IFT 598: Middleware Prog & Database Sec (2022 Fall)

Module 3 Assignment: Data Access

Name: Jyoshnapriya Buddhi

Class: 2022Fall-P-IFT458-IFT598-82341-82345

Date: 10-02-2022

**Npm init:**

Text

Description automatically generated

**Package.json:Text

Description automatically generated**

**Installing necessary modules:**

npm install express dotenv mssql morgan cors --save

Text

Description automatically generated

**Updated package.json:**

Text

Description automatically generated

**Installing MSSQL Extension:**

Graphical user interface, text

Description automatically generated

**Connecting to SQL Server:**

Server: sqlservercentralpublic.database.windows.net (Links to an external site.)

Database: AdventureWorks

Select SQL login option from the Dropdown

User: sqlfamily

Password: sqlf@m1ly

**Proof of Installation:**

Graphical user interface, text

Description automatically generated

**Setup Express Webserver:**

App.js

Text

Description automatically generated

Code:

const express = require('express');

const morgan = require('morgan');

const cors = require("cors");

const app = express();

//This is CORS Ref:https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

var corsOptions = {

origin: "http://localhost:8080"

};

app.use(cors(corsOptions));

// 1) MIDDLEWARES Morgan is used for debugging

if (process.env.NODE\_ENV === 'development') {

app.use(morgan('dev'));

}

// 2)MIDDLEWARE json is used for injecting the body attribute in the pipeline

app.use(express.json());

// parse requests of content-type - application/x-www-form-urlencoded

app.use(express.urlencoded({ extended: true }));

app.use((req, res, next) => {

console.log('Hello from the middleware 👋');

next();

});

// 3) MIDDLE ROUTES loading

const salesRouter = require('./routes/salesRoutes');

const customersRouter = require('./routes/customersRoutes');

app.use('/api/v1/sales', salesRouter);

app.use('/api/v1/customers', customersRouter);

module.exports = app;

Server.js:Graphical user interface, text

Description automatically generated

Code:

const dotenv = require('dotenv');

dotenv.config({ path: './config.env' }); // this line has to come first since we need to tell where is the config file.

const app = require('./app');

const port = process.env.PORT || 3000;

app.listen(port, () => {

console.log(`App running on port ${port}...`);

});

Config.env

Graphical user interface, text, application

Description automatically generated

Code:

NODE\_ENV=development

PORT=8000

DBUSERNAME=sqlfamily

PASSWORD=sqlf@m1ly

DATABASE=AdventureWorks

Install MSSQL package Text

Description automatically generated

**SQL server credentials**

db.config.js

Text

Description automatically generated

Code:

const sqlConfig = {

server: "sqlservercentralpublic.database.windows.net",

user: process.env.DBUSERNAME,

password: process.env.PASSWORD,

database: process.env.DATABASE,

pool: {

idleTimeoutMillis: 60000

},

options:{

encrypted: true, // for azure

trustServerCertificate: false,

useUTC: true

}

}

module.exports = sqlConfig;

`

Db.manager.js

Text

Description automatically generated

Code:

const sql = require('mssql');

const dbConnection = require('./db.config');

console.log(dbConnection);

async function getSalesProducts(){

console.log(' Connecting to SQL....... Cloud Server');

let dbContext = await sql.connect(dbConnection);

console.log('The Databse connection was Successful');

console.log('Getting data');

let results = await dbContext.request()

.query(`

SELECT TOP(20)

productId,

name,

productNumber,

color

listPrice

FROM

salesLT.Product

`);

console.log(`Returned SQL results`);

return results;

}

async function getCustomersProducts(){

console.log(' Connecting to SQL....... Cloud Server');

let dbContext = await sql.connect(dbConnection);

console.log('The Databse connection was Successful');

console.log('Getting data');

let results = await dbContext.request()

.query(`

SELECT TOP(20)

CustomerID,

NameStyle,

Title,

FirstName,

MiddleName,

LastName

FROM

salesLT.Customer

`);

console.log(`Returned SQL results`);

return results;

}

//Export

module.exports = {getSalesProducts :getSalesProducts, getCustomersProducts:getCustomersProducts};

**Creating routes folder:**

customersRouter.js:

Text, application, chat or text message

Description automatically generated

Code:

const express = require('express');

const router = express.Router();

const customersController = require('../controllers/customersController');

router

.route('/')

.get(customersController.getAllCustomers)

router

.route('/:id')

.get(customersController.getCustomersByID)

module.exports = router;

salesRouter.js

Graphical user interface, text, application, chat or text message

Description automatically generated

**Creating Controllers folder:**

customersController.js:

Text, chat or text message

Description automatically generated

Code:

const db = require('../config/db.manager');

exports.getAllCustomers = function (req, res) {

const customersProduct = db.getCustomersProducts().then(results=>{

console.log(results);

res.status(200).json({

status: 'successfull',

data: results.recordsets[0]

}); // send all the data

});

}

exports.getCustomersByID = function( req, res){

const {id} = req.params; // get id

res.status(200).json({

status: 'no implemented'

});

}

salesContoller.js:

Text

Description automatically generated

**Connecting to SQL server:**

Text

Description automatically generated

**Checking for /api/v1/customers:**

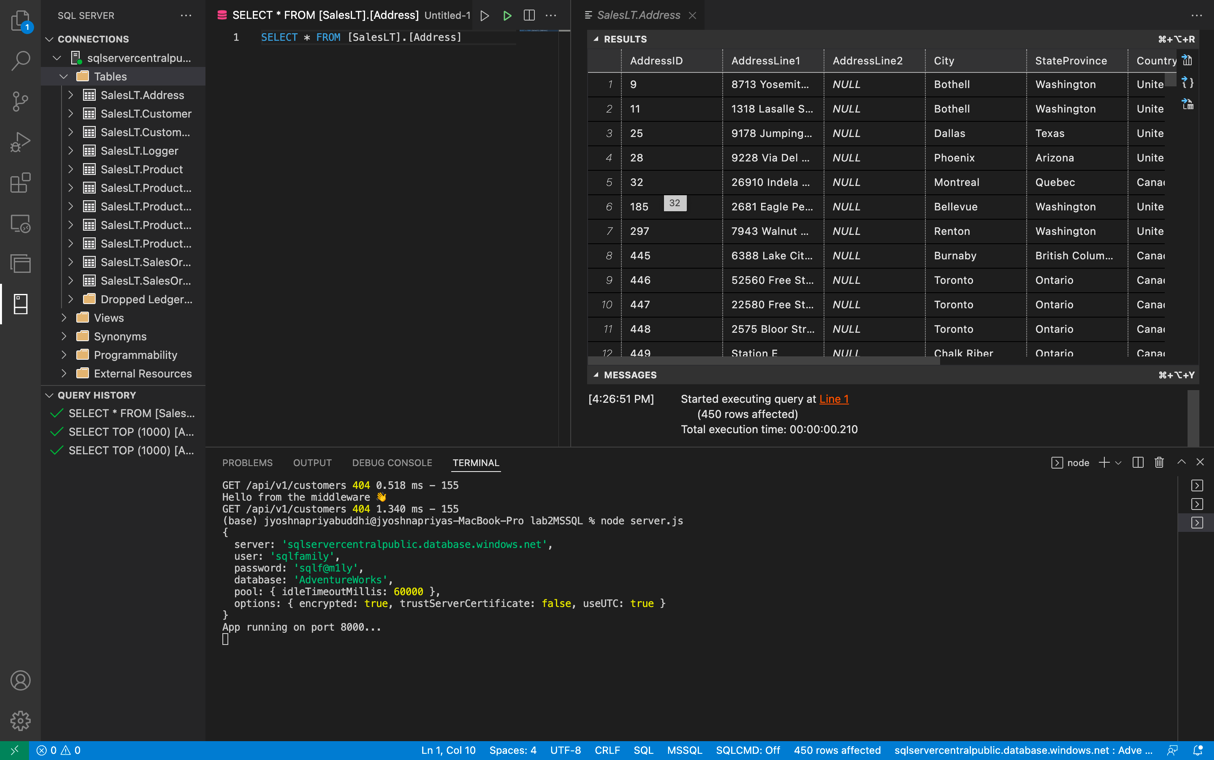
**Text

Description automatically generated**

**SELECT from 4 tables along with table view:**

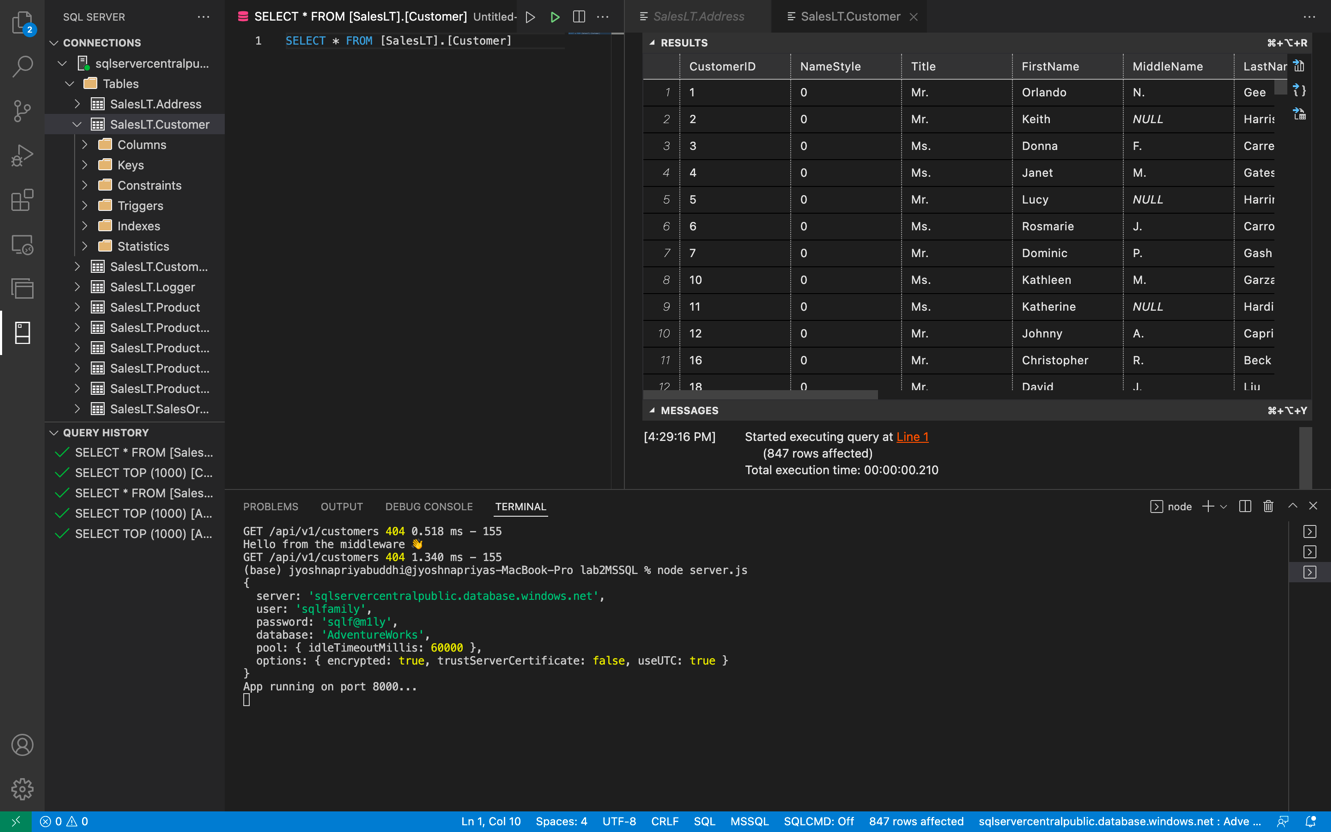
Select everything from address table:

SELECT \* FROM [SalesLT].[ Address]



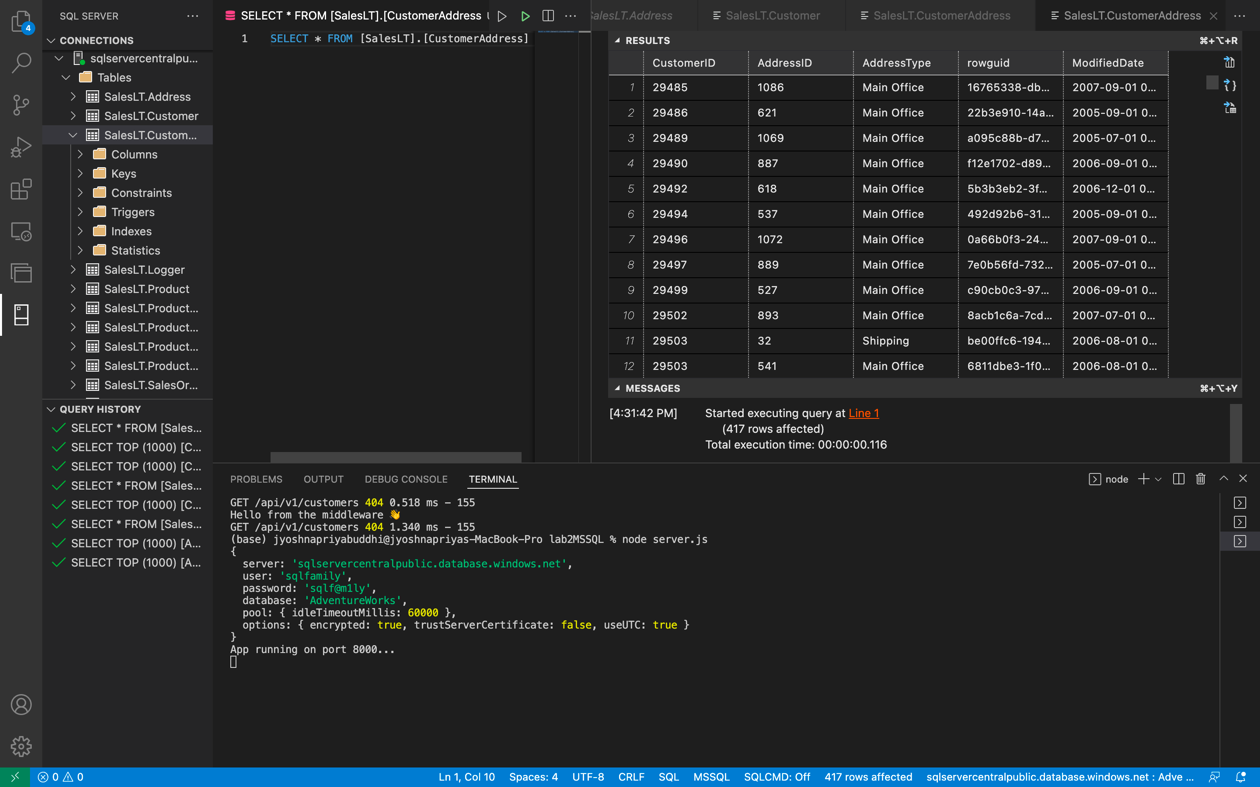
Select everything from customer table:

SELECT \* FROM [SalesLT].[ Customer]



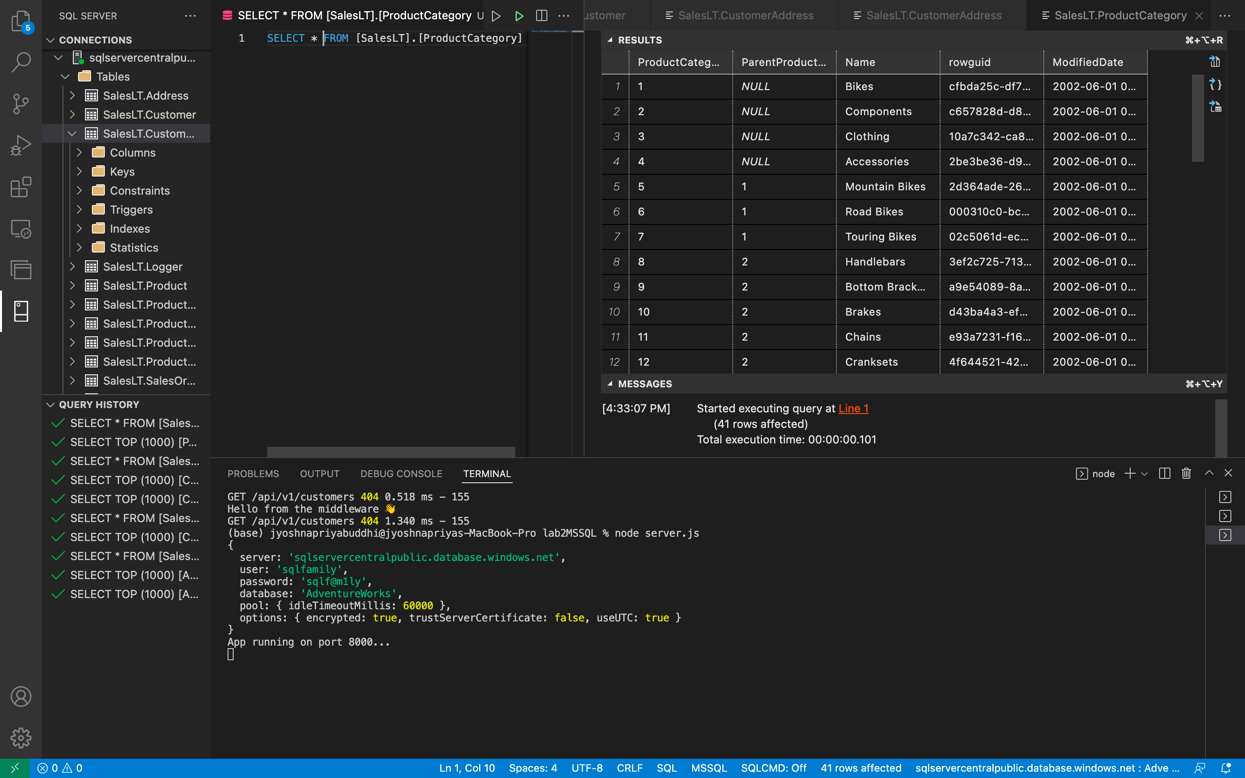
Select everything from CustomerAddress table:

SELECT \* FROM [SalesLT].[ CustomerAddress]



Select everything form Product Category Table

SELECT \* FROM [SalesLT].[ProductCategory]



**JOIN TWO TABLES:**

**A screenshot of a computer

Description automatically generated with medium confidence**

Code:

SELECT

SalesLT.Customer.FirstName AS First\_name,

SalesLT.Customer.MiddleName AS Middle\_name,

SalesLT.Customer.LastName AS Last\_name,

SalesLT.CustomerAddress.AddressID AS Address\_ID,

SalesLT.CustomerAddress.AddressType AS Address\_type

FROM SalesLT.Customer

JOIN SalesLT.CustomerAddress ON SalesLT.Customer.CustomerID= SalesLT.CustomerAddress.CustomerID;