University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: ____2____

Date: _____March 1, 2023_____

Group Number: _____40_____

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Parker Lee	35909605	u4s5k	parkerlee247@gmail.com
Drea Sy-Quia	96799671	l5m7y	dreasyquia@gmail.com
Jason Liu	16585622	n9n4k	jl1u02asap@gmail.com

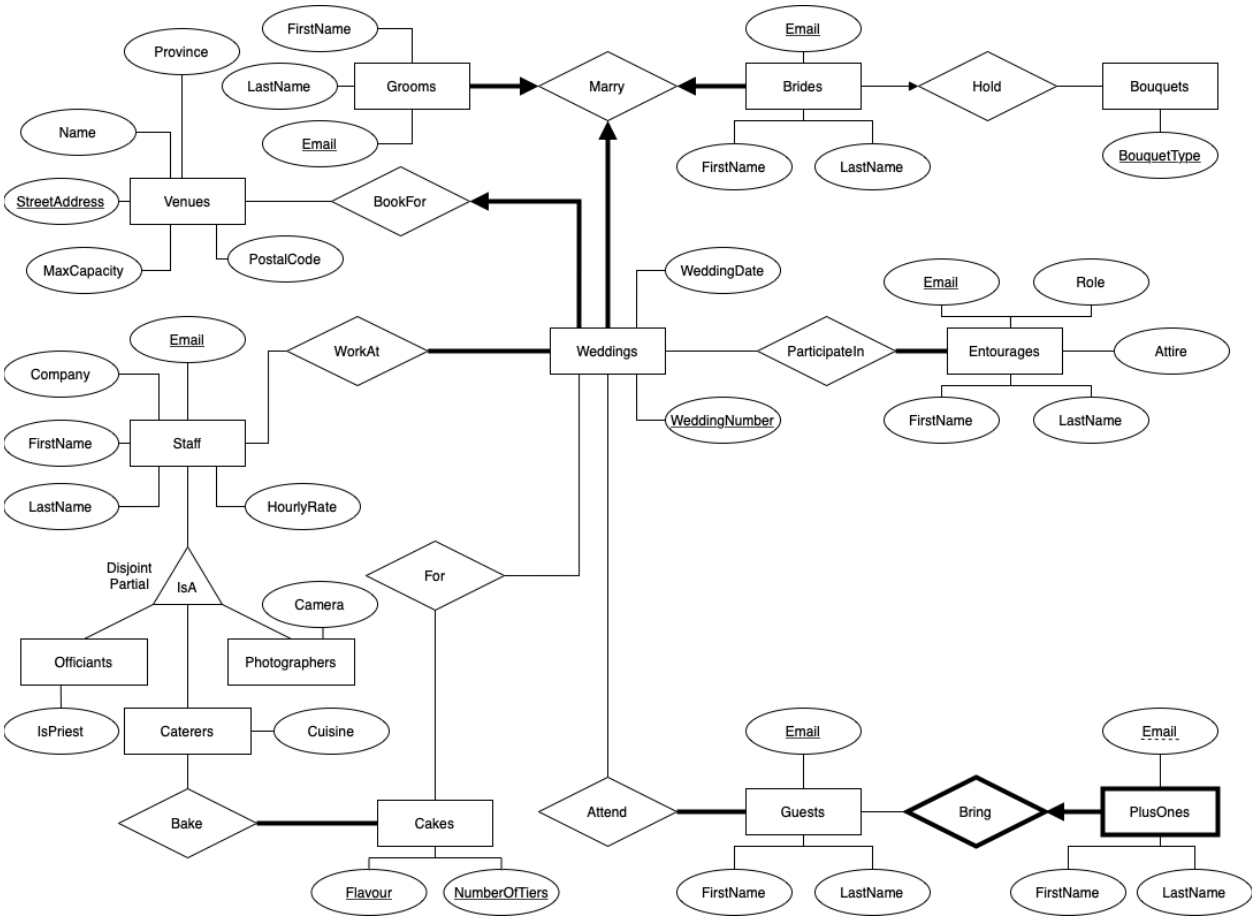
By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address,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

Project Summary

WedUP is a wedding planner – built for wedding planners! WedUP solves the difficulties of managing personnel by modeling wedding staff and guests in the database. WedUP reduces the need for planners to be bogged down by the overwhelming amount of logistics, and instead focus on delivering a spectacular wedding service for their clients.

ER Diagram



- Added “Attire” attribute to Entourages entity to have a functional dependency not identified by a primary key or a candidate key
- Added “Province” and “PostalCode” attributes to Venues entity to have a functional dependency not identified by a primary key or a candidate key
- Added “HourlyRate” attribute to Staff entity to have a functional dependency not identified by a primary key or a candidate key
- Added covering and joining constraint for ISA entity since it was missing in M1
- Moved “WeddingDate” attribute from BookFor relationship to Weddings entity since it feels more a part of the latter rather than the former after further consideration

Schema

Note: no candidate keys exist for any of the relations

- WeddingsBookFor(WeddingNumber: Integer, **StreetAddress: String,** WeddingDate: Date)
    - StreetAddress is not null
- BridesHolds(Email: String, FirstName: String, LastName: String, **BouquetType: String**)
- GroomsMarry(Email: String, FirstName: String, LastName: String, **BrideEmail: String, WeddingNumber: Integer**)
    - BrideEmail is unique and not null
    - WeddingNumber is unique and not null
- Bouquets(BouquetType: String)
- Entourages(Email: String, Role: String, FirstName: String, LastName: String, Attire: String)
- Venues(StreetAddress: String, Name: String, MaxCapacity: Integer, PostalCode: String, Province: String)
- Staff(Email: String, Company: String, FirstName: String, LastName: String, HourlyRate: Integer)
- Officiants(**Email: String**, IsAPriest: Boolean)
- Caterers(**Email: String**, Cuisine: String)
- Photographers(**Email: String**, Camera: String)
- Cakes(Flavour: String, NumberOfTiers: Integer)
- Guests(Email: String, FirstName: String, LastName: String)
- PlusOnesBring(**GuestEmail: String**, PlusOneEmail: String, FirstName: String, LastName: String)
- ParticipateIn(**WeddingNumber: Integer**, **EntourageEmail: String**)
- WorksAt(**WeddingNumber: Integer**, **StaffEmail: String**)
- Attends(**WeddingNumber: Integer**, **GuestEmail: String**)
- Bakes(**CatererEmail: String**, **Flavour: String**, **NumberOfTiers: Integer**)
- For(**Flavour: String**, **NumberOfTiers: Integer**, **WeddingNumber: Integer**)

Functional Dependencies

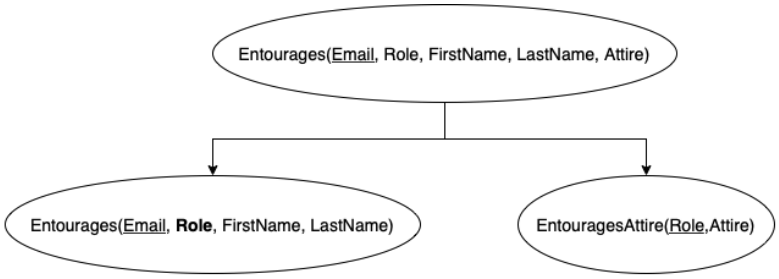
- WeddingsBookFor
    - WeddingNumber → StreetAddress, WeddingDate
- BridesHolds
    - Email → FirstName, LastName, BouquetType

- GroomsMarry
    - Email → FirstName, LastName, BrideEmail, WeddingNumber
- Bouquets
    - BouquetType → BouquetType
- Entourages
    - Email → Role, FirstName, LastName, Attire
    - Role → Attire
- Venues
    - StreetAddress → Name, MaxCapacity, PostalCode, Province
    - PostalCode → Province
- Staff
    - Email → Company, FirstName, LastName, HourlyRate
    - Company → HourlyRate
- Officiants
    - Email → IsAPriest
- Caterers
    - Email → Cuisine
- Photographers
    - Email → Camera
- Cakes
    - Flavour, NumberOfTiers → Flavour, NumberOfTiers
- Guests
    - Email → FirstName, LastName
- PlusOnesBring
    - GuestEmail, PlusOneEmail → FirstName, LastName
- ParticipateIn
    - WeddingNumber, EntourageEmail → WeddingNumber, EntourageEmail
- WorksAt
    - WeddingNumber, StaffEmail → WeddingNumber, StaffEmail
- Attends
    - WeddingNumber, GuestEmail → WeddingNumber, GuestEmail
- Bakes
    - CatererEmail, Flavour, NumberOfTiers → CatererEmail, Flavour, NumberOfTiers
- For
    - Flavour, NumberOfTiers, WeddingNumber → Flavour, NumberOfTiers, WeddingNumber

Normalization

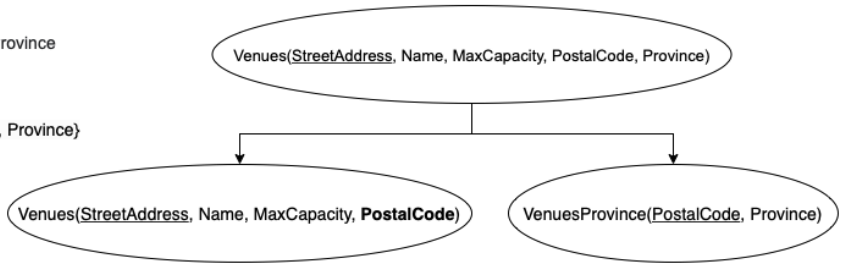
FDs:
Email → Role, FirstName, LastName, Attire
Role → Attire

Closures:
$\{Email\}^+ = \{Role, FirstName, LastName, Attire\}$
$\{Role\}^+ = \{Attire\}$



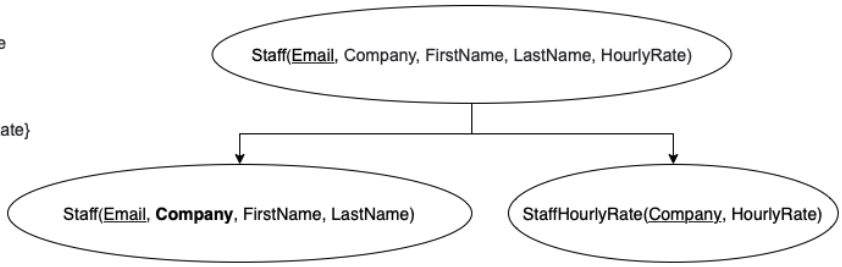
FDs:
StreetAddress → Name, MaxCapacity, PostalCode, Province
PostalCode → Province

Closures:
$\{StreetAddress\}^+ = \{Name, MaxCapacity, PostalCode, Province\}$
$\{PostalCode\}^+ = \{Province\}$



FDs:
Email → Company, FirstName, LastName, HourlyRate
Company → HourlyRate

Closures:
$\{Email\}^+ = \{Company, FirstName, LastName, HourlyRate\}$
$\{Company\}^+ = \{HourlyRate\}$



Note: no candidate keys exist for any of the relations

- WeddingsBookFor(WeddingNumber: Integer, **StreetAddress: String,** WeddingDate: Date)
    - StreetAddress is not null
- BridesHolds(Email: String, FirstName: String, LastName: String, **BouquetType: String**)
- GroomsMarry(Email: String, FirstName: String, LastName: String, **BrideEmail: String, WeddingNumber: Integer**)
    - BrideEmail is unique and not null
    - WeddingNumber is unique and not null
- Bouquets(BouquetType: String)
- Entourages(Email: String, **Role: String**, FirstName: String, LastName: String)
- EntouragesAttire(Role: String, Attire: String)
- Venues(StreetAddress: String, Name: String, MaxCapacity: Integer, **PostalCode: String**)

- VenuesProvince(PostalCode: String, Province: String)
- Staff(Email: String, **Company: String**, FirstName: String, LastName: String)
- StaffHourlyRate(Company: String, HourlyRate: Integer)
- Officiants(**Email: String**, IsAPriest: Boolean)
- Caterers(**Email: String**, Cuisine: String)
- Photographers(**Email: String**, Camera: String)
- Cakes(Flavour: String, NumberOfTiers: Integer)
- Guests(Email: String, FirstName: String, LastName: String)
- PlusOnesBring(**GuestEmail: String**, PlusOneEmail: String, FirstName: String, LastName: String)
- ParticipateIn(**WeddingNumber: Integer**, **EntourageEmail: String**)
- WorksAt(**WeddingNumber: Integer**, **StaffEmail: String**)
- Attends(**WeddingNumber: Integer**, **GuestEmail: String**)
- Bakes(**CatererEmail: String**, **Flavour: String**, **NumberOfTiers: Integer**)
- For(**Flavour: String**, **NumberOfTiers: Integer**, **WeddingNumber: Integer**)

SQL DDL Statements

```
CREATE TABLE WeddingsBookFor(
WeddingNumber INTEGER PRIMARY KEY,
StreetAddress CHAR(30) NOT NULL,
WeddingDate DATE,
FOREIGN KEY (StreetAddress) REFERENCES Venues(StreetAddress)
ON DELETE CASCADE
ON UPDATE CASCADE
);

CREATE TABLE BridesHolds(
Email CHAR(30) PRIMARY KEY,
FirstName CHAR(30),
LastName CHAR(30),
BouquetType CHAR(30),
FOREIGN KEY (BouquetType) REFERENCES Bouquets(BouquetType)
ON UPDATE CASCADE
);

CREATE TABLE GroomsMarry(
Email CHAR(30) PRIMARY KEY,
FirstName CHAR(30),
LastName CHAR(30),
BrideEmail CHAR(30) UNIQUE NOT NULL,
WeddingNumber INTEGER UNIQUE NOT NULL,
FOREIGN KEY (BrideEmail) REFERENCES BridesHolds(Email)
```

```
ON DELETE CASCADE
ON UPDATE CASCADE
FOREIGN KEY (WeddingNumber) REFERENCES WeddingsBookFor(WeddingNumber)
ON DELETE CASCADE
ON UPDATE CASCADE
);

CREATE TABLE Bouquets(
BouquetType CHAR(30) PRIMARY KEY
);

CREATE TABLE EntouragesAttire(
Role CHAR(30) PRIMARY KEY,
Attire CHAR(30)
);

CREATE TABLE Entourages(
Email CHAR(30) PRIMARY KEY,
Role CHAR(30),
FirstName CHAR(30),
LastName CHAR(30),
FOREIGN KEY (Role) REFERENCES EntouragesAttire(Role)
ON DELETE CASCADE
ON UPDATE CASCADE
);

CREATE TABLE VenuesProvince(
PostalCode CHAR(30) PRIMARY KEY,
Province CHAR(30)
);

CREATE TABLE Venues(
StreetAddress CHAR(30) PRIMARY KEY,
Name CHAR(30),
MaxCapacity INTEGER,
PostalCode CHAR(30),
FOREIGN KEY (PostalCode) REFERENCES VenuesProvince(PostalCode)
ON DELETE CASCADE
ON UPDATE CASCADE
);

CREATE TABLE Staff(
Email CHAR(30) PRIMARY KEY,
Company CHAR(30),
FirstName CHAR(30),
LastName CHAR(30),
FOREIGN KEY (Company) REFERENCES StaffHourlyRate(Company)
ON DELETE CASCADE
```

```
ON UPDATE CASCADE
);

CREATE TABLE StaffHourlyRate(
Company CHAR(30) PRIMARY KEY,
HourlyRate INTEGER
);

CREATE TABLE Officiants(
Email CHAR(30) PRIMARY KEY,
IsAPriest BOOLEAN
);

CREATE TABLE Caterers(
Email CHAR(30) PRIMARY KEY,
Cuisine CHAR(30)
);

CREATE TABLE Photographers(
Email CHAR(30) PRIMARY KEY,
Camera CHAR(30)
);

CREATE TABLE Cakes(
Flavour CHAR(30),
NumberOfTiers INTEGER,
PRIMARY KEY (Flavour, NumberOfTiers)
);

CREATE TABLE Guests(
Email CHAR(30) PRIMARY KEY,
FirstName CHAR(30),
LastName CHAR(30),
);

CREATE TABLE PlusOnesBring(
GuestEmail CHAR(30),
PlusOneEmail CHAR(30),
FirstName CHAR(30),
LastName CHAR(30),
PRIMARY KEY (GuestEmail, PlusOneEmail),
FOREIGN KEY (GuestEmail) REFERENCES Guests(Email)
);

CREATE TABLE ParticipateIn(
WeddingNumber INTEGER,
EntourageEmail CHAR(30),
PRIMARY KEY (WeddingNumber, EntourageEmail),
FOREIGN KEY (WeddingNumber) REFERENCES WeddingsBookFor(WeddingNumber),
```

```
FOREIGN KEY (EntourageEmail) REFERENCES Entourages(Email)
);

CREATE TABLE WorksAt(
WeddingNumber INTEGER,
StaffEmail CHAR(30),
PRIMARY KEY (WeddingNumber, StaffEmail),
FOREIGN KEY (WeddingNumber) REFERENCES WeddingsBookFor(WeddingNumber),
FOREIGN KEY (StaffEmail) REFERENCES Staff(Email)
);

CREATE TABLE Attends(
WeddingNumber INTEGER,
GuestEmail CHAR(30),
PRIMARY KEY (WeddingNumber, GuestEmail),
FOREIGN KEY (WeddingNumber) REFERENCES WeddingsBookFor(WeddingNumber),
FOREIGN KEY (GuestEmail) REFERENCES Guests(Email)
);

CREATE TABLE Bakes(
CatererEmail CHAR(30),
Flavour CHAR(30),
NumberOfTiers INTEGER,
PRIMARY KEY (CatererEmail, Flavour, NumberOfTiers),
FOREIGN KEY (CatererEmail) REFERENCES Caterers(Email),
FOREIGN KEY (Flavour, NumberOfTiers) REFERENCES Cakes(Flavour, NumberOfTiers)
);

CREATE TABLE For(
Flavour CHAR(30),
NumberOfTiers INTEGER,
WeddingNumber INTEGER,
PRIMARY KEY (Flavour, NumberOfTiers, WeddingNumber),
FOREIGN KEY (Flavour, NumberOfTiers) REFERENCES Cakes(Flavour, NumberOfTiers),
FOREIGN KEY (WeddingNumber) REFERENCES WeddingsBookFor(WeddingNumber)
);
```

INSERT Statements

```
INSERT
INTO WeddingsBookFor(WeddingNumber, Address, WeddingDate)
VALUES
(1, 601 Smithe St, 2023-08-23),
(2, 6301 Crescent Rd, 2023-09-24),
(3, 5251 Oak St, 2023-10-25),
(4, 845 Avison Way, 2023-11-26),
(5, 405 Spray Ave, 2023-12-27);
```

```
INSERT
INTO GroomsMarry(Email, FirstName, LastName, BrideEmail, WeddingNumber)
VALUES
(rickblaine@gmail.com, Rick, Blaine, ilsalund@gmail.com, 1),
(jackdawson@gmail.com, Jack, Dawson, rosedewittbukater@gmail.com, 2),
(sebwilder@gmail.com, Sebastian, Wilder, miadolan@gmail.com, 3),
(forrestgump@gmail.com, Forrest, Gump, jennycurran@gmail.com, 4),
(carlfredrickson@gmail.com, Carl, Fredrickson, elliedocter@gmail.com, 5);

INSERT
INTO BridesHolds(Email, FirstName, LastName, BouquetType)
VALUES
(ilsalund@gmail.com, Ilsa, Lund, cascade),
(rosedewittbukater@gmail.com, Rose, DeWitt Bukater, posy),
(miadolan@gmail.com, Amelia, Dolan, hand-tied),
(jennycurran@gmail.com, Jenny, Curran, round),
(elliedocter@gmail.com, Elizabeth, Docter, pomander);

INSERT
INTO Bouquets(BouquetType)
VALUES
(cascade),
(posy),
(hand-tied),
(round),
(pomander),
(composite),
(nosegay);

INSERT
INTO Entourages(Email, Role, FirstName, LastName)
VALUES
(captrenault@gmail.com, best man, Louis, Renault),
(victorlaszlo@gmail.com, usher, Victor, Laszlo),
(ruthdewittbukater@gmail.com, bridesmaid, Ruth, DeWitt Bukater),
(lizzyclavert@gmail.com, flower girl, Lizzy, Calvert),
(tommyryan@gmail.com, best man, Tommy, Ryan),
(laurawilder@gmail.com, bridesmaid, Laura, Wilder),
(ltdan@gmail.com, best man, Dan, Taylor),
(bubba@gmail.com, groomsman, Benjamin Buford, Blue),
(charlesmuntz@gmail.com, usher, Charles, Muntz),
(russelnagai@gmail.com, ring bearer, Russel, Nagai);

INSERT
INTO EntouragesAttire(Role, Attire)
VALUES
(best man, suit),
(groomsman, suit),
```

```
(bridesmaid, dress),
(flower girl, dress),
(usher, suit),
(ring bearer, suit);

INSERT
INTO Venues(StreetAddress, Name, MaxCapacity, PostalCode)
VALUES
(601 Smithe St, The Orpheum, 2780, V6B 3L4),
(6301 Crescent Rd, UBC Rose Garden, 250, V6T 1Z2),
(5251 Oak St, VanDusen Botanical Garden, 400, V6M 4H1),
(845 Avison Way, Vancouver Aquarium, 1000, V6G 3E2),
(405 Spray Ave, Fairmont Banff Springs, 2500, T1L 1J4);

INSERT
INTO VenuesProvince(PostalCode, Province)
VALUES
(V6B 3L4, BC),
(V6T 1Z2, BC),
(V6M 4H1, BC),
(V6G 3E2, BC),
(T1L 1J4, Alberta);

INSERT
INTO Staff(Email, Company, FirstName, LastName)
VALUES
(michaelcurtiz@gmail.com, Casablanca Co., Michael, Curtiz),
(maxsteiner@gmail.com, Casablanca Co., Max, Steiner),
(jamescameron@gmail.com, Titanic Studio, James, Cameron),
(jameshorner@gmail.com, Titanic Studio, James, Horner),
(celinedion@gmail.com, Titanic Studio, Celine, Dion),
(damienchazelle@gmail.com, La La Land Ltd., Damien, Chazelle),
(johnlegend@gmail.com, La La Land Ltd., John Legend),
(brianrobins@gmail.com, Paramount, Brian, Robins),
(robertzemeckis@gmail.com, Forrest Gump and Co., Robert, Zemeckis),
(petedocter@gmail.com, Up Services, Pete, Docter);

INSERT
INTO StaffHourlyRate(Company, HourlyRate)
VALUES
(Casablanca Co., 30),
(Titanic Studio, 28),
(La La Land Ltd., 22),
(Paramount, 20),
(Forrest Gump and Co., 23),
(Up Services, 22);

INSERT
```

```
INTO Officiants(Email, IsAPriest)
VALUES
(michaelcurtiz@gmail.com, true),
(jamescameron@gmail.com, false),
(damienchazelle@gmail.com, false);

INSERT
INTO Caterers(Email, Cuisine)
VALUES
(maxsteiner@gmail.com, austrian),
(jameshorner@gmail.com, dessert),
(brianrobins@gmail.com, dessert),
(johnlegend@gmail.com, american);

INSERT
INTO Photographers(Email, Camera)
VALUES
(celinedion@gmail.com, canon),
(petedocter@gmail.com, nikon);

INSERT
INTO Cakes(Flavour, NumberOfTiers)
VALUES
(tiramisu, 3),
(tiramisu, 2),
(red velvet, 7),
(black forest, 4),
(creme brulee, 1);

INSERT
INTO Guests(Email, FirstName, LastName)
VALUES
(signorferrari@gmail.com, Signor, Ferrari),
(brocklovett@gmail.com, Brock, Lovett),
(caledonhockley@gmail.com, Caledon, Hockley),
(captsmith@gmail.com, Edward John, Smith),
(amybrandt@gmail.com, Amy, Brandt);

INSERT
INTO PlusOnesBring(GuestEmail, PLusOneEmail, FirstName, LastName)
VALUES
(signorferrari@gmail.com, majstrasser@gmail.com, Heinrich, Strasser),
(signorferrari@gmail.com, janbrandel@gmail.com, Jan, Brandel),
(brocklovett@gmail.com, fabrizioderossi@gmail.com, Fabrizio, De Rossi),
(caledonhockley@gmail.com, spicerlovejoy@gmail.com, Spicer, Lovejoy),
(captsmith@gmail.com, archibaldgracie@gmail.com, Archibald, Gracie);

INSERT
```

```
INTO ParticipateIn(WeddingNumber, EntourageEmail)
VALUES
(captrenault@gmail.com, 1),
(victorlaszlo@gmail.com, 1),
(ruthdewittbukater@gmail.com, 2),
(lizzyclavert@gmail.com, 2),
(tommyryan@gmail.com, 2),
(laurawilder@gmail.com, 3),
(ltdan@gmail.com, 4),
(bubba@gmail.com, 4),
(charlesmuntz@gmail.com, 5),
(russelnagai@gmail.com, 5);


INSERT
INTO WorksAt(WeddingNumber, StaffEmail)
VALUES
(1, michaelcurtiz@gmail.com),
(1, maxsteiner@gmail.com),
(2, jamescameron@gmail.com),
(2, jameshorner@gmail.com),
(2, celinedion@gmail.com),
(3, damienchazelle@gmail.com),
(3, johnlegend@gmail.com),
(3, brianrobins@gmail.com),
(4, brianrobins@gmail.com),
(4, robertzemeckis@gmail.com),
(5, petedocter@gmail.com);

INSERT
INTO Attends(WeddingNumber, GuestEmail)
VALUES
(1, signorferrari@gmail.com),
(2, brocklovett@gmail.com),
(2, caledonhockley@gmail.com),
(2, captsmith@gmail.com),
(3, amybrandt@gmail.com);

INSERT
INTO Bakes(CatererEmail, Flavour, NumberOfTiers)
VALUES
(jameshorner@gmail.com, tiramisu, 3),
(jameshorner@gmail.com, black forest, 4),
(brianrobins@gmail.com, tiramisu, 2),
(brianrobins@gmail.com, red velvet, 7),
(brianrobins@gmail.com, creme brulee, 1);

INSERT
```

```
INTO For(Flavour, NumberOfTiers, WeddingNumber)
VALUES
(tiramisu, 3, 1),
(black forest, 4, 1),
(tiramisu, 2, 4),
(red velvet, 7, 3),
(creme brulee, 1, 3);
```