Pooling

There are a lot of different things to keep in mind with pooling, but what is pooling? Simply put, if you have a single connection, you have to connect, make the query, then disconnect for every query. This can cause a huge server load. So what you can do is pool the connections. Basically, you have (in the example below) 10 open connections. If one person calls the API, they use a pre-opened pooled connection, make their query and then it is released back to the pool. If there are 10 people using the connections and then an 11th connects, they're added to a queue, and then when one of the 10 in use is released they will use that one, and the pattern will continue.

- In the Database conf file:

```
import mysql from 'mysql';
let pool = mysql.createPool({
  connectionLimit: 10,
  host: process.env.DB_HOST,
  user: process.env.DB_USER,
  password: process.env.DB_PASSWORD,
  database: process.env.DB_DATABASE,
});

pool.getConnection((err, connection) => {
  if (err) {
    if (err.code === "PROTOCOL_CONNECTION_LOST") {
      console.error("Database connection was closed.");
    }
    if (err.code === "ER_CON_COUNT_ERROR") {
      console.error("Database has too many connections.");
    }
    if (err.code === "ECONNREFUSED") {
      console.error("Database connection was refused.");
    }
  }

  if (connection) connection.release();
  return;
});
```

- Then any route can access it with the following code:

```
//At the top
let pool = require("../mysql.conf.js"); // Or path to above code.

//Obviously the query can be anything you want it to be, this is just an
example.
pool.query(
  "SELECT * FROM POSTS WHERE userId = ?",
```

```
  req.params.userId,
  (err, results, field) => {
    res.send(results);
  }
);
```

With the above, rather than opening a connection, querying and then ending the connection, the `pool.query` does all of that work for you.

- Ideally you would move the queries into their own functions specific to the posts or users or whatever.
- This might look like:

```
getPostsById(res, userId){
    pool.query('SELECT * FROM POSTS WHERE userId = ?', userId, (err,
results, field)=>{
        if (err){
            res.send(err);
        }
        else{
            res.send(results);
        }
    })
}
```

- Then in the routes function for `GET /posts/:userId`

```
//At the top import the posts functions from the file

router.get("/posts/:userId", (req, res) => {
  //The below is an example but it can be whatever way you import it you'd
like
  postsFunctions.getPostsById(res, req.params.userId);
});
```