COLUMBIA UNIVERSITY, BIG DATA ANALYTICS 6893

Homework 3: Data visualization

Yi Yang (yy3089)

November 4, 2021

1 PART 1

1.1 SVG Coordinate Space and Mathematical / Graph Coordinate Space

SVG coordinate Spave is similar to Mathematical / Graph Coordinate Space except that the Y coordinate of SVG Coordinate increases from top to bottom.

1.2 enter() and exit()

’enter()’ creates the initial join of data to elements, creating one circle element for every data element in the array. ’exit()’ removes any circle elements no longer needed and create any new circle elements by using ’enter()’.

1.3 transform() and translate()

SVG provides options to transform a single SVG shape element or group of SVG elements. SVG transform supports Translate, Scale, Rotate and Skew. Translate takes two options, tx refers translation along the x-axis and ty refers to the translation along the y-axis.

1.4 Anonymous function

In JavaScript, an anonymous function is that type of function that has no name or we can say which is without any name. When we create an anonymous function, it is declared without any identifier. In this question, the result should be [5,6,7,8,9]

1.5 part1_samplecode.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Homework 3 Question 1</title>
</head>
```

```

<style>body{}.bar-chart{background-color: #D2FFBD;}</style>
<svg class="bar-chart"></svg>

<script src="https://d3js.org/d3.v5.min.js"></script>
<script src="./visualizer.js"></script>

</body>
</html>
```

1.6 visualizer.js

```
var data = [80, 100, 56, 120, 180, 30, 40, 120, 160];
var svgWidth = 500, svgHeight = 300;

var barPaddingWidth = 5;
var textPaddingWidth = 2;
var barWidth = svgWidth / data.length - barPaddingWidth;
// The required padding between bars is 5px.
// The label must locate 2px above the middle of each bar.

var svg = d3.select('svg')
    .attr("width", svgWidth)
    .attr("height", svgHeight);

var barChart = svg.selectAll("rect")
    .data(data)
    .enter();

barChart.append("rect")
    .attr("class", "bar")
    .attr("height", function(d) {return d;})
    .attr("width", barWidth)
    .attr("transform", function(d, i) {
        return "translate(" + (barWidth+barPaddingWidth)*i
                            + ","
                            + (svgHeight - d)
                            + ")";
    })
    .attr("fill", "#CC6450");

barChart.append("text")
    .text(function(d) {return d;})
    .attr("transform", function(d, i) {
        return "translate(" + ((barWidth+barPaddingWidth)*i+barWidth/2)
                            + ","
                            + (svgHeight-d-textPaddingWidth)
                            + ")";
    })
    .style("text-anchor", "middle");
```

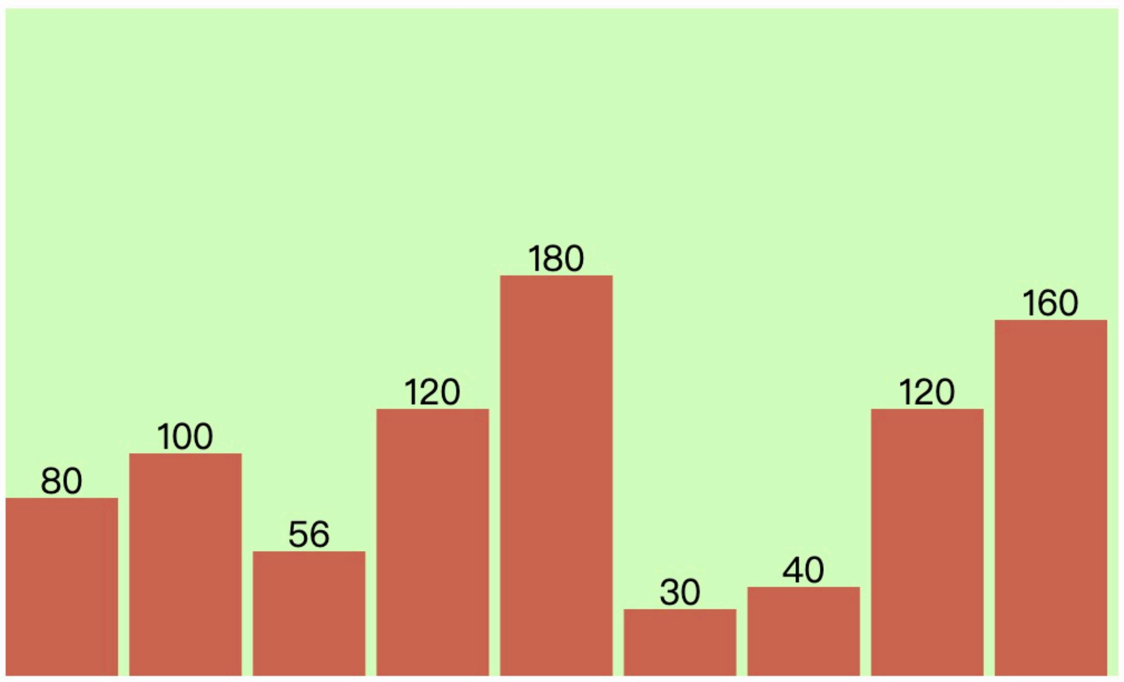
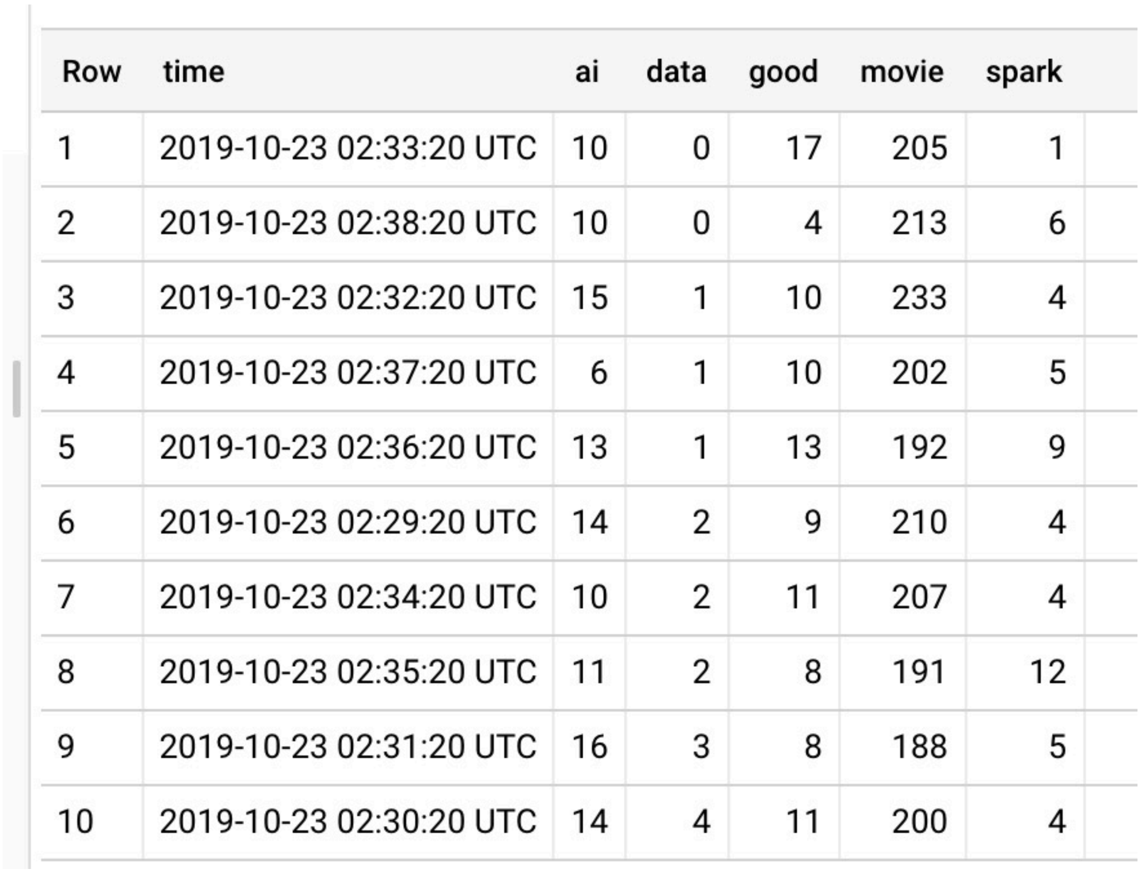


Figure 1.1: Screenshot for *'part1_ samplecode.html'*

2 Part 2

2.1 Data processing

Here is the data stored in BigQuery



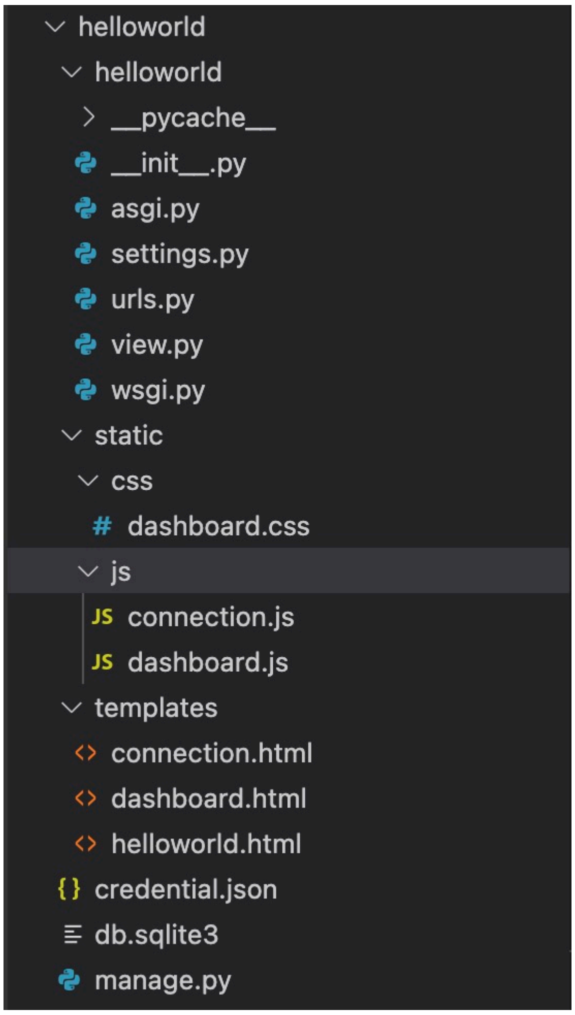
Row	time	ai	data	good	movie	spark
1	2019-10-23 02:33:20 UTC	10	0	17	205	1
2	2019-10-23 02:38:20 UTC	10	0	4	213	6
3	2019-10-23 02:32:20 UTC	15	1	10	233	4
4	2019-10-23 02:37:20 UTC	6	1	10	202	5
5	2019-10-23 02:36:20 UTC	13	1	13	192	9
6	2019-10-23 02:29:20 UTC	14	2	9	210	4
7	2019-10-23 02:34:20 UTC	10	2	11	207	4
8	2019-10-23 02:35:20 UTC	11	2	8	191	12
9	2019-10-23 02:31:20 UTC	16	3	8	188	5
10	2019-10-23 02:30:20 UTC	14	4	11	200	4

Figure 2.1: Screenshot for *'hashtag-bigquery'*

```
SELECT * FROM big_data_analysis.data_wordcount;

create table big_data_analysis.ai as
(select time, count as ai from big_data_analysis.data_wordcount
where word = 'ai');

create table big_data_analysis.data as
(select time, count as data from big_data_analysis.data_wordcount
where word = 'data');

create table big_data_analysis.good as
(select time, count as good from big_data_analysis.data_wordcount
where word = 'good');

create table big_data_analysis.movie as
(select time, count as movie from big_data_analysis.data_wordcount
where word = 'movie');

```

```
create table big_data_analysis.spark as
(select time, count as spark from big_data_analysis.data_wordcount
where word = 'spark');

create table word_count_res as (select
    ai.time,
    ai,
    ifnull(data, 0) as data,
    ifnull(good,0) as good,
    ifnull(movie, 0) as movie,
    ifnull(spark, 0) as spark
from
((((ai left join data on ai.time = data.time)
left join good on ai.time = good.time)
left join movie on ai.time = movie.time)
left join spark on ai.time = spark.time));
```

2.2 Create Django project



```
helloworld
  helloworld
    __pycache__
    __init__.py
    asgi.py
    settings.py
    urls.py
    view.py
    wsgi.py
  static
    css
      dashboard.css
    js
      connection.js
      dashboard.js
  templates
    connection.html
    dashboard.html
    helloworld.html
  credential.json
  db.sqlite3
  manage.py
```

Figure 2.2: Screenshot for *'Django project'*

2.3 HelloWorld

Hello World!

Figure 2.3: Screenshot for *'Hello World'*

2.4 Dashboard

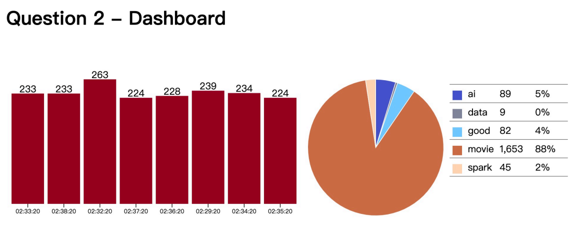
Changes in view.py:

```python
def dashboard(request):
    pandas_gbq.context.credentials = credentials
    pandas_gbq.context.project = "big-data-analytics-6893"

    SQL = "select␣time,␣ai,␣data,␣good,␣movie,␣spark␣\
␣␣␣␣␣␣␣␣␣␣␣from␣`wordcount.word_count_res`␣limit␣8"
    df = pandas_gbq.read_gbq(SQL)
    # print(df)

    records = df.to_dict(orient='records')
    # print("records:", records)

    data = {}

    l = []

    for record in records:
        d = {}
        d["Time"] = record["time"].strftime(format="%H:%M:%S")
        record.pop("time")
        d["count"] = record
        # print(d)
        l.append(d)

    data["data"] = l

    return render(request, 'dashboard.html', data)
```

Changes in dashboard.js:

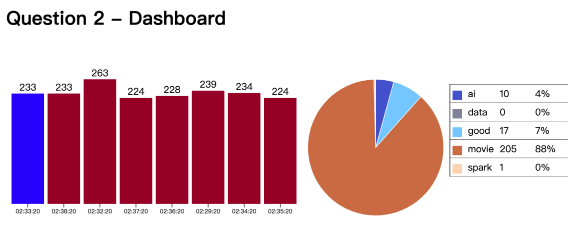
```javascript
// Choose color for each word:
function segColor(c) {
    cmap = {ai: "#4753CC", data: "#828499",
    good: "#73C9FF", movie: "#CC6E47", spark: '#FFD4B3'};
    return cmap[c]; /* TO FINISH */
}

// compute total for each state.
fData.forEach(function (d) {
    d.total = Object.keys(d.count).reduce((sum, key)
    => sum + d.count[key], 0); /* TO FINISH */
});

//create the rectangles.
bars.append("rect")
    .attr("x", function (d) {return x(d[0]);}) /* TO FINISH */
    .attr("y", function (d) {return y(d[1]);}) /* TO FINISH */
    .attr("width", x.rangeBand())
    .attr("height", function (d) {
        return hGDim.h - y(d[1]);
    })
    .attr('fill', barColor)
    .on("mouseover", mouseover)
    .on("mouseout", mouseout);

//Create the frequency labels ABOVE the rectangles.
bars.append("text").text(function (d) {
    return d3.format(",")(d[1])
})
    .attr("x", function (d) {
        return x(d[0]) + x.rangeBand() / 2;
    }) /* TO FINISH */
    .attr("y", function (d) {
        return y(d[1]) - 2;
    }) /* TO FINISH */
    .attr("text-anchor", "middle");

// transition the height and color of rectangles.
bars.select("rect").transition().duration(500)
    .attr("y", function (d) {
        return y(d[1])
    }) /* TO FINISH */
    .attr("height", function (d) {
        return hGDim.h - y(d[1]);
    })
    .attr("fill", color);

var tF = ['ai', 'data', 'good', 'movie', 'spark']
```

```
.map(function (d) {
    return {
        type: d, count: d3.sum(fData.map(function (t) {
            return t.count[d]/* TO FINISH */;
        }))
    };
});
```


Question 2 – Dashboard

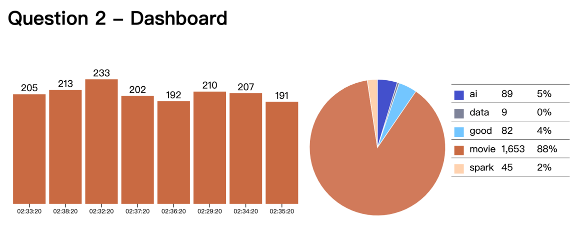
🟦	ai	89	5%
▨	data	9	0%
🟦	good	82	4%
🟧	movie	1,653	88%
🟧	spark	45	2%

Figure 2.4: Screenshot for *'dashboard'*


Question 2 – Dashboard

🟦	ai	10	4%
▨	data	0	0%
🟦	good	17	7%
🟧	movie	205	88%
🟧	spark	1	0%

Figure 2.5: Screenshot for *'dashboard'*



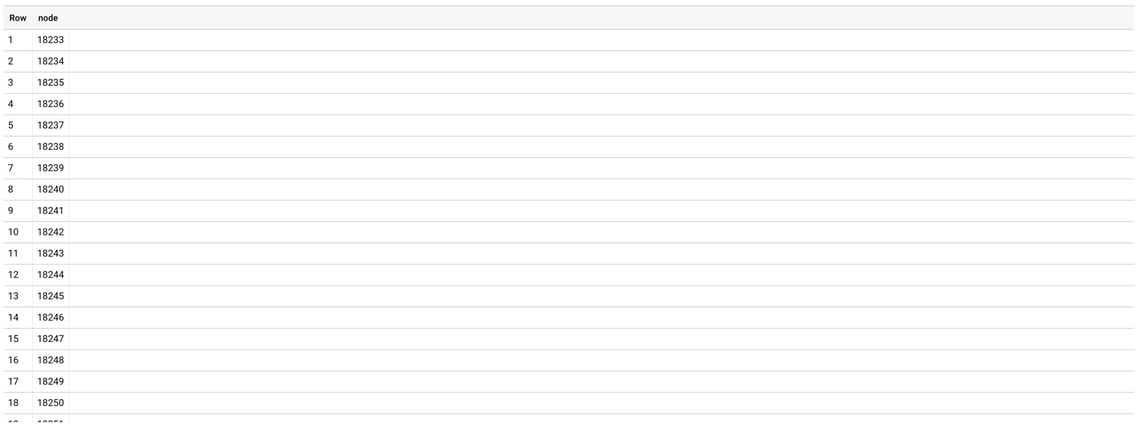
🟦	ai	89	5%
⬛	data	9	0%
🟦	good	82	4%
🟧	movie	1,653	88%
🟧	spark	45	2%

Figure 2.6: Screenshot for *'dashboard'*

3 Part 3

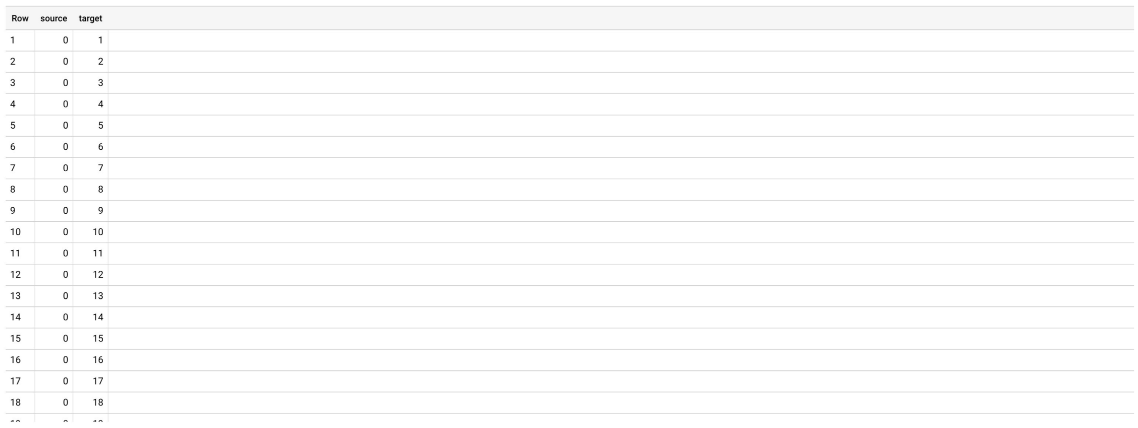
3.1 Date processing

```python
import csv
import os

data = []
txt_file = "data_103079215141.txt"

with open(txt_file, 'r') as f:
    data_ = f.read()

data_ = data_[2:-2].split("),␣(")

for d in data_:

    d = d[1:-1].split("',␣'")
    data.append((d[0], d[1]))

node_file = "nodes.csv"
edge_file = "edges.csv"

if os.path.isfile(node_file):
    os.remove(node_file)

if os.path.isfile(edge_file):
    os.remove(edge_file)

with open("nodes.csv", "a") as f:
    csv_writer = csv.writer(f)
    csv_writer.writerow(["node"])

with open("edges.csv", "a") as f:
    csv_writer = csv.writer(f)
    csv_writer.writerow(["source", "target"])

nodes = {}

order = 0
for _ in data:
    p1 = _[0]
    p2 = _[1]

    if p1 not in nodes.keys():
        nodes[p1] = order
        with open("nodes.csv", "a") as f:
            csv_writer = csv.writer(f)
            csv_writer.writerow([p1])
        order += 1

    if p2 not in nodes.keys():
        nodes[p2] = order
        with open("nodes.csv", "a") as f:
            csv_writer = csv.writer(f)
            csv_writer.writerow([p2])
        order += 1

    with open("edges.csv", "a") as f:
        csv_writer = csv.writer(f)
        csv_writer.writerow([nodes[p1], nodes[p2]])
```



Row	node
1	18233
2	18234
3	18235
4	18236
5	18237
6	18238
7	18239
8	18240
9	18241
10	18242
11	18243
12	18244
13	18245
14	18246
15	18247
16	18248
17	18249
18	18250

Figure 3.1: Screenshot for *'nodes'*



Row	source	target
1	0	1
2	0	2
3	0	3
4	0	4
5	0	5
6	0	6
7	0	7
8	0	8
9	0	9
10	0	10
11	0	11
12	0	12
13	0	13
14	0	14
15	0	15
16	0	16
17	0	17
18	0	18

Figure 3.2: Screenshot for *'edges'*

3.2 Connection

Changes in view.py:

```
def connection(request):
    pandas_gbq.context.credentials = credentials
    pandas_gbq.context.project = "big-data-analytics-6893"

    SQL1 = 'select␣node␣from␣`nodes_and_edges.nodes`'
    df1 = pandas_gbq.read_gbq(SQL1)
    # print(df1)

    SQL2 = 'select␣source,␣target␣from␣`nodes_and_edges.edges`'
    df2 = pandas_gbq.read_gbq(SQL2)
    # print(df2)

    data = {}

    data['n'] = df1.to_dict(orient='records')
    data['e'] = df2.to_dict(orient='records')

    # print(data['n'])
    # print(data['e'])

    return render(request, 'connection.html', data)
```

Changes in connection.js:

```
var svg = d3.select("body")
            .append("svg")
            .attr("height", height) /* TO FINISH */
            .attr("width", width); /* TO FINISH */

var svg_edges = svg.selectAll("line")
                   .data(edges) /* TO FINISH */
                   .enter()
                   .append("line") /* TO FINISH */
                   .style("stroke","#ccc")
                   .style("stroke-width",1);

var svg_nodes = svg.selectAll("circle")
                   .data(nodes) /* TO FINISH */
                   .enter()
                   .append("circle") /* TO FINISH */
                   .attr("r",20)
                   .style("fill", function (d) {return color(d.index)})
                    /* TO FINISH */
                   .call(force.drag);

var svg_texts = svg.selectAll("text")
                   .data(nodes)
                   .enter()
                   .append("text")
                   .style("fill", "black")
                   .attr("dx", 20)
                   .attr("dy", 8)
                   .text(function (d) {return d.node}); /* TO FINISH */

force.on("tick", function(){
   svg_edges.attr("x1", function (d) {return d.source.x}) /* TO FINISH */
            .attr("y1", function (d) {return d.source.y}) /* TO FINISH */
```

```
                .attr("x2", function (d) {return d.target.x}) /* TO FINISH */
                .attr("y2", function (d) {return d.target.y}); /* TO FINISH */
```

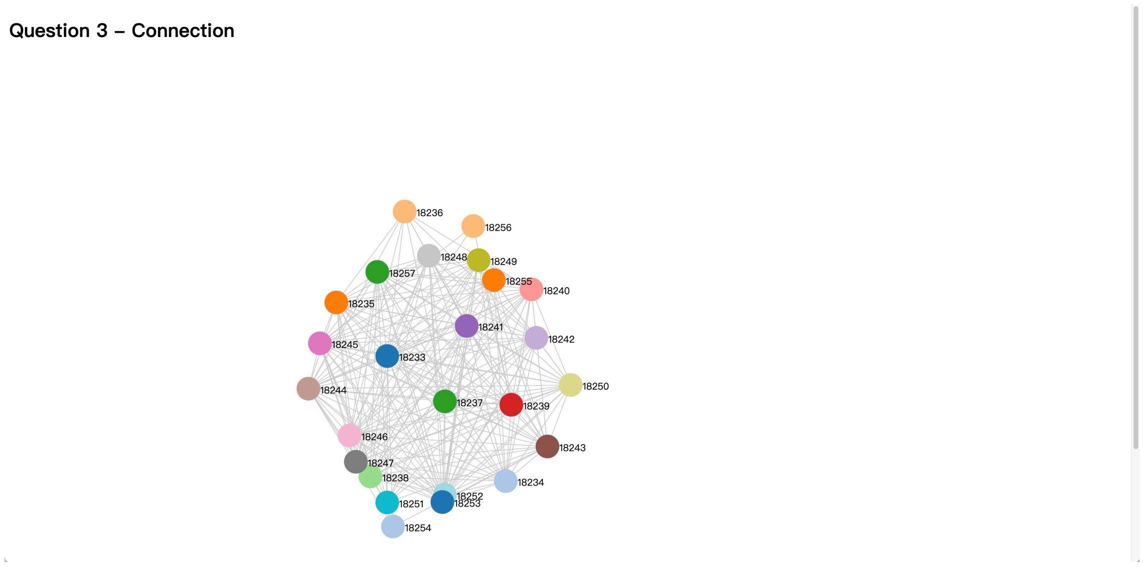


Figure 3.3: Screenshot for *'Connection'*