Knex Seeds and Migrations

Objectives

- Describe the purpose of migrations and seeds
- Create migrations with schema definitions for a single table
- Create seeds for a single table
- Create seeds for 1 to many tables
- Create seeds for many to many tables

Migrations

- A way to handle a streamline database across many environments
- Ensures all changes to the Schema are reproducible
  - Prevents inconsistencies from user error
  - Team can have same schema
  - Can make a change once and team will have it
- Manages incremental changes in a schema
  -You can add, remove and change columns to existing databases
  -You don’t have to blow away a production database to change the schema
- Allows you to rollback to specific times in your schema
  -You can undo a migration if it broke your database

Seeding

- Populate default data into a database
  - A countries table should come populated with countries
  - Any lookup table should have some lookups predefined
- Great for testing
  - Put test data into a database for testing purposes
  - Everyone can have some data to work with when developing and testing

Creating Migrations

First install Knex

```
npm install -g knex
```

In package.json

```
{
    "scripts" : {
    "knex": "knex"
    }
}
```

Creating Migrations

On the Command Line:

- knex migrate:make **<name>**

** Make sure you run this command in your project directory**

Creating Migrations

In newly created migration file

```
exports.up = function(knex, Promise) {

};

exports.down = function(knex, Promise) {

};
```

Creating Migrations

Add your schema

```
exports.up = function(knex, Promise) {
  return knex.schema.createTable('countries', table => {
    table.increments()
    table.string('name').notNullable().defaultsTo('')
    table.integer('population').notNullable().defaultsTo(0)
    table.timestamps(true, true)
  })
};

exports.down = function(knex, Promise) {
  return knex.schema.dropTable('countries')
};
```

Creating Migrations

Lastly...

In command Line

-knex migrate:latest

Seeding Data

In the command Line:
- knex seed:make **<name>**

NOTE when creating table names know the order these tables may have to be joined. Your database reads them in your file alphabetically and numerically.

Seeding Data

```
exports.seed = function (knex, Promise) {
  // Deletes ALL existing entries
  return knex('countries').del()
    .then(() => {
      // Inserts seed entries
      return knex('countries').insert([
        { name: 'Germany', population: 82114224 },
        { name: 'Puerto Rico', population: 3663131 },
        { name: 'Qatar', population: 2639211 }
      ])
    })
}
```

Seeding Data

Lastly

- knex seed:run

Seeding and IDs

```
exports.seed = function (knex, Promise) {
  // Deletes ALL existing entries
  return knex('countries').del()
    .then(() => {
      // Inserts seed entries
      return knex('countries').insert([
        { name: 'Germany', population: 82114224 },
        { name: 'Puerto Rico', population: 3663131 },
        { name: 'Qatar', population: 2639211 }
      ])
    }).then(() => {
      return knex.raw(
        `SELECT setval('countries_id_seq', (SELECT MAX(id) FROM countries));`
      );
    })
}
```

Dealing with Primary and Foreign Keys

```
table.integer('continent_id').notNullable()
table.foreign('continent_id').references('continents.id').onDelete('CASCADE')
```

Heroku

- now let's complete your checklist for heroic

```
heroku create:addons heroku-postgresql
```

Objectives

- Describe the purpose of migrations and seeds
- Create migrations with schema definitions for a single table
- Create seeds for a single table
- Create seeds for 1 to many tables
- Create seeds for many to many tables

Exercise

- Will be split into pairs
- Fork and Clone Repo
- Work through skit and creating migrations and seeds
- GitHub