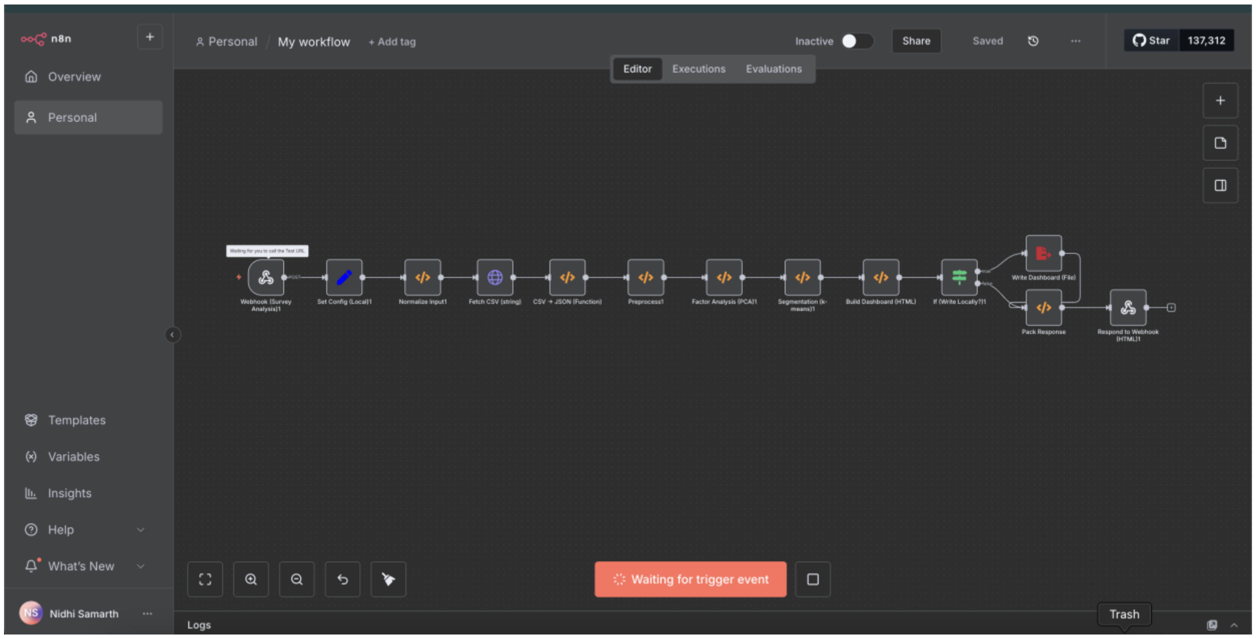
# Weekly Report- 1

**Nidhi Samarth**

As per today's meeting, we were supposed to share screenshot. Please find my screenshot of my work below.



My Json code file:

```
{
 "name": "My workflow 2",
 "nodes": [
  {
   "parameters": {
    "url": "={{$json.csvUrl}}",
    "responseFormat": "string",
    "options": {}
   },
   "id": "26ac9d3d-d969-48eb-9b84-935ec851b508",
   "name": "Fetch CSV (string)",
   "type": "n8n-nodes-base.httpRequest",
    "typeVersion": 2,
    "position": [
      -1952,
      32
    ]
  },
  {
    "parameters": {
      "functionCode": "// Parse CSV (no Spreadsheet node needed)\nfunction parseCSV(text){\n  const rows=[]; let
i=0, field='', rec=[], inQ=false;\n  const push=()=>{ rec.push(field); field=''; };\n  const endRec=()=>{ rows.push(rec);
rec=[]; };\n  while(i<text.length){\n    const ch=text[i];\n    if(inQ){\n      if(ch==='\"'){\n        if(text[i+1]==='\"'){ field+='\"';
i+=2; } else { inQ=false; i++; }\n      } else { field+=ch; i++; }\n    } else {\n      if(ch==='\"'){ inQ=true; i++; }\n      else
if(ch===','){ push(); i++; }\n      else if(ch==='\\n'){ push(); endRec(); i++; if(text[i]==='\\r') i++; }\n      else if(ch==='\\r'){
push(); endRec(); i++; if(text[i]==='\\n') i++; }\n      else { field+=ch; i++; }\n    }\n  }\n  push(); if(rec.length) endRec();\n
return rows;\n}\nconst text = items[0].json;\nconst raw = typeof text === 'object' && text.body ? text.body : (typeof text
=== 'string' ? text : items[0].json.body);\nconst rows = parseCSV(String(raw||''));\nif(!rows.length) return
[{json:{error:'Empty CSV'}}];\nconst headers = rows[0].map(h=>String(h||'').trim());\nconst data =
rows.slice(1).filter(r=>r.length===headers.length).map(r=>{\n  const o={}; for(let j=0;j<headers.length;j++){
o[headers[j]] = r[j]; } return o;\n});\nreturn data.map(d=>({json:d}));"
    },
    "id": "0dae2ed5-3526-43a7-9f06-76ae85b10bdd",
    "name": "CSV → JSON (Function)",
    "type": "n8n-nodes-base.function",
    "typeVersion": 1,
    "position": [
      -1760,
      32
    ]
  },
  {
    "parameters": {
      "functionCode": "// Build HTML dashboard\nconst { rows, numericColumns, scores2D, labels, centroids,
explained, components } = items[0].json;\nconst outRows =
rows.map((r,i)=>({...r,__PC1:scores2D[i][0],__PC2:scores2D[i][1],__segment:labels[i]}));\nconst k =
Math.max(...labels)+1; const counts = Array(k).fill(0); for(const g of labels) counts[g]++;\nconst loadings =
(components||[])[0] ?
numericColumns.map((name,j)=>({feature:name,PC1:components[0][j]||0,PC2:components[1]?.[j]||0})) : [];\nconst
vlData = outRows.map(r=>({PC1:r.__PC1, PC2:r.__PC2, segment:String(r.__segment)}));\nconst vlCentroids =
centroids.map((c,idx)=>({PC1:c[0],PC2:c[1],segment:String(idx)}));\nconst spec = { $schema:
\"https://vega.github.io/schema/vega-lite/v5.json\", width: 720, height: 440, layer:[ { data:{name:\"points\"},
mark:{type:\"circle\",opacity:0.6,size:60}, encoding:{ x:{field:\"PC1\",type:\"quantitative\"},
y:{field:\"PC2\",type:\"quantitative\"}, color:{field:\"segment\",type:\"nominal\"},
tooltip:[{field:\"segment\"},{field:\"PC1\"},{field:\"PC2\"}] } }, { data:{name:\"centroids\"},
mark:{type:\"point\",filled:true,size:220,shape:\"triangle-up\"}, encoding:{ x:{field:\"PC1\",type:\"quantitative\"},
y:{field:\"PC2\",type:\"quantitative\"}, color:{field:\"segment\",type:\"nominal\"},
tooltip:[{field:\"segment\"},{field:\"PC1\"},{field:\"PC2\"}] } } ] };\nconst html = `<!doctype html><html><head><meta
charset=\\\"utf-8\\\"/><title>Survey Segments</title><script
src=\\\"https://cdn.jsdelivr.net/npm/vega@5\\\"></script><script src=\\\"https://cdn.jsdelivr.net/npm/vega-
lite@5\\\"></script><script src=\\\"https://cdn.jsdelivr.net/npm/vega-embed@6\\\"></script><style>body{font-family:ui-
sans-serif,system-ui;margin:24px}.grid{display:grid;grid-template-columns:2fr 1fr;gap:24px}table{border-
collapse:collapse;width:100%}th,td{border:1px solid #ddd;padding:6px 8px;font-
size:12px}th{background:#f5f5f5}.small{font-size:12px;color:#444}</style></head><body><h1>Survey
Segmentation</h1><p class=\\\"small\\\">Explained variance — PC1 ${(explained[0]*100).toFixed(1)}%, PC2
${(explained[1]*100).toFixed(1)}%</p><div class=\\\"grid\\\"><div><div
id=\\\"vis\\\"></div></div><div><h3>Segments</h3><table><thead><tr><th>Segment</th><th>Count</th></tr></thea
d><tbody>${counts.map((c,i)=>`<tr><td>${i}</td><td>${c}</td></tr>`).join('')}</tbody></table><h3 style=\\\"margin-
top:16px\\\">Top
Loadings</h3><table><thead><tr><th>Feature</th><th>PC1</th><th>PC2</th></tr></thead><tbody>${loadings.ma
p(l=>`<tr><td>${l.feature}</td><td>${l.PC1.toFixed(3)}</td><td>${l.PC2.toFixed(3)}</td></tr>`).join('')}</tbody></table
></div></div><script>const spec=${JSON.stringify(spec)};const points=${JSON.stringify(vlData)};const
centroids=${JSON.stringify(vlCentroids)};vegaEmbed('#vis',spec,{actions:false}).then(res=>{res.view.change('points',
vega.changeset().remove(()=>true).insert(points)).run();res.view.change('centroids',vega.changeset().remove(()=>tru
e).insert(centroids)).run();});</script></body></html>`;\nconst buff = Buffer.from(html,'utf8');\nreturn [{ json:{
...items[0].json, outRows, counts, loadings, html }, binary:{ dashboard:{ data: buff,
fileName:'madison_survey_dashboard.html', mimeType:'text/html' } } }];"
   },
   "id": "89ea83cc-8bba-46e5-a3ec-c86bc66ab88f",
   "name": "Build Dashboard (HTML)",
   "type": "n8n-nodes-base.function",
   "typeVersion": 1,
   "position": [
     -928,
     32
   ]
```

```
  },
  {
    "parameters": {
      "fileName": "={{$json.localPath}}",
      "dataPropertyName": "dashboard",
      "options": {}
    },
    "id": "0fc4eccd-8504-485f-bbcc-be1f8c1a1333",
    "name": "Write Dashboard (File)",
    "type": "n8n-nodes-base.writeBinaryFile",
    "typeVersion": 1,
    "position": [
      -496,
      -32
    ]
  },
  {
    "parameters": {
      "functionCode": "const src = $items('Build Dashboard (HTML)', 0, 0).json; return [{ json: { html: src.html } }];"
    },
    "id": "35c96730-781e-47fd-9c31-8d995b01c924",
    "name": "Pack Response",
    "type": "n8n-nodes-base.function",
    "typeVersion": 1,
    "position": [
      -496,
      112
    ]
  },
  {
    "parameters": {
      "httpMethod": "=POST",
      "path": "madison/research/survey",
      "responseMode": "responseNode",
      "options": {}
    },
    "id": "1840aa15-57cc-4a58-9c6b-7ab09736a45a",
   "name": "Webhook (Survey Analysis)1",
   "type": "n8n-nodes-base.webhook",
   "typeVersion": 1,
   "position": [
    -2560,
    32
   ],
   "webhookId": "7fa5e6ee-8bc0-4fb4-bca1-342c2fcc9097"
  },
  {
   "parameters": {
    "keepOnlySet": true,
    "values": {
     "boolean": [
      {
       "name": "writeFile",
       "value": true
      }
     ],
     "string": [
      {
       "name": "defaultCsvUrl",
       "value": "https://people.sc.fsu.edu/~jburkardt/data/csv/airtravel.csv"
      },
      {
       "name": "localPath",
       "value": "/data/madison_survey_dashboard.html"
      },
      {
       "name": "defaultIdColumn"
      }
     ],
     "number": [
      {
       "name": "defaultK",
       "value": 3
      }
```

```
        ]
      },
      "options": {}
    },
    "id": "3ab4c650-425c-4b47-abc3-199a3b243ef6",
    "name": "Set Config (Local)1",
    "type": "n8n-nodes-base.set",
    "typeVersion": 2,
    "position": [
      -2352,
      32
    ]
  },
  {
    "parameters": {
      "functionCode": "const body = items[0].json || {};\nconst csvUrl = body.csvUrl || $json.defaultCsvUrl;\nconst k =
Number(body.k != null ? body.k : $json.defaultK);\nconst idColumn = (body.idColumn != null ? body.idColumn :
$json.defaultIdColumn).trim();\nconst runId = `mads_${Date.now()}_${Math.random().toString(36).slice(2,8)}`;\nreturn
[{ json: { runId, csvUrl, k, idColumn, writeFile: !!$json.writeFile, localPath: $json.localPath, maxIterations: 50 } }];"
    },
    "id": "8fc158cc-a787-46cd-be51-6e4f4a5a41fd",
    "name": "Normalize Input1",
    "type": "n8n-nodes-base.function",
    "typeVersion": 1,
    "position": [
      -2144,
      32
    ]
  },
  {
    "parameters": {
      "functionCode": "// Preprocess: detect numeric cols, impute mean, z-score\nfunction toNum(v){
if(v===null||v===undefined||v==='') return NaN; if(typeof v==='number') return v; const
n=Number(String(v).replace(/,/,'.')); return Number.isFinite(n)?n:NaN; }\nconst rows =
items.map(i=>i.json);\nif(!rows.length) return [{json:{error:'No rows'}}];\nconst allCols =
Object.keys(rows[0]||{});\nconst idColumn = $json.idColumn||'';\nfunction isNumericCol(c){ let t=0, ok=0; for(const r of
rows){ const v=r[c]; if(v!==''&&v!=null){ t++; if(Number.isFinite(toNum(v))) ok++; } } return t>0 ? (ok/t)>=0.8 : false;
```

```
}\nconst numericColumns = allCols.filter(c=>c!==idColumn && isNumericCol(c));\nif(!numericColumns.length) return
[{json:{error:'No numeric columns'}}];\nconst X = rows.map(r=>numericColumns.map(c=>toNum(r[c])));\nconst
n=X.length, d=numericColumns.length;\nconst means=Array(d).fill(0);\nfor(let j=0;j<d;j++){ let s=0,c=0; for(let
i=0;i<n;i++){ const v=X[i][j]; if(Number.isFinite(v)){ s+=v; c++; } } means[j]=c?s/c:0; }\nfor(let i=0;i<n;i++){ for(let
j=0;j<d;j++){ if(!Number.isFinite(X[i][j])) X[i][j]=means[j]; } }\nconst stds=Array(d).fill(0);\nfor(let j=0;j<d;j++){ let s2=0;
for(let i=0;i<n;i++){ const z=X[i][j]-means[j]; s2+=z*z; } stds[j]=Math.sqrt(s2/Math.max(1,n-1))||1; }\nfor(let i=0;i<n;i++){
for(let j=0;j<d;j++){ X[i][j]=(X[i][j]-means[j])/(stds[j]||1); } }\nreturn [{json:{rows, X, numericColumns, means, stds,
idColumn, k:$json.k, maxIterations:$json.maxIterations||50, runId:$json.runId, writeFile:$json.writeFile,
localPath:$json.localPath}}];"
   },
   "id": "a6b6854b-7ccc-4435-a074-4f7ca2886fc5",
   "name": "Preprocess1",
   "type": "n8n-nodes-base.function",
   "typeVersion": 1,
   "position": [
    -1552,
    32
   ]
  },
  {
   "parameters": {
    "functionCode": "// PCA (top 2) via power iteration\nconst { X, numericColumns } = items[0].json;\nfunction
dot(a,b){ let s=0; for(let i=0;i<a.length;i++) s+=a[i]*b[i]; return s; }\nfunction norm(a){ return Math.sqrt(dot(a,a))||1;
}\nfunction cov(M){ const n=M.length,d=M[0].length; const C=Array(d).fill(0).map(()=>Array(d).fill(0)); for(let
i=0;i<n;i++){ for(let j=0;j<d;j++){ const vj=M[i][j]; for(let k=0;k<d;k++){ C[j][k]+=vj*M[i][k]; } } } for(let j=0;j<d;j++){ for(let
k=0;k<d;k++){ C[j][k]/=Math.max(1,n-1); } } return C; }\nfunction powerIteration(A,it){ const d=A.length; let
v=Array(d).fill(0).map(()=>Math.random()); let nv=norm(v); for(let i=0;i<d;i++) v[i]/=nv; let lambda=0; for(let
t=0;t<(it||300);t++){ const Av=Array(d).fill(0).map((_,i)=>A[i].reduce((s,a,idx)=>s+a*v[idx],0)); lambda=dot(v,Av); const
nrm=norm(Av); for(let i=0;i<d;i++) v[i]=Av[i]/(nrm||1); } return {vec:v,val:lambda}; }\nfunction deflate(A,vec,val){ const
d=A.length; for(let i=0;i<d;i++){ for(let j=0;j<d;j++){ A[i][j]-=val*vec[i]*vec[j]; } } }\nlet C=cov(X); const comps=[], vals=[];
for(let k=0;k<2;k++){ const {vec,val}=powerIteration(C,400); comps.push(vec); vals.push(Math.max(0,val));
deflate(C,vec,val); }\nconst scores2D = X.map(row=>comps.map(v=>dot(row,v)));\nconst explained =
vals.map(v=>v/(numericColumns.length));\nreturn [{json:{...items[0].json, scores2D, components:comps,
explained}}];"
   },
   "id": "ba629d25-9fee-4277-84fd-0fc25c71eae6",
   "name": "Factor Analysis (PCA)1",
    "type": "n8n-nodes-base.function",
    "typeVersion": 1,
    "position": [
      -1344,
      32
    ]
  },
  {
    "parameters": {
      "functionCode": "// k-means on PCA scores\nconst { scores2D } = items[0].json;\nconst k =
Number($json.k||3);\nconst maxIter = Number($json.maxIterations||50);\nfunction d2(a,b){ const dx=a[0]-b[0],
dy=a[1]-b[1]; return dx*dx+dy*dy; }\nfunction init(X,k){ const n=X.length; const cent=[ X[Math.floor(Math.random()*n)]
]; for(let m=1;m<k;m++){ const dists=X.map(p=>Math.min(...cent.map(c=>d2(p,c)))); const
sum=dists.reduce((a,b)=>a+b,0)||1; let r=Math.random()*sum, idx=0; for(let i=0;i<n;i++){ r-=dists[i]; if(r<=0){ idx=i;
break; } } cent.push(X[idx]); } return cent.map(c=>[c[0],c[1]]); }\nlet centroids=init(scores2D,k); let
labels=Array(scores2D.length).fill(0);\nfor(let it=0; it<maxIter; it++){\n  let changed=false;\n  for(let
i=0;i<scores2D.length;i++){\n    const p=scores2D[i]; let best=0,bd=Infinity; for(let j=0;j<k;j++){ const
d=d2(p,centroids[j]); if(d<bd){ bd=d; best=j; } }\n    if(labels[i]!==best){ labels[i]=best; changed=true; }\n  }\n  const
sums=Array(k).fill(0).map(()=>[0,0,0]);\n  for(let i=0;i<scores2D.length;i++){ const g=labels[i];
sums[g][0]+=scores2D[i][0]; sums[g][1]+=scores2D[i][1]; sums[g][2]++; }\n  for(let j=0;j<k;j++){ if(sums[j][2]>0){
centroids[j]=[sums[j][0]/sums[j][2], sums[j][1]/sums[j][2]]; } }\n  if(!changed) break;\n}\nreturn [{json:{...items[0].json,
labels, centroids}}];"
    },
    "id": "08050032-8f6c-42f0-8855-8a19895a5a0a",
    "name": "Segmentation (k-means)1",
    "type": "n8n-nodes-base.function",
    "typeVersion": 1,
    "position": [
      -1136,
      32
    ]
  },
  {
    "parameters": {
      "conditions": {
        "boolean": [
          {
```

```
          "value1": "={{$json.writeFile === true || $json.writeFile === 'true'}}"
        }
      ]
    }
  },
  "id": "fd4faf8a-4837-4572-b151-1fe8ecc11c1a",
  "name": "If (Write Locally?)1",
  "type": "n8n-nodes-base.if",
  "typeVersion": 1,
  "position": [
    -720,
    32
  ]
 },
 {
  "parameters": {
    "options": {}
  },
  "id": "a053fa6f-ad88-4ed7-a79c-b16eeded2def",
  "name": "Respond to Webhook (HTML)1",
  "type": "n8n-nodes-base.respondToWebhook",
  "typeVersion": 1,
  "position": [
    -272,
    112
  ]
 }
],
"pinData": {},
"connections": {
 "Fetch CSV (string)": {
  "main": [
    [
      {
        "node": "CSV → JSON (Function)",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"CSV → JSON (Function)": {
  "main": [
    [
      {
        "node": "Preprocess1",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Build Dashboard (HTML)": {
  "main": [
    [
      {
        "node": "If (Write Locally?)1",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Write Dashboard (File)": {
  "main": [
    [
      {
        "node": "Pack Response",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
  "Pack Response": {
   "main": [
    [
     {
      "node": "Respond to Webhook (HTML)1",
      "type": "main",
      "index": 0
     }
    ]
   ]
  },
  "Webhook (Survey Analysis)1": {
   "main": [
    [
     {
      "node": "Set Config (Local)1",
      "type": "main",
      "index": 0
     }
    ]
   ]
  },
  "Set Config (Local)1": {
   "main": [
    [
     {
      "node": "Normalize Input1",
      "type": "main",
      "index": 0
     }
    ]
   ]
  },
  "Normalize Input1": {
   "main": [
    [
     {
        "node": "Fetch CSV (string)",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Preprocess1": {
  "main": [
    [
      {
        "node": "Factor Analysis (PCA)1",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Factor Analysis (PCA)1": {
  "main": [
    [
      {
        "node": "Segmentation (k-means)1",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Segmentation (k-means)1": {
  "main": [
    [
      {
        "node": "Build Dashboard (HTML)",
        "type": "main",
        "index": 0
      }
```

```
        ]
      ]
    },
    "If (Write Locally?)1": {
      "main": [
        [
          {
            "node": "Write Dashboard (File)",
            "type": "main",
            "index": 0
          }
        ],
        [
          {
            "node": "Pack Response",
            "type": "main",
            "index": 0
          }
        ]
      ]
    }
  },
  "active": false,
  "settings": {
    "executionOrder": "v1"
  },
  "versionId": "3a4a3458-47b5-4cdb-896e-0dcb6bb73ba9",
  "meta": {
    "instanceId": "a2e93776a6561cf91088c56d633ce5983f60798351f141b409c407ebc82b6740"
  },
  "id": "MypbblqzeyGZUBVn",
  "tags": []
}
```