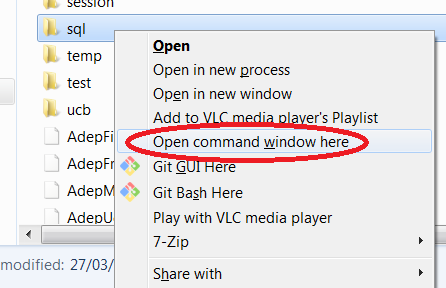
Create New Project

In this section create a new VSC (Visual Studio Code) project.

1. Using file Windows Explore, in D:\Project\VS create new folder sql
2. Select folder sql, Shift and RHC



Context menu should open

1. Select ‘Open command window here’. This should open a command promot with path d:\Project\VS\sql.
2. At command prompt type:

npm init

and accept all defaults. Results should be a package.json file

1. At command prompt start Visual Studio Code by typing (note the use of the full-stop):

code .

1. Close the command prompt (no longer required)
2. In VSC open an integrated terminal
3. Install some of the required external modules by typing:



This should update package.json & and create folder node\_module

1. Create file index.js and type in

var express = require('express');

var app = express();

var PORT = 1337;

// Default route

app.get('/', (req,res) => {

res.send('Hello SQL Soon');

});

app.listen(PORT, () => {

console.log('Listening on port: ' + PORT);

});

1. Start node server by typing

nodemon

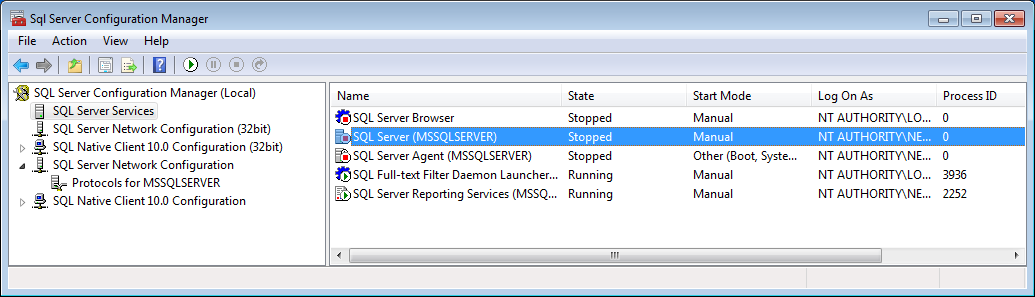
(nodemon will automatically detect .js file changes)

From a browser ensure you are getting ‘Hello SQL Soon’ message

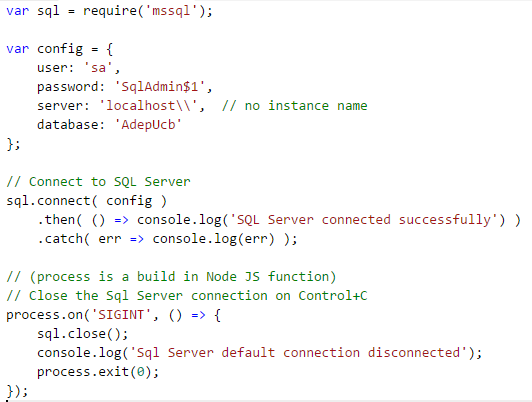
Connect to SQL Server

In this section you will turn on the SQL Server services and update your Node application to connect to SQL Server and run a simple query to test you can retrive results.

1. Turn on SQL Server services from Windows Start -> SQL Server Configuration Manager
2. SQL Server Network Configuration -> Protocols for MSSQLSERVER -> TCP/IP-> (RHC) Enabled



1. SQL Server Services -> SQL Server (MSSQLSERVER) -> (RHC) Start
2. SQL Server Services -> SQL Server Browser -> (RHC) Start
3. Back in VSC create a folder routes
4. Inside routes create a file sqlServerSetup.js and type in following:

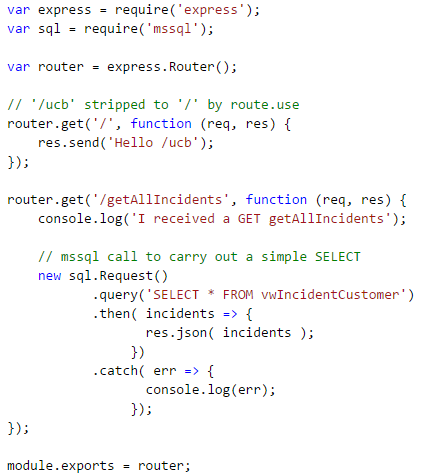


1. Back inside index.js add following line near top of file:

var sql = require('./routes/sqlServerSetup');

In terminal window should see SQL Server connected successfully

1. Inside routes folder create file ucb.js and type in following:



Execution of multiple Queries

In this section you will use Promises to return results from more than one query

1. In the Integrated Terminal stop nodemon (Ctrl +C, Y)
2. Install the Promise module q by typing:

npm install q --save

1. Restart nodemon
2. In file ucb.js add near the top enter:

var Q = require('q');

1. Next copy following ‘incorrect’ approach to return two query results:

router.get('/bad', function (req, res) {

console.log('I received a GET bad');

let firstQuery;

// Get 1st query - cound be long running

new sql.Request()

.query('select \* from Incident')

.then( results => {

firstQuery = results;

})

.catch(function (err) {

console.log(err);

});

// Get 2nd query - cound be short running

new sql.Request()

.query('select \* from Title')

.then( results => {

// Send results back

res.json( {category: firstQuery, title: results} );

})

.catch(function (err) {

console.log(err);

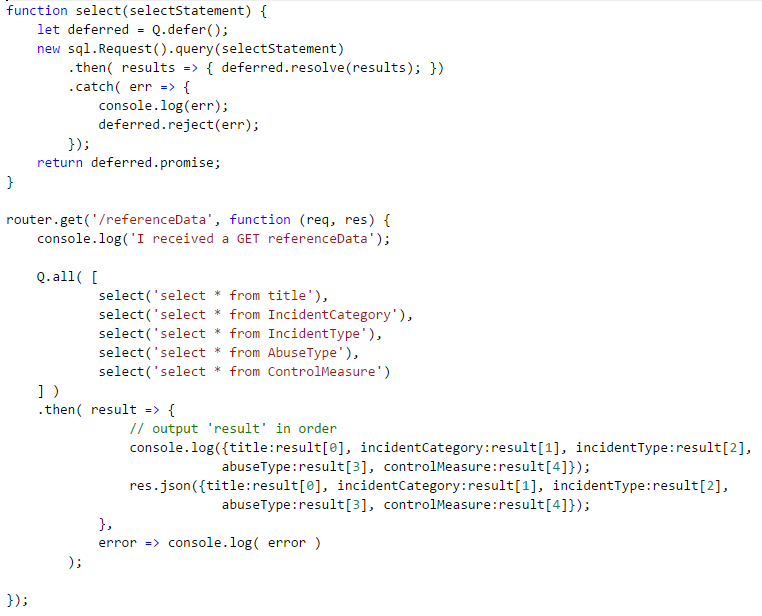
});

});

1. In browser navigate to ‘ucb/bad’. Note:

* Sometimes you will get all the {incidents + all Titles}
* Try refreshing (F5) and you may only get {Titles} because {Incidents} results have not been returned when 2nd query completed.

1. Now copy following code which uses Promises to execute 5 asyn queries and returns all the results when all the Promises have been resolved:



1. In browser navigate to ‘ucb/referenceData’. You should get all 5 sets of ref data.