INSTITUTE OF REMOTE SENSING
ANNA UNIVERSITY, CHENNAI - 600 025.

LAB MANUAL
CUSTOMIZATION LABORATORY

TABLE OF CONTENTS

Exp. No. 1	CREATION OF SPATIAL DATA USING POSTGRESQL USING POSTGIS EXTENSION	Date

AIM

Create user/role, table view, spatial data and query using SQL on postgresql.

SOFTWARE USED

- Command prompt - psql
- QGIS

PROCEDURE

- Create a user account in psql with your roll_no and default password

  Create User/Role

  #create user g21nnn with password 'secret';

  Assign database to user

  #create database g21nnn owner g21nnn;

  GRANT

  #grant all on database g21nnn to g21nnn;

- Creating Postgis extensions

  # \connect g21nnn
  # create extension postgis;
  # create extension postgis_topology;

- Connect the PostgreSQL and Change your Password.

  #psql –h 192.168.5.2 –U g21nnn –p 5432

- The password can be changed using: *\password;*

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

An Instance of Sailors

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

An Instance of Reserves

bid	*bname*	*color*
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

An Instance of Boats

- **Using Create Table command, Create Three tables, namely Sailor, Reserves, and Boats.**

```
create table sailors( sid integer, sname varchar(30),
        rating integer,
        age real,
        primary key(sid)
        );

create table boats(
        bid integer,
        bname varchar(30),
        color varchar(10),
        primary key(bid)
        );

create table reserves(
        sid integer,
        bid integer,
        day date,
        primary key(sid,bid,day),
        foreign key(sid) references sailors,
        foreign key(bid) references boats
        );
```

- **Using insert command, insert all values into the 3 tables.**

```
insert into sailors (sid, sname, rating, age)
```

```
        values (71, 'Zorba', 10, 16.0);
insert into sailors (sid, sname, rating, age)
        values (74, 'Horatio', 9, 35.0);

insert into boats (bid, bname, color)
        values (101, 'Interlake', 'blue');
insert into boats (bid, bname, color)
        values (102, 'Interlake', 'red');

insert into reserves (sid, bid, day)
        values (22, 101, '10/10/98');
insert into reserves (sid, bid, day)
        values (22, 102, '10/10/98');
```

- Perform the following SQL queries:
  1. Find the name and age of all sailors
  2. Find all sailors with rating above 7
  3. Find the names of sailors who have reserved boat no.103
  4. Find SID of sailor who have reserved a red boat
  5. Find names of sailors who have reserved a red boat
  6. Find colours of boat reserved by Lubber
  7. Find names of sailors who have reserved atleast one boat
  8. Find the age of sailors whose name begin and end with B and has 3 characters
  9. Find the names of sailors who have reserved a red or green boat
  10. Find the names of sailors who have reserved both red and green boat

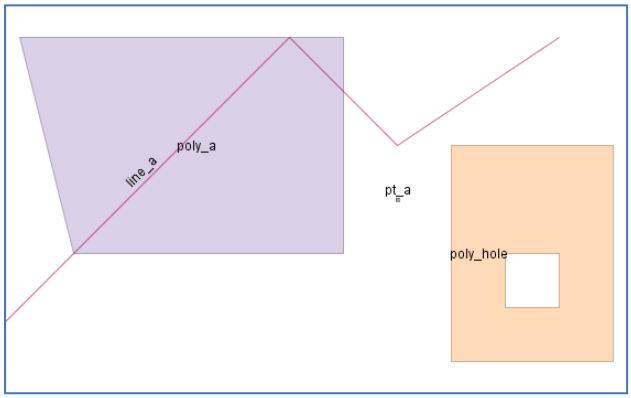
- **Spatial data creation:**

  The spatial data can be stored on the database by creating a new schema and store the data as tables for point, line and polygon.

- **Create Schema**

```
create schema spa1;
```

We will learn to create spatial data as postgresql table. The following fig contains basic geometry features poit, line and polygon. We now create these in postgresql table



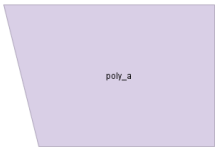
Procedure

(i) Polygon

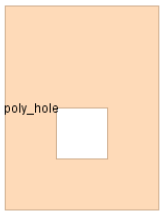
- **Create Table**

```
create table spa1.sample_poly( pid integer primary key,
pname varchar(10),
the_geom geometry('POLYGON'));
```

- **Insert Table**




```
insert into spa1.sample_poly(pid,pname,the_geom)
values (1,'poly_a','POLYGON((3 3, 8 3, 8 7, 2 7, 3 3))' );
```




```
insert into spa1.sample_poly(pid,pname,the_geom)
values (2,'poly_hole','POLYGON((10 1, 13 1, 13 5, 10 5, 10 1),
(11 3, 12 3, 12 2, 11 2, 11 3))');
```

- **Create index**

```
CREATE INDEX idx_sample_poly_geom ON spa1.sample_poly
USING gist (the_geom);
```

- **Queries on spatial table**

```
select pname, ST_AsText(the_geom) from spa1.sample_poly;

select * from geometry_columns where f_table_name='sample_poly';

SELECT pname, ST_GeometryType(the_geom), ST_NDims(the_geom),
ST_SRID(the_geom)  FROM spa1.sample_poly;
```

(ii) Line

- **Create Table**

```
create table spa1.sample_line( lid integer primary key,
                    lname varchar(10),
                    the_geom geometry(LINESTRING));
```

- **Insert Table**

```
insert into spa1.sample_line(lid,lname,the_geom) values
(1,'line_a','LINESTRING(1 1, 7 7, 9 5, 12 7)');
```

- **Create index**

```
CREATE INDEX idx_sample_line_geom ON spa1.sample_line USING
gist (the_geom);
```

- **Queries on spatial table**

```
select lname, ST_AsText(the_geom) from spa1.sample_line;

select * from geometry_columns where f_table_name='sample_line';

SELECT lname, ST_GeometryType(the_geom), ST_NDims(the_geom),
ST_SRID(the_geom)
FROM spa1.sample_line;
```

(iii) Point

- **Create Table**

```
create table spa1.sample_point( ptid integer primary key, ptname varchar(10),the_geom
geometry(POINT));
```

- **Insert Table**

```
insert into spa1.sample_point(ptid,ptname,the_geom) values
(1,'pt_a','POINT(9 4)');
```

- **Create index**

```
CREATE INDEX idx_sample_point_geom ON spa1.sample_point USING
gist (the_geom);
```

- **Queries on spatial table**

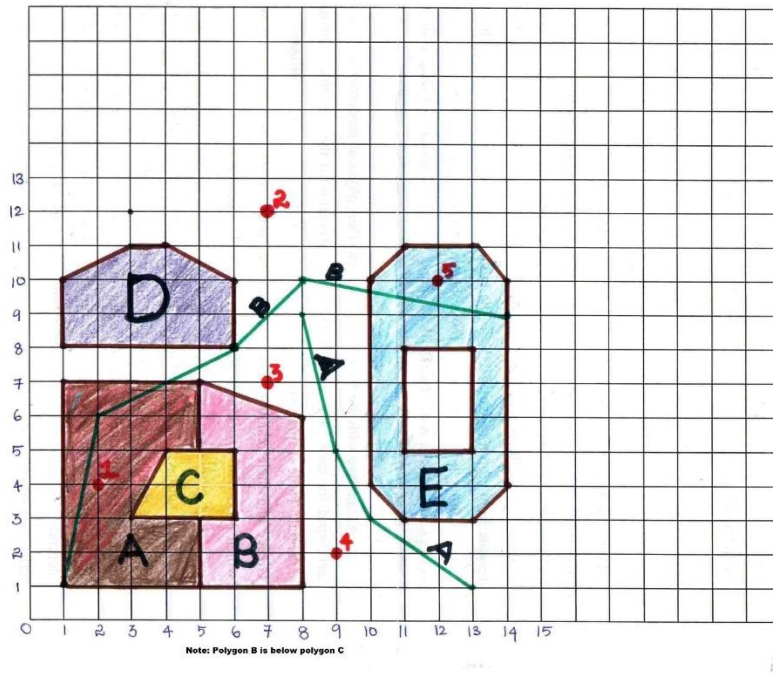
```
select ptname, ST_AsText(the_geom) from spa1.sample_point;

select * from geometry_columns where f_table_name='sample_point';

SELECT ptname, ST_GeometryType(the_geom), ST_NDims(the_geom),
ST_SRID(the_geom)
FROM spa1.sample_point;
```

- **Viewing on QGIS**.

Creating new postgis connection in QGIS:

- Open QGIS software.
- In the browser tab, select 'PostgreSQL' and right-click it.
- Click 'New Connection.' A dialog box appears.
- Fill the database name (e.g., roll_no), host as 192.168.56.2, and provide a name for it (e.g., your roll_no).
- In Authentication, change to 'Basic.'
- Enter the username and password for the PostgreSQL database. Click on 'Test Connection' and click OK.
- Now the Postgis database appears below the PostgreSQL on the browser tab.
- Open the schema on which the spatial data are created.
- Open all the layers.
- Take a screenshot of the final displayed layers and save it on your folder.

Exercise:
Create my_poly,my_line and my_pont from the following fig



Exp No.2	IMPORT AND EXPORT ON POSTGIS	Date

AIM

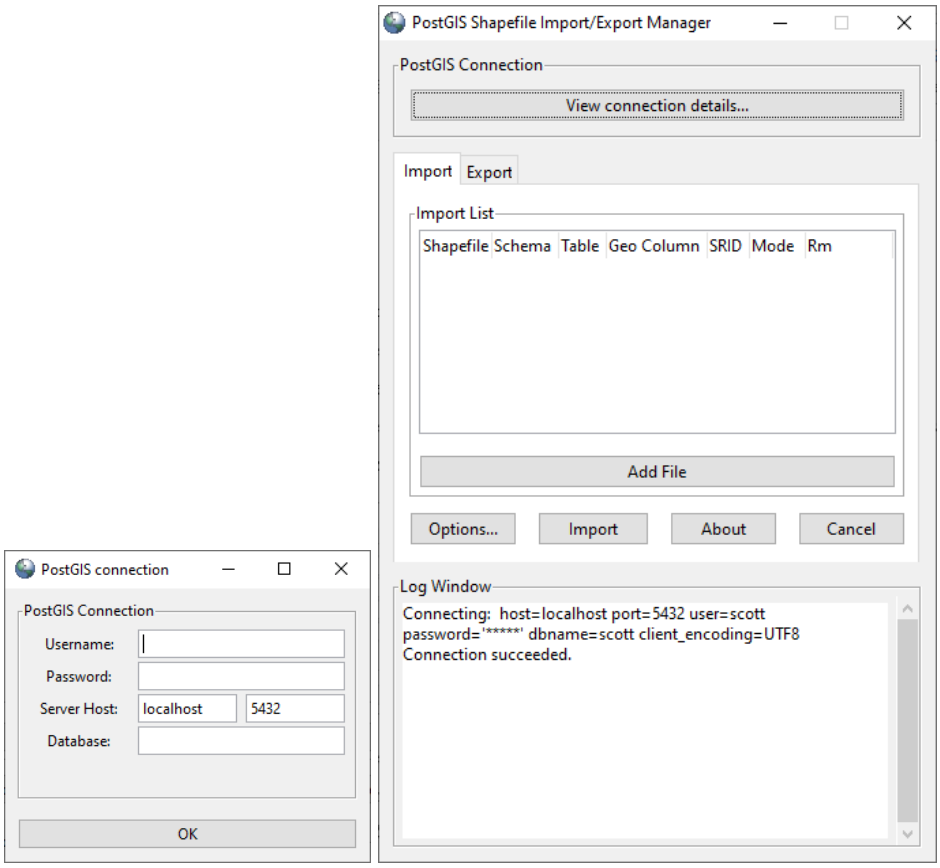
To import and export data on PostGIS.

SOFTWARE USED

QGIS

PROCEDURE

- Create a new folder ‘geo7’ in any drive other than the 'C' drive.
- Copy the files from the share folder \\192.168.5.3\g7data and paste them into the folder created.
- From the copied files, open 'Postgisgui.' folder In that file, open 'shp2pgsql-gui', a dialog box appears.



- Click view connection details and enter the PostGIS connection username, and password and click OK.

- Follow the steps in order to import files:
    - ❖ Select import tab (to send .shape file to postgis, Export is for copy from postgis)
    - ❖ Click 'Add File,' and add all four files from the DIN_GCS.
    - ❖ Change the geometry column value to the_geom and set SRID as 4326.
    - ❖ Click 'OK.' All files are imported.
- Now open QGIS. In the browser tab, select the 'PostgreSQL,' right-click, and select 'Add New Connection.'
- Enter the host as 192.168.5.2, username, and password, and click 'Test Connection,' then click 'OK.'
- Add all the four layers imported in QGIS.
- Right-click each layer and click 'Export - Save Feature As.' A dialog box appears; Set the format as 'ESRI Shapefile,' and in the folder, enter the location and name for the layer where it has to be saved. Then click 'OK.'
- Repeat the steps to export the other three layers as well.

Exp. No. 3	PUBLISHING SHAPEFILE ON GEOSERVER	Date

AIM

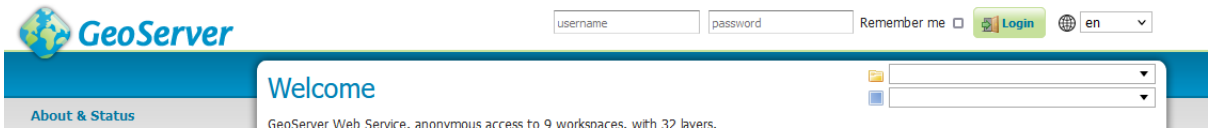
To publish shapefile on the geoserver.

SOFTWARE USED

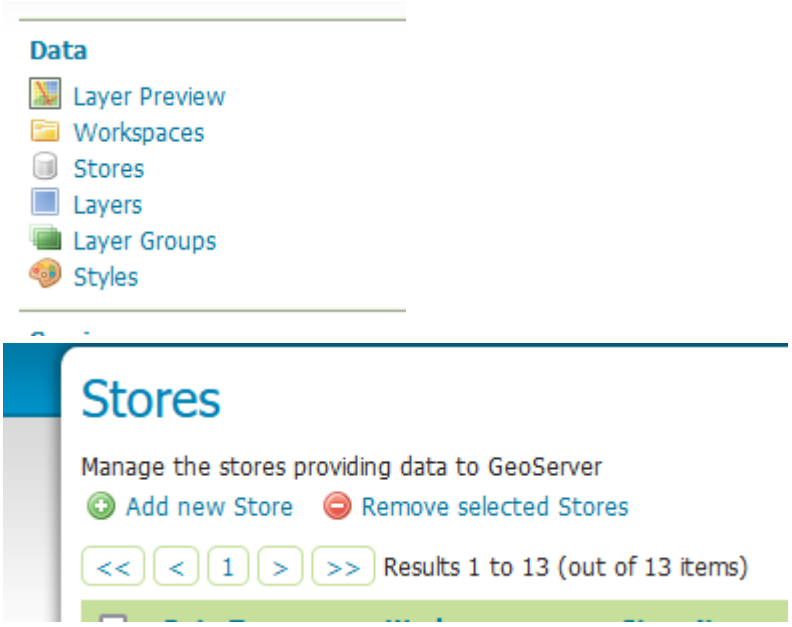
Geoserver

PROCEDURE

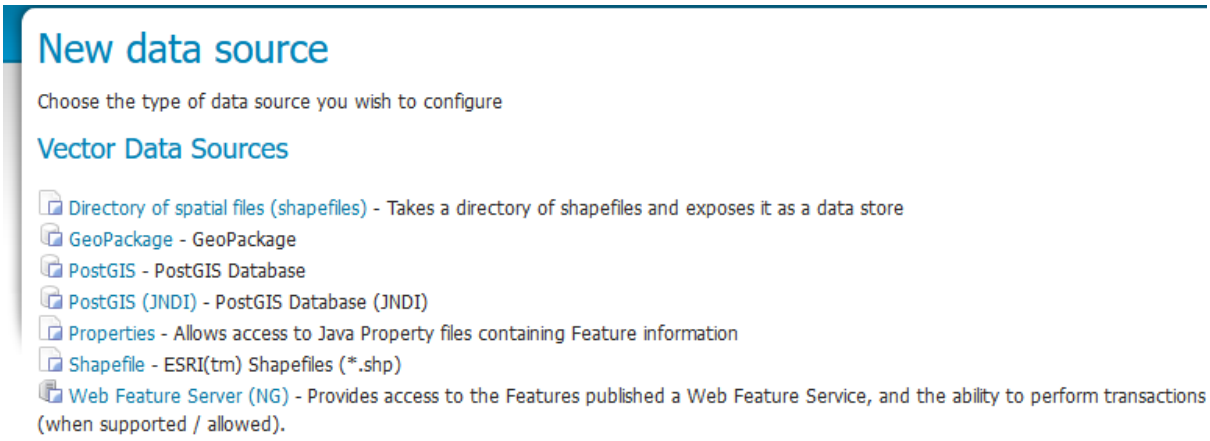
- Open the URL "*198.168.5.2:8080/geoserver/web*"in a browser. This opens the geoserver page.
- click on "Login" and enter the username and password which is the same as that of PostgreSQL.



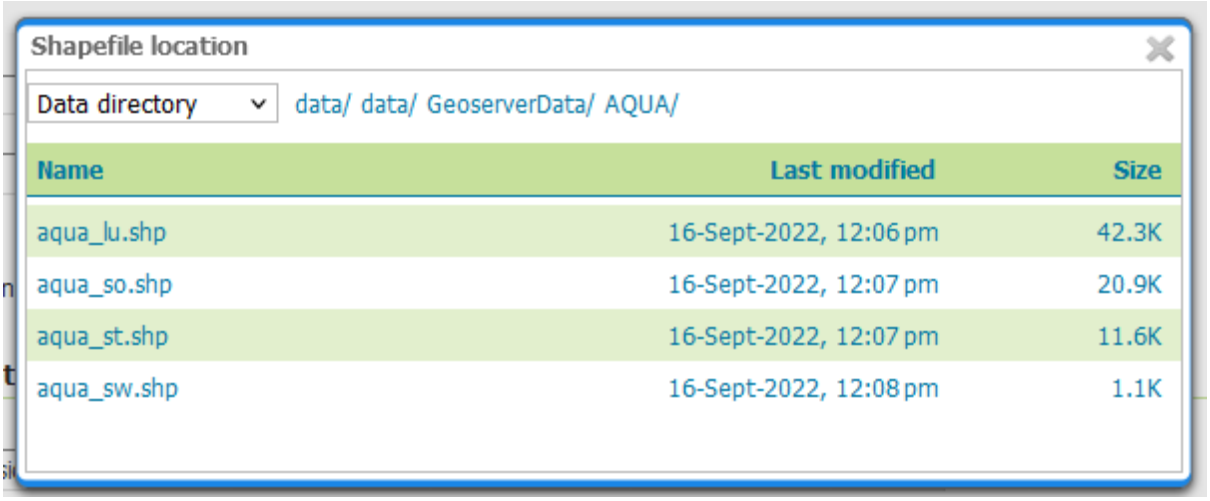
- In the Data, tab, select "Layer preview" which shows the all layers created previously.
- In the Data tab. Go to stores and click "Add new stores".



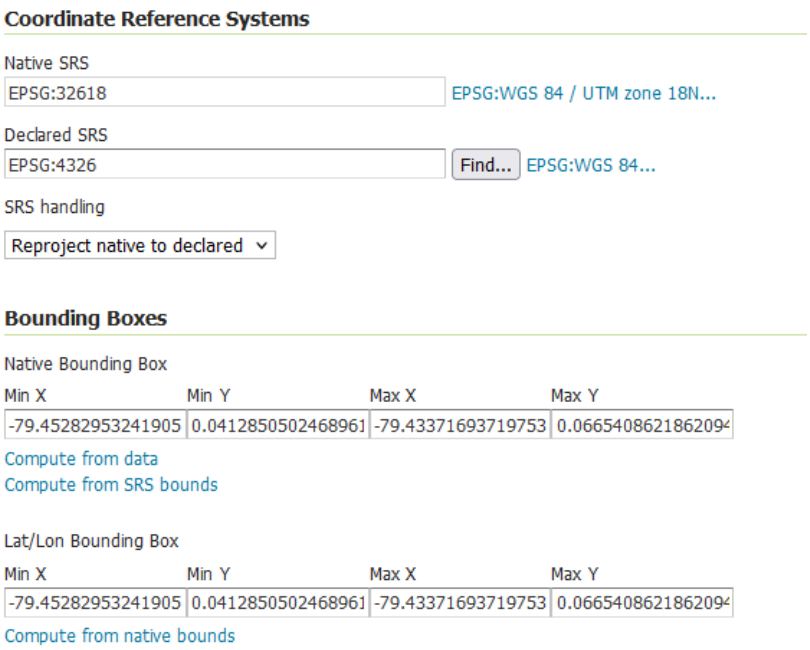
- From the appearing Vector data source. select “Add new shapefile” A new vector source dialog box appears.



- Choose the workspace, Data source name (eg: Scott 24), Browse for the shapefile in the data directory (Data/data/geoserver data/Aqua) in connection parameter. Click Save



- Choose layer - > add new layer, choose WS then layer
- Then click publish button, another dialog. Box appears. Provide a Meaning Title.



- Set the coordinate reference system as WGS UTM-18N" change the declared SRS to 4326.
- Set the srs handling as " reproject native to declared"
- click on "Compute from data" for native bounding and Compute from native bounds for lat/long bounding box.
- Once done, click Save.
- Repeat the above steps for all the four layers in Aqua and for three layers in Mulb.
- Now, again in the Data tab, Go to "Layer groups".
- Click on "Add new layer group" provide name and title for the layer group.
- Using the "Add layer" option, add all the required layer click on "Generate Bound" and click "save". Thus, Group layer is created.
- In Similar way, Create layer group for Mulb and Aqua.
- Once the layer group is created go to workspace in data tab and select the Layer Created to view it.

Exp. No.4	PUBLISHING SHAPEFILE FROM POSTGIS ON GEOSERVER	Date

AIM

To publish shapefile from the postgis on the Geoserver.

SOFTWARE USED

QG1S

PROCEDURE

1) CHANGING PROJECTION:

- Open the share folder and Copy the "Geoserver data" and paste it in your folder.
- In the given Geoserver data there are three data namely Aqua, mulb, thiru. In which Aqua has a wrong projection.
- Now open the QGIS software in the browser tab itself navigate to your Aqua data folder.
- Now right click one layer at a time and select "export layer" option then go to file option, a dialog box appears.
- In the dialog box change the CRS to "EPSG:42 WGS84" and in the file name give a name and the folder where output has to be saved. Then click OK.
- Repeat the above step for all the layers or file present in the Aqua folder.

2) CREATE NEW SCHEME ON POSTGIS:

- In the QGIS browser tab, go to the PostgreSQL.
- Open the postgres database created already.
- Now right click the database and then click "new schema".
- Give a name to the schema and then click OK.

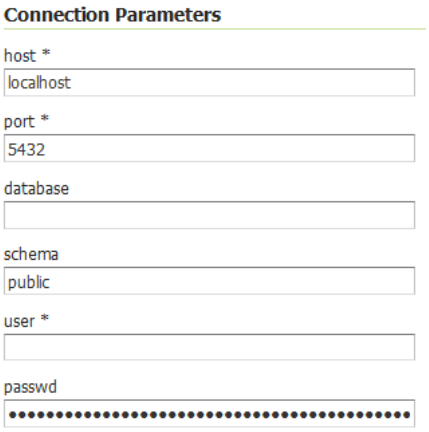
3) ADD SHAPE FILE TO THE SCHEMA:

- Add all the shapefile in QGIS (Aqua, Mulb, Thiru) it will appear in the layer tab.
- Keep the new schema created on the browser tab.

- Drag the shape file from the layer tab to the new schema in the browser tab Thus all the shapefile are added to the schema.

4) PUBLISHING ON GEOSERVER:

- In the browser enter the URL "*192.168.5.2:8080/geoserver/web*" .
- Enter the username and password which is the same as that of PSQL.
- Go to the stores "add new store".
- From the vector data source click "postgis".



- Give a name to it. In the username enter the database (eg: roll_no) and enter the schema as (ex4) where all the layers are saved. Enter the password and click save
- Now it shows all the shapefile in the schema with an option "publish".
- Now we can publish the layer one by one click "publish" for the first layer. A dialog box appears.
- Provide a meaningful title, set the SRS handling "native to be declared" and ensure whether the coordinates reference system is on 4326.
- Click "compute from data" for native bounding box and "compute from native bounds" for lat/long bounding box. Click save on the bottom
- Now to publish the second layer, go to the layer tab. In the set "select layer from" to postgis store name (eg: ex4pg)
- Now it shows all the layer available in the schema. we can publish the next layer by clicking "publish".
- Repeat the above process and publish all the shapefile in the schema.

Exp No.5	GEOSERVER STYLE	Date

AIM

To apply styles to the previously published layers on the geoserver.

SOFTWARE USED

Open Layer Jump

QGIS

PROCEDURE

OPEN JUMP

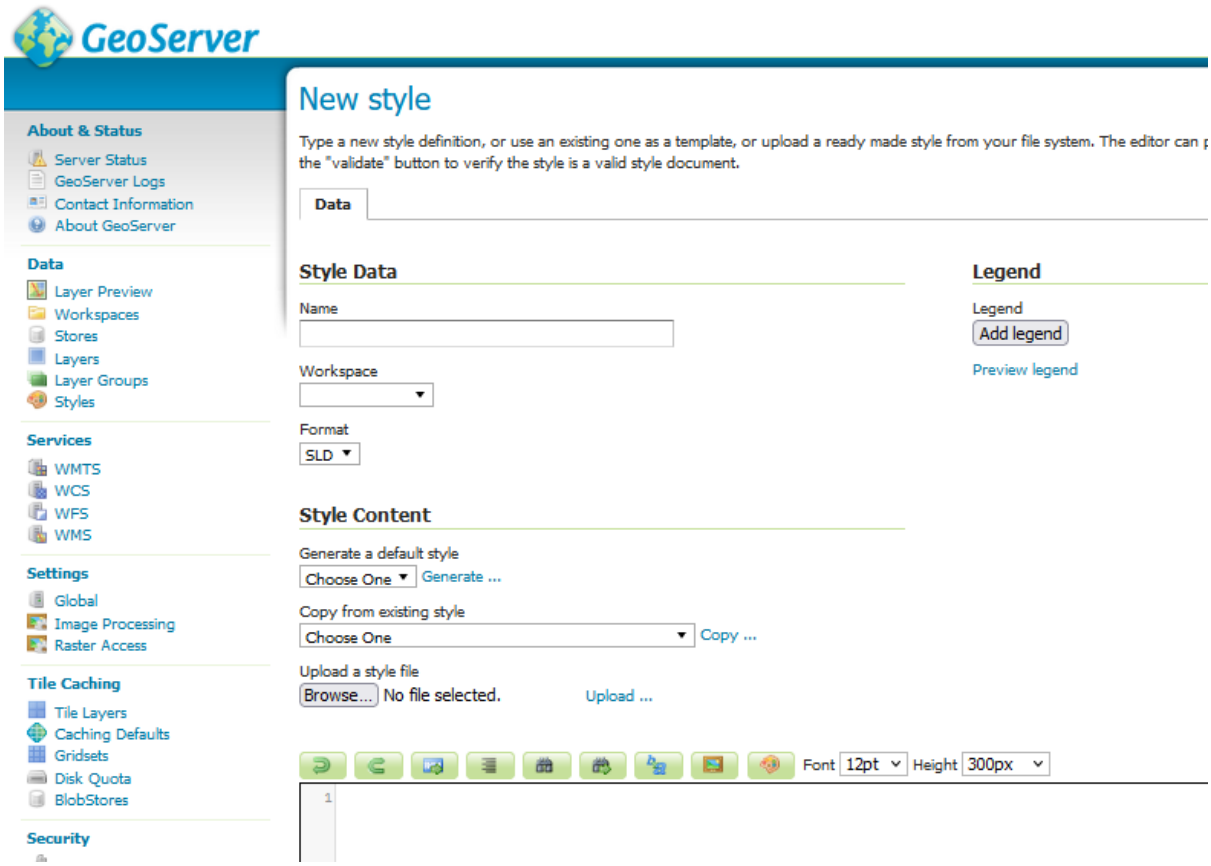
- In the webbrowser, emter the url "*192.168.5.2:8080/geoserver/web*"
- Enter the username and password to login.
- Open "open jump". Right click and add the data.
- Right click the layer added and select style. Then change style.
- The color ramp is applied to the shapefile. Similarly 'label' is clicked and applied.
- Once completed, right click select style, then export layer.
- Set the type as "spatial data descriptor".
- In "sld parameter" window change the geometry, as "the_geom",and click OK. The sld file is saved.

QGIS

- Open the QGIS software. Add the required layers.
- Right click the required layer select "properties". Now click "symbology".
- At the top, select "categorized". Go to labels and let it be displayed.
- Once style is completed. At the bottom click "style" button
- Select "save style". Save the style as ".sld".

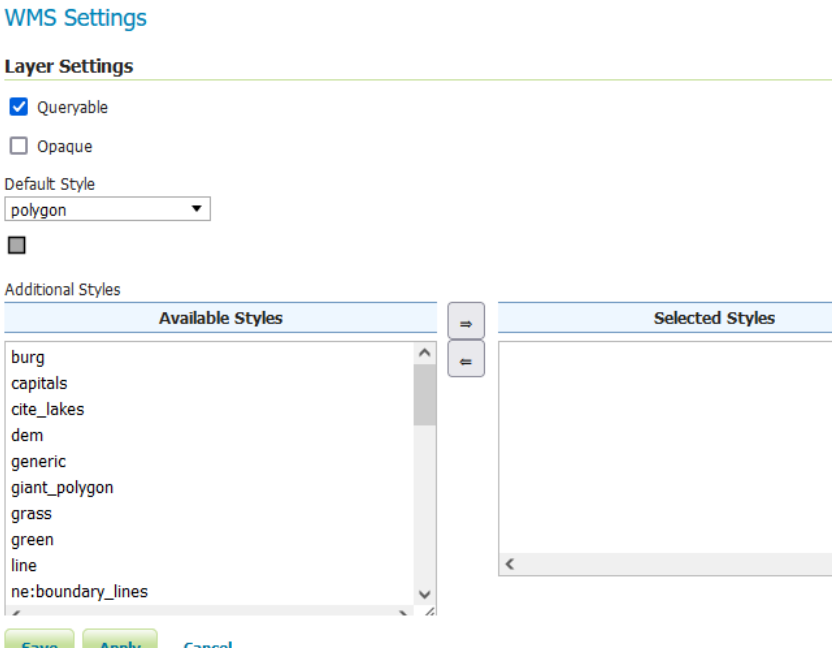
APPLYING IN GEOSERVER

- In geoserver click “style” at the left under Data tab.



- Click “new style”. Name the style with the layer’s name for which you are going to apply the color.
- Click “upload a style” and upload the sld file (which is generated either from open jump or QGIS). Click upload.
- Click validate as you scroll down. Then “save and apply”.
- As you go to “layers” click “add new layers” and select “roll_no-pg” store. Click on any one of the files and publish again.
- Change its name as layer’s name has to be unique. In the SRS handling set “reproject native to declared” and in bounding boxes, click “compute from native bound”.

- Go to ‘publishing tab’. In default style, choose the style you created for the layer and below that click on the style you created.



- Go to layer preview and search the styled layer and click on ‘open layer’ and the styled map is displayed.
- Perform this step for all the files and publish it with styles.

Exp No.6	HTML AND CSS	Date

AIM

To create and style a webpage using various HTML and CSS tags.

SOFTWARE USED

Visual studio

PROCEDURE

- create a html page that contains the biodata with external css.
- create a html webpage that replicates the observation note index page and while selecting the experiment name, it should take to another webpage that contains the experiment description and from that webpage, it should return back to index page.
- Create a webpage that contain all the html and css tags along with its description used for creating the above webpages
- Must use tag with CSS position , display properties in any one of the above

RESULT

Hence the biodata, index page and tags page has been created using html and css

Exp. No.7	JAVA SCRIPT	Date

AIM

To learn and Perform simple Javascript programs.

SOFTWARE USED

Visual studio

PROCEDURE

- In the web browser open the url '*192.168.5.2\JS'*, it opens the javascript code pdf.
- Open the Visual Studio code, type the given code one by one in a new text file.
- Save the file as html files with '.html' as extension and javascript files as '.js' in a new folder (EXP 7).
- Once all the code are typed, the output of the code can be viewed by right clicking the code on visual studio code and select "open in default browser" or "open in live stream".
- It will open the output in a new webpage.
- Once all the code are done, share the folder.

Demo programs

- To make the Background color of the Page as Green.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8" />
   <title>Example 1</title>
</head>
<body bgcolor="white">
   <p> Paragaraph 1</p>
   <script>
      document.bgColor = "green";
   </script>
</body>
</html>
```

- To make a Paragraph with limit and alert ‘script blocks’ if the letter limit of the paragraph exceeds and create a new.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8" />
   <title> Example 2</title>
</head>
<body bgcolor="white">
   <p> Paragraph 1</p>
   <script>
      //script block 1
      alert("First Script Block");
   </script>
   <p>Paragraph 2</p>
   <script>
      //script block 2
      alert ("Second Script Block");
   </script>
   <p> Paragraph 3</p>
</body>
</html>
```

- To Print the text using document.getElementById

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8" />
   <title>Example 3</title>
</head>
<body>
   <p id="results"></p>
   <script>
      document.getElementById("results").innerHTML="Hello World";
   </script>
</body>
</html>
```

- Usage of var and variables.

```
<html lang="en">
<head>
   <meta charset="utf-8" />
   <title> Example 4</title>
</head>
<body>
   <script>
      var myFirstVariable;
      myFirstVariable = "Hello";
```

```
            alert(myFirstVariable);
            myFirstVariable = 54321;
            alert(myFirstVariable);
        </script>
    </body>
```

- Arithmetic expression example by Celsius to Fahrenheit Temperature Conversion.

```
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title> Example 5</title>
</head>
<body>
  <script>
    // Equation is C=5/9 (F-32)
    var degFahren=prompt("Enter the degrees in Fahrenheit",50);
    var degCent;
    degCent=5/9*(degFahren-32);
    alert(degCent);
    //alternative more friendly
    alert(degFahren + "\xB0 Fahrenheit is " + degCent + "\xB0 centigrade");
  </script>
</body>
</html>
```

- User input

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title> Example 6</title>
</head>
<body>
  <script>
    var greetingString = "Hello";
    var myName = prompt("Please enter your name", "");
    var concatString;

    document.write(greetingString + " " + myName + "<br/>");
    concatString = greetingString + " " + myName;
    document.write(concatString);
  </script>
</body>
</html>
```

- String and number conversion parseInt, parsefloat

```
<!DOCTYPE html>
<html lang="en">

<head>
```

```
    <meta charset="utf-8" />
    <title> Example 7</title>
</head>
<body>
    <script>
        var myString = "56.02 degrees centrigrade";
        var myInt;
        var myFloat;
        document.write("\"" + myString + "\" is " + parseInt(myString, 10) + " as an integer" +
"<br/>");
        myInt = parseInt(myString, 10);
        document.write("\"" + myString +
            "\" when converted to an integer equals " + myInt + "<br/>");
        myFloat = parseFloat(myString);
        document.write("\"" + myString +
            "\" when converted to a floating point number equals " + myFloat);
    </script>
</body>
</html>
```

- Get two numbers from user input and add them.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Example 8</title>
</head>
<body>
    <script>
        var firstNumber = prompt("Enter the first number", "");
        var secondNumber = prompt("Enter the second number", "");
        var theTotal = firstNumber + secondNumber;
        document.write(firstNumber + " added to " + secondNumber +
            " equals " + theTotal);
        // use parsint
        document.write("<br/>");
        var theTotal1 = parseInt(firstNumber) + parseInt(secondNumber);
        document.write("After conversion " + firstNumber + " added to " + secondNumber +
            " equals " + theTotal1);

    </script>
</body>
</html>
```

- Switch .. case example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Example 9</title>
</head>
<body>
```

```
  <script>
    var secretNumber = prompt("Pick a number between 1 and 5:", "");
    secretNumber = parseInt(secretNumber, 10);
    switch (secretNumber) {
      case 1:
        document.write("Too low!");
        break;
      case 2:
        document.write("Too low!");
        break;
      case 3:
        document.write("You guessed the secret number!");
        break;
      case 4:
      case 5:
        document.write("Too high!");
        break;
      default:
        document.write("You did not enter a number between 1 and 5.");
        break;
    }
    document.write("<br />Execution continues here");
  </script>
</body>
</html>
```

- If ... else .. else example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Example 10</title>
</head>
<body>
  <script>
    var myNumber = parseInt(prompt("Enter the a number", ""), 10);
    if (myNumber < 0) {
      document.write(myNumber + " is Negative ");
    } else if (myNumber == 0) {
      document.write(myNumber + " is Zero");
    } else {
      document.write(myNumber + " is Positive");
    }
  </script>
</body>
</html>
```

- for loop, while loop, do while loop example.

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
   <title>Example 11 Multiplication table</title>
</head>
<body>
   <script>
      var tableNumber;
      var tCount;
      var result;
      var tableNumber = prompt("Enter Table Number", "");
      tableNumber = parseInt(tableNumber, 10);

      document.write("using for loop </br>");

      for (tCount = 1; tCount <= 12; tCount++) {
         result = tCount * tableNumber;

         document.write(tableNumber + " * " + tCount + " = " + result + "</br>");
      }
      document.write("</br/>using while loop </br>");

      tableNumber++;
      tCount = 1;
      while (tCount <= 12) {
         result = tCount * tableNumber;
         document.write(tableNumber + " * " + tCount + " = " + result + "</br>");
         tCount++
      }

            document.write("</br/>using do ..while loop </br>");

      tableNumber++;
      tCount = 1;
      do {
         result = tCount * tableNumber;
         document.write(tableNumber + " * " + tCount + " = " + result + "</br>");
         tCount++
      }while (tCount <= 12)


   </script>
</body>
</html>
```

- Function example.

```
<!DOCTYPE html>
<html lang="en">

<head>
   <title>Example 12</title>
</head>
```

```
<body>
   <script>
      function findBig(a, b) {
         if (a > b) {
            return a;
         } else
            return b;
      }
      var a = prompt("Enter A", "");
      a = parseInt(a, 10);
      var b = parseInt(prompt("Enter B", ""), 10);
      var big = findBig(a, b);
      document.write(" The big is " + big + " <br />");

   </script>
</body>
```

- Change style properties using id

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Example 13</title>
   <style>
      #divAdvert {
         font: 12pt arial;
      }
   </style>
</head>
<body>
   <div id="divGIS">
      IRS GIS Lab
   </div>
   <script>
      var divAdvert = document.getElementById("divGIS");
      divGIS.style.fontStyle = "italic";
      divGIS.style.color="red";
      divGIS.style.textDecoration = "underline";

   </script>
</body>
</html>
```

- Event click example.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Example 14</title>
</head>
<body>
```

```
  <form action="" name="form1">
    <input type="button" name="myButton" value="Button clicked 0 times" />
  </form>
  <script>
    var myButton = document.form1.myButton;
    var numberOfClicks = 0;

    function myButtonClick() {
      numberOfClicks++;
      myButton.value = "Button clicked " + numberOfClicks + " times";
    }
    myButton.addEventListener("click", myButtonClick);
  </script>
</body>
</html>
```

- Event on the mouse go up and down by the user input.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Example 15</title>
</head>

<body>
  <form action="" name="form1">
    <input type="button" name="myButton" value="Mouse goes up" />
  </form>
  <script>
    var myButton = document.form1.myButton;

    function myButtonMouseup() {
      myButton.value = "Mouse Goes Up";
    }

    function myButtonMousedown() {
      myButton.value = "Mouse Goes Down";

    }
    myButton.addEventListener("mousedown", myButtonMousedown);
    myButton.addEventListener("mouseup", myButtonMouseup);
  </script>
</body>
</html>
```

- External script example.

```
<html>
<head>
  <title>Example 16</title>
   <h1>Addition</h1>
   <script src="addition.js">
```

```
   </script>
</head>

<body>
   <p id="input">Enter First Number: <input id="one">
      <br /><br />
      Enter Second Number: <input id="two">
   </p>
   <p><button onclick="fun_addition()">ADD</button></p>
   <p id="res" style="display.none;">
      Addition Result = <input id="addition"><br /><br />
</body>
</html>
```

---

addition.js

```
var numOne,numTwo,res,temp;
function fun_addition()
{
  numOne=parseInt(document.getElementById("one").value);
  numTwo=parseInt(document.getElementById("two").value);
  if(numOne && numTwo)
  {
    temp=document.getElementById("res");
          temp.style.display="block";
          res=numOne + numTwo;
          document.getElementById("addition").value=res;
          }
}
```

Exercise

Standard exercises problems given during lab class

Exp. No 8	WMS	Date

AIM

1. To explore the web mapping services requests.
2. To create an HTML page that contains the URL of all the request.

PROCEDURE

Common WMS requests

GetCapabilities	Retrieves metadata about the service, including supported operations and parameters, and a list of the available layers
GetMap	Retrieves a map image for a specified area and content
GetFeatureInfo	Retrieves the underlying data, including geometry and attribute values, for a pixel location on a map
DescribeLayer	Indicates the WFS or WCS to retrieve additional information about the layer.
GetLegendGraphic	Retrieves a generated legend for a map

Open a web browser and type the following request urls in address one by one and check the results.

Request urls

- **Get Capabilities**

http://localhost:8080/geoserver/ows?
service=wms&
version=1.1.1&
request=GetCapabilities

- **Open Layers**

http://192.168.5.2:8080/geoserver/wms?
service=WMS&
version=1.1.0&
request=GetMap&
layers=scott%3AThiru_lu&
bbox=78.95884322696476%2C10.796515025175719%2C79.17195471507927%2C10.935436224928557&
width=768&
height=500&
srs=EPSG%3A4008&
styles=&

format=application/openlayers

- **PNG**

http://192.168.5.2:8080/geoserver/wms?
service=WMS&
version=1.1.0&
request=GetMap&
layers=scott%3Athiru_lu&
bbox=78.95884322696476%2C10.796515025175719%2C79.17195471507927%2C10.93543
6224928557&
width=768&
height=500&
srs=EPSG%3A4008&
styles=&
format=image/png

- **Feature information in Text format**(GetFeatureInfo) **:**

http://192.168.5.2:8080/geoserver/wms?
service=WMS&
version=1.1.0&
request=GetFeatureInfo&
layers=scott%3AThiru_lu&
bbox=78.95884322696476%2C10.796515025175719%2C79.17195471507927%2C10.93543
6224928557&
width=768&
height=500&
srs=EPSG%3A4008&
styles=&
format=image/png&
QUERY_LAYERS=scott%3AThiru_lu&
INFO_FORMAT=text/plain&
X=350&
Y=250

- **Feature information in format GeoJSON**:

http://192.168.5.2:8080/geoserver/wms?
service=WMS&
version=1.1.0&
request=GetFeatureInfo&
layers=scott%3AThiru_lu&
bbox=78.95884322696476%2C10.796515025175719%2C79.17195471507927%2C10.93543
6224928557&
width=768&
height=500&
srs=EPSG%3A4008&
styles=&
format=image/png&

QUERY_LAYERS=scott%3AThiru_lu&
INFO_FORMAT=application/json&
X=350&
Y=250

- **Feature information in format HTML**

http://192.168.5.2:8080/geoserver/wms?
service=WMS&
version=1.1.0&
request=GetFeatureInfo&
layers=scott%3AThiru_lu&
bbox=78.95884322696476%2C10.796515025175719%2C79.17195471507927%2C10.93543
6224928557&
width=768&
height=500&
srs=EPSG%3A4008&
styles=&
format=image/png&
QUERY_LAYERS=scott%3AThiru_lu&
INFO_FORMAT=text/html&
X=350&
Y=250

- **Describe Layer**

http://192.168.5.2:8080/geoserver/wms?
service=WMS&
version=1.1.1&
request=DescribeLayer&
layers=scott:Thiru_lu &
output_format=application/json

Or
&output_format=text/xml

- Legend

http://192.168.5.2:8080/geoserver/wms?
REQUEST=GetLegendGraphic&
VERSION=1.0.0&
FORMAT=image/png
&WIDTH=20&
HEIGHT=20&
LAYER=scott:Thiru_lu

Exercise

Prepare an html document having all the above request urls as anchor tags under unordered list

Exp No 9	OPEN LAYERS	Date

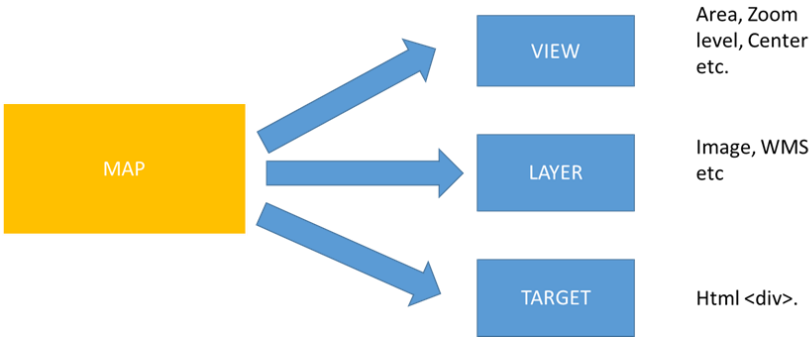
AIM

To learn the concepts of OpenLayers using various sample codes for the concepts like center and zoom, Basemap, WMS, WFS, and digitization using Node.js.

SOFTWARE USED

Visual studio, nodejs

PROCEDURE



Main components of Openlayers javascript API.

The sample map application codes are available at the lan web serever URL *http://192.168.5.2/oljs24*

- Copy the folder 'node_ol_lab' from file server and paste it in the working folder.
- Open this folder on Visual Studio Code.
- Open the browser and type the URL *http://192.168.5.2/oljs24* for the code. Map1 code sample given below. Other 6 maps can be viewed on the server.

Map1: main.js

```
import './style.css';
import {Map, View} from 'ol';
import TileLayer from 'ol/layer/Tile';
import OSM from 'ol/source/OSM';

const map = new Map({
  target: 'map',
  layers: [
    new TileLayer({
      source: new OSM()
    })
```

```
  ],
  view: new View({
    center: [0, 0],
    zoom: 2
  })
});
```

Map1:index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/x-icon" href="https://openlayers.org/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Simple Map</title>
  </head>
  <body>
    <div id="map"></div>
    <script type="module" src="./main.js"></script>
  </body>
</html>
```

Map1:style.css

```
@import "node_modules/ol/ol.css";

html, body {
  margin: 0;
  height: 100%;
}
#map {
  position: absolute;
  top: 0;
  bottom: 0;
  width: 100%;
}
```

Map 1: Simple map showing OSM layer using HTML, CSS and java script

- Create and type the Map1 code on the index.html, main.js, and style.css.

Preview at VS code

- Once the code is ready, open at VS code terminal and type ‘npm start’. Once done, click ‘q’to quit.

Build the application

- To build a distribution type ‘npm run build’ to create a dist file.
- For publishing create a new folder 'Map1**'**, copy the html, js, style.css, and dist files and paste them into the Map1 folder.

Check at VS code HTTP local server

- In order to run dist, type ‘npx http-server dist’ in the VS code terminal.

Check cmd with python

- Copy the dist folder of all map codes and paste it in a folder.
- Open the folder in the path, add **'cmd'** to the beginning of the path to open it in the command prompt.
- Then type:’ *C:\Python27\ArcGIS10.8\python.exe -m SimpleHTTPServer* ‘. This creates a local host for the folder opened.
- Open all the files and copy the URL and paste it in the html code.

Exp. No.10	PYTHON SCRIPTING	Date

AIM

Perform basic operations on the Map Layers in QGIS using python codes (script).

SOFTWARE USED

QGIS

PROCEDURE

- Create a folder (Exp.10) in the working Directory.
- Copy the ‘g7data’ from the 198.168.5.2. labdata directory and paste it into the folder.
- In the browser type the url “*192.168.5.2/Qpy*” for the python code.
- Open the QGIS application.
- Ensure whether pre-processing toolbox is available. Otherwise form the ‘plugin’ toolbox select the pre-processing toolbox.
- Open the pre-processing toolbox, Then open the ‘Python script’ and select the new script.
- In the opened script type the Python code as of in the given python code.
- Change the input and output directories, filenames as required in the code.
- Once done, click ‘ Run Script’ to see the output in the qgis .

QUESTIONS

1. Loading vector layers in QGIS using iface.addVectorLayer() and addMapLayer()

q1.py

```
import sys
os.chdir('D:/g7data')
from qgis.utils import iface
from qgis.core import QgsVectorLayer, QgsProject
layer=iface.addVectorLayer("citiesx020.shp","cities","ogr")
#Defining Layer with QgsVectorLayer
co_layer=QgsVectorLayer("countries_simpl.shp","countries","ogr")
mj_layer=QgsVectorLayer("mjcities.shp","mjcities","ogr")
states_layer=QgsVectorLayer("statesp020.shp","states","ogr")
#Add a list of layer
QgsProject.instance().addMapLayers([co_layer,mj_layer])
#Add a single layer
QgsProject.instance().addMapLayer(states_layer)
```

2. Loading vector layers in QGIS using addMapLayers() at once

q2.py

```
import sys
from qgis.utils import iface
from qgis.core import QgsVectorLayer, QgsProject,QgsVectorFileWriter
os.chdir('D:/g7data')
# load the layer
layer=iface.addVectorLayer("citiesx020.shp","cities","ogr")
# As there is no projection set it
crs1=layer.crs()
crs1.createFromId(4326)
layer.setCrs(crs1)
# save layer as new nname
QgsVectorFileWriter.writeAsVectorFormat(layer, "output/cities_new.shp", "UTF-8",
layer.crs(), "ESRI Shapefile")
# remove cities
QgsProject.instance().removeMapLayer(layer)
# add new cities
layer=iface.addVectorLayer("output/cities_new.shp","citiesNew","ogr")
#Defining Layer with QgsVectorLayer
co_layer=QgsVectorLayer("countries_simpl.shp","countries","ogr")
mj_layer=QgsVectorLayer("mjcities.shp","mjcities","ogr")
# As there is no projection set it
crs2=mj_layer.crs()
crs2.createFromId(4326)
mj_layer.setCrs(crs2)
# save layer as new nname
QgsVectorFileWriter.writeAsVectorFormat(layer, "output/mjNew.shp", "UTF-8",
mj_layer.crs(), "ESRI Shapefile")
# remove mj cities
QgsProject.instance().removeMapLayer(mj_layer)
# add new cities mj cities
mj_layer=QgsVectorLayer("output/mjNew.shp","mjNew","ogr")
states_layer=QgsVectorLayer("statesp020.shp","states","ogr")
#Add a list of layer
QgsProject.instance().addMapLayers([co_layer,mj_layer])
#Add a single layer
QgsProject.instance().addMapLayer(states_layer)
```

Following python codes can be viewed on the server

3. Update the CRS of the given shapefile.
4. Select by attributes and Save the selected features
5. Attribute Handling
6. Load the data from csv file
7. Create Buffers
8. Process the buffer
9. reading and writing on csv file
10. Union processing

Exp. No.11	R SCRIPTING	Date

AIM

To perform basic operations on the map layers in QGIS using R script.

SOFTWARE USED

QGIS

PROCEDURE

- Open the "g7 folder" in run and copy the ‘sample points’ data to the working directory.
- In a web browser, type the URL "*192.168.5.2/QR-providers24/#1*" to get the R scripts.
- Open QGIS software. In the plugin, check whether the "Processing R Provider" is installed (version 4.1); else install it and reload QGIS. Also check whether the processing uses 64 bit version.
- Open the processing tab, click "R" and select "Create new R script".
- Type the R script provided and save.
- In the given R scripts, Change the group name as your roll number and the name ending with the last three digits of roll number.
- Once saved, the "R" appears on the processing toolbox.
- Within the "R," roll_no folder and within that the script (Sample_random_points) are available.
- Double-click it to open it. Now load the layers and provide it as input and then run. Save the output in the working folder.

QUESTIONS

- Create sample random points

  script1.rsx

  ```
  ##Point pattern analysis=group
  ##Sample random points=name
  ##Layer=vector
  ##Size=number 10
  ##Output= output vector
  ```

```
Layer_sp = as_Spatial(Layer)
pts = spsample(Layer_sp,Size,type="random")
Output = SpatialPointsDataFrame(pts, as.data.frame(pts))
```

- Perform krigging

```
script2.rsx

##Basic statistics=group
##Krige value=name
##Layer=vector
##Field=Field Layer
##Output=output raster
library("automap")
library("sp")
Layer_sp = as_Spatial(Layer)
table = as.data.frame(Layer_sp)
coordinates(table)= ~coords.x1+coords.x2
c = Layer[[Field]]
kriging_result = autoKrige(c~1, table)
prediction = raster(kriging_result$krige_output)
Output = prediction
```

Following rstat scripts can be viewed on the server

- Find the minimum and maximum value from the data
- Generate graphs
- Perform expressions
- Generate statistical table

Using R script, perform the above-mentioned questions on QGIS using the data given.

Exp No.12	Mini Project	

Objective: Building a small GIS customization application involving above learned tools along with any additional scripting/database tools as an individual project.