**sequel dating**

July 9, 2018

Bryan Reznicek and Jeremy Cooley

|  |
| --- |
|  |
| Database Design (E-R Diagram) |
|  |
| Executive Summary |

Database Design (E-R Diagram)

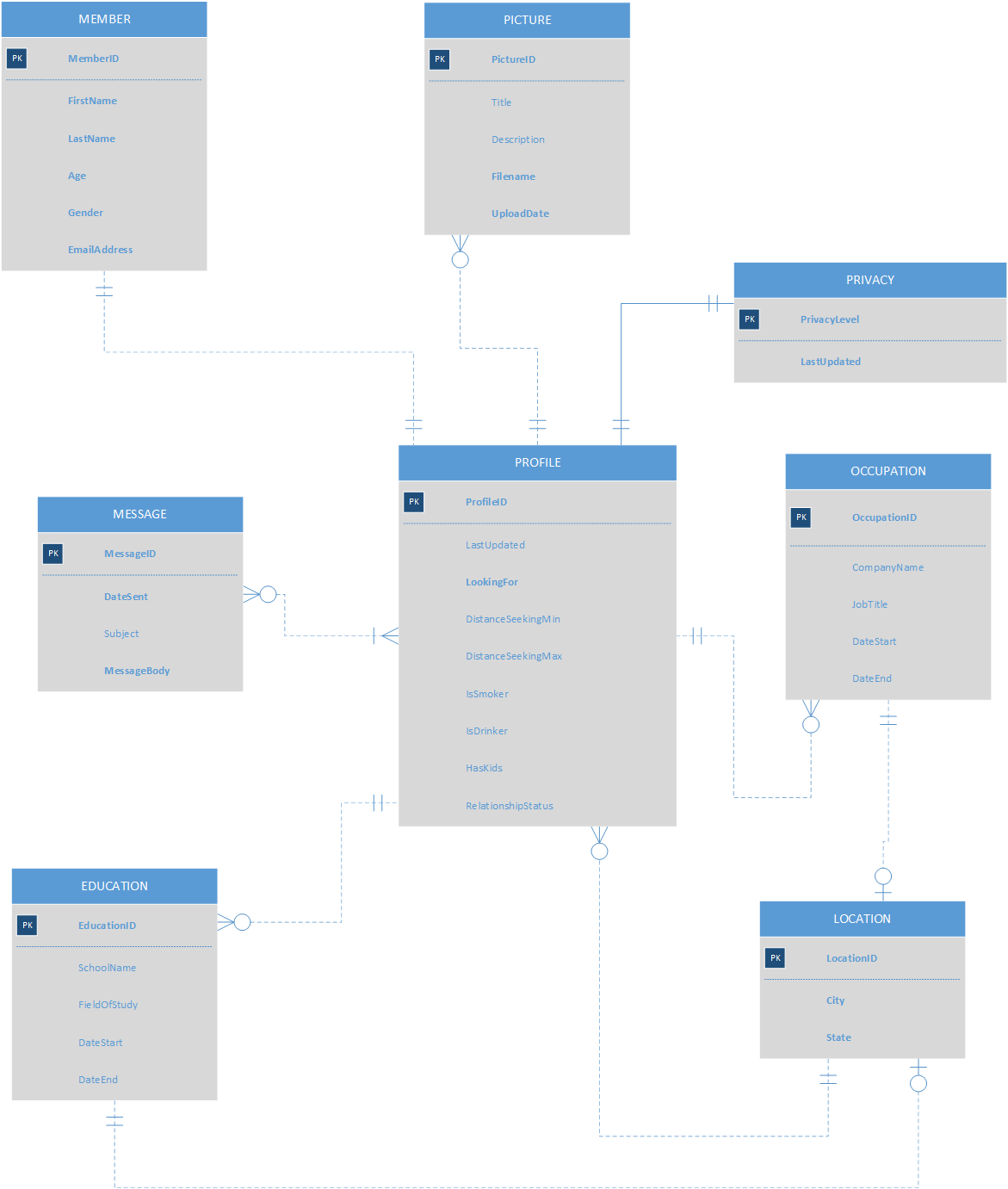
Executive Summary

The database enclosed is to be used to set up an online dating/matchmaking website in a social media-type foundation. The social media layout will be utilized to bring more familiarity and fluidity to the user experience, while simultaneously allowing for easier implementation of data for the developers. The users’ preferences will be used to generate numerous queries on the database that will match users based on a finite number of factors such as education, occupation, whether they smoke, whether they drink, if they have kids, and if they are married, single, divorced, or complicated. The database may also be used on the developers’ end to generate reports to view data such as the profiles most recently updated, profiles in a particular location, or users in a certain age bracket. The database will also be implemented using MySQL utilizing the same design detailed in this summary.

The implementation of this database contains seven (7) tables that provide information about the ***Profiles, Location, Pictures, Education, Occupation, Messages,*** and ***Privacy*** (Settings). These tables are the primary entities of the website and are used to define the relationships amongst the other entities. The ***Profiles*** table contains the associated member’s ID, first name, last name, location, profile ID, email address, and their gender, distance seeking minimum, distance seeking maximum, smoker, drinker, has kids or not (0 for no/false, 1 for yes/true) , and relationship status. (int – 1. Single 2. Married 3. Open Relationship). The ***Location*** table provides the city and state for the given location. The ***Pictures*** table provides information such as the title, description, filename, and upload date. The ***Education*** table contains information (if applicable) about a user’s educational background to include the high school or college name, location ID, start and graduation dates. The ***Occupation*** table contains information about the user’s work experience (if applicable) such as company name, company location ID, industry type, job title, date started, and date ended (if applicable). The ***Messages*** table contains information such as date sent, subject, and message body. The ***Privacy*** table contains information about a profile’s privacy settings utilizing a level-scale ranging from 0-3 (OFF to HIGH).

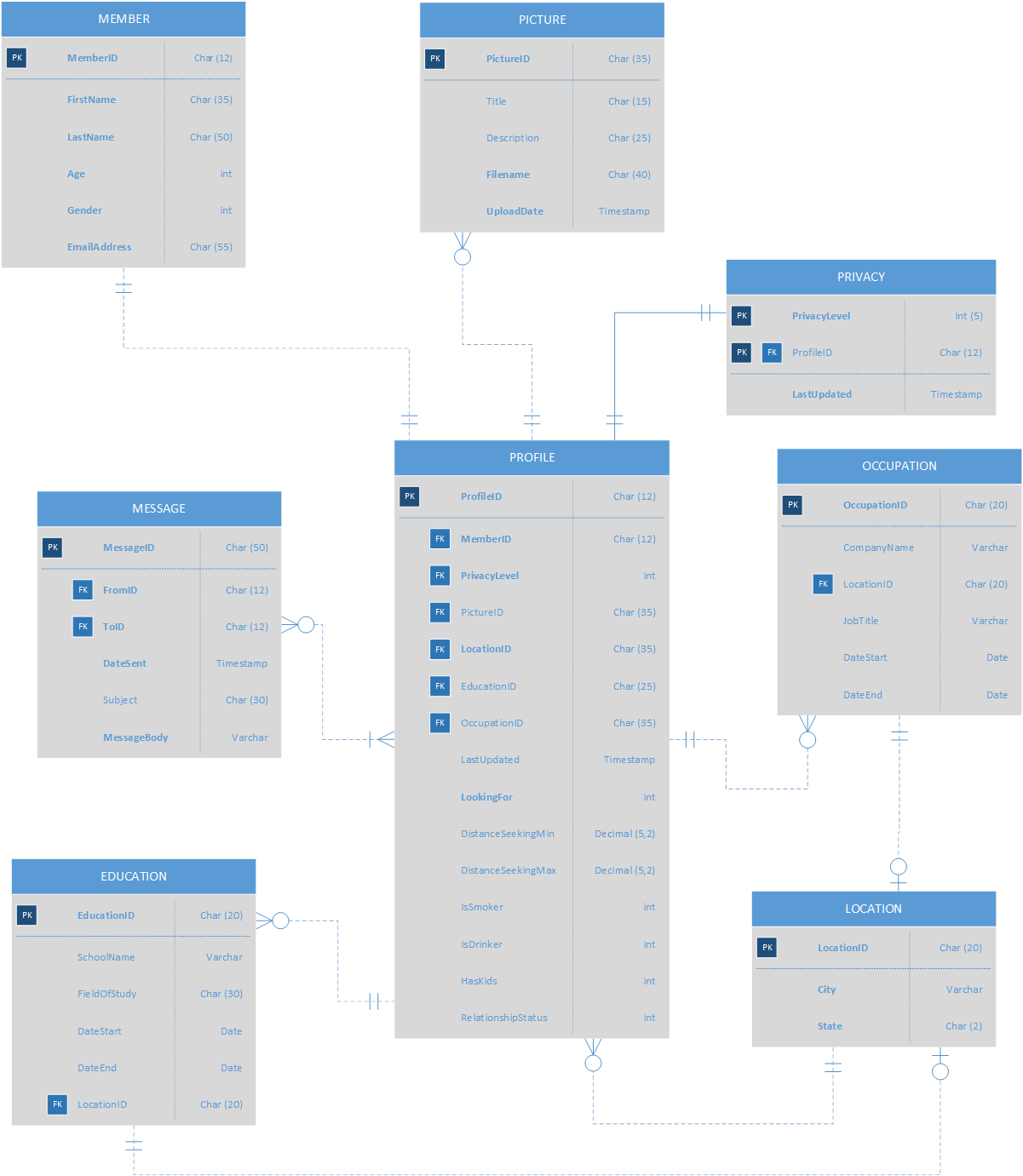
The usage of this database greatly improves the ability to match users based on their entries to their profiles. For instance, a user may execute a search query in the website (or a developer running a search query utilizing the DBMS) to find *single* users of the opposite sex, in an age range, possess some college education, are currently employed, don’t smoke, but drink on social occasions because of the way the database will be implemented. The website will have as many options and preferences as deemed necessary to ensure the matchmaking algorithms will be maximized to their fullest potential, resulting in a satisfactory and, hopefully, life-long, partner.

Database Model (E-R Diagram)



The ***Profile*** and ***Picture*** tables will have a zero-to-many relationship as each profile can have *none or many* pictures, but each picture can belong to *one and only one* profile. The ***Profile*** and ***Privacy*** tables will have a one-to-one relationship as each profile can only have one privacy setting and *must* have privacy settings, and each privacy setting is unique to one and only one profile. The ***Profile*** and ***Occupation*** tables will have a zero-to-many relationship as each profile can have zero or more occupations, but each occupation can be associated with one and only one profile. The ***Profile*** and ***Location*** tables will share a one-to-many relationship as each profile must have one and only one location, and each location may be associated with zero or more profiles. The ***Profile*** and ***Education*** tables will share a one-to-many relationship due to each profile having zero or more educational levels, whereas each educational instance may be associated with one and only one profile. The ***Profile*** and ***Message*** tables will share an intersection table relationship because each profile may have sent zero or more messages, but each message belongs to at least one other profile, specifically at least 2 profiles. The ***Occupation*** and ***Education*** tables each share a zero-to-one relationship with the ***Location*** table due to each occupation or educational instance do not require a location to be associated with them but, if they do, may only have one location associated with that record, whereas each location may be associated with numerous occupations and educational instances.

Database Design (E-R Diagram)



Progressing from the database model to the design, data types for the tables’ fields have been specified to reasonably and efficiently accommodate, store, and process the data given. In the **Profile** table, ProfileID is to be used as the primary key, that acts as foreign keys in some of the other tables. The other entities with references to Profile’s primary key will provide additional information for the user without having to have an over encumbered profile table. Profile will also require information such as the first name, last name, age, gender, isSmoker, isDrinker, hasKids, RelationshipStatus, and DistanceSeeking Max and Min, along with the user’s email address and password; none of these fields may be left blank by the user, so that every user should have a complete profile. *LocationID* serves as a Foreign Key in Profile so that multiple users are allowed in a given city or state. The **Privacy** table will have PrivacyID as a primary key, PrivacyLevel as a field, and *ProfileID* as a foreign key to reference the Profile table. The **Location** table,and its primary key, LocationID, will also be used as a foreign key in several other tables to provide the geographical data requested, like City and State, while minimizing redundancy. The **Picture** table, utilizing PictureID as its primary key, stores basic information about uploaded photos, notably requiring at a minimum of a filename, a *ProfileID* foreign key to link it to, and an automatically generated timestamp for the upload. Each photo will also have which profile it was uploaded from saved to each record. The **Message** table will contain MessageID as its primary key, and each message is required to have a subject, message body, and timestamp. The **Message\_Profile** table serves as an intersection table between Message and Profile that saves unique combinations of MessageIDs and ProfileIDs. This allows each Message to be stored separately from its sender and recipient, so each message will be referenced twice separately in Message\_Profile, connecting to the associated *ProfileID* from which it originates from or is addressed to. The **Education** table, using EducationID as its primary key, will also use LocationID as a foreign key to populate the geographical location of the user-input educational institution they went to or are attending, and ProfileID as a foreign key to tie a specific user’s educational history to them. The **Occupation** table will used OccupationID as its primary key, with *ProfileID* as a foreign key to reference a specific person, and *LocationID* as a foreign key to more easily pull geographical information and allow users to have or have had jobs in states different from where they currently live.

The primary changes from the model to the design phase were the addition and specifications of foreign keys in the tables. No additional tables needed to be created during the transformation to the database design, however data types were specified in reasonable amounts as to minimize unnecessarily large storage of data values such as using constricted char- or int-types as opposed to all left as varchar. The use of bit for isSmoker, isDrinker, and hasKids allows us to designate them as simple, yes/no, true/false, fields.

Schema

**Profile** (ProfileID, FirstName, LastName, Age, Gender, EmailAddress, Password, DistanceSeekingMin, DistanceSeekingMax, IsSmoker, IsDrinker, HasKids, RelationshipStatus, *LocationID*)

FK\_LocationID 🡪 LOCATION

**Picture** (PictureID, Title, Description, Filename, UploadDate, *ProfileID*)

FK\_ProfileID 🡪 PROFILE

**Privacy** (PrivacyLevel, LastUpdated, *ProfileID*)

FK\_ProfileID 🡪 PROFILE

**Message** (MessageID, Subject, MessageBody, DateSent)

**Message\_Profile**(*MessageID, ProfileID*)

FK\_ProfileID 🡪 PROFILE

**Education** (EducationID, SchoolName, FieldOfStudy, DateStart, DateEnd, *LocationID, ProfileID*)

FK\_ProfileID 🡪 PROFILE

**Occupation** (OccupationID, CompanyName, *LocationID, ProfileID*, JobTitle, DateStart, DateEnd)

FK\_ProfileID 🡪 PROFILE

**Location** (LocationID, City, State)

SQL Implementation

Create Statements:

CREATE TABLE LOCATION(

LocationID int(24) NOT NULL AUTO\_INCREMENT,

    City varchar(20) NOT NULL,

    State char(2) NOT NULL,

    CONSTRAINT LOCATION\_PK PRIMARY KEY(LocationID)

    );

CREATE TABLE PROFILE(

ProfileID int(12) NOT NULL AUTO\_INCREMENT,

    FirstName varchar(20) NOT NULL,

    LastName varchar(30) NOT NULL,

    Age int(2) NOT NULL,

    Gender int(1) NOT NULL,

Email varchar(35) NOT NULL,

    Pass varchar(20) NOT NULL,

    LookingFor int(1) NOT NULL,

    IsSmoker bit NULL,

    IsDrinker bit NULL,

    HasKids bit NULL,

    RelationshipStatus int(1) NULL,

    LocationID int(24) NOT NULL,

    CONSTRAINT PROFILE\_PK PRIMARY KEY(ProfileID),

    CONSTRAINT PROF\_LOC\_FK FOREIGN KEY(LocationID)

REFERENCES LOCATION(LocationID)

        ON UPDATE CASCADE

    );

CREATE TABLE PICTURE(

PictureID int(24) NOT NULL AUTO\_INCREMENT,

    Description varchar(50) NULL,

    Filename varchar(15) NOT NULL,

    UploadDate date NOT NULL,

    ProfileID int(12) NOT NULL,

    CONSTRAINT PICTURE\_PK PRIMARY KEY(PictureID),

    CONSTRAINT PIC\_PROF\_FK FOREIGN KEY(ProfileID)

REFERENCES Profile(ProfileID)

        ON UPDATE CASCADE

        ON DELETE NO ACTION

    );

CREATE TABLE PRIVACY(

PrivacyLevel int(1) NOT NULL AUTO\_INCREMENT,

    ProfileID int(12) NOT NULL,

    LastUpdated date NOT NULL,

    CONSTRAINT PRIV\_PROF\_FK FOREIGN KEY(ProfileID)

REFERENCES PROFILE(ProfileID)

ON UPDATE CASCADE

        ON DELETE NO ACTION,

    CONSTRAINT PRIVACY\_PK PRIMARY KEY(PrivacyLevel, ProfileID)

    );

CREATE TABLE OCCUPATION(

OccupationID int(20) NOT NULL AUTO\_INCREMENT,

    CompanyName varchar(30) NULL,

    LocationID int(20) NULL,

    JobTitle varchar(20) NOT NULL,

    StartDate date NULL,

    EndDate date NULL,

    ProfileID int(12) NOT NULL,

    CONSTRAINT OCCUPATION\_PK PRIMARY KEY(OccupationID),

    CONSTRAINT OCC\_LOCATION\_FK FOREIGN KEY(LocationID)

REFERENCES LOCATION(LocationID)

        ON UPDATE CASCADE,

CONSTRAINT OCC\_PROFILE\_FK FOREIGN KEY(ProfileID)

REFERENCES PROFILE(ProfileID)

ON UPDATE CASCADE

    );

CREATE TABLE EDUCATION(

EducationID int(24) NOT NULL AUTO\_INCREMENT,

    SchoolName varchar(20) NOT NULL,

    Field char(20) NULL,

    GradDate date NULL,

    LocationID int(20) NOT NULL,

    ProfileID int(12) NOT NULL,

    CONSTRAINT EDUCATION\_PK PRIMARY KEY(EducationID),

    CONSTRAINT EDU\_LOCATION\_FK FOREIGN KEY(LocationID)

REFERENCES LOCATION(LocationID)

ON UPDATE CASCADE,

CONSTRAINT EDU\_PROFILE\_FK FOREIGN KEY(ProfileID)

REFERENCES PROFILE(ProfileID)

        ON UPDATE CASCADE

    );

CREATE TABLE MESSAGE(

MessageID int(24) NOT NULL AUTO\_INCREMENT,

Subject char(20) NULL,

    MessageBody varchar(255) NOT NULL,

    DateSent date NOT NULL,

    CONSTRAINT MESSAGE\_PK PRIMARY KEY(MessageID)

    );

CREATE TABLE MESSAGE\_PROFILE(

MessageID int(24) NOT NULL,

    ProfileID int(12) NOT NULL,

    CONSTRAINT MESSAGE\_PROFILE\_PK PRIMARY KEY(MessageID, ProfileID),

    CONSTRAINT MP\_MESSAGE\_FK FOREIGN KEY(MessageID)

REFERENCES MESSAGE(MessageID)

        ON UPDATE CASCADE,

CONSTRAINT MP\_PROFILE\_FK FOREIGN KEY(ProfileID)

REFERENCES PROFILE(ProfileID)

        ON UPDATE CASCADE

    );

Insert Statements:

INSERT INTO LOCATION(City, State) VALUES('Saint Louis', 'MO');

INSERT INTO LOCATION(City, State) VALUES('Atlanta', 'GA');

INSERT INTO LOCATION(City, State) VALUES('San Diego', 'CA');

INSERT INTO LOCATION(City, State) VALUES('Miami', 'FL');

INSERT INTO LOCATION(City, State) VALUES('Jacksonville', 'FL');

INSERT INTO LOCATION(City, State) VALUES('Roswell', 'GA');

INSERT INTO LOCATION(City, State) VALUES('Duluth', 'GA');

INSERT INTO LOCATION(City, State) VALUES('Lawrenceville', 'GA');

INSERT INTO LOCATION(City, State) VALUES('Nashville', 'TN');

INSERT INTO LOCATION(City, State) VALUES('Memphis', 'TN');

INSERT INTO Message (MessageID, Subject, MessageBody, DateSent) VALUES( ‘1’, ‘Hey there cutie!’, ‘Hi, I thought your eyes were simply gorgeous and had to say hello! :3’, ‘7/15/2018’);

INSERT INTO Message VALUES( ‘2’, ‘Your Smile Tho’, ‘Hey man, your smile is so fun and cute!’, ‘7/15/2018’);

INSERT INTO Message VALUES( ‘3’, ‘Coffee?’, ‘Thanks! I’d love to get coffee with you sometime!’, ‘7/16/2018’);

INSERT INTO Message VALUES( ‘4’, ‘Mad Respect’, ‘I see you’re a Harvard graduate AND you can play guitar!? Marry me, please!!!! ROFL’, ‘7/18/2018’);

INSERT INTO Message VALUES( ‘5’, ‘Can I pet your dog?’, ‘Your corgi is SO CUTE I’M DYING. What’s his name? ’, ‘7/18/2018’);

INSERT INTO PROFILE(ProfileID, FirstName, LastName, Age, Gender, Email, Pass, LookingFor, IsSmoker, IsDrinker, HasKids, RelationshipStatus, LocationID)

VALUES(1, 'John', 'Smith', 28, 1, 'john.smith@website.org', 'Password1234', 0, 0, 0, 1, 1, 2);

INSERT INTO PROFILE(ProfileID, FirstName, LastName, Age, Gender, Email, Pass, LookingFor, IsSmoker, IsDrinker, HasKids, RelationshipStatus, LocationID)

VALUES(2, 'Jerry', 'Seinfeld', 47, 1, 'J.Seinfeld@nbc.com', '238ZSDknjf', 0, 0, 0, 0, 0, 5);

INSERT INTO PROFILE(ProfileID, FirstName, LastName, Age, Gender, Email, Pass, LookingFor, IsSmoker, IsDrinker, HasKids, RelationshipStatus, LocationID)

VALUES(3, 'Elaine', 'Bennis', 45, 0, 'E.Bennis@nbc.com', 'sefnSDC234', 1, 0, 1, 1, 0, 6);

INSERT INTO PROFILE(ProfileID, FirstName, LastName, Age, Gender, Email, Pass, LookingFor, IsSmoker, IsDrinker, HasKids, RelationshipStatus, LocationID)

VALUES(4, 'George', 'Costanza', 48, 1, 'Festivus4Restuvus@gmail.com', 'SEFsvnd9322', 0, 1, 1, 0, 0, 8);

INSERT INTO PROFILE(ProfileID, FirstName, LastName, Age, Gender, Email, Pass, LookingFor, IsSmoker, IsDrinker, HasKids, RelationshipStatus, LocationID)

VALUES(5, 'Cosmo', 'Kramer', 39, 1, 'AnywhereUSA@nbc.com', 'ASCnksdflf', 0, 0, 1, 1, 0, 1);#5

INSERT INTO PROFILE(ProfileID, FirstName, LastName, Age, Gender, Email, Pass, LookingFor, IsSmoker, IsDrinker, HasKids, RelationshipStatus, LocationID)

VALUES(6, 'Terry', 'Crews', 21, 1, 'WastedMilk@somewhere.com', 'ERf20ds9mvd', 0, 1, 0, 1, 1, 7);

INSERT INTO PROFILE(ProfileID, FirstName, LastName, Age, Gender, Email, Pass, LookingFor, IsSmoker, IsDrinker, HasKids, RelationshipStatus, LocationID)

VALUES(7, 'Etta', 'James', 28, 0, 'oldsoul@gmail.com', 'ASD3w@#3f', 1, 0, 0, 0, 0, 4);

INSERT INTO PROFILE(ProfileID, FirstName, LastName, Age, Gender, Email, Pass, LookingFor, IsSmoker, IsDrinker, HasKids, RelationshipStatus, LocationID)

VALUES(8, 'Gabrielle', 'Union', 23, 0, 'GabUnion@gmail.com', 'AEF@$vmsdvs', 1, 1, 1, 0, 0, 8);

INSERT INTO PROFILE(ProfileID, FirstName, LastName, Age, Gender, Email, Pass, LookingFor, IsSmoker, IsDrinker, HasKids, RelationshipStatus, LocationID)

VALUES(9, 'George', 'Benson', 33, 1, 'jazzytracks@record.com', '%sdmgfdr!', 0, 0, 0, 0, 1, 4);

INSERT INTO PROFILE(ProfileID, FirstName, LastName, Age, Gender, Email, Pass, LookingFor, IsSmoker, IsDrinker, HasKids, RelationshipStatus, LocationID)

VALUES(10, 'Anita', 'James', 35, 0, 'IGotThatSoul@gmail.com', '@43fmxldwep', 1, 1, 1, 0, 0, 2);

INSERT INTO PICTURE VALUES(1, 'Funny cat', 'funnycat.jpg', 20170710, 2);

INSERT INTO PICTURE VALUES(2, 'hithere', 'hithere.jpg', 20170402, 3);

INSERT INTO PICTURE VALUES(3, 'groupphoto', 'group\_photo.jpg', 20160302, 3);

INSERT INTO PICTURE VALUES(4, 'school', 'school.jpg', 20150102, 3);

INSERT INTO PICTURE VALUES(5, 'car', 'car.jpg', 20170102, 3);

INSERT INTO PICTURE VALUES(6, 'sunset', 'sunset.jpg', 20180502, 3);

INSERT INTO PICTURE VALUES(7, 'selfie', 'selfie.jpg', 20180502, 3);

INSERT INTO PICTURE VALUES(8, 'test3', 'test3.png', 20180327, 5);

INSERT INTO PICTURE VALUES(9, 'obama meme', 'obamameme.gif', 20180514, 3);

INSERT INTO PICTURE VALUES(10, 'Funny dog', 'funnydog.jpg', 20180705, 7);

INSERT INTO PRIVACY VALUES(1, 1, 20170111);

INSERT INTO PRIVACY VALUES(0, 2, 20180206);

INSERT INTO PRIVACY VALUES(3, 3, 20180302);

INSERT INTO PRIVACY VALUES(3, 4, 20170506);

INSERT INTO PRIVACY VALUES(3, 5, 20180916);

INSERT INTO PRIVACY VALUES(2, 6, 20171206);

INSERT INTO PRIVACY VALUES(2, 7, 20180506);

INSERT INTO PRIVACY VALUES(2, 8, 20170501);

INSERT INTO PRIVACY VALUES(1, 9, 20150908);

INSERT INTO PRIVACY VALUES(1, 10, 20140518);

INSERT INTO OCCUPATION(OccupationID, CompanyName, LocationID, JobTitle, ProfileID)

VALUES(1, 'Google', 1, 'Project Manager', 1);

INSERT INTO OCCUPATION(OccupationID, CompanyName, LocationID, JobTitle, ProfileID)

VALUES(2, 'Google', 3, 'Project Manager', 2);

INSERT INTO OCCUPATION(OccupationID, CompanyName, LocationID, JobTitle, ProfileID)

VALUES(3, 'HP', 6, 'Regional Manager', 3);

INSERT INTO OCCUPATION(OccupationID, CompanyName, LocationID, JobTitle, ProfileID)

VALUES(4, 'Self-Employed', 7, 'Project Manager', 4);

INSERT INTO OCCUPATION(OccupationID, CompanyName, LocationID, JobTitle, ProfileID)

VALUES(5, 'Dell', 10, 'Department Manager', 5);

INSERT INTO OCCUPATION(OccupationID, CompanyName, LocationID, JobTitle, ProfileID)

VALUES(6, 'Alibaba', 4, 'Project Manager', 6);

INSERT INTO OCCUPATION(OccupationID, CompanyName, LocationID, JobTitle, ProfileID)

VALUES(7, 'Amazon', 7, 'Database Manager', 7);

INSERT INTO OCCUPATION(OccupationID, CompanyName, LocationID, JobTitle, ProfileID)

VALUES(8, 'Comcast', 7, 'Shift Manager', 8);

INSERT INTO OCCUPATION(OccupationID, CompanyName, LocationID, JobTitle, ProfileID)

VALUES(9, 'GGC', 1, 'Office Assistant', 9);

INSERT INTO OCCUPATION(OccupationID, CompanyName, LocationID, JobTitle, ProfileID)

VALUES(10, 'GGC', 1, 'Professor', 10);

INSERT INTO EDUCATION VALUES(1, 'UKC', 'Law', NULL, 5, 8);

INSERT INTO EDUCATION VALUES(2, 'GGC', 'Info Systems', NULL, 3, 2);

INSERT INTO EDUCATION VALUES(3, 'Mizzou', 'Journalism', 20150530, 6, 1);

INSERT INTO EDUCATION VALUES(4, 'UNC', NULL, NULL, 8, 3);

INSERT INTO EDUCATION VALUES(5, 'FSU', 'Music Production', NULL, 4, 8);

INSERT INTO EDUCATION VALUES(6, 'GSU', NULL, 20180704, 7, 5);

INSERT INTO EDUCATION VALUES(7, 'UCLA', NULL, 20170504, 8, 7);

INSERT INTO Message VALUES( 1, 'Hey there cutie!', 'Hi, I thought your eyes were simply gorgeous and had to say hello!' , 20180715);

INSERT INTO Message VALUES( 2, 'Your Smile Tho', 'Hey man, your smile is so fun and cute!', 20180710);

INSERT INTO Message VALUES( 3, 'Coffee?', 'Thanks! I’d love to get coffee with you sometime!', 20180716);

INSERT INTO Message VALUES( 4, 'Mad Respect', 'I see you’re a Harvard graduate AND you can play guitar!? Marry me, please!!!! ROFL', 20180710);

INSERT INTO Message VALUES( 5, 'Can I pet your dog?', 'Your corgi is SO CUTE I’M DYING. What’s his name? ', 20180712);

// Remember the 1st number is the MessageID and the 2nd is one of the profiles the message is attached to, i.e. the sender or recipient

INSERT INTO Message\_Profile Values (1,1);

INSERT INTO Message\_Profile Values (1,2);

INSERT INTO Message\_Profile Values (2,3);

INSERT INTO Message\_Profile Values (2,7);

INSERT INTO Message\_Profile Values (3,1);

INSERT INTO Message\_Profile Values (3,2);

INSERT INTO Message\_Profile Values (4,4);

INSERT INTO Message\_Profile Values (4,6);

INSERT INTO Message\_Profile Values (5,8);

INSERT INTO Message\_Profile Values (5,9);

Database Design (E-R Diagram)

Table and Query Implementation (Deliverable 3)

### SELECT \* FROM Profile

WHERE RelationshipStatus = 1 AND HasKids = 0;

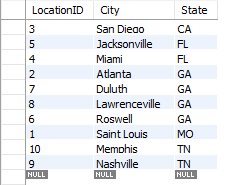
* Displays all profiles where the person is single and has no kids.



### SELECT \* FROM Location

ORDER BY State ASC THEN City ASC;

* Displays all locations sorted by State and then City, both in ascending order



### SELECT AVG(Age) AS AverageUserAge FROM Profile;

* Shows the average age of all members, displayed as AverageUserAge



### SELECT \* FROM Profile

WHERE ProfileID IN

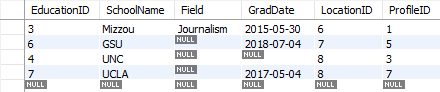
(SELECT ProfileID FROM Education

WHERE LocationID IN

(SELECT LocatonID FROM Location

WHERE State = ‘GA’));

* Shows all profiles/people that have gone to school in Georgia, even if the people no longer live in Georgia (which is why we access Location table thru the Education table)



### SELECT MessageID, Subject, MessageBody, ProfileID

### FROM Message

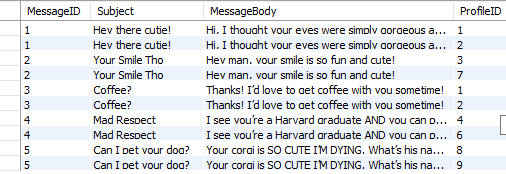
### JOIN Message\_Profile ON Message.MessageID = Message\_Profile.MessageID

### JOIN Profile ON Message\_Profile.ProfileID = Profile.ProfileID

### JOIN Privacy ON Profile.ProfileID = Privacy.ProfileID

### WHERE Privacy.PrivacyLevel < 3;

* Shows all messages and the users they are sent between where the users’ privacy levels are set to less than 3, the max, or “invisible” privacy.



### SELECT COUNT(\*) FROM Profile

### WHERE IsDrinker = 0 AND IsSmoker = 1;

* Shows the number of users that smoke but do not drink



### SELECT \* FROM Profile

GROUP BY LocationID;

* Returns all users grouped by their locationID

