CODE CHALLENGES

Code Challenge 1

Code Challenge 1

Instructions

1.

Using `db.each()`, select all the rows from a table from `Flower`. In the callback, check if the `petal_color` is 'blue' and `console.log` the row if it is.

Hint

Reference our `db.each()` exercise for the proper syntax.

app.js

```
const db = require('./db');

db.each("SELECT * FROM Flower", (error, row) => {
  if (row.petal_color === 'blue') {
    console.log(row)
  }
})
```

Code Challenge 2

Code Challenge 2

Instructions

1.

Use `db.run()` to create a new empty table called `City`.

Hint

Reference our Node table exercise for the proper syntax.

app.js

```
const db = require('./db');

db.run("CREATE TABLE City()")
```

Code Challenge 3

Code Challenge 3

Instructions

1.

SELECT the `superpower` column of the superhero in the `Superhero` table with an `id` of `12` and `console.log()` that `superpower` in the callback function.

Hint

Reference our Node exercise on retrieving a single row for the proper syntax.

Reference our SQL `WHERE` exercise for the syntax to select a row by a specific value.

app.js

```
const db = require('./db');

db.get("SELECT superpower FROM Superhero WHERE id = 12", (err, row) => {
  console.log(row.superpower)
})
```

Code Challenge 4

Code Challenge 4

Instructions

1.

The query in the workspace is going to return an error! Log the error to the console if it exists, otherwise log the retrieved rows.

Hint

Reference our exercise on error-handling for the proper syntax.

app.js

```
const db = require('./db');

db.all('SELECT * from NonexistentTable', (err, rows) => {
  if(err) {
    console.log(err)
  } else{
      console.log(rows)
```

```
  }
});
```

Code Challenge 5

Code Challenge 5

Instructions

1.

Use `db.each()` to find the `totalPrice` if you bought every shirt from the `Clothing` table. Select the `price` from each row where `item` is `'shirt'` and add the prices to `totalPrice`. Log `totalPrice` after they have all been added. Each row's `price` property is already a number, so you do not need to use `Number()` to convert it.

Hint

You'll need to use both callbacks in `db.each()`. You can reference our `db.each()` exercise for a refresher on syntax and use.

app.js

```
const db = require('./db');

let totalPrice = 0;
db.each("SELECT price FROM Clothing WHERE item = 'shirt'", (err, row) => {
    totalPrice += (row.price);
  },
  (err, numRows) =>{
    console.log(totalPrice)
  }
);
```

Code Challenge 6

Code Challenge 6

Instructions

1.

Find a way to wrap the queries in the workspace so that they run synchronously.

Hint

You'll probably want to serialize your queries.

app.js

```
const db = require('./db');

db.serialize(() => {
  db.run('CREATE TABLE Popcorn (id INTEGER PRIMARY KEY, type TEXT)');
  db.run('INSERT INTO POPCORN (type) VALUES ("cheddar")');
  db.run('INSERT INTO POPCORN (type) VALUES ("kettle corn")');
})
```

Code Challenge 7

Code Challenge 7

Instructions

1.

Find and print the `quantity` column of the spice `'paprika'` in a table called `SpiceRack` based on its `name`. `name`s are unique, so you only need to retrieve one row.

Hint

You can reference our `db.get()` exercise for the proper syntax.

app.js

```
const db = require('./db');

db.get("SELECT quantity FROM SpiceRack WHERE name = 'paprika'", (err, row) => {
  console.log(row.quantity)
})
```

Code Challenge 8

Code Challenge 8

Instructions

1.

Use the `genre` parameter in the function to find the `title` of every song in the `Song` database matching the `genre`.

Hint

You'll need to use placeholders to `SELECT` based on the `genre`.

app.js

```
const db = require('./db');

const selectByGenre = genre => {
  // Add your code in here
  db.all("SELECT title FROM Song WHERE genre = $genre",
  {
    $genre: genre
  })
}
```

Code Challenge 9

Code Challenge 9

Instructions

1.

Use `db.all()` to find every scientist from the `Scientist` table whose `field` is `'biology'` and select all columns. Log the list to the console.

Hint

You can reference our `db.all()` exercise for a refresher on syntax and use.

app.js

```
const db = require('./db');

db.all("SELECT * FROM Scientist WHERE field = 'biology'", (err, rows) => {
  console.log(rows)
})
```

Code Challenge 10

Code Challenge 10

Instructions

1.

Use `db.each()` to print the `height` of every character from the `CartoonCharacter` database where the `species` is `'mouse'`.

Hint
Reference our `db.each()` exercise for the proper syntax.

app.js

```
const db = require('./db');

db.each("SELECT * FROM CartoonCharacter WHERE species = 'mouse'", (err, row) => {
  console.log(row.height)
})
```

Code Challenge 11

Code Challenge 11

Instructions

1.

Drop the table `Furniture` if it exists, then create it again (in that order). Don't worry about defining a schema for `Furniture` when you create it.

Hint
Reference our table exercise for the syntax to create a table.

app.js

```
const db = require('./db');

db.serialize(() => {
  db.run("DROP TABLE IF EXISTS Furniture;");
  db.run("CREATE TABLE Furniture")
});
```

Code Challenge 12

Code Challenge 12

Instructions

1.

Write a function called `logCaffeineLevel` that takes the `name` of a tea and logs its `caffeine_level` from the `Tea` table.

Hint

You'll need to use placeholders with a `db.get()` to `SELECT` based on the `name` parameter to your function.

app.js

```
const db = require('./db');

const logCaffeineLevel = (name) => {
  db.get("SELECT * FROM Tea WHERE name = $name", {
    $name: name
  },
  (error, row) => {
    console.log(row.caffeine_level)
  }
  )
}
```

Code Challenge 13

Code Challenge 13

Instructions

1.

Insert a new bridge into the `Bridge` table, with the `name` `Brooklyn Bridge` and with an `established_year` value of `1883`.

Hint

You'll want to use `db.run()` to `INSERT` a new row. Reference our `db.run()` exercise for the proper syntax.

app.js

```
const db = require('./db');

db.run("INSERT INTO Bridge (name, established_year) VALUES ('Brooklyn Bridge', 1883);");
```

Code Challenge 14

Code Challenge 14

Instructions

1.
You want to know the number of people per month that go through the same train station as you use for your commute. Get the `traffic` property from the `TrainStation` table where the `station_id` is `38` and the `month` is the current month. Log the `traffic` property of the row to the console.

Hint
You'll need to structure a SQL query using `WHERE`.

app.js

```
const db = require('./db');

db.get("SELECT traffic from TrainStation WHERE station_id = 38 AND month = 'Febru
ary'", (err, row) => {
  console.log(row.traffic)
})
```

Code Challenge 15
Code Challenge 15

Instructions

1.
Use the parameter to find the `number_of_floors` column from the `Building` table at the user-given `address`.

Hint
You'll have to use placeholders inside your `logFloorsForAddress` function to select by `address`.

app.js

```
const db = require('./db');

const logFloorsForAddress = address => {
  // Your code here:
  db.get("SELECT number_of_floors FROM Building WHERE address = $address",
    {
      $address: address
    },
```

```
    (err, row) => {
      console.log(row.number_of_floors);
    }
    )
}
```

Code Challenge 16

Code Challenge 16

Instructions

1.

Add a row to the `BirdOfParadise` table with `scientific_name` `Cicinnurus regius` and with `king bird-of-paradise` as its `common_name`

Hint

You can reference our `INSERT` exercise for the proper syntax.

app.js

```
const db = require('./db');

db.run("INSERT INTO BirdOfParadise (scientific_name, common_name) VALUES ('Cicinn
urus regius', 'king bird-of-paradise');");
```

Code Challenge 17

Code Challenge 17

Instructions

1.

Complete the `addMovie` function to inserts a movie into the `Movie` table with columns named `title`, `publication_year`, and `director`. Use the style of placeholders using named parameter and an object as the second argument of `db.run()`.

Hint

You can reference our placeholders exercise for use and syntax.

app.js

```
const db = require('./db');
```

```
const addMovie = (title, publicationYear, director) => {
  db.run("INSERT INTO Movie (title, publication_year, director) VALUES ($title, $
publicationYear, $director)", {
    $title: title,
    $publicationYear: publicationYear,
    $director: director
  })
};
```

Code Challenge 18

Code Challenge 18

Instructions

1.

Use `db.each()` to list all of the beverage `name`s that have `'soda'` as their `type` from the `Minifridge` table.

Hint

Reference our `db.each()` exercise for the proper syntax.

app.js

```
const db = require('./db');

db.each("SELECT * FROM Minifridge WHERE type = 'soda'", (err, row) => {
  console.log(row.name)
})
```

Code Challenge 19

Code Challenge 19

Instructions

1.

Take the day off! Add a new holiday to the `Holiday` database. Set the `name` attribute to any name you like and set `work` to `false`.

Hint

You can reference our `INSERT` exercise for the proper syntax.

app.js

```
const db = require('./db');


db.run("INSERT INTO Holiday (name, work) VALUES ('San Pedro Regalado', false);");
```