```
{
  "name": "Lead Qualification & SDR Assignment System - FINAL",
  "nodes": [
    {
      "parameters": {
        "httpMethod": "POST",
        "path": "calendly-demo-signup",
        "responseMode": "responseNode"
      },
      "id": "calendly-webhook",
      "name": "Calendly Demo Signup",
      "type": "n8n-nodes-base.webhook",
      "typeVersion": 1,
      "position": [200, 400],
      "webhookId": "calendly-demo-webhook"
    },
    {
      "parameters": {
        "jsCode": "// Extract Calendly webhook data with error
handling\ntry {\n  const webhookData = $input.all()[0].json;\n\n
let leadData = {\n    email: webhookData.email || '',\n    name:
webhookData.name || '',\n    company: webhookData.company || '',\n
job_title: webhookData.job_title || '',\n    phone:
webhookData.phone || '',\n    received_at: new
Date().toISOString(),\n    session_id:
`lead_${Date.now()}_${Math.random().toString(36).substr(2, 9)}`,\n
processing_stage: 'data_extraction'\n  };\n\n  // Handle Calendly
webhook format if needed\n  if (webhookData.payload &&
webhookData.payload.invitee) {\n    const invitee =
webhookData.payload.invitee;\n    leadData.email = invitee.email
|| leadData.email;\n    leadData.name = invitee.name ||
leadData.name;\n    \n    // Extract from questions and answers\n
if (invitee.questions_and_answers) {\n
invitee.questions_and_answers.forEach(qa => {\n        const
question = qa.question.toLowerCase();\n        if
(question.includes('company')) {\n          leadData.company =
qa.answer;\n        } else if (question.includes('title') ||
question.includes('role')) {\n          leadData.job_title =
qa.answer;\n        }\n      });\n    }\n  }\n\n  // Validate
required fields\n  if (!leadData.email || !leadData.name) {\n
throw new Error('Missing required fields: email and name are
mandatory');\n  }\n\n  // Clean data\n  leadData.email =
leadData.email.toLowerCase().trim();\n  leadData.name =
leadData.name.trim();\n  leadData.company =
leadData.company.trim();\n  leadData.job_title =
leadData.job_title.trim();\n\n  return [{ json: leadData }];\n
\n} catch (error) {\n  return [{\n    json: {\n      error:
true,\n      error_message: error.message,\n      node_name:
```

```
'Extract Lead Data',\n      timestamp: new Date().toISOString()\n
}\n  }];\n}"
      },
      "id": "extract-lead-data",
      "name": "Extract Lead Data",
      "type": "n8n-nodes-base.code",
      "typeVersion": 2,
      "position": [400, 400]
    },
    {
      "parameters": {
        "url":
"https://nubela.co/proxycurl/api/linkedin/person/resolve",
        "sendQuery": true,
        "queryParameters": {
          "parameters": [
            {
              "name": "first_name",
              "value": "={{ $json.name.split(' ')[0] }}"
            },
            {
              "name": "last_name",
              "value": "={{ $json.name.split(' ').slice(1).join('
') }}"
            },
            {
              "name": "company_name",
              "value": "={{ $json.company }}"
            }
          ]
        },
        "options": {
          "response": {
            "response": {
              "responseFormat": "json"
            }
          }
        }
      },
      "id": "linkedin-lookup",
      "name": "LinkedIn Profile Lookup",
      "type": "n8n-nodes-base.httpRequest",
      "typeVersion": 4.1,
      "position": [600, 300],
      "continueOnFail": true
    },
    {
      "parameters": {
```

```
        "url": "https://company.clearbit.com/v1/domains/find",
        "sendQuery": true,
        "queryParameters": {
          "parameters": [
            {
              "name": "name",
              "value": "={{ $('Extract Lead
Data').item.json.company }}"
            }
          ]
        },
        "options": {
          "response": {
            "response": {
              "responseFormat": "json"
            }
          }
        }
      },
      "id": "company-enrichment",
      "name": "Company Data Enrichment",
      "type": "n8n-nodes-base.httpRequest",
      "typeVersion": 4.1,
      "position": [600, 500],
      "continueOnFail": true
    },
    {
      "parameters": {
        "operation": "read",
        "documentId": "your-icp-qualification-sheet-id",
        "sheetName": "ICP Criteria",
        "options": {}
      },
      "id": "fetch-icp-criteria",
      "name": "Fetch ICP Qualification Criteria",
      "type": "n8n-nodes-base.googleSheets",
      "typeVersion": 4,
      "position": [800, 400],
      "continueOnFail": true
    },
    {
      "parameters": {
        "jsCode": "// Robust Dynamic Qualification Engine with
Error Handling\ntry {\n  // Safely get data from previous nodes\n
let leadData = {};\n  let linkedinData = {};\n  let companyData =
{};\n  let icpCriteria = [];\n\n  try {\n    leadData = $('Extract
Lead Data').item.json;\n  } catch (error) {\n    throw new
Error('Extract Lead Data node failed or missing');\n  }\n\n  try
```

```
{\n    linkedinData = $('LinkedIn Profile Lookup').item.json || {};\n  } catch (error) {\n    console.log('LinkedIn lookup failed, continuing without LinkedIn data');\n    linkedinData = {};\n  }\n\n  try {\n    companyData = $('Company Data Enrichment').item.json || {};\n  } catch (error) {\n    console.log('Company enrichment failed, continuing without company data');\n    companyData = {};\n  }\n\n  try {\n    icpCriteria = $('Fetch ICP Qualification Criteria').item.json || [];\n  } catch (error) {\n    console.log('ICP criteria fetch failed, using fallback qualification');\n    icpCriteria = [];\n  }\n\n  // Initialize scoring\n  let score = 0;\n  let persona = 'P3';\n  let reasoning = [];\n\n  // Extract company size with safe defaults\n  let companySize = 50;\n  if (companyData && companyData.employees) {\n    companySize = companyData.employees;\n  } else if (linkedinData && linkedinData.experiences && linkedinData.experiences[0]) {\n    companySize = 100;\n  }\n\n  // Determine seniority level\n  const jobTitle = (leadData.job_title || '').toLowerCase();\n  let seniorityLevel = 'individual-contributor';\n  \n  if (jobTitle.includes('ceo') || jobTitle.includes('founder') || jobTitle.includes('chief')) {\n    seniorityLevel = 'c-level';\n  } else if (jobTitle.includes('vp') || jobTitle.includes('vice president')) {\n    seniorityLevel = 'vp';\n  } else if (jobTitle.includes('director') || jobTitle.includes('head')) {\n    seniorityLevel = 'director';\n  } else if (jobTitle.includes('manager') || jobTitle.includes('lead')) {\n    seniorityLevel = 'manager';\n  }\n\n  // Get industry\n  const industry = (companyData && companyData.category) || \n                   (linkedinData && linkedinData.industry) || \n                   'unknown';\n\n  // Use ICP criteria if available, otherwise use fallback\n  if (icpCriteria && Array.isArray(icpCriteria) && icpCriteria.length > 0) {\n    reasoning.push('Using ICP criteria from sheet');\n    \n    icpCriteria.forEach(criterion => {\n      if (!criterion) return;\n      \n      const criteriaName = criterion.Criteria || criterion.criteria || 'Unknown';\n      const weight = parseFloat(criterion.Weight || criterion.weight || 0);\n      const fieldToCheck = criterion.Field_To_Check || criterion.field_to_check || '';\n      \n      if (!weight || !fieldToCheck) return;\n      \n      let points = 0;\n      let matched = 'none';\n      \n      if (fieldToCheck === 'company_size') {\n        const p1Req = (criterion.P1_Requirement || criterion.p1_requirement || '').toString();\n        const p2Req = (criterion.P2_Requirement || criterion.p2_requirement || '').toString();\n        \n        if (p1Req.includes('>=')) {\n          const threshold = parseInt(p1Req.replace('>=', '').trim());\n          if (companySize >= threshold) {\n            points = weight;\n            matched = 'P1';\n          }\n        }\n        \n        if (matched === 'none' && p2Req.includes('-')) {\n          const
```

```
range = p2Req.split('-');\n          if (range.length === 2) {\n
const min = parseInt(range[0].trim());\n            const max =
parseInt(range[1].trim());\n            if (companySize >= min &&
companySize <= max) {\n              points = weight * 0.7;\n
matched = 'P2';\n            }\n          }\n        }\n        \n
if (matched === 'none') {\n          points = weight * 0.3;\n
matched = 'P3';\n        }\n      } else if (fieldToCheck ===
'seniority_level') {\n        const p1Req =
(criterion.P1_Requirement || criterion.p1_requirement ||
'').toLowerCase();\n        const p2Req =
(criterion.P2_Requirement || criterion.p2_requirement ||
'').toLowerCase();\n        \n        if
(p1Req.includes(seniorityLevel)) {\n          points = weight;\n
matched = 'P1';\n        } else if
(p2Req.includes(seniorityLevel)) {\n          points = weight *
0.7;\n          matched = 'P2';\n        } else {\n
points = weight * 0.3;\n          matched = 'P3';\n        }\n
} else if (fieldToCheck === 'industry') {\n        const p1Req =
(criterion.P1_Requirement || criterion.p1_requirement ||
'').toLowerCase();\n        const p2Req =
(criterion.P2_Requirement || criterion.p2_requirement ||
'').toLowerCase();\n        \n        if (p1Req &&
p1Req.split(',').some(ind =>
industry.toLowerCase().includes(ind.trim()))) {\n          points
= weight;\n          matched = 'P1';\n        } else if (p2Req &&
p2Req.split(',').some(ind =>
industry.toLowerCase().includes(ind.trim()))) {\n          points
= weight * 0.7;\n          matched = 'P2';\n        } else {\n
points = weight * 0.3;\n          matched = 'P3';\n        }\n
}\n      \n      score += points;\n      if (points > 0) {\n
reasoning.push(`${criteriaName}: ${matched} match
(+${Math.round(points)} points)`);\n      }\n    });\n  } else {\n
// Fallback scoring logic\n    reasoning.push('Using fallback
qualification (ICP sheet unavailable)');\n    \n    if
(companySize >= 500) {\n      score += 40;\n
reasoning.push(`Large company (${companySize} employees) = +40
points`);\n    } else if (companySize >= 50) {\n      score +=
25;\n      reasoning.push(`Mid-size company (${companySize}
employees) = +25 points`);\n    } else {\n      score += 10;\n
reasoning.push(`Small company (${companySize} employees) = +10
points`);\n    }\n    \n    if (seniorityLevel === 'c-level') {\n
score += 35;\n      reasoning.push('C-level title = +35
points');\n    } else if (seniorityLevel === 'vp') {\n      score
+= 30;\n      reasoning.push('VP level = +30 points');\n    } else
if (seniorityLevel === 'director') {\n      score += 25;\n
reasoning.push('Director level = +25 points');\n    } else if
(seniorityLevel === 'manager') {\n      score += 15;\n
reasoning.push('Manager level = +15 points');\n    } else {\n
```

```
score += 5;\n      reasoning.push('Individual contributor = +5
points');\n    }\n    \n    const targetIndustries =
['technology', 'software', 'saas', 'fintech'];\n    if
(targetIndustries.some(target =>
industry.toLowerCase().includes(target))) {\n      score += 25;\n
reasoning.push(`Target industry (${industry}) = +25 points`);\n
} else {\n      score += 5;\n      reasoning.push(`Non-target
industry (${industry}) = +5 points`);\n    }\n  }\n\n  //
Determine persona\n  if (score >= 70) {\n    persona = 'P1';\n  }
else if (score >= 40) {\n    persona = 'P2';\n  } else {\n
persona = 'P3';\n  }\n\n  const qualifiedLead = {\n
...leadData,\n    linkedin_data: {\n      profile_url:
(linkedinData && linkedinData.linkedin_profile_url) || '',\n
headline: (linkedinData && linkedinData.headline) || '',\n
industry: (linkedinData && linkedinData.industry) || ''\n    },\n
company_data: {\n      domain: (companyData && companyData.domain)
|| '',\n      employees: companySize,\n      industry: industry,\n
revenue: (companyData && companyData.revenue) || 'Unknown'\n
},\n    lead_characteristics: {\n      company_size:
companySize,\n      seniority_level: seniorityLevel,\n
industry: industry\n    },\n    qualification: {\n      persona:
persona,\n      score: Math.round(score),\n      reasoning:
reasoning,\n      qualified_at: new Date().toISOString()\n    },\n
processing_stage: 'qualified'\n  };\n\n  return [{ json:
qualifiedLead }];\n\n} catch (error) {\n  return [{\n    json: {\n
error: true,\n      error_message: error.message,\n
error_stack: error.stack,\n      node_name: 'Dynamic Lead
Qualification Engine',\n      timestamp: new
Date().toISOString()\n    }\n  }];\n}"
      },
      "id": "qualification-engine",
      "name": "Dynamic Lead Qualification Engine",
      "type": "n8n-nodes-base.code",
      "typeVersion": 2,
      "position": [1000, 400]
    },
    {
      "parameters": {
        "jsCode": "// SDR Assignment Engine with Your Team
Priority Order\ntry {\n  let lead = {};\n  try {\n    lead =
$('Dynamic Lead Qualification Engine').item.json;\n  } catch
(error) {\n    throw new Error('Dynamic Lead Qualification Engine
node failed or missing');\n  }\n\n  if (!lead ||
!lead.qualification || !lead.qualification.persona) {\n    throw
new Error('Invalid lead data: missing qualification or
persona');\n  }\n\n  const persona =
lead.qualification.persona;\n\n  // Your SDR team in priority
order\n  const sdrTeam = [\n    { \n      id: 'owais_tariq', \n
```

```
name: 'Owais Tariq', \n      email: 'owais@yourcompany.com', \n
calendar_id: 'owais@yourcompany.com', \n      priority: 1,\n
current_load: 10,\n      max_capacity: 15,\n      available:
true\n    },\n    { \n      id: 'sahil_gill', \n      name: 'Sahil
Gill', \n      email: 'sahil@yourcompany.com', \n
calendar_id: 'sahil@yourcompany.com', \n      priority: 2,\n
current_load: 12,\n      max_capacity: 15,\n      available:
true\n    },\n    { \n      id: 'wajahat_mirza', \n      name:
'Wajahat Mirza', \n      email: 'wajahat@yourcompany.com', \n
calendar_id: 'wajahat@yourcompany.com', \n      priority: 3,\n
current_load: 8,\n      max_capacity: 15,\n      available: true\n
},\n    { \n      id: 'prasanthi_basava', \n      name: 'Prasanthi
Basava', \n      email: 'prasanthi@yourcompany.com', \n
calendar_id: 'prasanthi@yourcompany.com', \n      priority: 4,\n
current_load: 9,\n      max_capacity: 15,\n      available: true\n
},\n    { \n      id: 'ashish_gururaj', \n      name: 'Ashish
Gururaj', \n      email: 'ashish@yourcompany.com', \n
calendar_id: 'ashish@yourcompany.com', \n      priority: 5,\n
current_load: 11,\n      max_capacity: 15,\n      available:
true\n    },\n    { \n      id: 'albert_nam', \n      name:
'Albert Nam', \n      email: 'albert@yourcompany.com', \n
calendar_id: 'albert@yourcompany.com', \n      priority: 6,\n
current_load: 7,\n      max_capacity: 15,\n      available: true\n
}\n  ];\n\n  // Assignment strategy based on persona\n  let
startIndex = 0;\n  let assignmentReason = '';\n  \n  if (persona
=== 'P1') {\n    startIndex = 0; // Start with Owais\n
assignmentReason = 'P1 lead - starting with highest priority
SDR';\n  } else if (persona === 'P2') {\n    startIndex = 1; //
Start with Sahil\n    assignmentReason = 'P2 lead - starting with
second priority SDR';\n  } else {\n    startIndex = 2; // Start
with Wajahat\n    assignmentReason = 'P3 lead - starting with
third priority SDR';\n  }\n\n  // Find first available SDR\n  let
selectedSDR = null;\n  let fallbackUsed = false;\n\n  for (let i =
startIndex; i < sdrTeam.length; i++) {\n    const sdr =
sdrTeam[i];\n    if (sdr.available && sdr.current_load <
sdr.max_capacity) {\n      selectedSDR = sdr;\n      break;\n
}\n  }\n\n  if (!selectedSDR) {\n    fallbackUsed = true;\n    for
(let i = 0; i < startIndex; i++) {\n      const sdr =
sdrTeam[i];\n      if (sdr.available && sdr.current_load <
sdr.max_capacity) {\n        selectedSDR = sdr;\n        break;\n
}\n    }\n  }\n\n  if (!selectedSDR) {\n    selectedSDR =
sdrTeam.sort((a, b) => a.current_load - b.current_load)[0];\n
fallbackUsed = true;\n  }\n\n  // Calculate meeting schedule\n
const scheduleDays = persona === 'P1' ? 0 : persona === 'P2' ? 1 :
2;\n  const scheduledDate = new Date();\n
scheduledDate.setDate(scheduledDate.getDate() + scheduleDays);\n
\n  if (scheduledDate.getHours() < 9 || scheduledDate.getHours()
>= 17) {\n    scheduledDate.setHours(14, 0, 0, 0);\n  }\n\n  const
```

```
meetingEnd = new Date(scheduledDate);\n
meetingEnd.setMinutes(meetingEnd.getMinutes() + 15); // All
meetings 15 minutes\n\n  const assignment = {\n    ...lead,\n
assignment: {\n      sdr_id: selectedSDR.id,\n      sdr_name:
selectedSDR.name,\n      sdr_email: selectedSDR.email,\n
sdr_calendar_id: selectedSDR.calendar_id,\n      sdr_priority:
selectedSDR.priority,\n      assigned_at: new
Date().toISOString(),\n      assignment_reason: `${persona} lead
assigned to ${selectedSDR.name} (Priority
#${selectedSDR.priority}, Load:
${selectedSDR.current_load}/${selectedSDR.max_capacity})`,\n
fallback_used: fallbackUsed,\n      original_strategy:
assignmentReason\n    },\n    meeting: {\n      start_time:
scheduledDate.toISOString(),\n      end_time:
meetingEnd.toISOString(),\n      duration_minutes: 15,\n
priority: persona === 'P1' ? 'high' : persona === 'P2' ? 'medium'
: 'low',\n      title: `${persona} Demo - ${lead.name ||
'Unknown'} (${lead.company || 'Unknown Company'})`,\n
description: `${persona} Priority Demo Call\\n\\nLead: ${lead.name
|| 'Unknown'}\\nCompany: ${lead.company || 'Unknown'}\\nTitle:
${lead.job_title || 'Unknown'}\\nScore:
${lead.qualification.score}/100\\n\\nAssigned to:
${selectedSDR.name} (Priority #${selectedSDR.priority})`,\n
scheduled_for_days: scheduleDays\n    },\n    processing_stage:
'assigned'\n  };\n\n  console.log(`Assigned ${persona} lead
(${lead.name}) to ${selectedSDR.name} (Priority
#${selectedSDR.priority})`);\n  \n  return [{ json: assignment
}];\n\n} catch (error) {\n  console.error('SDR Assignment Error:',
error.message);\n  \n  return [{\n    json: {\n      error:
true,\n      error_message: error.message,\n      error_stack:
error.stack,\n      node_name: 'SDR Assignment Engine',\n
timestamp: new Date().toISOString()\n    }\n  }];\n}"
      },
      "id": "sdr-assignment",
      "name": "SDR Assignment Engine",
      "type": "n8n-nodes-base.code",
      "typeVersion": 2,
      "position": [1200, 400]
    },
    {
      "parameters": {
        "jsCode": "// Mock Calendar Creation (No OAuth2
required)\ntry {\n  const assignment = $('SDR Assignment
Engine').item.json;\n\n  const mockEvent = {\n    ...assignment,\n
calendar_event: {\n      id: `mock_event_${Date.now()}`,\n
status: 'confirmed',\n      htmlLink:
'https://calendar.google.com/calendar/mock-event',\n
hangoutLink: 'https://meet.google.com/abc-defg-hij',\n
```

```
created: new Date().toISOString(),\n      summary:
assignment.meeting.title,\n      start: {\n        dateTime:
assignment.meeting.start_time\n      },\n      end: {\n
dateTime: assignment.meeting.end_time\n      },\n      attendees:
[\n        {\n          email: assignment.assignment.sdr_email,\n
displayName: assignment.assignment.sdr_name\n        },\n
{\n          email: assignment.email,\n          displayName:
assignment.name\n        }\n      ]\n    },\n    calendar_created:
true,\n    processing_stage: 'calendar_complete'\n  };\n\n
console.log(`Mock calendar event created for ${assignment.name}
with ${assignment.assignment.sdr_name}`);\n\n  return [{ json:
mockEvent }];\n  \n} catch (error) {\n  return [{\n    json: {\n
error: true,\n      error_message: error.message,\n
node_name: 'Mock Calendar Creation',\n      timestamp: new
Date().toISOString()\n    }\n  }];\n}"
      },
      "id": "create-calendar-event",
      "name": "Mock Calendar Creation",
      "type": "n8n-nodes-base.code",
      "typeVersion": 2,
      "position": [1400, 400]
    },
    {
      "parameters": {
        "operation": "append",
        "documentId": "your-lead-tracking-sheet-id",
        "sheetName": "Lead Pipeline",
        "columns": {
          "mappingMode": "defineBelow",
          "value": {
            "timestamp": "={{ $json.received_at }}",
            "session_id": "={{ $json.session_id }}",
            "name": "={{ $json.name }}",
            "email": "={{ $json.email }}",
            "company": "={{ $json.company }}",
            "job_title": "={{ $json.job_title }}",
            "phone": "={{ $json.phone }}",
            "persona": "={{ $json.qualification.persona }}",
            "score": "={{ $json.qualification.score }}",
            "company_size": "={{
$json.lead_characteristics.company_size }}",
            "seniority": "={{
$json.lead_characteristics.seniority_level }}",
            "industry": "={{ $json.lead_characteristics.industry
}}",
            "assigned_sdr": "={{ $json.assignment.sdr_name }}",
            "sdr_priority": "={{ $json.assignment.sdr_priority
}}",
```

```
            "meeting_time": "={{ $json.meeting.start_time }}",
            "meeting_priority": "={{ $json.meeting.priority }}",
            "duration_minutes": "15",
            "qualification_reasoning": "={{
$json.qualification.reasoning.join('; ') }}",
            "assignment_reason": "={{
$json.assignment.assignment_reason }}",
            "fallback_used": "={{ $json.assignment.fallback_used
}}",
            "linkedin_profile": "={{
$json.linkedin_data.profile_url }}",
            "calendar_event_id": "={{ $json.calendar_event.id }}",
            "status": "scheduled",
            "created_at": "={{ DateTime.now().toISO() }}"
          }
        }
      },
      "id": "update-tracking",
      "name": "Update Lead Tracking Sheet",
      "type": "n8n-nodes-base.googleSheets",
      "typeVersion": 4,
      "position": [1600, 300]
    },
    {
      "parameters": {
        "url":
"https://hooks.slack.com/services/YOUR/SLACK/WEBHOOK",
        "sendBody": true,
        "bodyParameters": {
          "parameters": [
            {
              "name": "text",
              "value": ":dart: *NEW {{ $json.qualification.persona
}} LEAD ASSIGNED*\\n\\n:bust_in_silhouette: *Lead Details:*\\n•
*Name:* {{ $json.name }} ({{ $json.job_title }})\\n• *Company:* {{
$json.company }} ({{ $json.lead_characteristics.company_size }}
employees)\\n• *Industry:* {{ $json.lead_characteristics.industry
}}\\n• *Score:* {{ $json.qualification.score
}}/100\\n\\n:technologist: *Assignment:*\\n• *SDR:* {{
$json.assignment.sdr_name }} (Priority #{{
$json.assignment.sdr_priority }})\\n• *Meeting:* {{
DateTime.fromISO($json.meeting.start_time).toFormat('EEE, MMM dd
at h:mm a') }}\\n• *Duration:* 15 minutes\\n• *Priority:* {{
$json.meeting.priority }}\\n\\n:clipboard: *Top
Qualifications:*\\n{{ $json.qualification.reasoning.slice(0,
2).join('\\n') }}\\n\\n:white_check_mark: {{
$json.assignment.assignment_reason }}\\n\\nReady for demo!
```

```
:rocket:\\n\\n_Contact: {{ $json.email }} | Phone: {{ $json.phone
|| 'Not provided' }}_"
            }
          ]
        }
      },
      "id": "slack-notification",
      "name": "Slack Notification",
      "type": "n8n-nodes-base.httpRequest",
      "typeVersion": 4.1,
      "position": [1600, 500]
    }
  ],
  "connections": {
    "Calendly Demo Signup": {
      "main": [
        [
          {
            "node": "Extract Lead Data",
            "type": "main",
            "index": 0
          }
        ]
      ]
    },
    "Extract Lead Data": {
      "main": [
        [
          {
            "node": "LinkedIn Profile Lookup",
            "type": "main",
            "index": 0
          },
          {
            "node": "Company Data Enrichment",
            "type": "main",
            "index": 0
          }
        ]
      ]
    },
    "LinkedIn Profile Lookup": {
      "main": [
        [
          {
            "node": "Fetch ICP Qualification Criteria",
            "type": "main",
            "index": 0
```

```
      }
    ]
  ]
},
"Company Data Enrichment": {
  "main": [
    [
      {
        "node": "Fetch ICP Qualification Criteria",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Fetch ICP Qualification Criteria": {
  "main": [
    [
      {
        "node": "Dynamic Lead Qualification Engine",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Dynamic Lead Qualification Engine": {
  "main": [
    [
      {
        "node": "SDR Assignment Engine",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"SDR Assignment Engine": {
  "main": [
    [
      {
        "node": "Mock Calendar Creation",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
```

```
    "Mock Calendar Creation": {
      "main": [
        [
          {
            "node": "Update Lead Tracking Sheet",
            "type": "main",
            "index": 0
          },
          {
            "node": "Slack Notification",
            "type": "main",
            "index": 0
          }
        ]
      ]
    }
  },
  "createdAt": "2024-01-01T00:00:00.000Z",
  "updatedAt": "2024-01-01T00:00:00.000Z",
  "settings": {},
  "staticData": null,
  "meta": {},
  "pinData": {},
  "versionId": "1",
  "active": false
}
```