Nome do Grupo: gaia
Integrantes: Claudia Alvim e Frances Tanure

I2A2-Agentes - Trabalho Final

Sumário Executivo

A **GAIA - Agentes EDA** é uma solução de agentes de análise exploratória e preditiva para dados fiscais (NF-e), implementado sobre n8n v1.113.3 (Render), PostgreSQL e OpenAI. Ele recebe perguntas em linguagem natural, traduz para um plano estruturado, executa consultas SQL no BD relacional e devolve uma página HTML responsiva com KPIs, tabela e gráficos (Chart.js).
Há dois fluxos complementares: Padrão (EDA) e Preditivo.

Tema:

Análise exploratória e preditiva de notas fiscais, consolidando receita, quantidade, número de notas, ticket médio, e consultas por UF, tempo, fornecedor, cliente, CFOP, com possibilidade de direcionar análises gerenciais e de governança fiscal. O tema se encaixa nos “Temas que podem ser tratados” do curso, em especial Ferramentas Gerenciais (relatórios personalizados, análises preditivas e simulações) e Automação Fiscal/Contábil.

Público-alvo:

Times financeiros, contábeis e de BI de empresas que precisam monitorar vendas ou receitas por período, UF e parceiro (fornecedor/cliente), além de nutrir decisões com tendências e cenários.

Justificativa:

Reduz tempo de fechamento e erros manuais, amplia visibilidade sobre drivers de receita e oferece uma camada assistida por IA para perguntas ad-hoc e análises gerenciais, diretamente citadas nos objetivos de projeto.

gaia

Arquitetura da Solução

Visão geral (camadas):

1. Interface Web (HTML/Chart.js)
   Formulário + renderização dinâmica do dashboard; retorna KPIs, gráfico de série temporal e tabela.

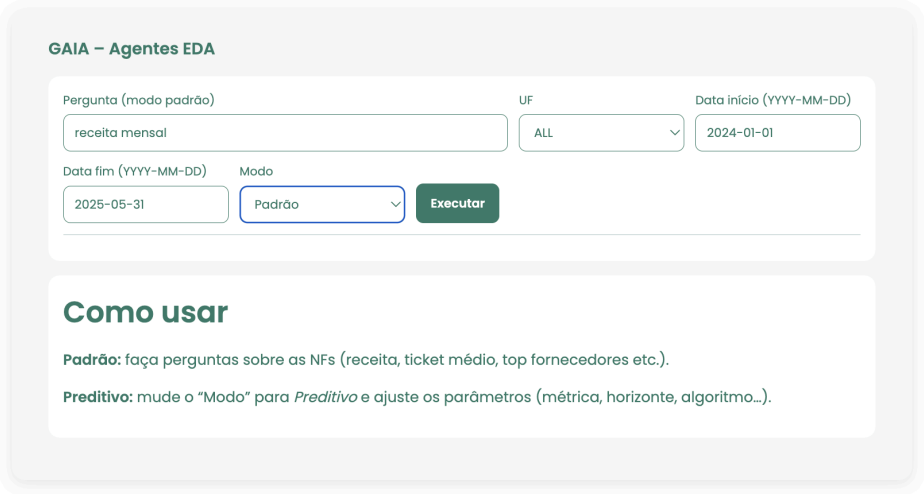
2. Orquestração (n8n – Render)
   São dois fluxos que se dividem a partir dos parâmetros de entrada: relatórios EDA e análise preditiva.

3. Dados (PostgreSQL)
   Armazena datasets (linhas normalizadas), schema, plan e histórico de perguntas/sessões, Agentes Autônomos, relatórios etc

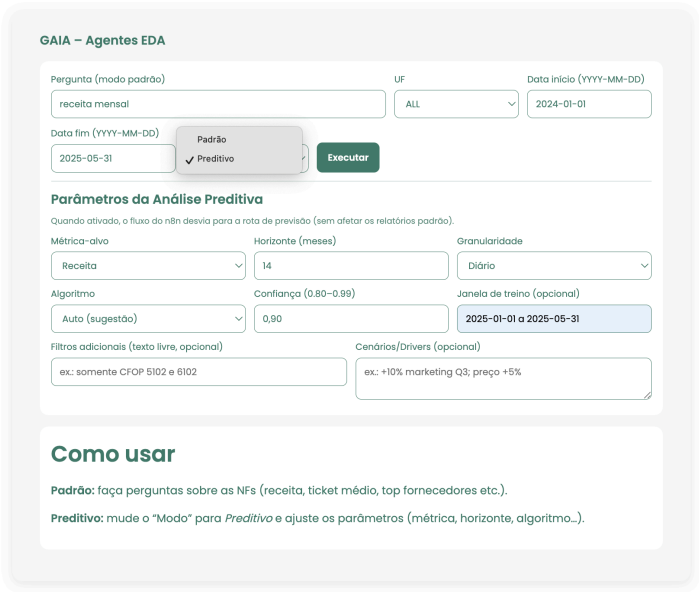
4. Modelo (OpenAI)
   Gera o plano JSON de análise (intent/metrics/viz/correlação), e, quando habilitado, insights.

Interface e Usabilidade

Front-end user friendly - com explicação de como usar, facilitando análises recorrentes.



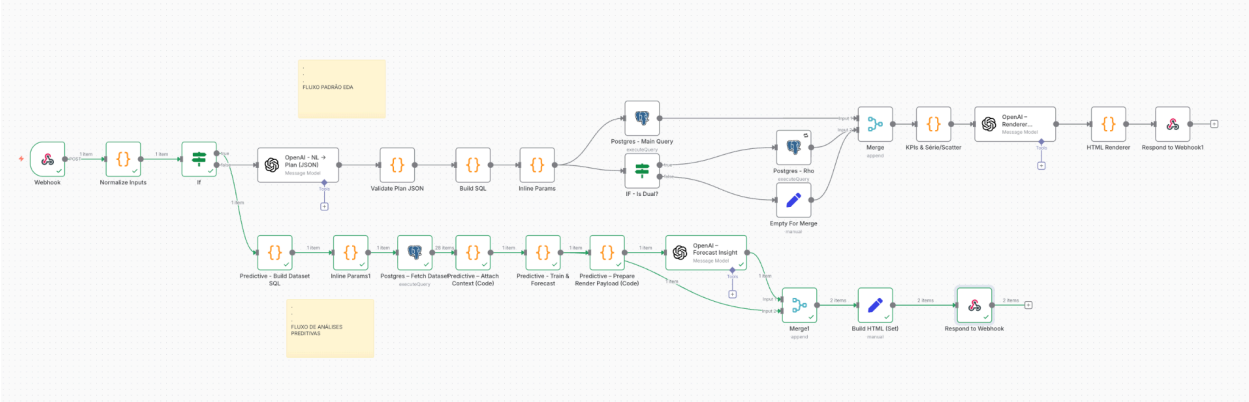
Se o usuário optar por análises preditivas, o formulário abre mais campos para preencher como granularidade, horizonte, confiança entre outros.



O formulário sempre é retornado junto ao resultado para que o usuário possa fazer perguntas recorrentes. O resultado mostra KPIs, Gráficos, Análises e detalhes das categorias.

Fluxos

Dependendo da escolha do usuário ao selecionar “Modo: Padrão ou Preditivo” na interface, o fluxo caminha para uma lógica diferente EDA ou análise preditiva.



FLUXO PADRÃO (EDA):

Webhook (entrada) — recebe os campos do formulário (pergunta, UF, datas, viz) e dispara o fluxo.

Normalize Inputs — padroniza os parâmetros (UF, `data_inicio`, `data_fim`, `user_viz`) e prepara o payload básico.

Planner (LLM) — interpreta a pergunta e propõe um `plan` (intent, group_by, metrics, viz, filtros).

Validate Plan — higieniza/normaliza o `plan` e injeta `data_inicio`, `data_fim`, `uf` e `user_viz` vindos do Normalize.

Build SQL (Code) — traduz o `plan` para SQL (seleções, agregações, filtros e ordenações).

Execute SQL — roda a query no dataset de NFs e retorna as linhas agregadas.

Post-Process (Code) — calcula campos derivados (ex.: ticket_médio), formata números e limita registros.

OpenAI – Insight — gera um parágrafo curto interpretando os resultados (tendências, destaques).

Build HTML — monta o HTML da UI padrão (título, filtros usados, tabela, gráficos conforme `viz`).

Respond to Webhook (Immediately) — devolve o HTML renderizável para o navegador/cliente.

FLUXO DE ANÁLISE PREDITIVA - RESUMIDA

Parâmetros (métrica-alvo, horizonte, granularidade, algoritmo, confiança) são exibidos e, quando o modo está ativo, o fluxo desvia para nós preditivos sem quebrar o padrão.

Webhook (entrada) — recebe os parâmetros do formulário (pergunta, UF, datas, modo) e inicia a execução.

Normalize Inputs — padroniza os campos e deriva `is_predictive` + `pred_params` (métrica, granularidade, horizonte, etc.).

Router / IF (Modo = Preditivo?) — desvia para a rota preditiva quando `is_predictive = true`.

Prepare Time Series (SQL) — extrai e agrega a série histórica conforme UF, período e granularidade.

Validate Series (Code) — confere se a série tem pontos suficientes; em caso negativo, monta payload explicativo.

Forecast (Model Selector) — escolhe o algoritmo (auto/prophet/arima/etc.) e calcula previsão e bandas de confiança.

OpenAI – Forecast Insight — gera um resumo em linguagem natural com os principais achados.

Build Payload (Code) — organiza `series`, `bands`, `kpi`, `table` e `insight` no formato único da UI.

Render HTML (Build HTML) — injeta `window.__PAYLOAD__` no template (Chart.js + KPIs + tabela).

Respond to Webhook (Last Node) — retorna o HTML final para o navegador/cliente.

DADOS E LIMITAÇÕES

- A base disponível, nesta entrega, não é rica/contínua: os dados mostrados concentram-se praticamente em jan/2024 e mai/2025, apenas dois meses, com salto expressivo na receita em maio/2025 (≈ R$ 3,06 bi). Isso restringe robustez de tendências/diagnósticos.

- Implicação: séries curtas comprometem modelos preditivos (horizontes longos e bandas de confiança ficam artificiais). Recomenda-se ampliar a amostragem temporal e cobertura de UFs/CFOPs/itens. Com está base, é recomendável testar a granularidade diária do mês de maio de 2025.

Catálogo de perguntas (EDA)

O agente entende perguntas em linguagem natural.

Exemplos de perguntas que o agente responde:

1. Agregações

- “Qual a receita mensal no período?”
- “Qual o ticket médio por mês/UF?”
- “Top 10 fornecedores por receita (período/UF)?”
- “Top itens por quantidade?”

2. Distribuição/Estatística

- “Qual a distribuição de notas por UF?” / “Qual a variabilidade (desvio, variância, mediana) da receita diária?”
  Agentes Autônomos – Relatório...

3. Tendência Temporal

- “Como evolui a receita ao longo do tempo?” / “Qual a tendência do ticket médio no período?”
  Agentes Autônomos – Relatório...

4. Correlação/Outliers

- “Há correlação entre receita e quantidade?”
- “Quais outliers de valor por nota?”

5. Filtros e Recortes

- “Receita mensal somente SP entre 2025-05-01 e 2025-06-30.”
- “Top 5 fornecedores por receita em RJ.”

6. Perguntas de governança

- “Quais CFOPs mais recorrentes?”
- “Quais clientes concentram maior receita?”

Testes e validação

- Pela URL - https://gaia-ft4y.onrender.com/index_trabalho_final.html
- Teste por cURL (produção):

```
curl -X POST 'https://n8n-service-to7f.onrender.com/webhook/trabalho_final' \
--data-urlencode 'pergunta=receita mensal' \
--data-urlencode 'uf=ALL' \
--data-urlencode 'data_inicio=2025-05-01' \
--data-urlencode 'data_fim=2025-06-30' \
--data-urlencode 'modo=padrão'
```

- Retorna a página HTML contendo KPIs/gráfico/tabela

Roadmap de evolução

- Dados: ampliar faixa histórica (≥ 12–24 meses), normalizar itens como inteiro, adicionar dimensões (produto, canal, CFOP, CST).

- Preditivo: habilitar Prophet/ARIMA/XGBoost com granularidade configurável e bandas de confiança; preservar rota do fluxo padrão (já previsto na UI).
  GAIA – Agentes EDA

- Insights LLM: reinstalar nó de Conclusões baseado em histórico de perguntas (consta no trabalho anterior).
- Acesso a bases externas com impostos que incidiram nas NFs.

Conclusão

O **GAIA - Agentes EDA** cumpre o papel de assistente analítico: agrega métricas-chave, apresenta gráficos de séries temporais com tendências e segmentações por UF/tempo/entidades e prepara o terreno para previsões e insights gerados por LLM. Portanto, cumpre seu objetivo de auxiliar executivos em tomadas de decisão.

Github

https://github.com/claudiabsa/gaia

JSON


```
{
  "name": "I2A2_trabalho_final fornecedor",
  "nodes": [
    {
      "parameters": {
        "httpMethod": "POST",
        "path": "trabalho_final",
        "responseMode": "responseNode",
        "options": {
          "responseHeaders": {
            "entries": [
```

```
        {
          "name": "Content-Type",
          "value": "text/html; charset=utf-8"
        },
        {
          "name": "Cache-Control",
          "value": "no-store"
        },
        {
          "name": "X-Content-Type-Options",
          "value": "nosniff"
        },
        {
          "name": "Referrer-Policy",
          "value": "no-referrer"
        }
      ]
    }
  }
},
"type": "n8n-nodes-base.webhook",
"typeVersion": 2.1,
"position": [
```

```
    -640,
    0
  ],
  "id": "9f24f78d-c477-4cb2-876e-66f5db3a3b27",
  "name": "Webhook",
  "webhookId": "c331ce41-9132-4977-8e63-537924e5a212"
},
{
  "parameters": {
    "respondWith": "text",
    "responseBody": "={{$json.html}}\n\n",
    "options": {
      "responseCode": 200,
      "responseHeaders": {
        "entries": [
          {
            "name": "Content-Type",
            "value": "text/html; charset=utf-8"
          },
          {
            "name": "Cache-Control",
            "value": "no-store"
          },
```

```
          {
            "name": "X-Content-Type-Options",
            "value": "nosniff"
          }
        ]
      }
    }
  },
  "type": "n8n-nodes-base.respondToWebhook",
  "typeVersion": 1.4,
  "position": [
    1904,
    400
  ],
  "id": "75f4116b-26fe-4912-9b88-a03e6d641c19",
  "name": "Respond to Webhook"
},
{
  "parameters": {
    "jsCode": "// --- Helpers ---\nfunction parseIntervalo(str) {\n  if (!str) return { start: null, end: null };\n  const m = String(str).trim().match(\n /(\\d{4}-\\d{2}-\\d{2})\\s*(?:a|até|to|\\-|–|—)\\s*(\\d{4}-\\d{2}-\\d{2})/i\n );\n  return m ? { start: m[1], end: m[2] } : { start: null, end: null };\n}\n\nfunction normGran(g) {\n  const s = String(g || '').toLowerCase();\n  if (['diario','diária','diaria','daily','day','d'].includes(s)) return 'diario';\n  if (['anual','yearly','year','a','y'].includes(s)) return 'anual';\n
```

return 'mensal';\n}\n\nfunction toInt(v, def=6) {\n const n = parseInt(String(v).trim(), 10);\n return Number.isFinite(n) ? n : def;\n}\n\nfunction toFloat(v, def=0.8) {\n if (v === null || v === undefined) return def;\n const s = String(v).replace(',', '.');\n const n = parseFloat(s);\n return Number.isFinite(n) ? n : def;\n}\n\n// --- Entrada do webhook já normalizada no seu nó ---\nconst body = ($json.body ?? {}); // ou o objeto que você já usa\nconst modo = String(body.modo || '').toLowerCase();\n\n// Captura e parse do intervalo de treino\nconst { start: treino_inicio, end: treino_fim } = parseIntervalo(body.treino_intervalo);\n\n// Normalizações\nconst is_predictive =\n modo === 'preditivo' || modo === 'predictive' || modo === 'on' ||\n String(body.is_predictive).toLowerCase() === 'true';\n\nconst pred_target = body.target_metric || 'receita';\nconst pred_gran = normGran(body.granularidade || 'mensal');\nconst pred_horizonte= toInt(body.horizonte, 6);\nconst pred_conf = toFloat(body.confianca, 0.8);\nconst pred_algo = String(body.algoritmo || 'auto').toLowerCase();\n\n// Monte (ou sobrescreva) pred_params no payload final:\nconst out = {\n ...$json,\n is_predictive,\n pred_params: {\n target_metric: pred_target,\n granularidade: pred_gran,\n horizonte: pred_horizonte,\n confianca: pred_conf,\n algoritmo: pred_algo,\n treino_inicio, // ← agora preenchido\n treino_fim, // ← agora preenchido\n grupo_dim: 'TOTAL'\n }\n};\n\nreturn [{ json: out }];\n"

    },

    "type": "n8n-nodes-base.code",

    "typeVersion": 2,

    "position": [

      -432,

      0

    ],

    "id": "1690a517-300d-4d21-b62c-f04cd7ef2bc5",

    "name": "Normalize Inputs"

  },

  {

    "parameters": {

"jsCode": "// === Validate Plan JSON (robusto) ===\n// Lê a saída do Planner (pode vir em vários formatos) e normaliza em\n// { plan, data_inicio, data_fim, uf, user_viz }.\n\n// ---------- Helpers ----------\nfunction safeParse(obj) {\n if (!obj) return null;\n if (typeof obj === 'object') return obj;\n if (typeof obj === 'string') {\n const s = obj.replace(/```json/gi, '```').replace(/```/g, '').trim();\n try { return JSON.parse(s); } catch { return null; }\n }\n return null;\n}\n\nfunction pickFirst(arr) {\n return Array.isArray(arr) && arr.length ? arr[0] : null;\n}\n\n// Tenta extrair um único objeto \"plano\" válido a partir de várias formas comuns\nfunction extractPlan(raw) {\n if (!raw) return null;\n\n // 1) Se já tem intent na raiz, ótimo\n if (typeof raw === 'object' && raw.intent) return raw;\n\n // 2) Se veio como array (ex.: [{ index, message: { content: {...} } }])\n const maybeArrFirst = pickFirst(raw);\n if (maybeArrFirst) {\n const p2 = extractPlan(maybeArrFirst);\n if (p2) return p2;\n }\n\n // 3) Se tem .data (alguns nodes colocam assim)\n if (raw.data) {\n const d = Array.isArray(raw.data) ? pickFirst(raw.data) : raw.data;\n const p3 = extractPlan(d);\n if (p3) return p3;\n }\n\n // 4) Caminhos típicos do OpenAI:\n // - message.content (objeto ou string JSON)\n // - choices[0].message.content\n if (raw.message && raw.message.content) {\n const p4 = safeParse(raw.message.content) || raw.message.content;\n if (p4 && typeof p4 === 'object' && p4.intent) return p4;\n }\n if (raw.choices && raw.choices[0] && raw.choices[0].message && raw.choices[0].message.content) {\n const p5 = safeParse(raw.choices[0].message.content) || raw.choices[0].message.content;\n if (p5 && typeof p5 === 'object' && p5.intent) return p5;\n }\n\n // 5) Última tentativa: parsear o próprio raw se for string JSON\n const p6 = safeParse(raw);\n if (p6 && p6.intent) return p6;\n\n return null;\n}\n\n// ---------- Captura do \"Normalize Inputs\" ----------\n// (NÃO MUDE O NOME do node)\nconst niItems = (() => { try { return $items(\"Normalize Inputs\"); } catch { return []; } })();\nconst Ni = (niItems && niItems.length ? niItems[0].json : {}) || {};\n// O Normalize coloca os campos do formulário dentro de body.\n// Fazemos o merge para que fiquem acessíveis no topo também.\nconst NiMerged = Ni.body ? { ...Ni, ...Ni.body } : Ni;\n\n// ---------- Tabelas de validação ----------\nconst intents = new Set([\n 'receita_por_produto',\n 'receita_por_cliente',\n 'receita_por_cfop',\n 'receita_mensal',\n 'ticket_medio',\n 'correlacao',\n 'top_fornecedor_receita',\n 'top_item_quantidade',\n 'top_item_receita',\n]);\nconst vizes = new Set(['auto','time_series','bar','pie','scatter']);\nconst corrVars = new Set(['receita','qtd','ticket_medio']);\n\n// ---------- Entrada do Planner ----------\nlet planRaw = ($json && $json.data) ? $json.data : $json;\n\n// Extrai o plano\nlet plan = extractPlan(planRaw) || {};\n\n// --------- Fallbacks/normalizações do plano ----------\nif (!intents.has(plan.intent)) plan.intent = 'receita_mensal';\n\n// group_by: padrão para intents mensais/temporal\nif (!Array.isArray(plan.group_by)) {\n plan.group_by = (plan.intent === 'receita_mensal' || plan.intent ===

'ticket_medio') ? ['mes'] : [];\n}\n\n// metrics padrão\nif (!Array.isArray(plan.metrics) || !plan.metrics.length) {\n plan.metrics = ['receita','qtd','notas','ticket_medio'];\n}\n\n// filtros UF (com saneamento do escopo)\nif (!plan.filters) plan.filters = {};\nconst validScopes = new Set(['DESTINATARIO','REMETENTE']);\nif (!validScopes.has(plan.filters.uf_scope)) {\n plan.filters.uf_scope = 'DESTINATARIO';\n}\nif (!plan.filters.uf_value) {\n plan.filters.uf_value = NiMerged.uf || 'ALL';\n}\n\n// viz (respeita user_viz/viz do Normalize, se vier)\nif (!vizes.has(plan.viz)) {\n plan.viz = NiMerged.user_viz || NiMerged.viz || 'auto';\n}\n\n// limit\nif (typeof plan.limit !== 'number') plan.limit = 50;\n\n// Correlação\nif (plan.intent === 'correlacao') {\n plan.viz = 'scatter';\n if (!plan.correlation) plan.correlation = { x:'receita', y:'qtd' };\n if (!corrVars.has(plan.correlation.x)) plan.correlation.x = 'receita';\n if (!corrVars.has(plan.correlation.y)) plan.correlation.y = 'qtd';\n}\n\n// ---------- Saída ----------\nreturn [{\n json: {\n   plan,\n   data_inicio: NiMerged.data_inicio,\n   data_fim: NiMerged.data_fim,\n   uf: NiMerged.uf ?? 'ALL',\n   user_viz: NiMerged.user_viz ?? NiMerged.viz ?? 'auto'\n }\n}];\n"

    },

    "type": "n8n-nodes-base.code",

    "typeVersion": 2,

    "position": [

      320,

      16

    ],

    "id": "4998879b-cd14-45c1-a152-9ff2ccdae73c",

    "name": "Validate Plan JSON"

  },

  {

    "parameters": {

      "jsCode": "/**\n * Consolida resultados do(s) Postgres em payload para render.\n * Lê (sempre via $items, não $prevNode):\n * - Build SQL: viz/label/mode/context\n * - Validate Plan JSON: plan (para scatter x/y)\n * - Postgres - Main Query: linhas\n * -

```
Postgres - Rho: rho (se correlação)\n *  - Normalize Inputs: pergunta (para mostrar no HTML)\n */\nfunction num(n){ const x = Number(n); return isFinite(x) ? x : 0; }\nfunction fmtDate10(s){ return (s || '').toString().slice(0,10); }\n\n// --- Fetch robusto via $items(...)\nconst getOne = (name) => {\n  try {\n    const it = $items(name);\n return (it && it.length ? it[0].json : {}) || {};\n  } catch { return {}; }\n};\nconst getMany = (name) => {\n  try { return $items(name).map(i => i.json || {}); }\n  catch { return []; }\n};\n\n// Nós de onde lemos\nconst buildNode = getOne(\"Build SQL\");          // { viz,label,mode,context, ... }\nconst planNode  = getOne(\"Validate Plan JSON\");      // { plan, ... }\nconst niNode    = getOne(\"Normalize Inputs\");          // { pergunta, ... }\n\nconst rows      = getMany(\"Postgres - Main Query\");   // linhas de dados\nconst rhoArr    = getMany(\"Postgres - Rho\");         // [{ rho }]\nconst rho       = (rhoArr.length && rhoArr[0].rho != null) ? Number(rhoArr[0].rho) : null;\n\n// Metadados\nconst viz = buildNode.viz || 'time_series';\nconst label   = buildNode.label || 'Relatório';\nconst mode        = buildNode.mode || 'single';\nconst context = buildNode.context || {};\nconst plan        = planNode.plan || {};\nconst corrX     = plan.correlation?.x || 'receita';\nconst corrY       = plan.correlation?.y || 'qtd';\nconst lastQuery = (niNode.pergunta || '');\n\n// KPIs\nlet total = 0, itens = 0, notas = 0;\nfor (const r of rows) {\n  if ('receita' in r) total += num(r.receita);\n  if ('qtd'       in r) itens += num(r.qtd);\n  if ('notas'   in r) notas  += num(r.notas);\n}\nconst ticket = notas ? (total / notas) : (itens ? (total / itens) : 0);\n\n// Estruturas para charts\nlet series  = { labels: [], data: [] };\nlet pie     = { labels: [], data: [] };\nlet scatter = { xKey: corrX, yKey: corrY, points: [], rho };\n\n// Mapeamento por viz\nif (viz === 'time_series') {\n  const agg = {};\n  for (const r of rows) {\n    const m = fmtDate10(r.mes);\n    if (!m) continue;\n    agg[m] = (agg[m] || 0) + num(r.receita);\n  }\n  const labels = Object.keys(agg).sort();\n  const data = labels.map(k => agg[k]);\n  series = { labels, data };\n\n} else if (viz === 'bar' || viz === 'pie') {\n  const topNDisplay = 12;\n  const sorted = rows\n    .filter(r => r.categoria != null)\n    .sort((a,b) => num(b.receita) - num(a.receita));\n  const labelsAll = sorted.map(r => String(r.categoria));\n  const dataAll   = sorted.map(r => num(r.receita));\n  series = { labels: labelsAll.slice(0, topNDisplay), data: dataAll.slice(0, topNDisplay) };\n\n  if (viz === 'pie' && dataAll.length > topNDisplay) {\n    const somaOutros = dataAll.slice(topNDisplay).reduce((s,x)=>s+num(x),0);\n    pie = { labels: series.labels.concat(['Outros']), data: series.data.concat([somaOutros]) };\n  } else {\n    pie = { labels: series.labels, data: series.data };\n  }\n\n} else if (viz === 'scatter') {\n  const pts = [];\n  for (const r of rows) {\n    const x = num(r[corrX]);\n    const y = num(r[corrY]);\n    const lbl = r.mes ? fmtDate10(r.mes) : (r.categoria ? String(r.categoria) : '');\n    if (isFinite(x) && isFinite(y)) pts.push({ x, y, label: lbl });\n  }\n  scatter = { xKey: corrX, yKey: corrY, points: pts, rho };\n\n} else {\n  // fallback: agrega por data/mês se vier algo diferente\n  const agg = {};\n  for (const r of rows) {\n    const m = fmtDate10(r.mes || r.data_emissao || '');\n    if (!m) continue;\n    agg[m] =
```

(agg[m] || 0) + num(r.receita);\n }\n const labels = Object.keys(agg).sort();\n const data = labels.map(k => agg[k]);\n series = { labels, data };\n}\n\n// Tabela\nconst tableLimit = 50;\nlet table = [];\nif (viz === 'bar' || viz === 'pie') {\n table = rows\n.map(r => ({\n categoria: r.categoria ?? '-',\n receita: num(r.receita),\n qtd: num(r.qtd ?? 0),\n notas: num(r.notas ?? 0),\n }))\n .sort((a,b) => b.receita - a.receita)\n .slice(0, tableLimit);\n} else if (viz === 'time_series') {\n table = rows.map(r => ({\n mes: fmtDate10(r.mes),\n receita: num(r.receita),\n qtd: num(r.qtd ?? 0),\n notas: num(r.notas ?? 0),\n })).slice(0, tableLimit);\n} else if (viz === 'scatter') {\n table = (scatter.points || []).map(p => ({\n ponto: p.label || '-',\n x: p.x, y: p.y,\n })).slice(0, tableLimit);\n} else {\n table = rows.slice(0, tableLimit);\n}\n\n// (Opcional) Campos de debug para o HTML\nconst rowsCount = rows.length;\n\nreturn [{\n json: {\n viz, label, plan, context,\n query: lastQuery,\n kpi: { total, itens, notas, ticket },\n series, pie, scatter, table,\n _debug: {\n rowsCount\n }\n }\n}];\n"

    },

    "type": "n8n-nodes-base.code",

    "typeVersion": 2,

    "position": [

      1792,

      -96

    ],

    "id": "1d168626-6d6e-4f2d-86b5-dcbd28578edc",

    "name": "KPIs & Série/Scatter"

  },

  {

    "parameters": {

      "jsCode": "/**\n * BUILD SQL — COMPLETO\n * Entrada (do node \"Validate Plan JSON\"):\n * - plan, data_inicio, data_fim, uf, user_viz\n * Saída:\n * - mode, label, viz\n * - sql, params\n * - sql2, params2 (se correlação)\n * - context: { start, end, uf_scope, uf_value }\n */\n\nconst Ni = $json;\nconst plan = Ni.plan || {};\nconst start

= Ni.data_inicio;\nconst end = Ni.data_fim;\nconst ufSel = Ni.uf || 'ALL';\n\n// ------- helpers -------\nfunction ufFilter(scope, value) {\n if (!value || value === 'ALL') return {clause:'', args:[]};\n const col = scope === 'EMITENTE' ? 'i.uf_emitente' : 'i.uf_destinatario';\n return {clause: ` AND ${col} = $X `, args:[value]};\n}\nfunction monthlySelect(extraAgg) {\n return `DATE_TRUNC('month', i.data_emissao)::date AS mes, ${extraAgg}`;\n}\nfunction dimFromIntent(intent){\n if (intent === 'receita_por_produto') return 'i.descricao_produto';\n if (intent === 'receita_por_cliente') return 'i.nome_destinatario';\n if (intent === 'receita_por_cfop') return 'i.cfop';\n return null;\n}\n// Se existir UF vira $4; se não, vira $3\nfunction limitIdx(hasUf) { return hasUf ? '$4' : '$3'; }\n\n// ------- viz final (respeita user_viz quando fizer sentido) -------\nlet viz = plan.viz || 'auto';\nif (Ni.user_viz && ['time_series','bar','pie','scatter','auto'].includes(Ni.user_viz)) {\n viz = (plan.intent === 'correlacao')\n ? 'scatter'\n : (Ni.user_viz === 'auto' ? (plan.viz || 'auto') : Ni.user_viz);\n}\nif (plan.intent === 'correlacao') viz = 'scatter';\nif (!['auto','time_series','bar','pie','scatter'].includes(viz)) viz = 'auto';\n\n// ------- UF -------\nconst scope = (plan.filters && plan.filters.uf_scope) || 'DESTINATARIO';\nconst ufVal = (plan.filters && plan.filters.uf_value) || ufSel;\nconst ufCnd = ufFilter(scope, ufVal);\n\n// ------- rascunhos -------\nlet label = '';\nlet sql = '';\nlet params = [start, end]; // $1 = inicio, $2 = fim\nlet mode = 'single';\nlet sql2 = '';\nlet params2 = [];\n\n// ===== rotas por intent =====\nswitch (plan.intent) {\n /* ---------- RANKINGS ESPECÍFICOS ---------- */\n case 'top_fornecedor_receita': {\n label = 'Top Fornecedores por Receita';\n const hasUf = !!ufCnd.args.length;\n const ufIdx = hasUf ? '$3' : null;\n const limIdx = limitIdx(hasUf);\n\n // Usa campos existentes (não existe nome_emitente); tenta i.*, senão c.*\n sql = `\n SELECT\n CONCAT(\n COALESCE(i.razao_social_emitente, c.razao_social_emitente, 'Sem Razão'),\n ' (', COALESCE(i.cnpj_emitente, c.cnpj_emitente, 's/ CNPJ'), ')'\n) AS categoria,\n SUM(i.valor_total) AS receita,\n SUM(i.quantidade) AS qtd,\n COUNT(DISTINCT i.chave_acesso) AS notas\n FROM nf_itens i\n LEFT JOIN nf_cabecalhos c ON c.chave_acesso = i.chave_acesso\n WHERE i.data_emissao BETWEEN $1::date AND $2::date\n ${ufIdx ? ufCnd.clause.replace('$X', ufIdx) : ''}\n GROUP BY 1\n ORDER BY receita DESC\n LIMIT ${limIdx};\n `;\n params = [start, end];\n if (hasUf) params.push(ufCnd.args[0]); // $3\n params.push(Number(plan.limit || 50)); // $3 ou $4\n if (viz === 'auto') viz = 'bar';\n break;\n }\n\n case 'top_item_quantidade': {\n label = 'Top Itens por Quantidade';\n const hasUf = !!ufCnd.args.length;\n const ufIdx = hasUf ? '$3' : null;\n const limIdx = limitIdx(hasUf);\n\n sql = `\n SELECT\n i.descricao_produto AS categoria,\n SUM(i.quantidade) AS qtd,\n SUM(i.valor_total) AS receita,\n COUNT(DISTINCT i.chave_acesso) AS notas\n FROM nf_itens i\n WHERE i.data_emissao BETWEEN $1::date AND $2::date\n ${ufIdx ? ufCnd.clause.replace('$X', ufIdx) : ''}\n GROUP BY 1\n ORDER BY qtd

```
DESC\n LIMIT ${limIdx};\n `;\n params = [start, end];\n if (hasUf) params.push(ufCnd.args[0]);\n params.push(Number(plan.limit || 50));\n if (viz === 'auto') viz = 'bar';\n break;\n }\n\n case 'top_item_receita': {\n label = 'Top Itens por Receita';\n const hasUf = !!ufCnd.args.length;\n const ufIdx = hasUf ? '$3' : null;\n const limIdx = limitIdx(hasUf);\n\n sql = `\n SELECT\n i.descricao_produto AS categoria,\n SUM(i.valor_total) AS receita,\n SUM(i.quantidade) AS qtd,\n COUNT(DISTINTCT i.chave_acesso) AS notas\n FROM nf_itens i\n WHERE i.data_emissao BETWEEN $1::date AND $2::date\n ${ufIdx ? ufCnd.clause.replace('$X', ufIdx) : ''}\n GROUP BY 1\n ORDER BY receita DESC\n LIMIT ${limIdx};\n `;\n // correção de typo COUNT DISTINCT\n sql = sql.replace('COUNT(DISTINTCT', 'COUNT(DISTINCT');\n\n params = [start, end];\n if (hasUf) params.push(ufCnd.args[0]);\n params.push(Number(plan.limit || 50));\n if (viz === 'auto') viz = 'bar';\n break;\n }\n\n /* ---------- AGRUPAMENTOS PADRÃO ---------- */\n case 'receita_por_produto':\n case 'receita_por_cliente':\n case 'receita_por_cfop': {\n const dim = dimFromIntent(plan.intent);\n label = (plan.intent === 'receita_por_produto') ? 'Receita por Produto'\n : (plan.intent === 'receita_por_cliente') ? 'Receita por Cliente'\n : 'Receita por CFOP';\n\n const hasUf = !!ufCnd.args.length;\n const ufIdx = hasUf ? '$3' : null;\n const limIdx = limitIdx(hasUf);\n\n sql = `\n SELECT\n ${dim} AS categoria,\n SUM(i.valor_total) AS receita,\n SUM(i.quantidade) AS qtd,\n COUNT(DISTINCT i.chave_acesso) AS notas\n FROM nf_itens i\n WHERE i.data_emissao BETWEEN $1::date AND $2::date\n ${ufIdx ? ufCnd.clause.replace('$X', ufIdx) : ''}\n GROUP BY 1\n ORDER BY receita DESC\n LIMIT ${limIdx};\n `;\n params = [start, end];\n if (hasUf) params.push(ufCnd.args[0]);\n params.push(Number(plan.limit || 50));\n if (viz === 'auto') viz = 'bar';\n break;\n }\n\n /* ---------- TICKET MÉDIO ---------- */\n case 'ticket_medio': {\n label = 'Ticket Médio Mensal';\n const hasUf = !!ufCnd.args.length;\n const ufIdx = hasUf ? '$3' : null;\n\n sql = `\n SELECT\n ${monthlySelect(`SUM(i.valor_total) AS receita,\n COUNT(DISTINCT i.chave_acesso) AS notas,\n SUM(i.quantidade) AS qtd`)}\n FROM nf_itens i\n WHERE i.data_emissao BETWEEN $1::date AND $2::date\n ${ufIdx ? ufCnd.clause.replace('$X', ufIdx) : ''}\n GROUP BY 1\n ORDER BY 1 ASC;\n `;\n params = [start, end];\n if (hasUf) params.push(ufCnd.args[0]);\n if (viz === 'auto') viz = 'time_series';\n break;\n }\n\n /* ---------- CORRELAÇÃO ---------- */\n case 'correlacao': {\n label = 'Correlação';\n const x = (plan.correlation && plan.correlation.x) || 'receita';\n const y = (plan.correlation && plan.correlation.y) || 'qtd';\n\n const hasUf = !!ufCnd.args.length;\n const ufIdx = hasUf ? '$3' : null;\n const ufIdx2 = hasUf ? '$3' : null;\n\n // 1) pontos para scatter (por mês)\n sql = `\n SELECT\n DATE_TRUNC('month', i.data_emissao)::date AS mes,\n SUM(i.valor_total) AS
```

receita,\n SUM(i.quantidade) AS qtd,\n (SUM(i.valor_total) / NULLIF(COUNT(DISTINCT i.chave_acesso),0)) AS ticket_medio\n FROM nf_itens i\n WHERE i.data_emissao BETWEEN $1::date AND $2::date\n ${ufIdx ? ufCnd.clause.replace('$X', ufIdx) : ''}\n GROUP BY 1\n ORDER BY 1 ASC;\n `;\n params = [start, end];\n if (hasUf) params.push(ufCnd.args[0]);\n\n // 2) rho (correlação de Pearson)\n sql2 = `\n SELECT\n corr(x::double precision, y::double precision) AS rho\n FROM (\n SELECT\n DATE_TRUNC('month', i.data_emissao)::date AS mes,\n SUM(i.valor_total) AS receita,\n SUM(i.quantidade) AS qtd,\n (SUM(i.valor_total) / NULLIF(COUNT(DISTINCT i.chave_acesso),0)) AS ticket_medio\n FROM nf_itens i\n WHERE i.data_emissao BETWEEN $1::date AND $2::date\n ${ufIdx2 ? ufCnd.clause.replace('$X', ufIdx2) : ''}\n GROUP BY 1\n) s\n CROSS JOIN LATERAL (\n SELECT\n CASE WHEN $${hasUf ? 4 : 3} = 'receita' THEN s.receita\n WHEN $${hasUf ? 4 : 3} = 'qtd' THEN s.qtd\n ELSE s.ticket_medio\n END AS x,\n CASE WHEN $${hasUf ? 5 : 4} = 'receita' THEN s.receita\n WHEN $${hasUf ? 5 : 4} = 'qtd' THEN s.qtd\n ELSE s.ticket_medio\n END AS y\n) m;\n `;\n params2 = [start, end];\n if (hasUf) params2.push(ufCnd.args[0]); // vira $3\n params2.push(x, y); // últimos parâmetros são as variáveis\n\n mode = 'dual';\n viz = 'scatter';\n break;\n }\n\n /* ---------- DEFAULT: RECEITA MENSAL ---------- */\n case 'receita_mensal':\n default: {\n label = 'Receita Mensal';\n const hasUf = !!ufCnd.args.length;\n const ufIdx = hasUf ? '$3' : null;\n\n sql = `\n SELECT\n ${monthlySelect(`SUM(i.valor_total) AS receita,\n COUNT(DISTINCT i.chave_acesso) AS notas,\n SUM(i.quantidade) AS qtd`)}\n FROM nf_itens i\n WHERE i.data_emissao BETWEEN $1::date AND $2::date\n ${ufIdx ? ufCnd.clause.replace('$X', ufIdx) : ''}\n GROUP BY 1\n ORDER BY 1 ASC;\n `;\n params = [start, end];\n if (hasUf) params.push(ufCnd.args[0]);\n if (viz === 'auto') viz = 'time_series';\n break;\n }\n}\n\nreturn [{\n json: {\n mode,\n label,\n viz,\n sql, params,\n sql2, params2,\n context: { start, end, uf_scope: scope, uf_value: ufVal }\n }\n}];\n"

},

"type": "n8n-nodes-base.code",

"typeVersion": 2,

"position": [

528,

```
    16
   ],
   "id": "45eb7956-e7f7-42bb-9ff6-39eb42b10a17",
   "name": "Build SQL"
  },
  {
   "parameters": {
    "operation": "executeQuery",
    "query": "{{$json.sql2_final}}",
    "options": {}
   },
   "type": "n8n-nodes-base.postgres",
   "typeVersion": 2.6,
   "position": [
    1408,
    -32
   ],
   "id": "587672e2-8fbe-434e-909c-de5032ae9279",
   "name": "Postgres - Rho",
   "retryOnFail": true,
   "credentials": {
    "postgres": {
     "id": "XKpWJjbwdcEEMiIq",
```

```
      "name": "Postgres account"
    }
  }
},
{
  "parameters": {
    "operation": "executeQuery",
    "query": "{{$json.sql_final}}\n",
    "options": {}
  },
  "type": "n8n-nodes-base.postgres",
  "typeVersion": 2.6,
  "position": [
    992,
    -112
  ],
  "id": "4ad37372-1c44-4c19-8101-f9b9bed07b8a",
  "name": "Postgres - Main Query",
  "credentials": {
    "postgres": {
      "id": "XKpWJjbwdcEEMiIq",
      "name": "Postgres account"
    }
```

```
  }
},
{
  "parameters": {
    "modelId": {
      "__rl": true,
      "value": "gpt-4o-mini",
      "mode": "list",
      "cachedResultName": "GPT-4O-MINI"
    },
    "messages": {
      "values": [
        {
          "content": "Você é um planejador de consultas. Sua função é ler a pergunta do usuário e retornar UM ÚNICO objeto JSON **válido** que descreve o plano de consulta permitido (intent, métricas, dimensões, filtros e visualização).\n\nNUNCA gere SQL. NUNCA retorne textos fora do JSON. NUNCA use campos fora do formato especificado.\n\nFormato OBRIGATÓRIO da resposta (apenas um objeto JSON):\n{\n \"intent\": \"<'receita_por_produto'|'receita_por_cliente'|'receita_por_cfop'|'receita_mensal'|'ticket_medio'|'correlacao'|'top_fornecedor_receita'|'top_item_quantidade'|'top_item_receita'>\",\n \"group_by\": [\"<dim 1>\", \"<dim 2 opcional>\"],\n \"metrics\": [\"receita\",\"qtd\",\"notas\",\"ticket_medio\"],\n \"correlation\": { \"x\": \"<receita|qtd|ticket_medio>\", \"y\": \"<receita|qtd|ticket_medio>\" },\n \"viz\": \"<'auto'|'time_series'|'bar'|'pie'|'scatter'>\",\n \"filters\": { \"uf_scope\": \"<'ALL'|'EMITENTE'|'DESTINATARIO'>\", \"uf_value\": \"<sigla UF ou 'ALL'>\" },\n \"limit\": 50,\n \"explanacao\": \"<justificativa curta do plano>\"\n}\n\nRegras de mapeamento:\n- Se a pergunta menciona “produto” → intent='receita_por_produto' e group_by=['descricao_produto'].\n- Se menciona “cliente” ou “destinatário” → intent='receita_por_cliente' e group_by=['cnpj_destinatario','nome_destinatario'].\n-
```

Se menciona “CFOP” → intent='receita_por_cfop' e group_by=['cfop'].\n- Se não houver pista clara → intent='receita_mensal' com group_by=['mes'].\n- “ticket médio” → intent='ticket_medio' (time series por mês).\n- “correlação/relacionar X e Y” → intent='correlacao', viz='scatter' e defina correlation.x e correlation.y ∈ {receita,qtd,ticket_medio}.\n\nNovos intents:\n- “fornecedor/emitente com maior montante/receita”, “top N fornecedores” → intent='top_fornecedor_receita', group_by=['razao_social_emitente','cnpj_emitente'], viz='bar'.\n- “item com maior quantidade/volume entregue”, “top N por quantidade” → intent='top_item_quantidade', group_by=['descricao_produto'], viz='bar'.\n- “item com maior receita”, “top N itens por receita” → intent='top_item_receita', group_by=['descricao_produto'], viz='bar' (ou 'pie' se pedir percentual).\n\nVisualização sugerida:\n- Se a pergunta menciona “série temporal/ao longo do tempo/mês a mês” → viz='time_series'.\n- Se menciona “barras/por categoria/top N” → viz='bar'.\n- Se menciona “pizza/percentual/participação” → viz='pie'.\n\nFiltros:\n- Se a pergunta citar explicitamente “estado do emitente” → uf_scope='EMITENTE'.\n- Se citar “estado do destinatário” → uf_scope='DESTINATARIO'.\n- Caso contrário, use o valor fornecido (ALL ou uma UF) como padrão em 'DESTINATARIO'.\n\nUse APENAS dimensões do schema recebido e respeite [data_inicio, data_fim]. Não invente valores.\nResponda APENAS com o JSON, sem markdown, sem comentários, sem texto adicional.\n",

"role": "system"

},

{

"content": "==Dados para planejar a consulta (NÃO gere SQL):\n\npergunta: {{ $json.pergunta }}\ndata_inicio: {{ $json.data_inicio }}\ndata_fim: {{ $json.data_fim }}\nuf: {{ $json.uf }}\nviz_preferida_do_usuario: {{ $json.user_viz }}\nschema: {{ JSON.stringify($json.schema) }}\n\nRetorne SOMENTE o objeto JSON no formato exigido pelo system.\n"

}

]

},

"jsonOutput": true,

"options": {

```
      "maxTokens": 500,
      "temperature": 0.2
    }
  },
  "type": "@n8n/n8n-nodes-langchain.openAi",
  "typeVersion": 1.8,
  "position": [
    -16,
    16
  ],
  "id": "22da19ef-5ea5-4de6-99e3-d6394549ee0b",
  "name": "OpenAI - NL → Plan (JSON)",
  "credentials": {
    "openAiApi": {
      "id": "SedjdE4QVcnu8x3b",
      "name": "OpenAi account 2"
    }
  }
},
{
  "parameters": {
    "conditions": {
      "options": {
```

```
      "caseSensitive": false,
      "leftValue": "",
      "typeValidation": "loose",
      "version": 2
    },
    "conditions": [
      {
        "id": "4700c6b0-33b4-4e9c-87d6-bb5dadd5f12b",
        "leftValue": "=={{$prevNode[\"Build SQL\"].json.mode}}",
        "rightValue": "dual",
        "operator": {
          "type": "string",
          "operation": "equals"
        }
      }
    ],
    "combinator": "and"
  },
  "looseTypeValidation": true,
  "options": {
    "ignoreCase": true
  }
},
```

```
    "type": "n8n-nodes-base.if",
    "typeVersion": 2.2,
    "position": [
      992,
      32
    ],
    "id": "70fdc365-4e92-416f-b785-ade788bc9471",
    "name": "IF - Is Dual?"
  },
  {
    "parameters": {
          "jsCode": "/**\n * HTML Renderer — robusto\n * - Busca o payload k via $items(\"KPIs & Série/Scatter\") para evitar $prevNode vazio.\n * - Extrai insight de $json.text ou choices[0].message.content (OpenAI simplify:false).\n * - Formulário sempre no topo (produção).\n */\n\nfunction safeItems(nodeName) {\n  try { return $items(nodeName) || []; } catch { return []; }\n}\n\n// 1) Pegar o payload consolidado do nó KPIs & Série/Scatter\nconst kItems = safeItems(\"KPIs & Série/Scatter\");\nconst k = (kItems.length ? (kItems[0].json || {}) : {}) || {};\n\n// 2) Extrair campos do k com defaults\nconst viz     = k.viz     || 'time_series';\nconst label   = k.label   || 'Relatório';\nconst kpi     = k.kpi     || { total:0, itens:0, notas:0, ticket:0 };\nconst series  = k.series  || { labels:[], data:[] };\nconst pie     = k.pie     || series;\nconst scatter = k.scatter || { xKey:'receita', yKey:'qtd', points:[], rho:null };\nconst table   = k.table   || [];\nconst context = k.context || {}; // { start, end, uf_scope, uf_value }\n\n// 3) Últimos inputs para preencher o form\nconst niItems = safeItems(\"Normalize Inputs\");\nconst ni = (niItems.length ? (niItems[0].json || {}) : {}) || {};\nconst perguntaUsed   = ni.pergunta   || '';\nconst ufUsed         = (ni.uf || 'ALL').toUpperCase();\nconst dataInicioUsed = ni.data_inicio || '';\nconst dataFimUsed    = ni.data_fim    || '';\n\n// 4) Insight: tenta $json.text, depois OpenAI choices[0].message.content\nlet insight = '(sem insight)';\nif (typeof $json?.text === 'string') {\n  insight = $json.text;\n} else if ($json?.choices?.[0]?.message?.content) {\n  insight = String($json.choices[0].message.content);\n} else if (typeof $json ===
```

'string') {\n insight = $json;\n}\n\n// 5) Helpers de render\nfunction esc(s){ return String(s ?? '').replace(/[&<>\\\"']/g, m => ({'&':'&','<':'<','>':'>','\\':'"','"':'''}[m])); }\nfunction fmtBRL(n){ try { return Number(n||0).toLocaleString('pt-BR',{style:'currency',currency:'BRL'}); } catch { return `${n}`; } }\nfunction fmtISO(s){ return (s||'').toString().slice(0,10); }\n\n// 6) UF options\nconst UF_LIST = ['ALL','AC','AL','AP','AM','BA','CE','DF','ES','GO','MA','MT','MS','MG','PA','PB','PR','PE','PI','RJ','RN','RS','RO','RR','SC','SP','SE','TO'];\nconst ufOptions = UF_LIST.map(uf => {\n const selected = uf === ufUsed ? 'selected' : '';\n return `<option value=\"${uf}\" ${selected}>${uf}</option>`;\n}).join('');\n\n// 7) Cabeçalho/linhas da tabela\nlet tableHead = '', tableRows = '';\nif (viz === 'bar' || viz === 'pie') {\n tableHead = `<tr><th>Categoria</th><th>Receita</th><th>Qtd</th><th>Notas</th></tr>`;\n tableRows = table.map(r => `<tr><td>${esc(r.categoria)}</td><td>${fmtBRL(r.receita)}</td><td>${r.qtd||0}</td><td>${r.notas||0}</td></tr>`).join('');\n} else if (viz === 'time_series') {\n tableHead = `<tr><th>Mês</th><th>Receita</th><th>Qtd</th><th>Notas</th></tr>`;\n tableRows = table.map(r => `<tr><td>${esc(r.mes)}</td><td>${fmtBRL(r.receita)}</td><td>${r.qtd||0}</td><td>${r.notas||0}</td></tr>`).join('');\n} else if (viz === 'scatter') {\n tableHead = `<tr><th>Ponto</th><th>${esc(scatter.xKey)}</th><th>${esc(scatter.yKey)}</th></tr>`;\n tableRows = table.map(r => `<tr><td>${esc(r.ponto)}</td><td>${r.x}</td><td>${r.y}</td></tr>`).join('');\n} else {\n const cols = Object.keys(table[0] || {coluna:'valor'});\n tableHead = `<tr>${cols.map(c=>`<th>${esc(c)}</th>`).join('')}</tr>`;\n tableRows = table.map(row => `<tr>${Object.keys(row).map(c=>`<td>${esc(row[c])}</td>`).join('')}</tr>`).join('');\n}\n\n// 8) Chart config\nconst chartType = (viz === 'time_series') ? 'line'\n : (viz === 'bar') ? 'bar'\n : (viz === 'pie') ? 'pie'\n : (viz === 'scatter') ? 'scatter' : 'line';\nconst labels = series.labels || [];\nconst data   = series.data || [];\nconst points = scatter.points || [];\nconst rhoTxt = (scatter.rho===null || scatter.rho===undefined) ? '' : `Coef. de correlação (ρ): <b>${Number(scatter.rho).toFixed(3)}</b>`;\nconst badgeUF = context.uf_value ? `${context.uf_scope || 'DEST'}: ${context.uf_value}` : `UF: ${ufUsed || 'ALL'}`;\nconst badgePer= `${fmtISO(context.start || dataInicioUsed)} → ${fmtISO(context.end || dataFimUsed)}`;\n\n// 9) Debug opcional (aparece acima dos KPIs)\nconst debugHtml = `\n <div class=\"section\" style=\"background:#fff8;border-left:4px solid #2E7D68\">\n Debug
\n Intent/Viz: ${esc(k?.plan?.intent || 'desconhecido')} / ${esc(viz)}
\n Período usado: ${esc(badgePer)}
\n UF: ${esc(badgeUF)}
\n Linhas exibidas (tabela): ${Array.isArray(table) ? table.length : 0}\n </div>\n`;\n\n// 10) HTML\nconst html = `\n<!doctype html>\n<html lang=\"pt-br\">\n<head>\n <meta

```
charset=\"utf-8\"/>\n <title>${esc(label || 'GAIA – Agentes EDA')}</title>\n <meta name=\"viewport\" content=\"width=device-width, initial-scale=1\"/>\n <link href=\"https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap\" rel=\"stylesheet\"/>\n <script src=\"https://cdn.jsdelivr.net/npm/chart.js\"></script>\n <style>\n body{background:#fff;font-family:Poppins,Arial;color:#2E7D68;display:flex;justify-content:center;align-items:flex-start;margin:0}\n .container{background:rgba(46,125,104,.05);padding:30px 40px;border-radius:15px;max-width:980px;width:100%;box-shadow:0 4px 15px rgba(0,0,0,.1);margin:24px}\n h1{margin:0 0 8px;text-transform:capitalize}\n h2{margin:0 0 6px;font-size:18px}\n label{font-size:13px;display:block;margin-bottom:4px}\n input,select,button{font-family:Poppins,Arial}\n input,select{width:100%;padding:10px 12px;border:1px solid #2E7D68;border-radius:8px;color:#2E7D68}\n button{padding:12px 14px;background:#2E7D68;color:#fff;border:none;border-radius:8px;font-weight:600;cursor:pointer}\n .section{background:#fff;padding:16px;border-radius:12px;margin:16px 0}\n .row{display:flex;gap:12px;flex-wrap:wrap;align-items:flex-end}\n .col{flex:1 1 240px;min-width:220px}\n .col-narrow{flex:0 0 150px}\n .kpi{background:#f6faf8;padding:14px 16px;border-radius:12px;margin:10px 0 16px}\n .kpi b{color:#1f5c4b}\n .badge{padding:4px 8px;border-radius:8px;background:#e1f2ec;display:inline-block;margin:0 6px 8px 0}\n table{width:100%;border-collapse:collapse}\n th,td{padding:8px;border-bottom:1px solid #eaeaea;text-align:left}\n .muted{opacity:.8}\n .rho{margin-top:8px}\n </style>\n</head>\n<body>\n <div class=\"container\">\n <h2>GAIA – Agentes EDA</h2>\n\n <!-- Form sempre no topo (produção) -->\n <form method=\"POST\" action=\"https://n8n-service-to7f.onrender.com/webhook-test/12a2_final\" class=\"section\">\n <div class=\"row\">\n <div class=\"col\">\n <label>Pergunta</label>\n <input type=\"text\" name=\"pergunta\" placeholder=\"Ex.: soma total das notas\" value=\"${esc(perguntaUsed)}\">\n </div>\n <div class=\"col-narrow\">\n <label>UF</label>\n <select name=\"uf\">${ufOptions}</select>\n </div>\n <div class=\"col-narrow\">\n <label>Data início (YYYY-MM-DD)</label>\n <input type=\"text\" name=\"data_inicio\" placeholder=\"YYYY-MM-DD\" value=\"${esc(dataInicioUsed)}\">\n </div>\n <div class=\"col-narrow\">\n <label>Data fim (YYYY-MM-DD)</label>\n <input type=\"text\" name=\"data_fim\" placeholder=\"YYYY-MM-DD\" value=\"${esc(dataFimUsed)}\">\n </div>\n <div class=\"col-narrow\">\n <button type=\"submit\"
```

title=\"Executar nova consulta\">Executar</button>\n \n \n </form>\n\n ${debugHtml}\n\n <h1>${esc(label || 'Relatório')}</h1>\n <div class=\"badge\">${esc(fmtISO(context.start || dataInicioUsed))} → ${esc(fmtISO(context.end || dataFimUsed))}</div>\n <div class=\"badge\">${esc(context.uf_value ? `${context.uf_scope || 'DEST'}: ${context.uf_value}` : `UF: ${ufUsed || 'ALL'}`)} \n\n <div class=\"kpi\">\n Total: ${fmtBRL(kpi.total)} • Itens: ${kpi.itens||0} • Notas: ${kpi.notas||0} • Ticket Médio: <b>${fmtBRL(kpi.ticket||0)} \n ${viz==='scatter' && rhoTxt ? `<div class=\"rho\">${rhoTxt} ` : ``}\n </div>\n\n <div class=\"section\">\n <canvas id=\"c1\"></canvas>\n </div>\n\n <div class=\"section\">\n Insight\n <p>${esc(insight)}</p>\n </div>\n\n <div class=\"section\">\n <h3>Detalhes</h3>\n <table>\n <thead>${tableHead}</thead>\n <tbody>${tableRows || `<tr><td class=\"muted\" colspan=\"4\">Sem dados para exibir</td></tr>`}</tbody>\n </table>\n </div>\n </div>\n\n <script>\n (function(){\n const ctx = document.getElementById('c1');\n const type = ${JSON.stringify(chartType)};\n const labels = ${JSON.stringify(labels)};\n const data = ${JSON.stringify(data)};\n const points = ${JSON.stringify(points)};\n const xKey = ${JSON.stringify(scatter.xKey || 'x')};\n const yKey = ${JSON.stringify(scatter.yKey || 'y')};\n\n let config;\n if (type === 'scatter') {\n config = {\n type: 'scatter',\n data: { datasets: [{ label: 'Correlação', data: points, pointRadius: 4 }] },\n options: {\n plugins: { legend: { display: false } },\n scales: {\n x: { title: { display: true, text: xKey } },\n y: { title: { display: true, text: yKey } }\n }\n }\n };\n } else if (type === 'pie') {\n config = {\n type: 'pie',\n data: { labels, datasets: [{ data }] },\n options: { plugins: { legend: { position: 'bottom' } } }\n };\n } else {\n config = {\n type,\n data: { labels, datasets: [{ label: 'Receita', data, borderWidth: 2, fill: false }]},\n options: { plugins: { legend: { display: false } }, scales: { y: { beginAtZero: true } } }\n };\n }\n new Chart(ctx, config);\n })();\n </script>\n</body>\n</html>\n`;\n\nreturn [{ json: { data: html } }];\n"

},

"type": "n8n-nodes-base.code",

"typeVersion": 2,

"position": [

2272,

```
    -96
  ],
  "id": "22dc93a9-11e3-4ca7-a782-afd1a322c1f8",
  "name": "HTML Renderer"
},
{
  "parameters": {
    "options": {}
  },
  "type": "n8n-nodes-base.set",
  "typeVersion": 3.4,
  "position": [
    1408,
    112
  ],
  "id": "77165803-6a5d-49ed-994e-967e24f8c438",
  "name": "Empty For Merge"
},
{
  "parameters": {},
  "type": "n8n-nodes-base.merge",
  "typeVersion": 3.2,
  "position": [
```

```
      1632,
      -96
    ],
    "id": "8de98d3b-fafa-4c56-b5b1-190df1d5b640",
    "name": "Merge"
  },
  {
    "parameters": {
          "jsCode": "function isNumberLike(v) {\n  return typeof v === 'number' || (/^-?\\d+(\\.\\d+)?$/.test(String(v)));\n}\nfunction sqlQuote(v) {\n  return `'${String(v).replace(/'/g, \"''\")}'`;\n}\nfunction inlineParams(sql, params) {\n  if (!sql) return '';\n  let out = sql;\n  if (!Array.isArray(params)) return out;\n  // substitui do maior para o menor ($10 antes de $1)\n  for (let i = params.length; i >= 1; i--) {\n const p = params[i-1];\n  const replacement = isNumberLike(p) ? String(p) : sqlQuote(p);\n  const re = new RegExp('\\\\$' + i + '(?!\\\\d)', 'g');\n  out = out.replace(re, replacement);\n  }\n  return out;\n}\n\nconst sql_final = inlineParams($json.sql, $json.params);\nconst sql2_final = inlineParams($json.sql2, $json.params2);\n\nreturn [{ json: { ...$json, sql_final, sql2_final } }];\n"
    },
    "type": "n8n-nodes-base.code",
    "typeVersion": 2,
    "position": [
      704,
      16
    ],
    "id": "e65bd096-b6c7-409c-a2bb-161ea9e33f44",
    "name": "Inline Params"
```

```
},
{
  "parameters": {
    "conditions": {
      "options": {
        "caseSensitive": true,
        "leftValue": "",
        "typeValidation": "loose",
        "version": 2
      },
      "conditions": [
        {
          "id": "16ec3d50-0f6e-4d69-8a7a-233ada73ead5",
          "leftValue": "={{ $json.body.modo }}",
          "rightValue": "=preditivo",
          "operator": {
            "type": "string",
            "operation": "equals"
          }
        }
      ],
      "combinator": "or"
    },
```

```
      "looseTypeValidation": true,
      "options": {}
    },
    "type": "n8n-nodes-base.if",
    "typeVersion": 2.2,
    "position": [
      -224,
      0
    ],
    "id": "0f3abba1-b073-4752-b73f-a26ea20c520f",
    "name": "If"
  },
  {
    "parameters": {
        "jsCode": "// Predictive – Build Dataset SQL\n// Lê datas de treino de pred_params, define granularidade e monta SQL + params para o dataset do preditivo.\n\nfunction isYMD(s) {\n  return typeof s === 'string' && /^\\d{4}-\\d{2}-\\d{2}$/.test(s);\n}\n\n// Fallbacks seguros se nada vier (evita null no Postgres)\nconst FALLBACK_START = '2000-01-01';\nconst FALLBACK_END   = '2100-12-31';\n\nconst j  = $json || {};\nconst pp = j.pred_params || {};\nconst body = j.body || {};\n\n// 1) Datas de treino (prioridade: pred_params -> body.treino_intervalo (se Normalize não tiver feito) -> data_inicio/fim -> fallbacks)\nlet start = pp.treino_inicio || j.data_inicio || body.data_inicio || null;\nlet end   = pp.treino_fim   || j.data_fim    || body.data_fim    || null;\n\n// Se ainda não tiver, tentar parsear \"YYYY-MM-DD a YYYY-MM-DD\" bruto do body.treino_intervalo\nif ((!start || !end) && body.treino_intervalo) {\n  const m = String(body.treino_intervalo).match(/(\\d{4}-\\d{2}-\\d{2})\\s*a\\s*(\\d{4}-\\d{2}-\\d{2})/i);\n  if (m) { start = start || m[1]; end = end || m[2]; }\n}\n\n// Sanear: garantir formato YYYY-MM-DD, senão usar fallback\nif (!isYMD(start)) start = FALLBACK_START;\nif
```

(!isYMD(end)) end = FALLBACK_END;\n\n// 2) Granularidade -> DATE_TRUNC\nconst granStr = (pp.granularidade || 'mensal').toLowerCase();\nlet dateTrunc = 'month';\nif (granStr === 'diario' || granStr === 'diária' || granStr === 'dia' || granStr === 'daily') {\n dateTrunc = 'day';\n} else if (granStr === 'anual' || granStr === 'ano' || granStr === 'yearly') {\n dateTrunc = 'year';\n} // 'mensal' é o default\n\n// 3) (Opcional) Categoria/Agrupamento — por enquanto TOTAL fixo\nconst categoria = 'TOTAL';\n\n// 4) Construir SQL parametrizado ($1,$2)\nconst sql_pred_dataset = `\n SELECT\n DATE_TRUNC('${dateTrunc}', i.data_emissao)::date AS ts,\n '${categoria}'::text AS categoria,\n SUM(i.valor_total) AS receita,\n SUM(i.quantidade) AS qtd,\n COUNT(DISTINCT i.chave_acesso) AS notas,\n (SUM(i.valor_total) / NULLIF(COUNT(DISTINCT i.chave_acesso), 0)) AS ticket_medio\n FROM nf_itens i\n LEFT JOIN nf_cabecalhos c ON c.chave_acesso = i.chave_acesso\n WHERE i.data_emissao BETWEEN $1::date AND $2::date\n GROUP BY 1, 2\n ORDER BY 1 ASC\n`.trim();\n\n// 5) Params para $1/$2\nconst params_pred_dataset = [start, end];\n\n// 6) Contexto, útil para debug\nconst pred_context = {\n target: (pp.target_metric || 'receita').toLowerCase(),\n gran: granStr,\n dim: categoria,\n treino_frac: 0.8,\n start,\n end,\n uf: j.uf || 'ALL'\n};\n\n// 7) Retorno preservando o restante do JSON\nreturn [\n {\n json: {\n ...j,\n sql_pred_dataset,\n params_pred_dataset,\n pred_context\n }\n }\n];\n"

},

"type": "n8n-nodes-base.code",

"typeVersion": 2,

"position": [

-16,

256

],

"id": "aedbe6b9-e732-44c7-b7e2-f46878659da9",

"name": "Predictive - Build Dataset SQL"

},

{

"parameters": {

```
    "operation": "executeQuery",
    "query": "{{$json.sql_pred_dataset_final}}",
    "options": {
      "queryReplacement": "=\n"
    }
  },
  "type": "n8n-nodes-base.postgres",
  "typeVersion": 2.6,
  "position": [
    368,
    256
  ],
  "id": "de3b7ffa-0965-4b46-a21c-f127b82dc42e",
  "name": "Postgres – Fetch Dataset",
  "credentials": {
    "postgres": {
      "id": "XKpWJjbwdcEEMiIq",
      "name": "Postgres account"
    }
  }
},
{
  "parameters": {
```

"jsCode": "/**\n * Predictive – Train & Forecast (Code)\n * Espera: input único com { history: [{ts,y},...], context:{...} }\n * Saída: { ok, history, forecast, intervals, context }\n */\n\nfunction sortByTs(a, b){ return new Date(a.ts) - new Date(b.ts); }\nfunction ensureDaily(history){\n // já vem diário; apenas valida ordem e tipos\n const clean = history\n .map(p => ({ ts: new Date(p.ts).toISOString(), y: Number(p.y) }))\n .filter(p => Number.isFinite(p.y));\n clean.sort(sortByTs);\n return clean;\n}\n\nconst inp = $json;\nconst history = Array.isArray(inp.history) ? ensureDaily(inp.history) : [];\nconst ctx = inp.context || {};\n\nconst minPoints = 7; // com 28 pontos você está ok\nif (history.length < minPoints) {\n return [{ json: {\n ok:false,\n error:`Série insuficiente (${history.length} ponto${history.length===1?'':'s'}). Amplie o intervalo, mude a granularidade ou verifique o dataset.`,\n history: [],\n context: {\n target: ctx.target || 'receita',\n gran: ctx.gran || 'diario',\n treino_frac: ctx.treino_frac ?? 0.8,\n start: ctx.start ?? null,\n end: ctx.end ?? null,\n uf: ctx.uf ?? 'ALL',\n horizonte: Number(ctx.horizonte ?? 14),\n passos: 'dias'\n }\n }}];\n}\n\n// Modelo simples: média móvel + tendência linear leve (baseline didático)\nfunction simpleForecast(history, steps){\n const ys = history.map(p => p.y);\n const n = ys.length;\n\n // média dos últimos k\n const k = Math.min(7, n);\n const tail = ys.slice(n - k);\n const mean = tail.reduce((a,b)=>a+b,0)/k;\n\n // tendência linear (OLS simples)\n const xs = ys.map((_,i)=>i+1);\n const xbar = xs.reduce((a,b)=>a+b,0)/n;\n const ybar = ys.reduce((a,b)=>a+b,0)/n;\n let num=0, den=0;\n for (let i=0;i<n;i++){ num += (xs[i]-xbar)*(ys[i]-ybar); den += (xs[i]-xbar)**2; }\n const slope = den===0 ? 0 : num/den;\n\n // último ts\n const last = new Date(history[history.length-1].ts);\n\n // gera próximos dias\n const out = [];\n for (let h=1; h<=steps; h++){\n const d = new Date(last);\n d.setDate(d.getDate()+h);\n const base = mean + slope*h; // baseline com leve tendência\n const yhat = Math.max(0, base); // clamp no mínimo 0\n out.push({ ts: d.toISOString(), yhat });\n }\n return out;\n}\n\n// Intervalos simples (bandas simétricas com base no desvio padrão da cauda)\nfunction computeIntervals(history, forecast, conf=0.9){\n const ys = history.map(p=>p.y);\n const k = Math.min(14, ys.length);\n const tail = ys.slice(ys.length - k);\n const mu = tail.reduce((a,b)=>a+b,0)/k;\n const sd = Math.sqrt(tail.reduce((a,b)=>a+(b-mu)**2,0)/k) || 0;\n\n // z aproximado para 90% ~ 1.64 ; 95% ~ 1.96\n const z = conf >= 0.95 ? 1.96 : 1.64;\n\n return forecast.map(p => {\n const half = z*sd;\n return { ts: p.ts, lo: Math.max(0, p.yhat - half), hi: p.yhat + half };\n });\n}\n\nconst steps = Number(ctx.horizonte ?? 14);\nconst forecast = simpleForecast(history, steps);\nconst intervals = computeIntervals(history, forecast, Number(ctx.conf ?? 0.9));\n\n// devolve pronto para render\nreturn [{\n json: {\n ok: true,\n history,\n forecast,\n intervals,\n context: {\n target: ctx.target || 'receita',\n gran: 'diario',\n dim: ctx.dim || 'TOTAL',\n horizonte: steps,\n passos: 'dias',\n conf: Number(ctx.conf ?? 0.9),\n algoritmo: ctx.algoritmo || 'auto'\n }\n }\n}];\n"

```
    },
    "type": "n8n-nodes-base.code",
    "typeVersion": 2,
    "position": [
      720,
      256
    ],
    "id": "adfcf0ec-cee2-4d45-aa2e-d307791ed8cf",
    "name": "Predictive - Train & Forecast"
  },
  {
    "parameters": {
      "jsCode": "function isNumberLike(v) {\n  if (v === null || v === undefined) return false;\n  return (typeof v === 'number') || (/^-?\\d+(\\.\\d+)?$/.test(String(v)));\n}\nfunction sqlQuote(v) {\n  return `'${String(v).replace(/'/g, \"''\")}'`;\n}\nfunction inlineParams(sql, params) {\n  if (!sql) return '';\n  let out = sql;\n  if (!Array.isArray(params) || params.length === 0) return out;\n  // Substitui do maior para o menor ($10 antes de $1)\n  for (let i = params.length; i >= 1; i--) {\n    const p = params[i - 1];\n    const replacement = isNumberLike(p) ? String(p) : sqlQuote(p);\n    const re = new RegExp('\\$' + i + '(?!\\d)', 'g');\n    out = out.replace(re, replacement);\n  }\n  return out;\n}\n\nconst inJson = $json;\n\n// Fluxo “padrão”\nconst sql  = inJson.sql ?? '';\nconst params= Array.isArray(inJson.params) ? inJson.params : [];\nconst sql2  = inJson.sql2 ?? '';\nconst params2 = Array.isArray(inJson.params2) ? inJson.params2 : [];\n\nconst sql_final  = inlineParams(sql, params);\nconst sql2_final = inlineParams(sql2, params2);\n\n// Fluxo “preditivo”\nconst sql_pred_dataset = inJson.sql_pred_dataset ?? '';\nconst params_pred_dataset = Array.isArray(inJson.params_pred_dataset) ? inJson.params_pred_dataset : [];\nconst sql_pred_dataset_final = inlineParams(sql_pred_dataset, params_pred_dataset);\n\n// Debug opcional\nconst _inline_debug = {\n  have_sql: Boolean(sql),\n  have_sql2: Boolean(sql2),\n  have_sql_pred_dataset: Boolean(sql_pred_dataset),\n  params_used_for_sql:
```

params,\n params_used_for_sql2: params2,\n params_used_for_pred_dataset: params_pred_dataset\n};\n\nreturn [{\n json: {\n ...inJson,\n sql_final,\n sql2_final,\n sql_pred_dataset_final,\n _inline_debug\n }\n}];\n"

},

"type": "n8n-nodes-base.code",

"typeVersion": 2,

"position": [

192,

256

],

"id": "8b6f2f6f-af13-478a-aba5-8c0f7d7fa1ff",

"name": "Inline Params1"

},

{

"parameters": {

"jsCode": "/**\n * Predictive – Prepare Render Payload (Code)\n * Entrada esperada: { ok, history:[{ts,y}], forecast:[{ts,yhat}], intervals:[{ts,lo,hi}], context:{...} }\n * Saída: payload com viz, kpi, series (hist + prev), bands e table — compatível com seu Render.\n */\n\nconst inp = $json;\nif (inp.ok !== true) {\n return [{ json: {\n viz: 'time_series',\n label: 'Previsão',\n kpi: { total_prev: 0, dias_previstos: 0, ultimo_observado: 0, conf: (inp.context?.conf ?? null) },\n series: { labels: [], hist: [], prev: [] },\n bands: { lo: [], hi: [] },\n table: [],\n insight: 'Série insuficiente ou outro erro na etapa de previsão.',\n _debug: { err: inp.error ?? 'unknown' }\n }}];\n}\n\n// Helpers\nfunction fmtDate(d){ return new Date(d).toISOString().slice(0,10); }\nfunction sum(arr){ return arr.reduce((a,b)=>a + (Number(b)||0), 0); }\n\nconst history = Array.isArray(inp.history) ? inp.history : [];\nconst forecast = Array.isArray(inp.forecast) ? inp.forecast : [];\nconst intervals = Array.isArray(inp.intervals) ? inp.intervals : [];\n\nhistory.sort((a,b)=> new Date(a.ts) - new Date(b.ts));\nforecast.sort((a,b)=> new Date(a.ts) - new Date(b.ts));\nintervals.sort((a,b)=> new Date(a.ts) - new Date(b.ts));\n\n// Constroi

eixos e séries\nconst labelsHist = history.map(p => fmtDate(p.ts));\nconst dataHist = history.map(p => Number(p.y) || 0);\n\nconst labelsPrev = forecast.map(p => fmtDate(p.ts));\nconst dataPrev = forecast.map(p => Number(p.yhat) || 0);\n\nconst lo = [];\nconst hi = [];\nconst bandsByDate = new Map(intervals.map(b => [fmtDate(b.ts), { lo: Number(b.lo)||0, hi: Number(b.hi)||0 }]));\nlabelsPrev.forEach(d => {\n const b = bandsByDate.get(d);\n lo.push(b ? b.lo : null);\n hi.push(b ? b.hi : null);\n});\n\n// KPIs\nconst totalPrev = sum(dataPrev);\nconst ultimoObs = dataHist.length ? dataHist[dataHist.length-1] : 0;\nconst conf = inp.context?.conf ?? null;\n\n// Tabela (histórico + previsão)\nconst table = [];\nhistory.forEach(p => table.push({ data: fmtDate(p.ts), tipo: 'hist', valor: Number(p.y)||0 }));\nforecast.forEach(p => {\n const d = fmtDate(p.ts);\n const band = bandsByDate.get(d) || {};\n table.push({\n data: d,\n tipo: 'prev',\n yhat: Number(p.yhat)||0,\n lo: (band.lo ?? null),\n hi: (band.hi ?? null)\n });\n});\n\n// Saída\nreturn [{\n json: {\n viz: 'time_series',\n label: `Previsão de ${inp.context?.target || 'receita'} (${inp.context?.gran || 'diario'})`,\n kpi: {\n total_prev: totalPrev,\n dias_previstos: dataPrev.length,\n ultimo_observado: ultimoObs,\n conf: conf\n },\n series: {\n labels: [...labelsHist, ...labelsPrev], // se quiser, pode manter separados\n hist: dataHist,\n prev: dataPrev\n },\n bands: {\n // bandas alinhadas com a série de previsão\n labels: labelsPrev,\n lo,\n hi\n },\n table,\n insight: null,\n _debug: {\n n_hist: history.length,\n n_prev: forecast.length\n }\n }\n}];\n"

    },

    "type": "n8n-nodes-base.code",

    "typeVersion": 2,

    "position": [

      896,

      256

    ],

    "id": "f88fb2fc-7735-4380-a3fb-da637376c54f",

    "name": "Predictive – Prepare Render Payload (Code)"

  },

```
{
  "parameters": {
    "modelId": {
      "__rl": true,
      "value": "gpt-4.1-mini",
      "mode": "list",
      "cachedResultName": "GPT-4.1-MINI"
    },
    "messages": {
      "values": [
        {
          "content": "Você é um analista de dados financeiro ultra sucinto. Gere um insight curto (2–4 frases) com base EXCLUSIVA nos dados recebidos. \nREGRAS:\n- NUNCA diga que não há dados se houver itens em \"series.hist\" ou \"series.prev\".\n- Não invente números; use os que recebeu.\n- Se \"series.prev\" existir, cite a direção da tendência prevista (alta/queda/estável) e o total previsto já calculado em \"kpi.total_prev\".\n- Se \"kpi.conf\" existir, mencione o nível de confiança de forma simples (ex.: “IC 90%”).\n- Evite jargão estatístico pesado; escreva para executivos.\n- Saída OBRIGATORIAMENTE em JSON no formato {\"insight\": \"<texto>\"}.\n",
          "role": "system"
        },
        {
          "content": "=Dados:\n- Label da visualização: {{ $json.label }}\n- Métrica alvo: receita\n- Granularidade: {{ $json.label.match(/\\((.+)\\)/)?.[1] || \"n/d\" }}\n- KPI:\n  - total_prev: {{ $json.kpi.total_prev }}\n  - dias_previstos: {{ $json.kpi.dias_previstos }}\n  - ultimo_observado: {{ $json.kpi.ultimo_observado }}\n  - conf: {{ $json.kpi.conf }}\n\nSéries:\n- Histórico (series.labels e series.hist) têm {{ $json.series.hist.length }} pontos.\n- Previsão (series.prev) tem {{ $json.series.prev.length }} pontos.\n\nBandas
```

(se houver):\n- Limites superiores presentes para {{ $json.bands.hi.length || 0 }} pontos de previsão.\n\nInstruções específicas:\n- Se houver pelo menos 1 ponto em \"series.hist\", comente brevemente sobre o nível recente (use \"kpi.ultimo_observado\" como referência).\n- Se houver \"series.prev\", comente a direção geral dos próximos dias e a ordem de grandeza (pode usar \"kpi.total_prev\" como soma prevista).\n- Se \"kpi.conf\" existir, inclua “(IC {{ Math.round($json.kpi.conf*100) }}%)”.\n- NÃO mude granularidade, NÃO peça mais dados, NÃO use condicional “se/então” excessivo.\n\nRetorne SOMENTE:\n{\"insight\": \"...\"}\n"

```
          }

        ]

      },

      "jsonOutput": true,

      "options": {

        "maxTokens": 300,

        "temperature": 0

      }

    },

    "type": "@n8n/n8n-nodes-langchain.openAi",

    "typeVersion": 1.8,

    "position": [

      1104,

      256

    ],

    "id": "d3b65840-0e20-45ee-9f7b-a656fabbfa77",

    "name": "OpenAI – Forecast Insight",
```

```
    "credentials": {
      "openAiApi": {
        "id": "SedjdE4QVcnu8x3b",
        "name": "OpenAi account 2"
      }
    }
  },
  {
    "parameters": {
        "jsCode": "/**\n * Predictive – Attach Context (Code)\n * Lê as linhas do Postgres (preditivo) -> monta history [{ts,y}]\n * Recupera pred_params/pred_context de nós anteriores (nomes flexíveis)\n * Retorna um ÚNICO item: { history: [...], context: {...} }\n */\n\n// 1) History a partir do Postgres – Fetch Dataset (Preditivo)\nconst rows = $input.all()\n  .map(it => {\n    const ts = new Date(it.json.ts);\n    const y = Number(it.json.receita);\n    return (ts.toString() !== 'Invalid Date' && Number.isFinite(y))\n      ? { ts: ts.toISOString(), y }\n      : null;\n  })\n  .filter(Boolean);\n\n// Helper: tenta pegar o .json[0] de vários nomes possíveis\nfunction pickNodeJson(candidates) {\n  for (const name of candidates) {\n    try {\n      const items = $items(name); // não estoura se o nome não existir (cai no catch)\n      if (Array.isArray(items) && items[0]?.json) return items[0].json;\n    } catch (_) {}\n  }\n  return null;\n}\n\n// 2) Tenta achar os nós pelos nomes que você pode ter usado\nconst normJson = pickNodeJson([\n  'Normalize Input',\n  'Normalize Inputs1',\n  'Normalize Inputs',\n  'Normalize'\n]);\n\nconst buildSqlJson = pickNodeJson([\n  'Predictive - Build Dataset SQL',      // hífen normal\n  'Predictive – Build Dataset SQL',      // travessão\n  'Build Dataset SQL (Preditivo)',\n  'Predictive Build SQL'\n]);\n\n// 3) Extrai parâmetros de onde existir\nconst uiPred = normJson?.pred_params ?? {};\nconst built  = buildSqlJson?.pred_context ?? {};\n\n// 4) Granularidade e passos (aceita 'diario', 'diária', 'daily', etc.)\nconst granRaw = (uiPred.granularidade || built.gran || 'diario').toString().toLowerCase();\nconst isDaily = ['diario', 'diária', 'diaria', 'daily', 'day'].includes(granRaw);\nconst gran   = isDaily ? 'diario' : 'mensal';\nconst passos  = isDaily ? 'dias' : 'meses';\n\n// 5) Monta contexto final (com defaults seguros)\nconst context = {\n  target: uiPred.target_metric || built.target || 'receita',\n  gran,\n  dim: uiPred.grupo_dim || built.dim || 'TOTAL',\n
```

treino_frac: built.treino_frac ?? 0.8,\n start: uiPred.treino_inicio || built.start || null,\n end: uiPred.treino_fim || built.end || null,\n uf: built.uf || 'ALL',\n horizonte: Number(uiPred.horizonte ?? 14),\n passos,\n conf: Number(uiPred.confianca ?? 0.9),\n algoritmo: uiPred.algoritmo || 'auto'\n};\n\n// 6) Retorna UM item com history + context\nreturn [\n {\n   json: {\n   history: rows,\n   context\n   }\n }\n];\n"

    },

    "type": "n8n-nodes-base.code",

    "typeVersion": 2,

    "position": [

      528,

      256

    ],

    "id": "10d0db2c-2686-4f27-9665-0b62ce33630f",

    "name": "Predictive – Attach Context (Code)"

  },

  {

    "parameters": {

      "modelId": {

        "__rl": true,

        "value": "gpt-4o-mini",

        "mode": "list",

        "cachedResultName": "GPT-4O-MINI"

      },

      "messages": {

```
"values": [
  {
    "content": "Você é um analista de dados gerenciais. Sua tarefa é escrever um insight curto, claro e acionável, SEM inventar números fora do que recebeu. \n- Use SOMENTE o conteúdo do JSON fornecido (kpi, series/pie/scatter, viz, table, label).\n- NÃO crie métricas que não existam no JSON.\n- Não repita o JSON; produza texto humano.\n- Tamanho: 1–2 parágrafos (máx. ~120 palavras no total).\n- Se viz='time_series': comente tendência, sazonalidade e mês/delta mais marcante.\n- Se viz='bar' ou 'pie': destaque top categorias e concentração (ex.: “top 3 respondem por ~X%” se claro no JSON).\n- Se viz='scatter': interprete o coeficiente de correlação (ρ) qualitativamente: \n  |ρ| ≥ 0.7 (forte), 0.4–0.7 (moderada), 0.2–0.4 (fraca), < 0.2 (quase nula); sinal (+/−) importa.\n- Sempre finalize com 1 ação recomendada objetiva (ex.: revisar mix, ajustar preço, priorizar clientes/regiões).\n- Se os dados forem insuficientes, diga isso explicitamente e sugira a próxima coleta/recorte.\n",
    "role": "system"
  },
  {
    "content": "=Contexto do relatório (NÃO invente nada fora do JSON):\n{{ JSON.stringify($json) }}\n\nPergunta original:\n{{ $prevNode[\"Normalize Inputs\"].json.pergunta }}\n"
  }
 ]
},
"simplify": false,
"options": {
  "maxTokens": 350,
  "temperature": 0.3
}
```

```
  },
  "type": "@n8n/n8n-nodes-langchain.openAi",
  "typeVersion": 1.8,
  "position": [
    1952,
    -96
  ],
  "id": "093b8843-ac78-45f2-b7c1-d3a30ea3603b",
  "name": "OpenAI – Renderer (HTML)",
  "credentials": {
    "openAiApi": {
      "id": "SedjdE4QVcnu8x3b",
      "name": "OpenAi account 2"
    }
  }
},
{
  "parameters": {
    "assignments": {
      "assignments": [
        {
          "id": "0eb50eda-082d-4f12-a375-3f964c2f568c",
          "name": "html",
```

"value": "={{ ` <!doctype html><html lang=\"pt-br\"><head> <meta charset=\"utf-8\"/><title>GAIA – Agentes EDA</title> <meta name=\"viewport\" content=\"width=device-width, initial-scale=1\"/> <link href=\"https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap\" rel=\"stylesheet\"/> <style> :root{--gaia-green:#2E7D68;--gaia-soft:rgba(46,125,104,.05)} body{font-family:Poppins,Arial,sans-serif;margin:0;display:flex;justify-content:center} .container{background:#fff;max-width:980px;width:100%;padding:30px 40px;margin:24px;border:1px solid #eee;border-radius:12px;box-shadow:0 4px 15px rgba(0,0,0,.06)} h1{margin:0 0 12px}h2{margin:16px 0 8px} .card{background:var(--gaia-soft);border:1px solid rgba(46,125,104,.12);border-radius:10px;padding:14px 16px;margin:12px 0} .muted{color:#666}.chart-wrap{background:#fff;border:1px solid #eee;border-radius:12px;padding:16px} .warn{padding:12px;background:#fff7f7;border:1px solid #ffdede;border-radius:8px;color:#a40000;margin:12px 0} details{margin-top:14px}code{background:#f6f6f6;border:1px solid #eee;border-radius:6px;padding:2px 6px} </style></head><body><div class=\"container\"> <h1>Relatório Preditivo</h1> <p class=\"muted\">Gerado pelo fluxo n8n <h2>Insight</h2> <div class=\"card\"> ${ (function(){ const it0 = (($items()[0]||{}).json)||{}; return (it0.message && it0.message.content && it0.message.content.insight) ? it0.message.content.insight : '—'; })() } <h2>Contexto</h2> <div class=\"card\"> ${ (function(){ const it1 = (($items()[1]||{}).json)||{}; const ctx = it1.context || {}; const gran = ctx.gran || ctx.granularidade || '—'; const hor = ctx.horizonte || ctx.horizon || '—'; const tgt = ctx.target || 'receita'; return ' Granularidade: '+gran+' · Horizonte: '+hor+' · Métrica: '+tgt+' '; })() } <h2>Série temporal</h2> <div class=\"chart-wrap\"> <div id=\"chart\" style=\"width:100%;height:260px\"> <div id=\"warn\" class=\"warn\" style=\"display:none\">Sem dados suficientes para plotar. Verifique o dataset e o intervalo. <details><summary>Debug</summary><div id=\"dbg\" style=\"font-size:13px;color:#444\"> </details> <script>\n // Dados resolvidos no servidor n8n e injetados como constantes\n const IT1 = ${ JSON.stringify((($items()[1]||{}).json)||{}) };\n\n // 1) Histórico {ts,y}\n const history = Array.isArray(IT1.history) ? IT1.history.map(p => ({ ts:String(p.ts), y:Number(p.y) })) : [];\n\n // 2) Previsão {ts,yhat} → {ts,y}\n const forecast = Array.isArray(IT1.forecast) ? IT1.forecast.map(p => ({ ts:String(p.ts), y:Number(p.yhat) })) : [];\n\n // 3) Intervalos {ts,lo,hi} → lower/upper\n const lower = Array.isArray(IT1.intervals) ? IT1.intervals.map(p => ({ ts:String(p.ts), y:Number(p.lo) })) : [];\n const upper = Array.isArray(IT1.intervals) ? IT1.intervals.map(p => ({ ts:String(p.ts), y:Number(p.hi) })) :

[];\n\n // 4) Labels e alinhamento\n const labels = history.map(p=>p.ts).concat(forecast.map(p=>p.ts));\n const pick = (arr)=>labels.map(ts => {\n const f = (arr||[]).find(p => p.ts===ts);\n return (f && isFinite(f.y)) ? Number(f.y) : null;\n });\n const yHist = pick(history);\n const yFcst = pick(forecast);\n const yLower = pick(lower);\n const yUpper = pick(upper);\n\n // Debug visível ao abrir o <details>\n const dbg = document.getElementById('dbg');\n const short = arr => (arr||[]).slice(0,3);\n dbg.innerHTML =\n ' history: '+(history.length||0)+' – '+JSON.stringify(short(history))+' '+\n ' forecast: '+(forecast.length||0)+' – '+JSON.stringify(short(forecast))+' '+\n ' intervals: '+((IT1.intervals&&IT1.intervals.length)||0)+' '+\n ' labels: '+(labels.length||0)+' – '+JSON.stringify(labels.slice(0,6))+' ';\n\n // 5) Desenha SVG (sem libs)\n const el = document.getElementById('chart');\n const W = Math.max(el.getBoundingClientRect().width || el.clientWidth || 920, 320);\n const H = el.clientHeight || 260;\n const PAD = 28, plotW = W - PAD*2, plotH = H - PAD*2;\n\n // Escalas\n function minMax(...series){\n const vals = series.flat().filter(v => v!=null && isFinite(v));\n const mn = Math.min.apply(null, vals);\n const mx = Math.max.apply(null, vals);\n return [mn, mx];\n }\n const [yMin, yMax] = minMax(yHist, yFcst, yLower, yUpper);\n const xStep = labels.length > 1 ? plotW/(labels.length-1) : plotW;\n const yScale = (v) => {\n if (!isFinite(yMin) || !isFinite(yMax) || yMax===yMin) return plotH/2;\n return plotH - ((v - yMin) / (yMax - yMin)) * plotH;\n };\n const X = (i)=> PAD + i*xStep;\n const Y = (v)=> PAD + yScale(v);\n\n // Helpers\n function makePath(arr){\n let d = '', started = false;\n arr.forEach((y,i)=>{\n if (y==null) { started = false; return; }\n const x = X(i), yy = Y(y);\n d += (started ? ' L ' : ' M ') + x + ' ' + yy;\n started = true;\n });\n return d.trim();\n }\n function bandPath(lowerArr, upperArr){\n const top = [], bottom = [];\n for (let i=0;i<labels.length;i++){\n const lo = lowerArr[i], hi = upperArr[i];\n if (lo!=null && hi!=null){ top.push([X(i), Y(hi)]); bottom.push([X(i), Y(lo)]); }\n }\n if (!top.length) return '';\n const pts = top.concat(bottom.reverse());\n return 'M '+pts.map(p=>p[0]+' '+p[1]).join(' L ')+' Z';\n }\n function dots(arr, r, color){\n let s = '';\n arr.forEach((y,i)=>{\n if (y!=null) s += '<circle cx=\"'+X(i)+'\" cy=\"'+Y(y)+'\" r=\"'+r+'\" fill=\"'+color+'\" />';\n });\n return s;\n }\n\n const hasAny = labels.length>0 && (yHist.some(v=>v!=null) || yFcst.some(v=>v!=null));\n if (!hasAny) document.getElementById('warn').style.display = 'block';\n\n const pathHist = makePath(yHist);\n const pathFcst = makePath(yFcst);\n const pathBand = (yLower.some(v=>v!=null) && yUpper.some(v=>v!=null)) ? bandPath(yLower,yUpper) : '';\n\n const svg =\n '<svg width=\"'+W+'\" height=\"'+H+'\" viewBox=\"0 0 '+W+' '+H+'\" xmlns=\"http://www.w3.org/2000/svg\">'+\n // Eixos\n '<g stroke=\"#e9e9e9\" stroke-width=\"1\">'+\n '<line x1=\"'+PAD+'\" y1=\"'+PAD+'\" x2=\"'+PAD+'\" y2=\"'+(H-PAD)+'\"/>'+\n '<line x1=\"'+PAD+'\" y1=\"'+(H-PAD)+'\"

x2=\"'+(W-PAD)+'\" y2=\"'+(H-PAD)+'\"/>'+\n '</g>'+\n // Banda\n (pathBand ? '<path d=\"'+pathBand+'\" fill=\"rgba(46,125,104,0.12)\" stroke=\"none\"/>' : '')+\n // Histórico\n (pathHist ? '<path d=\"'+pathHist+'\" fill=\"none\" stroke=\"#2E7D68\" stroke-width=\"2\"/>' : '')+\n // Previsão\n (pathFcst ? '<path d=\"'+pathFcst+'\" fill=\"none\" stroke=\"#8EC9B3\" stroke-width=\"2\" stroke-dasharray=\"6 6\"/>' : '')+\n // Pontos (facilitam ver que desenhou)\n dots(yHist, 2, '#2E7D68') +\n dots(yFcst, 2, '#8EC9B3') +\n '</svg>';\n\n el.innerHTML = svg;\n\n // Sanity check: se ambas as linhas ficaram vazias, mostra aviso\n if (!pathHist && !pathFcst) {\n document.getElementById('warn').style.display = 'block';\n dbg.innerHTML += ' sanity: pathHist/pathFcst vazios ';\n }\n</script>\n </body></html> `\n}}",

"type": "string"

}

]

},

"includeOtherFields": "={{ false }}",

"options": {}

},

"type": "n8n-nodes-base.set",

"typeVersion": 3.4,

"position": [

1632,

400

],

"id": "848e0dfa-ce41-426f-8480-b8fe83d0014f",

"name": "Build HTML (Set)"

},

```
{
  "parameters": {
    "respondWith": "text",
    "responseBody": "={{$json.data}}",
    "options": {
      "responseCode": 200,
      "responseHeaders": {
        "entries": [
          {
            "name": "Content-Type",
            "value": "text/html; charset=utf-8"
          },
          {
            "name": "Cache-Control",
            "value": "no-store"
          },
          {
            "name": "X-Content-Type-Options",
            "value": "nosniff"
          },
          {
            "name": "Referrer-Policy",
            "value": "no-referrer"
```

```
          }
        ]
      }
    }
  },
  "type": "n8n-nodes-base.respondToWebhook",
  "typeVersion": 1.4,
  "position": [
    2448,
    -96
  ],
  "id": "dee3ef97-9080-4732-b19e-73926402c6dc",
  "name": "Respond to Webhook1"
},
{
  "parameters": {},
  "type": "n8n-nodes-base.merge",
  "typeVersion": 3.2,
  "position": [
    1424,
    400
  ],
  "id": "e4311940-2114-47f8-9f95-eb250fcb1bf5",
```

```
  "name": "Merge1"
},
{
  "parameters": {
    "content": "\n\n\nFLUXO PADRÃO EDA"
  },
  "type": "n8n-nodes-base.stickyNote",
  "typeVersion": 1,
  "position": [
    96,
    -224
  ],
  "id": "26e8915a-da72-478e-a688-47e5e15b9fce",
  "name": "Sticky Note"
},
{
  "parameters": {
    "content": "\n\n\nFLUXO DE ANÁLISES PREDITIVAS"
  },
  "type": "n8n-nodes-base.stickyNote",
  "typeVersion": 1,
  "position": [
    64,
```

```
      432
    ],
    "id": "81ec427c-bc1e-4db1-9c46-e8a0317c4fb4",
    "name": "Sticky Note1"
  }
],
"pinData": {},
"connections": {
  "Webhook": {
    "main": [
      [
        {
          "node": "Normalize Inputs",
          "type": "main",
          "index": 0
        }
      ]
    ]
  },
  "Normalize Inputs": {
    "main": [
      [
        {
```

```
        "node": "If",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Validate Plan JSON": {
  "main": [
    [
      {
        "node": "Build SQL",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"KPIs & Série/Scatter": {
  "main": [
    [
      {
        "node": "OpenAI – Renderer (HTML)",
```

```
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Build SQL": {
  "main": [
    [
      {
        "node": "Inline Params",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Postgres - Rho": {
  "main": [
    [
      {
        "node": "Merge",
        "type": "main",
```

```
        "index": 1
      }
    ]
  ]
},
"Postgres - Main Query": {
  "main": [
    [
      {
        "node": "Merge",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"OpenAI - NL → Plan (JSON)": {
  "main": [
    [
      {
        "node": "Validate Plan JSON",
        "type": "main",
        "index": 0
```

```
      }
    ]
  ]
},
"IF - Is Dual?": {
  "main": [
    [
      {
        "node": "Postgres - Rho",
        "type": "main",
        "index": 0
      }
    ],
    [
      {
        "node": "Empty For Merge",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"HTML Renderer": {
```

```
  "main": [
    [
      {
        "node": "Respond to Webhook1",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Empty For Merge": {
  "main": [
    [
      {
        "node": "Merge",
        "type": "main",
        "index": 1
      }
    ]
  ]
},
"Merge": {
  "main": [
```

```
    [
      {
        "node": "KPIs & Série/Scatter",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Inline Params": {
  "main": [
    [
      {
        "node": "Postgres - Main Query",
        "type": "main",
        "index": 0
      },
      {
        "node": "IF - Is Dual?",
        "type": "main",
        "index": 0
      }
    ]
```

```
  ]
},
"If": {
  "main": [
    [
      {
        "node": "Predictive - Build Dataset SQL",
        "type": "main",
        "index": 0
      }
    ],
    [
      {
        "node": "OpenAI - NL → Plan (JSON)",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Predictive - Build Dataset SQL": {
  "main": [
    [
```

```
      {
        "node": "Inline Params1",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Postgres – Fetch Dataset": {
  "main": [
    [
      {
        "node": "Predictive – Attach Context (Code)",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Predictive - Train & Forecast": {
  "main": [
    [
      {
```

```
        "node": "Predictive – Prepare Render Payload (Code)",
        "type": "main",
        "index": 0
      },
      {
        "node": "Merge1",
        "type": "main",
        "index": 1
      }
    ]
  ]
},
"Inline Params1": {
  "main": [
    [
      {
        "node": "Postgres – Fetch Dataset",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
```

```
"Predictive – Prepare Render Payload (Code)": {
  "main": [
    [
      {
        "node": "OpenAI – Forecast Insight",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"OpenAI – Forecast Insight": {
  "main": [
    [
      {
        "node": "Merge1",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Predictive – Attach Context (Code)": {
```

```
  "main": [
    [
      {
        "node": "Predictive - Train & Forecast",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"OpenAI – Renderer (HTML)": {
  "main": [
    [
      {
        "node": "HTML Renderer",
        "type": "main",
        "index": 0
      }
    ]
  ]
},
"Build HTML (Set)": {
  "main": [
```

```
      [
       {
         "node": "Respond to Webhook",
         "type": "main",
         "index": 0
       }
      ]
    ]
  },
  "Merge1": {
    "main": [
      [
       {
         "node": "Build HTML (Set)",
         "type": "main",
         "index": 0
       }
      ]
    ]
  }
},
"active": true,
"settings": {
```

```
    "executionOrder": "v1"
  },
  "versionId": "1f8e2749-e37b-4937-a0ec-21ecab35ab05",
  "meta": {
    "templateCredsSetupCompleted": true,
                                                                    "instanceId":
"3d3847ebf65ddab79f5da58d36912f2345e8fc147182b168485331ee5bb28fc6"
  },
  "id": "BwPbCyg8xbGNt0N3",
  "tags": []
}
```