PRACTICAL TEST 3
Nodejs & PostgreSQL

- **Due Sunday July 08 11.59pm on Blackboard**
- **Demo your app on Friday 13th in the lab**

PostgreSQL

- open source Object-relational database,
- Supports JSON datatypes
- Can define your own datatypes so its a good choice to use in a project.
- When Nodejs is used as backend a PostgreSQL database driver is needed to connect with NodeJS.
- node-postgres [pg] is a great PostgreSQL database connection driver
- You may connect PostgreSQL with NodeJS with **stored procedure** and **connection pooling**.

REQUIREMENTS PRACTICAL TEST 3

i.   Use a higher version of PostgreSQL either 9.6 or higher
ii. Do **NOT** use PostgreSQL version 6.0.1
iii. **Modify the provided web application codes** to work with the higher version of PostgreSQL in (ii) above.
iv. Create two versions of your app:
   a) Local version will run locally by connecting to the locally installed PostgreSQL 9.6 or higher   **[4 marks]**
   b) Heroku version will run on Heroku by connecting to the provisioned database.
                [4 marks]

WHAT YOU WILL SUBMIT

v.   Submit by Sunday July 8th 11.59pm, the local version app and Heroku version app in one .zip folder. That is: put the local app folder and the Heroku app folder in one .zip folder which you submit on Blackboard.
vi. **Paste Heroku version app URL** in the Comments Section of the submissions page – same as you do in assignments. (a missing URL will attract a –2 penalty.
vii. **Demo your app on Friday 13th in the lab. [2 marks]**

Preparation

- Download and install PostgreSQL 9.6 or higher.
- Download and install pgAdmin 4 (UI management tool for PostgreSQL)
- Provision Heroku postgres database (see instructions in week 7)
- Nodejs and npm (node package manager)

Sources of Help

Week 7 learning content: **web322.ca**

Blackboard content plus your researched online sources

Getting Started:

Folder Structure

```
1 |-- node_modules
2 |-- package.json
3 |-- server.js
```

See the package.json file.

```
 {
1   "name": "node-pg",
2   "author": "Javascript Employee",
3   "dependencies": {
4     "express": "^4.14.0",
5     "pg": "^6.0.1"
6   }
7 }
8
```

Create table in Postgresql

Now go to pgAdmin and use postgresql database and create a dummy table `Employee`.

```
1) CREATE TABLE Employee(
2) empid int not null,
3) name text not null,
4) deptnumber int not null
   );
```

Insert some dummy record into the table so later we will fetch through nodejs application.

```
1  INSERT INTO Employee values(1,'Billy John',10001);
2  INSERT INTO Employee values(2,'Smith George',10002);
3  INSERT INTO Employee values(3,'Ernest Cook',10003);
4  INSERT INTO Employee values(4,'Marshall Ballard',10004);
5  INSERT INTO Employee values(5,'Joann Riley',10005);
6  INSERT INTO Employee values(6,'Pearl Pearson',10006);
```

We have created a table and inserted some records . Now we are going to create our nodejs app using command

```
1 npm init
```

Install nodejs postgresql driver [pg]

Install expressjs and postgresql nodejs driver [pg] in our nodejs application using command

```
1 npm install express pg --save
```

this will install modules and write into our package.json file. Now we are going to use pg module to connect with database before that we have to create pg connection string. A pg database connection string made of with this format.

```
1 postgres://dbusername:passwrod@server:port/database
```


you can change your pg connection string as per your configuration.

here is my server.js file using pg module.

```
1   var express = require('express');
2   var pg = require("pg");
3   var app = express();
4
5   var connectionString = "postgres://postgres:123@localhost:5432/postgres";
6
7   app.get('/', function (req, res, next) {
8       pg.connect(connectionString,function(err,client,done) {
9           if(err){
10              console.log("not able to get connection "+ err);
11              res.status(400).send(err);
12          }
13          client.query('SELECT * FROM Employee whereempid= $1',
14   [1],function(err,result)      {
15              done(); // closing the connection;
16              if(err){
17                  console.log(err);
18                  res.status(400).send(err);
19              }
20              res.status(200).send(result.rows);
21          });
22      });
23  });
24
25  app.listen(3000, function () {
26      console.log('Server is running.. on Port 3000');
    });
```

We are using pg.connect method and passing connection string to get a connection, in
callback. catch error if any otherwise call postgresql query using client.query method, in this
method you can also pass parameter to query. run server.js file and go to
url http://localhost:3000/ , you will see Employee withempid= 1


```
[{"id":1,"name":"Billy John","rollnumber":10001}]
```

Stored Procedure in PostgreSQL

Create a simple stored procedure name GetAllEmployee to fetch all Employee record and
execute in pgAdmin

```
 1 -- Get All Record from Employee Table
 2 CREATE OR REPLACE FUNCTION GetAllEmployee()
 3 RETURNS setof Employee AS
 4 $BODY$
 5 BEGIN
 6 RETURN QUERY
 7 select * from Employee;
 8 END;
 9 $BODY$
10 LANGUAGE plpgsql;
```

Now add one new endpoint /sp and call stored procedure with pg module in our server.js file

```
 1 app.get('/sp', function (req, res, next) {
 2     pg.connect(connectionString,function(err,client,done) {
 3        if(err){
 4            console.log("not able to get connection "+ err);
 5            res.status(400).send(err);
 6        }
 7        client.query('SELECT * from GetAllEmployee()' ,function(err,result) {
 8            done(); // closing the connection;
 9            if(err){
10                console.log(err);
11                res.status(400).send(err);
12            }
13            res.status(200).send(result.rows);
14        });
15     });
16 });
```

add this in server.js, run node application and check url http://localhost:3000/sp , you will see all Employee records in browser.
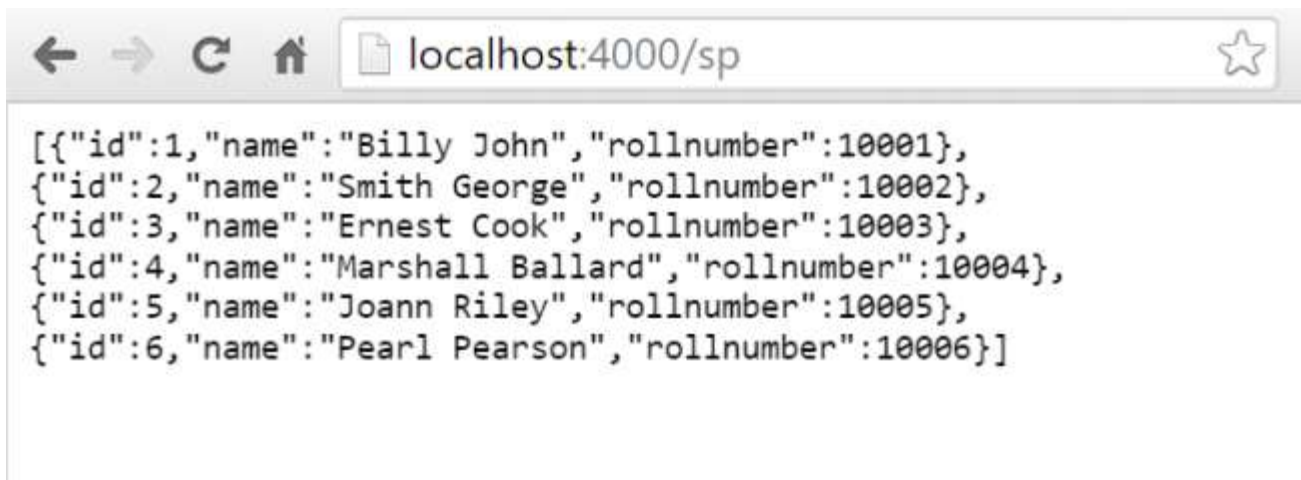
Till here, you are able to use postgresql query and stored procedure through nodejs application but if you are running a web server where you are getting 100 requests at a time then this code will open 100 connection to your postgresql database and slowly it will start to throw no memory error because of so many connections open at a time, to overcome this situation use connection pool, this will create connection but will not close till some

threshold and reuse for further requests. I will show how to use connection pooling in postgresql with nodejs.

Connection Pooling in PostgreSQL

create a new config object for connection pool like this

```
1  var config = {
2    user: 'postgres',
3    database: 'postgres',
4    password: '123',
5    port: 5432,
6    max: 10, // max number of connection can be open to database
7    idleTimeoutMillis: 30000, // how long a client is allowed to remain idle
   before being closed
8  };
```

now we will call pg.pool method to create connection pool using

```
1  var pool = new pg.Pool(config);
```

now add new endpoint /pool , and use pool variable to connect with database

```
1  app.get('/pool', function (req, res) {
2     pool.connect(function(err,client,done) {
3        if(err){
4            console.log("not able to get connection "+ err);
5            res.status(400).send(err);
6        }
7        client.query('SELECT * from GetAllEmployee()' ,function(err,result)
8  {
9            //call `done()` to release the client back to the pool
10           done();
11           if(err){
12               console.log(err);
13               res.status(400).send(err);
14           }
15           res.status(200).send(result.rows);
```

```
16            });
17        });
    });
```

Head to the browser and type

 http://localost:3000/pool, Enter

What do your get? Hopefully you'll get all Employees records.

Show it to your Professor