TP 4 : NodeJS et Base de données

Prérequis :

- NodeJS
- SGBD MySQL

Exercice 1:

1) Créer les tables suivantes :
   - Etudiants(id, nom, prénom, id_cours) ;
   - Cours(id_cours, dicipline, date_d,date_f) ;

   Ajouter les lignes suivantes à chaque tables :
   Table Etudiants
   Ligne 1 :12, « John », « Smith », 1
   Ligne 2 :13, « Jack »,« Smith » , 2
   Ligne 3 :14, « Mary », «Smith », 1
   Table Cours :
   Ligne 1 :1, « Maths », « 1/1/2020 », « 1/03/2020 »
   Ligne 1 :2, « physics », « 1/1/2020 », « 1/03/2020 »
   Ligne 1 :3, « Electronics », « 1/02/2020 », « 1/05/2020 »
2) Ecrire un programme NodeJS qui permet d'afficher la liste des étudiants inscrit dans le cours maths.

```
const mysql = require('mysql');
   const con = mysql.createConnection({host: "localhost",
   user: "nom_utilisateur",
   password: "mot_de_passe_utilisateur"
   database : "mabdd"
 });
  con.connect(function(err) {
   if (err) throw err;
   console.log("Connecté à la base de données MySQL!");
   con.query("SELECT nom,prenom from Etudiants where id_cours=1 ; ",
function (err, result) {
       if (err) throw err;
       console.log(result);
     });
 });
```

3) Ecrire un programme NodeJS qui permet d'insérer un nouvel étudiant «15, Jack, Peterson, 1 » à la fin du programme, afficher un message de succès ou d'erreur.

```
const mysql = require('mysql');

const con = mysql.createConnection({
   host: "localhost",
   user: "nom_utilisateur",
   password: "mot_de_passe_utilisateur"
   database : "mabdd"
 });
data={ id :"15", nom : " Jack", prénom : " Peterson", id_cours : "
```

```
1"}

  con.connect(function(err) {
   if (err) throw err;
   console.log("Connecté à la base de données MySQL!");
   con.query("INSERT INTO Etudiants SET ? ;",data, function (err,
result) {
       if (err) throw err;
       console.log(« insertion réussie »);
     });
 });
```

4) Ecrire un programme NodeJS qui permet de supprimer un cours à la fin du programme, afficher un message de succès ou d'erreur.

```
const mysql = require('mysql');

const con = mysql.createConnection({
   host: "localhost",
   user: "nom_utilisateur",
   password: "mot_de_passe_utilisateur"
   database : "mabdd"
 });

  con.connect(function(err) {
   if (err) throw err;
   console.log("Connecté à la base de données MySQL!");
   con.query("DELETE FROM Cours WHERE id_cours=3 ;" function (err,
result) {
       if (err) throw err;
       console.log(« suppression réussie »);
     });
 });
```

Exercice 2:

1) Créer la table employee(id, nom, prénom, désignation)
2) Insèrer quelques lignes dans la table employee.

```
table const express = require("express");
const mysql = require("mysql");
// Create connection
const db = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "simplilearn",
  database: "nodemysql",
});
db.connect((err) => {

  if (err) {
    throw err;
  }

  console.log("MySql Connected");
```

```
});
// Create table

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

  let sql =

    "CREATE TABLE employee(id int AUTO_INCREMENT, name
VARCHAR(255),firstname VARCHAR(255),designation VARCHAR(255), PRIMARY
KEY(id))";

  db.query(sql, (err) => {

    if (err) {

      throw err;

    }

    res.send("Employee table created");

  });

});
// Insert employee 1

// Insert employee 1

app.get("/employee1", (req, res) => {

  let post = { name: "Jake Smith", designation: "Chief Executive Officer"
};

  let sql = "INSERT INTO employee SET ?";

  let query = db.query(sql, post, (err) => {

    if (err) {

      throw err;

    }

    res.send("Employee 1 added");

  });

});
```

3) Ecrire un programme qui permet de mettre à jour le champ désignation de la table employee en récupérant l'identifiant de la requête http.

```
// Update employee

app.get("/updateemployee/:id", (req, res) => {

  let newName = "Updated name";

  let sql = `UPDATE employee SET name = '${newName}' WHERE id =
```

```
${req.params.id}`;

  let query = db.query(sql, (err) => {

    if (err) {

      throw err;

    }

    res.send("Post updated...");

  });

});
```

4) Ecrire un programme qui permet de supprimer un employée dont l'identifiant est inclus dans la requête http.

```
// Delete employee

app.get("/deleteemployee/:id", (req, res) => {

  let sql = `DELETE FROM employee WHERE id = ${req.params.id}`;

  let query = db.query(sql, (err) => {

    if (err) {

      throw err;

    }

    res.send("Employee deleted");

  });

});

app.listen("3000", () => {

  console.log("Server started on port 3000");

});

In the end, we set the server to listen at Port 3000.
app.listen("3000", () => {

  console.log("Server started on port 3000");

});
```