
('S765432', 'Anmol', 'Male', 'UBC. Year 3. International Relations.', '1988-12-12'),  
('S012120', 'Jaimin', 'Male', 'UBC. Year 4. International Relations.', '1990-07-23');

INSERT INTO Floors (BuildingName, NumberOfFloors) VALUES

('Marine Drive', 10),  
('Walter Gage', 20),  
('Ponderosa', 15),  
('Orchard Commons', 24),  
('Totem Park', 12);

INSERT INTO Building (Address, BuildingName) VALUES

('123 Main St', 'Marine Drive'),  
('456 Elm St', 'Orchard Commons'),  
('789 Oak St', 'Totem Park'),

('101 Pine St', 'Ponderosa'),  
('202 Birch St', 'Marine Drive');

INSERT INTO Res (ResID, Address, Amenities, NumberOfRooms) VALUES

('res01', '123 Main St', 'Pool, Gym, Parking', 3),  
('res02', '456 Elm St', 'Gym, Parking', 2),  
('res03', '789 Oak St', 'Pool, Parking', 4),  
('res04', '101 Pine St', 'Gym', 1),  
('res05', '202 Birch St', 'Parking', 3);

INSERT INTO Lister (ListerID, ResID, Rent) VALUES

('L123456', 'res01', 1500),  
('L101010', 'res02', 1200),  
('L765432', 'res03', 2000),  
('L345678', 'res04', 1000),  
('L012120', 'res05', 1800);

INSERT INTO Seeker (SeekerID, MaximumRentBudget, SeekingType) VALUES

('S123456', 1600, 'Short Term'),  
('S101010', 1100, 'Long Term'),  
('S987654', 1900, 'Short Term'),  
('S765432', 900, 'Long Term'),  
('S012120', 1700, 'Short Term');

INSERT INTO LookingFor (ResID, UserID) VALUES

('res01', 'S123456'),  
('res02', 'S101010'),  
('res03', 'S987654'),  
('res04', 'S765432'),  
('res05', 'S012120');

INSERT INTO Preferences (PreferencesID, PreferredGender, PreferredAgeRange, PreferredLifestyle,  
PreferredLocation) VALUES

('pref01', 'Male', '25-35', 'Active', 'Off campus'),



```
('pref02', 'Female', '30-40', 'Quiet', 'On campus'),  
( 'pref03', 'Female', '20-30', 'Active', 'On campus'),  
( 'pref04', 'Male', '25-35', 'Quiet', 'Off campus'),  
( 'pref05', 'Female', '30-40', 'Active', 'On campus');
```

INSERT INTO Selects (UserID, PreferencesID) VALUES

```
('S123456', 'pref01'),  
( 'L101010', 'pref02'),  
( 'S987654', 'pref03'),  
( 'L012120', 'pref04'),  
( 'L345678', 'pref05');
```

INSERT INTO Requirements (RequirementsID, RequiredGender, RequiredAgeRange, RequiredLifestyle, RequiredLocation) VALUES

```
('req01', 'Male', '20-30', 'Active', 'On campus'),  
( 'req02', 'Female', '30-40', 'Quiet', 'Off campus'),  
( 'req03', 'Male', '25-35', 'Active', 'On campus'),  
( 'req04', 'Female', '20-30', 'Quiet', 'Off campus'),  
( 'req05', 'Male', '30-40', 'Active', 'On campus');
```

INSERT INTO Needs (UserID, RequirementsID) VALUES

```
('L123456', 'req01'),  
( 'S123456', 'req02'),  
( 'L101010', 'req03'),  
( 'S101010', 'req04'),  
( 'S987654', 'req05');
```

INSERT INTO SocialPageHas (EmailID, UserID, PhoneNumber, InstagramUsername) VALUES

```
('devin_proothi@gmail.com', 'L123456', '778-123-4567', 'devinproothi'),  
( 'aryaan_habib@gmail.com', 'S123456', '604-123-4567', 'aryaanhabib'),  
( 'nabeel_ali@gmail.com', 'L101010', '778-765-4321', 'nabeelali'),  
( 'kartikkeya_chaturvedi@gmail.com', 'S101010', '604-765-4321', 'kartikkeyachaturvedi'),  
( 'aaditya_desai@gmail.com', 'S987654', '778-987-6543', 'aadityadesai');
```

```
INSERT INTO SwipeRight (MatchUserID, UserID) VALUES
('L123456', 'S123456'),
('S123456', 'L101010'),
('L101010', 'S101010'),
('S101010', 'S987654'),
('S987654', 'L123456');
```

```
INSERT INTO ReceivesMatch (MatchID, UserID, MatchUserID) VALUES
('match01', 'L123456', 'S123456'),
('match02', 'S123456', 'L101010'),
('match03', 'L101010', 'S101010'),
('match04', 'S101010', 'S987654'),
('match05', 'S987654', 'L123456');
```

```
INSERT INTO GetsNotification (Number, MatchID, Message) VALUES
(1, 'match01', 'You have a new match!'),
(2, 'match02', 'New match found!'),
(3, 'match03', 'It is a match!'),
(4, 'match04', 'Match alert!'),
(5, 'match05', 'You just found the one!');
```

```
INSERT INTO SMessage (Score, Message) VALUES
('A+', 'Perfect Match!'),
('A', 'Great Match!'),
('B+', 'Good Match!'),
('B', 'Not Bad!'),
('C+', 'There's Potential!');
```

```
INSERT INTO CScore (MatchID, Score) VALUES
('match01', 'A'),
('match02', 'B+'),
('match03', 'A+'),
('match04', 'B'),
('match05', 'C+');
```