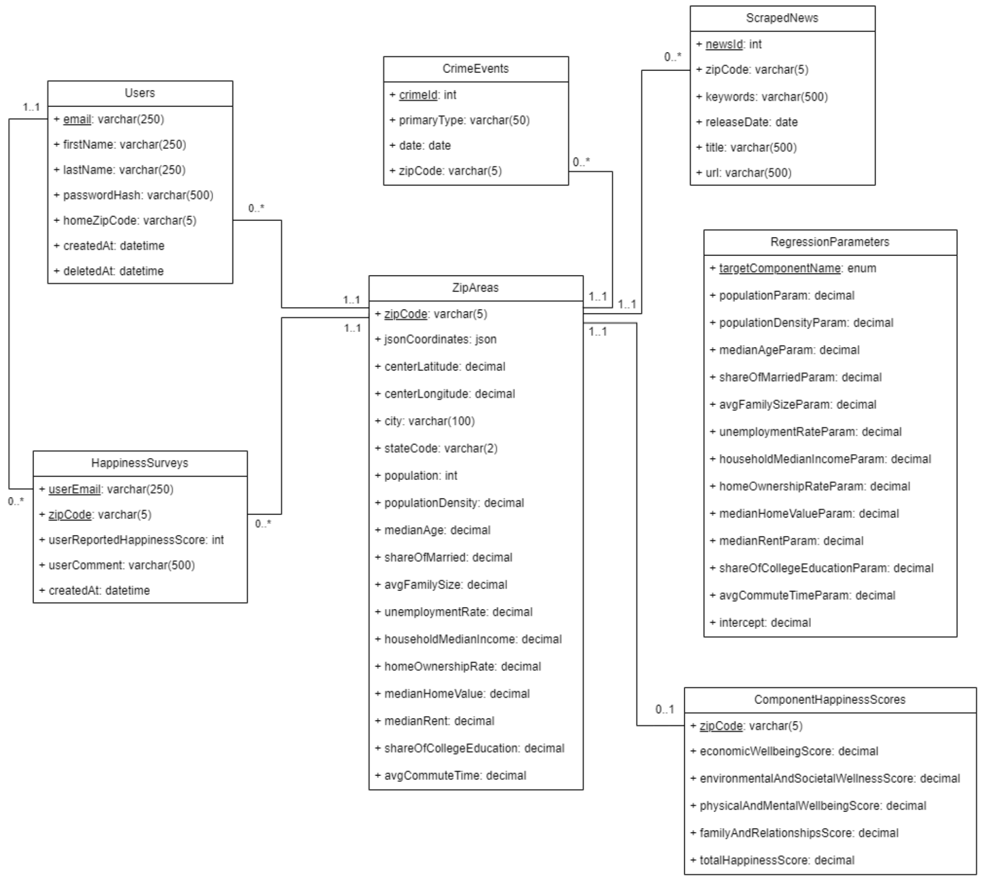
Database Design


ScrapedNews
+ newsId: int
+ zipCode: varchar(5)
+ keywords: varchar(500)
+ releaseDate: date
+ title: varchar(500)
+ url: varchar(500)
CrimeEvents
+ crimeId: int
+ primaryType: varchar(50)
+ date: date
+ zipCode: varchar(5)
Users
+ email: varchar(250)
+ firstName: varchar(250)
+ lastName: varchar(250)
+ passwordHash: varchar(500)
+ homeZipCode: varchar(5)
+ createdAt: datetime
+ deletedAt: datetime
RegressionParameters
+ targetComponentName: enum
+ populationParam: decimal
+ populationDensityParam: decimal
+ medianAgeParam: decimal
+ shareOfMarriedParam: decimal
+ avgFamilySizeParam: decimal
+ unemploymentRateParam: decimal
+ householdMedianIncomeParam: decimal
+ homeOwnershipRateParam: decimal
+ medianHomeValueParam: decimal
+ medianRentParam: decimal
+ shareOfCollegeEducationParam: decimal
+ avgCommuteTimeParam: decimal
+ intercept: decimal
ZipAreas
+ zipCode: varchar(5)
+ jsonCoordinates: json
+ centerLatitude: decimal
+ centerLongitude: decimal
+ city: varchar(100)
+ stateCode: varchar(2)
+ population: int
+ populationDensity: decimal
+ medianAge: decimal
+ shareOfMarried: decimal
+ avgFamilySize: decimal
+ unemploymentRate: decimal
+ householdMedianIncome: decimal
+ homeOwnershipRate: decimal
+ medianHomeValue: decimal
+ medianRent: decimal
+ shareOfCollegeEducation: decimal
+ avgCommuteTime: decimal
HappinessSurveys
+ userEmail: varchar(250)
+ zipCode: varchar(5)
+ userReportedHappinessScore: int
+ userComment: varchar(500)
+ createdAt: datetime
ComponentHappinessScores
+ zipCode: varchar(5)
+ economicWellbeingScore: decimal
+ environmentalAndSocietalWellnessScore: decimal
+ physicalAndMentalWellbeingScore: decimal
+ familyAndRelationshipsScore: decimal
+ totalHappinessScore: decimal
1..1
0..*
0..*
1..1
1..1
0..*
0..*
1..1
1..1
1..1
0..*
0..1

Relation Assumptions

Users

1. Users can register with one email only once and, thus, email can be used as a uniquely identifying key.
2. A user must submit one survey when signing up. However, there might be a delay between signing up and submitting a survey, and thus a user might momentarily have zero surveys linked to themselves. If a user moves locations, they must submit another survey and, thus, a user might have multiple HappinessSurveys linked to themselves. This leads to a 0..* relationship.
3. Each user can live in one place and hence are linked to one location defined by a homeZipCode. This links to the ZipAreas entity table, and serves as a way for the user to fetch data about their home location.

HappinessSurveys

1. Each survey submitted by a user is linked to the user by an email.
2. Each survey is about the happiness in one area and, thus, is linked to one ZipAreas entity.
3. A user can provide only one survey per zip code and a composite key of attributes userEmail and zipCode can be used to uniquely identify entries.

ZipAreas

1. Because we only consider the US mainland area, we can use the zipCode as a unique identifier for the entries.
2. The location boundaries of the ZipAreas are stored in json format. Whilst these could be stored in a separate table of coordinates, we have decided to keep these in json format in the table as our web application will need the boundary coordinates in json format for map overlay.
3. Each ZipAreas entry is linked to zero or one ComponentHappinessScores entry. Sometimes there is not sufficient data to calculate happiness scores and thus not every ZipCode has a score attributed to it. This is a one-to-one relationship, because we do not store historical changes in data and only update existing data if necessary.
4. ZipAreas may also have zero-to-many Users, HappinessSurveys, CrimeEvents or ScrapedNews linked to them.
5. This table also contains all the features (population, …, avgCommuteTime) which we will use to calculate the happiness score based on the zipCode.

CrimeEvents

1. This table contains data on crime events, including the location, date, and type of crime.
2. The app displays crime data to users based on the provided zip code.
3. Each crime event must be linked to one location and, thus, a one-to-one relationship to the ZipAreas table exists.
4. Each crime entry is given a unique identifier that determines all other attributes.

ScrapedNews

1. We will use a Python package to parse news from the website based on the provided zip code.

2. Once the user searches for the happiness score in a specific location, the app will display news articles relevant to that area.
3. Each scraped news article is given a unique identifier that determines all other attributes.
4. Each ScrapedNews entry must link to one ZipAreas entry via a zipCode. We will derive the zip code from the article contents if available or approximate it based on the article contents.

ComponentHappinessScores

1. Happiness of a location is estimated with five scores: EconomicWellbeingScore, EnvironmentalAndSocietalWellnessScore, PhysicalAndMentalWellbeingScore, familyAndRelationshipsScore, totalHappinessScore.
2. This table allows users to understand each aspect of the happiness score as well as the overall total happiness score.
3. Each entry in the ComponentHappinessScores table is linked to one ZipAreas entry, as one location can have just one happiness score.

RegressionParameters

1. This table contains the parameter values based on which the individual component scores are calculated. A linear equation representing the product-sum of features stored in ZipAreas and RegressionParameters will approximate the component happiness Scores. The parameters in this table are defined by a ML model trained behind-the-scenes on external data.
2. The table will have only 5 entries, identified by the targetComponentName. Each entry represents the parameters of the linear equation for a certain component happiness score.
3. The targetComponentName is an enum value where values are one of: 'economicWellbeingScore', 'environmentalAndSocietalWellnessScore', 'physicalAndMentalWellbeingScore', 'familyAndRelationshipsScore', or 'totalHappinessScore'.

3NF Normalization

For our schema to adhere to 3NF, each relation in the database should have only functional dependencies where the LHS of the equation is a superkey, or the RHS is part of a key. We have already considered this in our database design by separating functional dependencies in different relations. For example, to avoid transitive dependencies we have separated AreaFeatures and ComponentHappinessScores from

ZipAreas, and in each table there is one superkey that defines all attributes. We can prove that our database is in 3NF by observing the functional dependencies of each relation.

Users

In the Users table, the email attribute defines all the other attributes and is the only functional dependency.

HappinessSurveys

In the HappinessSurveys table, the component key (userEmail, zipCode) defines all the other attributes and is the only functional dependency. Neither attribute part of the composite key can alone define the rest of the attributes.

ZipAreas

In the ZipAreas table, the zipCode attribute defines all the other attributes and is the only functional dependency.

CrimeEvents

In the CrimeEvents table, the crimeId attribute defines all the other attributes and is the only functional dependency.

ScrapedNews

In the ScrapedNews table, the newsId attribute defines all the other attributes and is the only functional dependency.

ComponentHappinessScores

In the ComponentHappinessScores table, the zipCode attribute defines all the other attributes and is the only functional dependency.

RegressionParameters

In the RegressionParameters table, the targetComponentId attribute defines all the other attributes and is the only functional dependency.

Relational Schema

```
Users(
      email:varchar(250) [PK],
      firstName: varchar(250),
      lastName: varchar(250),
      passwordHash: binary(512),
      homeZipCode: varchar(5) [FK to ZipAreas.zipCode],
      createdAt: datetime,
      deletedAt: datetime
)

ZipAreas(
      zipCode: varchar(5) [PK],
      jsonCoordinates: JSON,
      centerLatitude: decimal,
      centerLongitude: decimal,
      city: varchar(100),
      stateCode: varchar(2),
      population: int,
      populationDensity: decimal,
      medianAge: decimal,
      shareOfMarried: decimal,
      avgFamilySize: decimal,
      unemploymentRate: decimal,
      householdMedianIncome: decimal,
      homeOwnershipRate: decimal,
      medianHomeValue: decimal,
      medianRent: decimal,
      shareOfCollegeEducation: decimal,
      avgCommuteTime: decimal
)

CrimeEvents(
      crimeId: int [PK],
      primaryType: varchar(50),
      Date: date,
      zipCode: varchar(5) [FK to ZipAreas.zipCode]
```

```
)

ScrapedNews(
      newsId: int [PK],
      zipCode: varchar(5) [FK to ZipAreas.zipCode],
      keywords: varchar(500),
      releaseDate: date,
      title: varchar(500),
      url: varchar(500)
)

HappinessSurveys(
      userEmail: varchar(250) [PK, FK to Users.email],
      zipCode: varchar(5) [PK, FK to ZipAreas.zipCode],
      userReportedHappinessScore: int,
      userComment: varchar(500),
      createdAt: datetime
)

ComponentHappinessScores(
      zipCode: varchar(5) [PK, FK to ZipAreas.zipCode],
      economicWellbeingScore: decimal,
      environmentalAndSocietalWellnessScore: decimal,
      physicalAndMentalWellbeingScore: decimal,
      familyAndRelationshipsScore: decimal,
      totalHappinessScore: decimal
)

RegressionParameters(
      targetComponentId: ENUM('economicWellbeingScore',
                              'environmentalAndSocietalWellnessScore',
                              'physicalAndMentalWellbeingScore',
                              'familyAndRelationshipsScore',
                              'totalHappinessScore')
                              [PK],
      populationParam: decimal,
      populationDensityParam: decimal,
      unemploymentRateParam: decimal,
      shareOfCollegeEducationParam: decimal,
      medianRentParam: decimal,
      homeOwnershipRateParam: decimal,
      householdMedianIncomeParam: decimal,
      avgFamilySizeParam: decimal,
      shareOfMarriedParam: decimal,
      medianAgeParam: decimal
)
```

Data Definition Language (DDL) commands

```
CREATE TABLE ZipAreas (
    zipCode VARCHAR(5),
    jsonCoordinates JSON,
    centerLatitude DECIMAL(9, 6),
    centerLongitude DECIMAL(9, 6),
    city VARCHAR(100),
    stateCode CHAR(2),
    population INT,
    populationDensity DECIMAL(10, 2),
    medianAge DECIMAL(5, 2),
    shareOfMarried DECIMAL(5, 2),
    avgFamilySize DECIMAL(5, 2),
    unemploymentRate DECIMAL(5, 2),
    householdMedianIncome DECIMAL(10, 2),
    homeOwnershipRate DECIMAL(5, 2),
    medianHomeValue DECIMAL(10, 2),
    medianRent DECIMAL(10, 2),
    shareOfCollegeEducation DECIMAL(5, 2),
    avgCommuteTime DECIMAL(5, 2),
    PRIMARY KEY (zipCode)
);


CREATE TABLE ComponentHappinessScores (
    zipCode VARCHAR(5),
    totalHappinessScore DECIMAL(5, 2),
    economicWellbeingScore DECIMAL(5, 2),
    familyAndRelationshipsScore DECIMAL(5, 2),
    physicalAndMentalWellbeingScore DECIMAL(5, 2),
    environmentalAndSocietalWellnessScore DECIMAL(5, 2),
    PRIMARY KEY (zipCode),
    FOREIGN KEY (zipCode) REFERENCES ZipAreas(zipCode)
                          ON DELETE CASCADE
                          ON UPDATE CASCADE
);


CREATE TABLE RegressionParameters (
    targetComponentName ENUM('economicWellbeingScore',
                             'environmentalAndSocietalWellnessScore',
                             'physicalAndMentalWellbeingScore',
                             'familyAndRelationshipsScore',
                             'totalHappinessScore'),
    populationParam DECIMAL(12, 10),
    populationDensityParam DECIMAL(12, 10),
    medianAgeParam DECIMAL(12, 10),
    shareOfMarriedParam DECIMAL(12, 10),
    avgFamilySizeParam DECIMAL(12, 10),
    unemploymentRateParam DECIMAL(12, 10),
    householdMedianIncomeParam DECIMAL(12, 10),
    homeOwnershipRateParam DECIMAL(12, 10),
    medianHomeValueParam DECIMAL(12, 10),
    medianRentParam DECIMAL(12, 10),
    shareOfCollegeEducationParam DECIMAL(12, 10),
    averageCommuteTimeParam DECIMAL(12, 10),
    intercept DECIMAL(12, 10),
    PRIMARY KEY (targetComponentName)
);


CREATE TABLE Users (
    email VARCHAR(250),
    firstName VARCHAR(250),
    lastName VARCHAR(250),
    passwordHash BINARY(512),
```

```sql
    homeZipCode VARCHAR(5) NOT NULL,
    createdAt DATETIME,
    deletedAt DATETIME,
    PRIMARY KEY (email),
    FOREIGN KEY (homeZipCode) REFERENCES ZipAreas(zipCode)
                              ON DELETE CASCADE
                              ON UPDATE CASCADE
);


CREATE TABLE HappinessSurveys (
    userEmail VARCHAR(250),
    zipCode VARCHAR(5),
    userReportedHappinessScore INT,
    userComment VARCHAR(500),
    createdAt DATETIME,
    PRIMARY KEY (userEmail, zipCode),
    FOREIGN KEY (userEmail) REFERENCES Users(email)
                            ON DELETE CASCADE
                            ON UPDATE CASCADE,
    FOREIGN KEY (zipCode) REFERENCES ZipAreas(zipCode)
                          ON DELETE CASCADE
                          ON UPDATE CASCADE
);


CREATE TABLE ScrapedNews (
    newsId INT,
    zipCode VARCHAR(5) NOT NULL,
    description VARCHAR(255),
    releaseDate DATE,
    title VARCHAR(255),
    url VARCHAR(255),
    PRIMARY KEY (newsId),
    FOREIGN KEY (zipCode) REFERENCES ZipAreas(zipCode)
                          ON DELETE CASCADE
                          ON UPDATE CASCADE
);
```

```sql
CREATE TABLE CrimeEvents (
    crimeId INT PRIMARY KEY,
    primaryType VARCHAR(50),
    date DATE,
    zipCode VARCHAR(5) NOT NULL,
    FOREIGN KEY (zipCode) REFERENCES ZipAreas(zipCode)
                          ON DELETE CASCADE
                          ON UPDATE CASCADE
);
```