University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: July 14th, 2024

Group Number: 31

Name	Student Number	CS Alias (Userid)	Preferred Email Address
Leyang Pan	93460962	yang0526	yangplypan@gmail.com
Binjie Ye	29415965	yebinjie	yebinjie2021@163.com
Cheryl Chen	44983922	cchen70	chen.q.cheryl@gmail.com

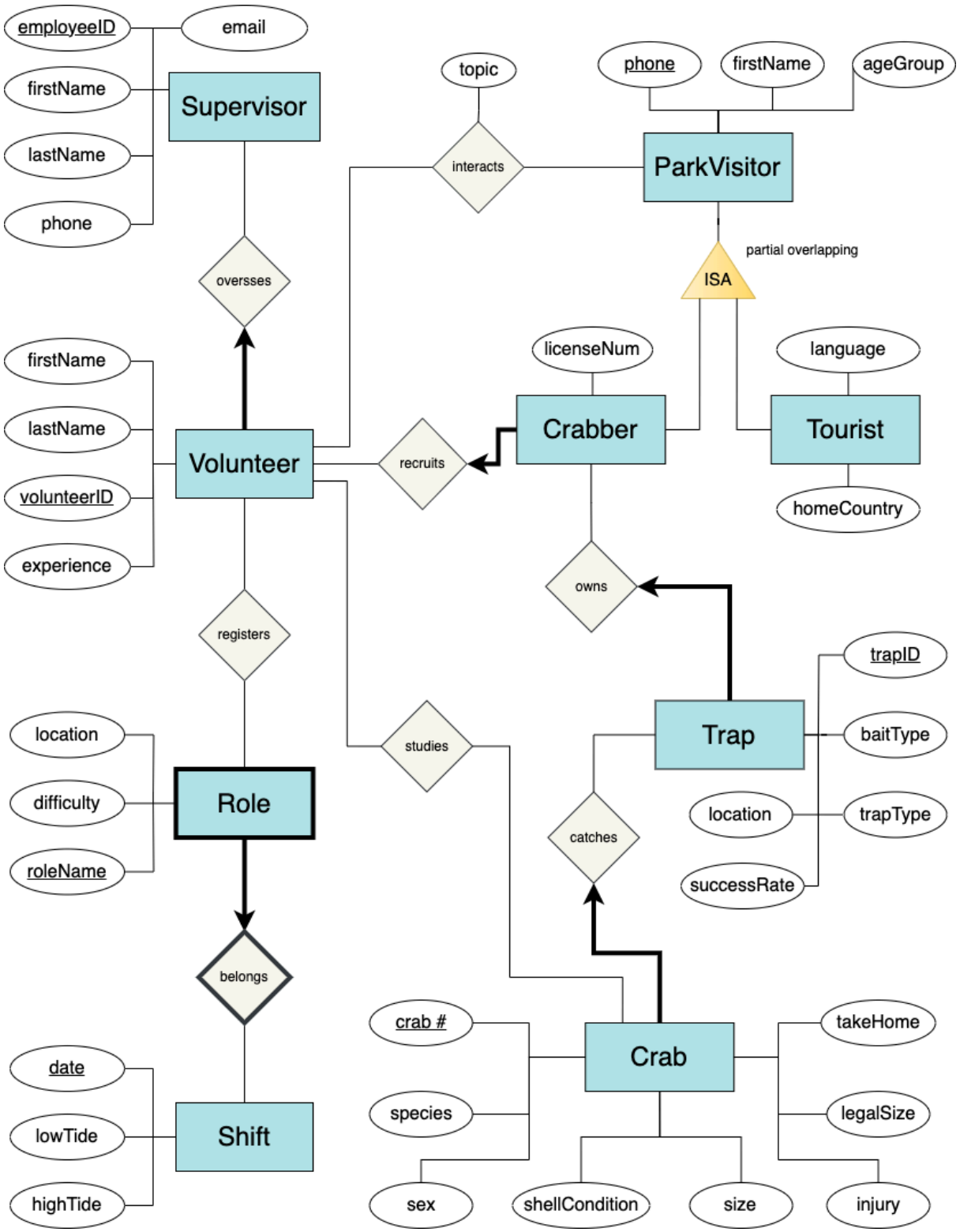
By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your email address, 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 Brief Summary:

This project models the Belcarra Beachkeepers summer program, in which volunteers partake in a research study on the crabs at the park, as well as educate park visitors about conservation. Entities include volunteers, supervisors, park visitors and crabbers, their crabs and traps, and logistics like roles and shifts. This project is intended to help staff schedule shifts and manage scientific data about the crabs, as well as insights into their public interactions.

3 ER Diagram:



Notes: 1) roleName is a partial key and should have a dotted underline (just wasn't possible on our editor), 2) From Milestone 1, we got rid of our ternary relationship and made Role a weak entity of Shift, so that Volunteer registers for a Role directly because it modeled the domain better. We added another subclass to

our IsA relationship and per our mentor's suggestion, changed the ParkVisitor primary key to be something actually unique, their phone number. We also incorporated the feedback that licenseNum should not be marked as a key because that would be improper IsA notation, and now just have it as a regular attribute with the UNIQUE constraint. 3) Some attributes have been tweaked just to be more clear and to represent the domain better.

---

4 Schema:

Supervisor(employeeID: INTEGER, firstName: VARCHAR, lastName: VARCHAR, phone: VARCHAR, email: VARCHAR)

- Primary key = employeeID

Volunteer(volunteerID: INTEGER, firstName: VARCHAR, lastName: VARCHAR, experience: INTEGER, **employeeID**: INTEGER)

- Primary key = volunteerID, employeeID is a FK to Supervisor and is NOT NULL

Role(roleName: VARCHAR, **date**: date, difficulty: INTEGER, location: VARCHAR)

- Role is a weak entity that relies on Shift, with partial key = roleName, and primary key = (roleName, date), where date is a FK to Shift and is NOT NULL

Registers(**volunteerID**: INTEGER, **roleName**: VARCHAR, **date**: date)

- Registers is a separate table because it represents many-to-many, primary key = (volunteerID, roleName, date), volunteerID is a FK to Volunteer, roleName and date are FKs to Role (two attributes because Role is a weak entity with a partial key)

Shift(date: date, lowTide: time, highTide: time)

- Primary key: date

ParkVisitor(phone: VARCHAR, firstName: VARCHAR, ageGroup: VARCHAR)

- Primary key: phone

Crabber(**phone**: VARCHAR, licenseNum: INTEGER, **volunteerID**: INTEGER)

- Primary key = phone (also a FK as this is superclass's PK), volunteerID is a FK to Volunteer and is NOT NULL, licenseNum must be UNIQUE

Tourist(**phone**: VARCHAR, language: VARCHAR, homeCountry: VARCHAR)

- Primary key = phone (also a FK as this is superclass's PK)

Interacts(**volunteerID**: INTEGER, **phone**: VARCHAR, topic: VARCHAR)

- Interacts is a separate table because it represents many-to-many, primary key = (volunteerID, phone), volunteerID is a FK to Volunteer, phone is a FK to ParkVisitor

---

Trap(<u>trapID</u>: INTEGER, baitType: VARCHAR, trapType: VARCHAR, location: VARCHAR, successRate: DOUBLE, **phone**: VARCHAR)

- Primary key = trapID, phone is a FK to Crabber and is NOT NULL

Crab(<u>crab#</u>: INTEGER, species: VARCHAR, sex: CHAR(1), shellCondition: VARCHAR, size: INTEGER, injury: CHAR(1), legalSize: INTEGER, takeHome: CHAR(1), **trapID**: INTEGER)

- Primary key = crab #, trapID is a FK to Trap and is NOT NULL

Studies(<u>**volunteerID**</u>: INTEGER, <u>**crab#**</u>: INTEGER)

- Studies is a separate table because it represents many-to-many, primary key = (volunteerID, crab #), volunteerID is a FK to Volunteer, crab # is a FK to Crab

5 Functional Dependencies (FDs):

Supervisor(employeeID, firstName, lastName, phone, email)
- employeeID → firstName, lastName, phone, email
- firstName, lastName → email

Volunteer(volunteerID, firstName, lastName, experience, **employeeID**)
- volunteerID → firstName, lastName, experience, employeeID

Role(roleName, **date**, difficulty, location)
- roleName, date → difficulty, location

Registers(**volunteerID**, **roleName**, **date**)
- Only trivial dependencies as all attributes are keys

Shift(date, lowTide, highTide)
- date → lowTide, highTide

ParkVisitor(phone, firstName, ageGroup)
- phone → firstName, ageGroup

Crabber(**phone**, licenseNum, **volunteerID**)
- phone → licenseNum, volunteerID

Tourist(**phone**, language, homeCountry)
- phone → language, homeCountry

Interacts(**volunteerID**, **phone**, topic)
- volunteerID, phone → topic

Trap(trapID, baitType, trapType, location, successRate, **phone**)
- trapID → baitType, trapType, location, phone
- baitType → successRate
- location → successRate

Crab(<u>crab#</u>, species, sex, shellCondition, size, injury, legalSize, takeHome, **trapID**)

- crab# → species, sex, shellCondition, size, injury, legalSize, takeHome, trapID
- species → legalSize
- species, sex → takeHome
- injury → shellCondition

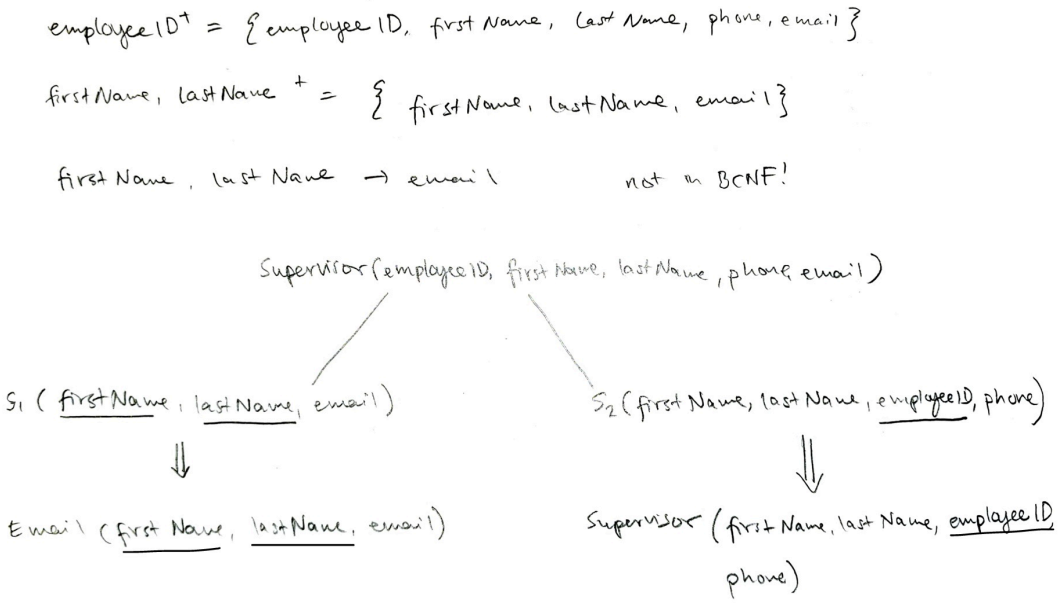
Studies(**<u>volunteerID</u>**, **<u>crab#</u>**)

- Only trivial dependencies as all attributes are keys

---

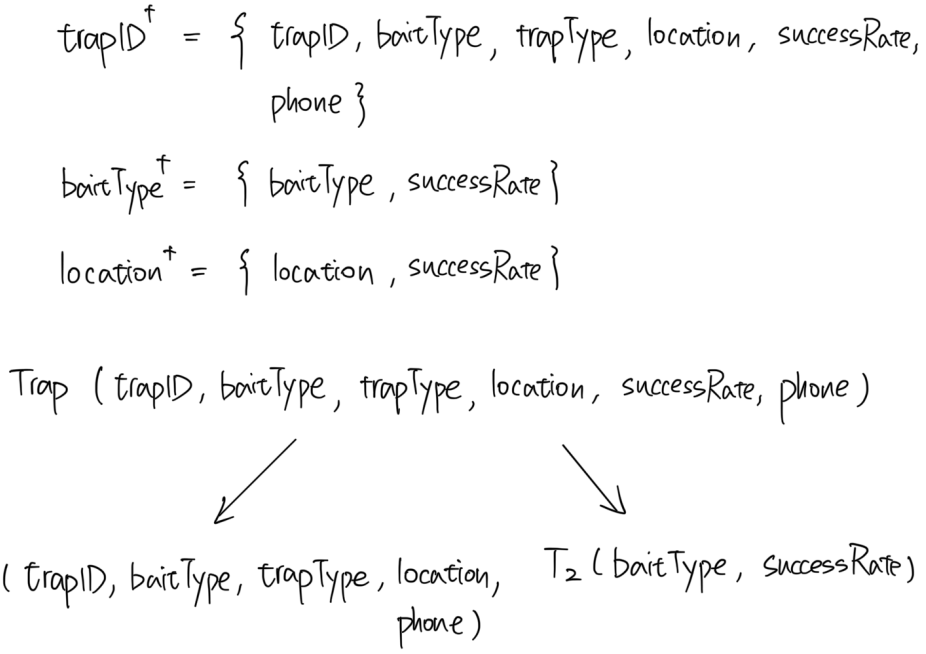
6 Normalization:

Supervisor(employeeID, firstName, lastName, phone, email)



- Email(firstName, lastName, email)
- Supervisor(employeeID, firstName, lastName, phone)

Trap(trapID, baitType, trapType, location, successRate, **phone**)



- Trap(trapID, baitType, trapType, location, **phone**)
- Bait(baitType, successRate)

University of British Columbia, Vancouver
Department of Computer Science

Crab(crab#, species, sex, shellCondition, size, injury, legalSize, takeHome, **trapID**)

For Crab, let's denote Crab(A, B, C, D, E, F, G, H, I) for convenience.

So we have FDs:

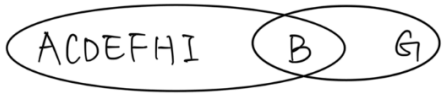
$A \to B, C, D, E, F, G, H, I$

$B \to G$

$B, C \to H$

$F \to D$

Decompose $B \to G$:



$C_1(A, B, C, D, EF, H, I)$ $C_2(B, G)$

Decompose $BC \to H$

$C_3(A, B, C, D, E, F, I)$ $C_4(B, C, H)$

Decompose $F \to D$

$C_5(A, B, C, E, F, I)$ $C_6(F, D)$

Therefore, the result is:

C_2(species, legalSize)

C_4(species, sex, takeHome)

C_5(crab#, species, sex, size, injury, trapID)

C_6(injury, shellCondition)

- Species(species, legalSize)
- Legal(species, sex, takeHome)
- Crab(crab#, species, sex, size, injury, **trapID**)

7 SQL DDL Statement:

```
CREATE TABLE Supervisor(
        employeeID          INTEGER,
        firstName           VARCHAR,
        lastName            VARCHAR,
        phone               VARCHAR,
        PRIMARY KEY (employeeID),
        UNIQUE (firstName, lastName)
);


CREATE TABLE Email(
        firstName           VARCHAR         NOT NULL,
        lastName            VARCHAR         NOT NULL,
        email               VARCHAR,
        PRIMARY KEY (firstName, lastName),
        FOREIGN KEY (firstName, lastName) REFERENCES Supervisor(firstName, lastName)
                ON DELETE CASCADE
);


CREATE TABLE Volunteer(
        volunteerID         INTEGER,
        firstName           VARCHAR,
        lastName            VARCHAR,
        experience          INTEGER,
        employeeID          INTEGER         NOT NULL,
        PRIMARY KEY (volunteerID),
        FOREIGN KEY (employeeID) REFERENCES Supervisor(employeeID)
                ON DELETE CASCADE
);


CREATE TABLE Crabber(
        phone               VARCHAR,
        licenseNum          INTEGER         UNIQUE,
        volunteerID         INTEGER         NOT NULL,
        FOREIGN KEY (volunteerID) REFERENCES Volunteer(volunteerID)
                ON DELETE CASCADE
);
```

```
CREATE TABLE Trap(
        trapID          INTEGER,
        baitType        VARCHAR,
        trapType        VARCHAR,
        location        VARCHAR,
        phone           VARCHAR         NOT NULL,
        PRIMARY KEY (trapID),
        FOREIGN KEY (phone) REFERENCES Crabber(phone)
                ON DELETE CASCADE
);

CREATE TABLE Bait(
        baitType        VARCHAR,
        successRate     DOUBLE,
        PRIMARY KEY (baitType)
);

CREATE TABLE Species(
        species         VARCHAR,
        legalSize       INTEGER,
        PRIMARY KEY (species)
);

CREATE TABLE Legal(
        species         VARCHAR,
        sex             CHAR(1),
        takeHome        CHAR(1),
        PRIMARY KEY (species, sex)
);

CREATE TABLE Crab(
        crabID          INTEGER,
        species         VARCHAR,
        sex             CHAR(1),
        size            INTEGER,
        injury          CHAR(1),
        trapID          INTEGER         NOT NULL,
        PRIMARY KEY (trapID),
        FOREIGN KEY (trapID) REFERENCES Trap(trapID)
                ON DELETE CASCADE
);
```

```
CREATE TABLE Shift(
        date                DATE,
        lowTide             TIME,
        highTide            TIME,
        PRIMARY KEY (date)
);

CREATE TABLE Role(
        roleName            VARCHAR,
        date                DATE                NOT NULL,
        difficulty          INTEGER,
        location            VARCHAR,
        PRIMARY KEY (roleName, date),
        FOREIGN KEY (date) REFERENCES Shift (date)
);

CREATE TABLE Registers(
        volunteerID         INTEGER,
        roleName            VARCHAR,
        date                DATE,
        PRIMARY KEY (volunteerID, roleName, date),
        FOREIGN KEY (volunteerID) REFERENCES Volunteer(volunteerID),
        FOREIGN KEY (roleName, date) REFERENCES Role(roleName, date)
);

CREATE TABLE ParkVisitor(
        phone               VARCHAR,
        firstName           VARCHAR,
        ageGroup            VARCHAR,
        PRIMARY KEY (phone)
);

CREATE TABLE Tourist(
        phone               VARCHAR,
        language            VARCHAR,
        homeCountry         VARCHAR,
        PRIMARY KEY (phone),
        FOREIGN KEY (phone) REFERENCES ParkVisitor(phone)
);
```

```
CREATE TABLE Interacts(
        volunteerID             INTEGER,
        phone                   VARCHAR,
        topic                   VARCHAR,
        PRIMARY KEY (volunteerID, phone),
        FOREIGN KEY (volunteerID) REFERENCES Volunteer(volunteerID),
        FOREIGN KEY (phone) REFERENCES ParkVisitor(phone)
);

CREATE TABLE Studies(
        volunteerID             INTEGER,
        crabID                  INTEGER,
        PRIMARY KEY (volunteerID, crabID),
        FOREIGN KEY (volunteerID) REFERENCES Volunteer(volunteerID),
        FOREIGN KEY (crabID) REFERENCES Crab(crabID)
);
```

---

8 INSERT statements:

Supervisor(employeeID, firstName, lastName, phone)

```
INSERT
INTO Supervisor (employeeID, firstName, lastName, phone)
VALUES
        (1, 'April', 'Ludgate', '778-945-1049'),
        (2, 'Keely', 'Langford', '604-802-3581'),
        (3, 'Chris', 'Mahoney', '778-381-4112'),
        (4, 'Danielle', 'Clark', '604-202-0733'),
        (5, 'Rachel', 'Green', '778-515-8466');
```

Email(firstName, lastName, email)

```
INSERT
INTO Email (firstName, lastName, email)
VALUES
        ('April', 'Ludgate', 'april.ludgate@gmail.com'),
        ('Keely', 'Langford', keely.langford@gmail.com),
        ('Chris', 'Mahoney', chris.mahoney@gmail.com),
        ('Danielle', 'Clark', danielle.clark@gmail.com),
        ('Rachel', 'Green', 'rachel.green@gmail.com');
```

Volunteer(volunteerID, firstName, lastName, experience, **employeeID**)

```
INSERT
INTO Volunteer (volunteerID, firstName, lastName, experience, employeeID)
VALUES
        (100, 'Elaine', 'Lu', 4, 1),
        (101, 'Liza', 'Kuz', 0, 5),
        (102, 'Cheryl', 'Cheng', 1, 5),
        (103, 'Calum', 'Lederat', 0, 2),
        (104, 'Erik', 'Dunce', 6, 3);
```

Role(roleName, **date**, difficulty, location)

```
INSERT
INTO Role (roleName, date, difficulty, location)
VALUES
        ('Measurer', '2023-06-24', 3, 'Wharf'),
        ('Recorder', '2023-07-09', 5, 'Wharf'),
        ('Handler', '2024-06-30', 4, 'Wharf'),
        ('Info Tent', '2024-07-01', 1, 'Picnic Area'),
        ('Measurer', '2024-07-13', 3, 'Wharf');
```

Registers(**volunteerID**, **roleName**, **date**)

```
INSERT
INTO Registers (volunteerID, roleName, date)
VALUES
        (104, 'Measurer', '2023-06-24'),
```

(con't on next page)

```
        (103, 'Measurer', '2023-06-24'),
        (100, 'Handler', '2024-06-30'),
        (102, 'Info Tent', '2024-07-01'),
        (101, 'Measurer', '2024-07-13');
```

Shift(date, lowTide, highTide)

```
    INSERT
    INTO Shift (date, lowTide, highTide)
    VALUES
        ('2023-06-24', '10:34:23', '21:59:00'),
        ('2023-07-09', '10:39:48', '22:11:08'),
        ('2024-06-30', '10:52:28', '23:18:47'),
        ('2024-07-01', '11:44:53', '24:42:12'),
        ('2024-07-13', '12:26:27', '00:33:06');
```

ParkVisitor(phone, firstName, ageGroup)

```
    INSERT
    INTO ParkVisitor (phone, firstName, ageGroup)
    VALUES
        ('778-284-9134', 'Maxim', 'Youth'),
        ('604-038-1435', 'Layla', 'Adult),
        ('604-678-4319', 'Raymond', 'Adult'),
        ('224-649-9683', 'Alan', 'Youth'),
        ('316-274-3485', 'Edouard', 'Adult'),
        ('856-462-3021', 'Ivan', 'Youth'),
        ('982-017-0112', 'Celeste', 'Adult'),
        ('245-924-5203', 'Elisa', 'Adult');
```

Crabber(**phone**, licenseNum, **volunteerID**)

```
    INSERT
    INTO Crabber (phone, licenseNum, volunteerID)
    VALUES
        ('778-284-9134', '825492638', 101),
        ('604-038-1435', '740274927', 102),
        ('604-678-4319', '312436442', 104),
        ('316-274-3485', '628365816', 102),
        ('245-924-5203', '028947552', 102);
```

Tourist(**phone**, language, homeCountry)

```
    INSERT
    INTO Tourist (phone, language, homeCountry)
    VALUES
        ('224-649-9683', 'English', 'Canada'),
        ('316-274-3485', 'French', 'Canada'),
        ('856-462-3021', 'Spanish', 'Chile'),
        ('982-017-0112', 'Mandarin', 'China),
        ('245-924-5203', 'Korea', 'South Korea');
```

Interacts(**volunteerID**, **phone**, topic)

```
INSERT
INTO Interacts (volunteerID, phone, topic)
VALUES
    (104, '224-649-9683', 'Starfish'),
    (101, '316-274-3485', 'Jellyfish Bloom'),
    (100, '604-678-4319', 'Sexing Crabs'),
    (100, '604-038-1435', 'Invasive Crabs'),
    (103, '316-274-3485', 'Smoking Area');
```

Trap(trapID, baitType, trapType, location, **phone**)

```
INSERT
INTO Trap (trapID, baitType, trapType, location, phone)
VALUES
    (1, 'Chicken', 'Clamshell', 'East', '778-284-9134'),
    (54, 'Chicken', 'Clamshell', 'Main', '604-038-1435'),
    (19, 'Turkey', 'Clamshell', 'North', '604-678-4319'),
    (72, 'Chicken', 'Box', 'South', '316-274-3485'),
    (3, 'Duck', 'Clamshell', 'South', '245-924-5203');
```

Bait(baitType, successRate)

```
INSERT
INTO Bait (baitType, successRate)
VALUES
    ('Chicken', 0.78),
    ('Beef', 0.14),
    ('Pork', 0.56),
    ('Duck', 0.63),
    ('Turkey', 0.92);
```

Species(species, legalSize)

```
INSERT
INTO Species (species, legalSize)
VALUES
    ('Dungeness', 165),
    ('Red Rock', 115),
    ('Graceful', NULL),
    ('Green', 1),
    ('Shore', NULL);
```

Legal(species, sex, takeHome)

```
INSERT
INTO Legal (species, sex, takeHome)
VALUES
    ('Dungeness', 'M', 'Y'),
    ('Graceful', 'M', 'N'),
    ('Dungeness', 'F', 'N'),
    ('Shore', 'M', 'N'),
    ('Red Rock', 'F', 'N');
```

Crab(<u>crab#</u>, species, sex, size, injury, **trapID**)

```
INSERT
INTO Crab (crab#, species, sex, size, injury, trapID)
VALUES
        (1,'Dungeness', 'M', 134, 'Y', 54),
        (2,'Graceful', 'F', 62, 'Y', 54),
        (3,'Red Rock', 'M', 99, 'Y', 54),
        (4,'Red Rock', 'F', 118, 'Y', 72),
        (5,'Dungeness', 'M', 155, 'Y', 3);
```

Studies(**<u>volunteerID</u>**, **<u>crab#</u>**)

```
INSERT
INTO Studies (volunteerID, crab#)
VALUES
        (101, 1),
        (103, 1),
        (104, 2),
        (104, 4),
        (105, 5);
```