Case Highway – Visio.AI

A ferramenta de visualização utilizada foi o Power BI.
Para demonstrar minha capacidade em SQL e tratamento de tabelas, criei uma query no Google BigQuery consolidando os dados em uma **One Big Table**. Mesmo assim, no Power BI eu poderia chegar ao mesmo resultado usando relacionamentos e cálculos. Optei por **importar as tabelas individualmente no Power BI** para manter maior granularidade e, ao final, comparei os valores agregados das duas abordagens para validar a consulta SQL.

No Power BI criei **tabelas dimensão** (como Dim_Loja, Dim_Delivery e Dim_Data) para estruturar os relacionamentos no modelo. Uma tabela dimensão nada mais é do que uma lista sem repetições — por exemplo, a Dim_Data contém todas as datas do período, sem duplicatas. Essa dimensão eu criei em **linguagem M no Power Query** a partir de um script que já tenho pronto, onde além das datas calculo também dias úteis, finais de semana etc.

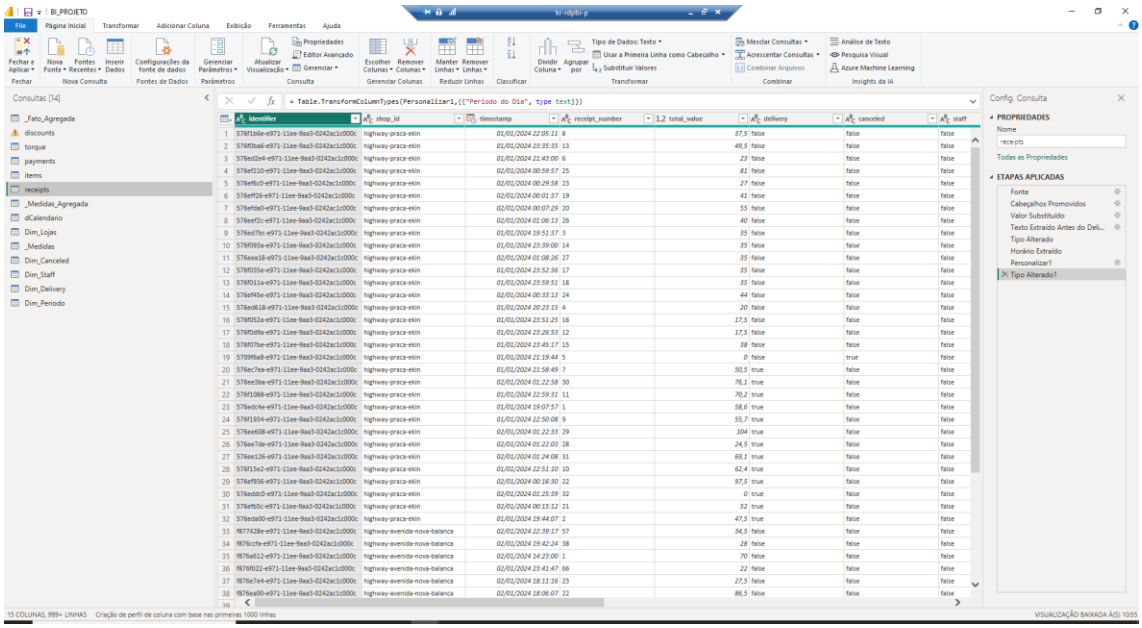
Em seguida criei **medidas em DAX** (Data Analysis Expressions) para todos os cálculos dinâmicos do painel, como faturamento total, score e indicadores financeiros. Também utilizei **fórmulas em M** no Power Query para pré-calcular alguns campos, como o período do dia para cada cupom.

Para o layout, desenhei os **backgrounds no Figma** e importei para o Power BI. Criei **botões de navegação** entre páginas, um **filtro lateral interativo** para filtrar/desfiltrar rapidamente, e indicadores com botões que alternam os visuais.

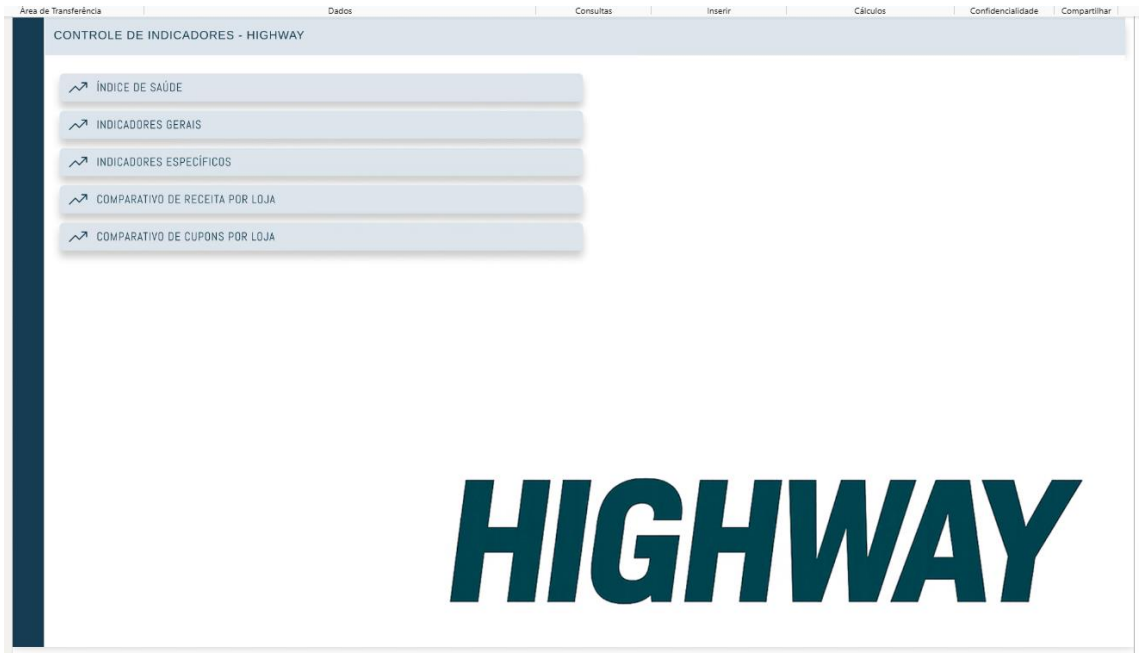
O painel ficou organizado assim:

- **Capa**: página inicial com direcionamento para as demais abas.
- **Índice de Saúde**: página principal para o Lorenzzo acompanhar a saúde financeira das lojas com base no índice construído.
- **Indicadores Gerais**: aba com faturamento e receita por loja e tipo (ex.: Delivery), evolução mensal e uma matriz resumo (receita, torque, descontos etc.) por loja. Nesta aba usei **cards (Big Numbers)** para mostrar receita, faturamento etc. Há um botão que leva para uma visão comparativa das lojas, com quatro gráficos de coluna + linha mostrando receita e ticket médio ao longo do tempo.
- **Indicadores Específicos**: aqui uso gráficos de barra horizontal para ver os itens mais vendidos, e análises temporais de número de cupons e torque. Também criei um botão para navegar para a análise detalhada por loja, além de medir cupons por dia da semana.

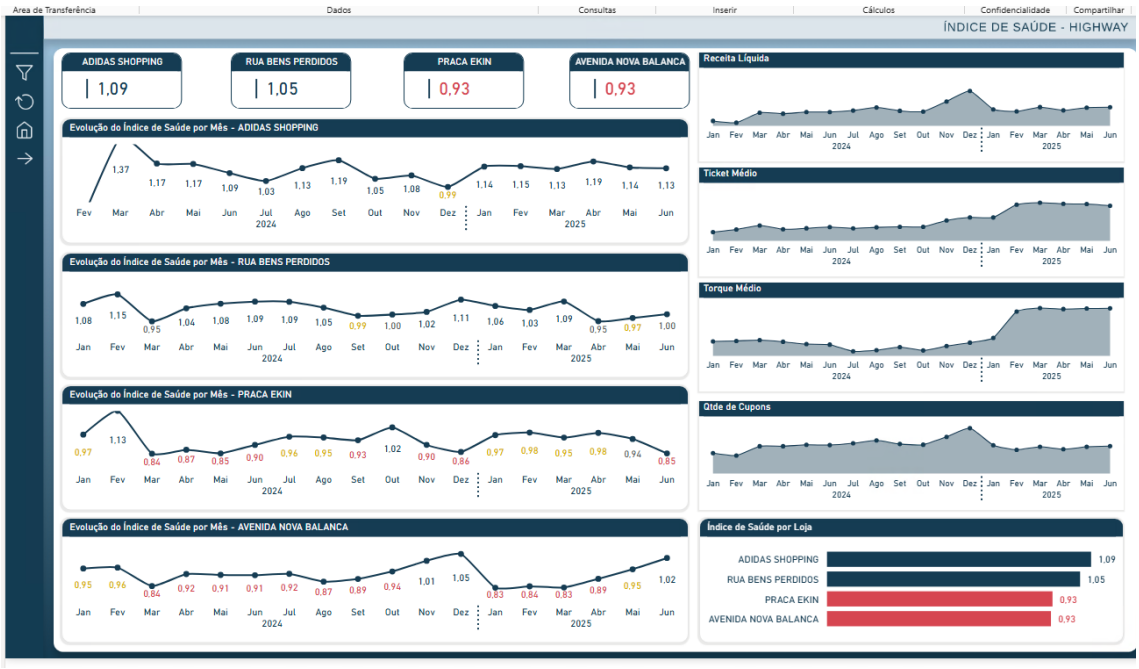
Tratamento no Power Query:



Capa:



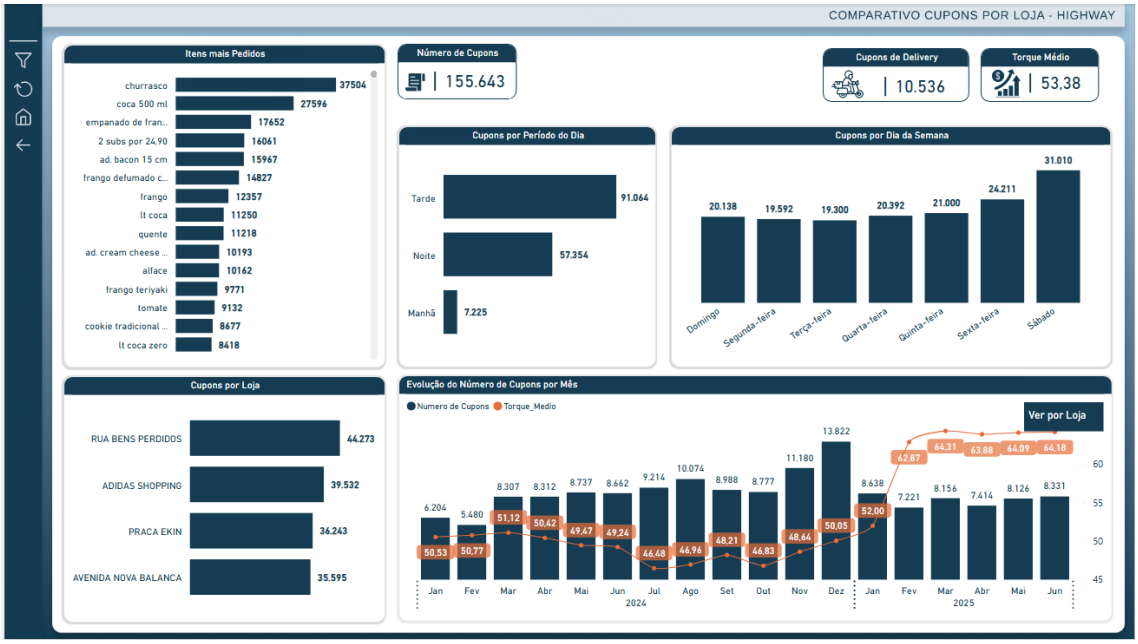
Indicadores de Saúde:



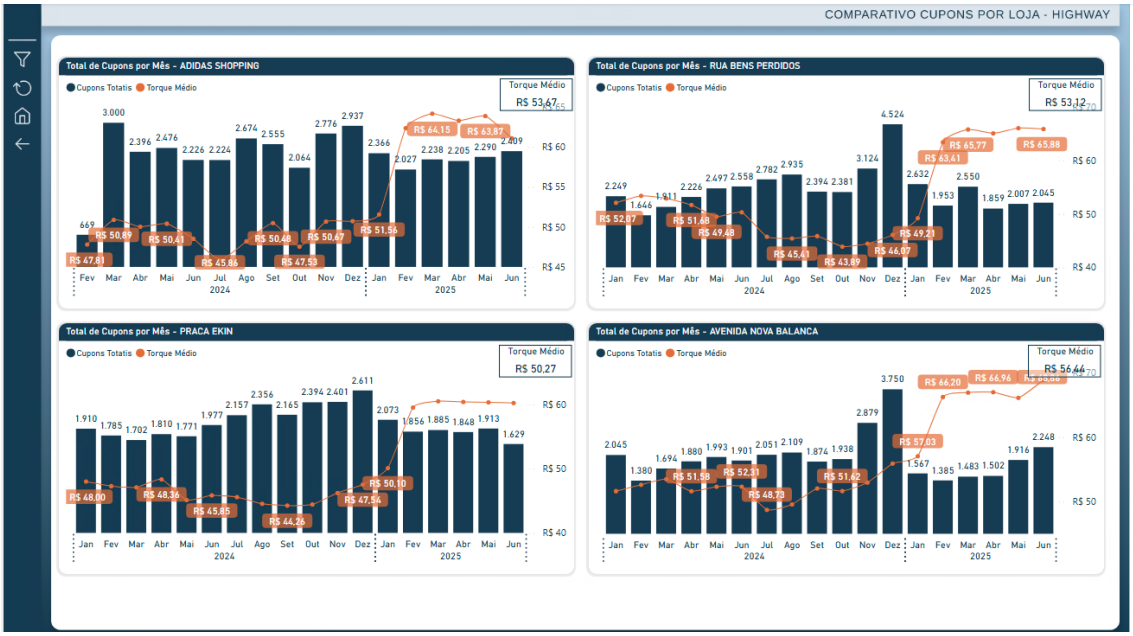
Indicadores Gerais:

Comparativo de Receita:

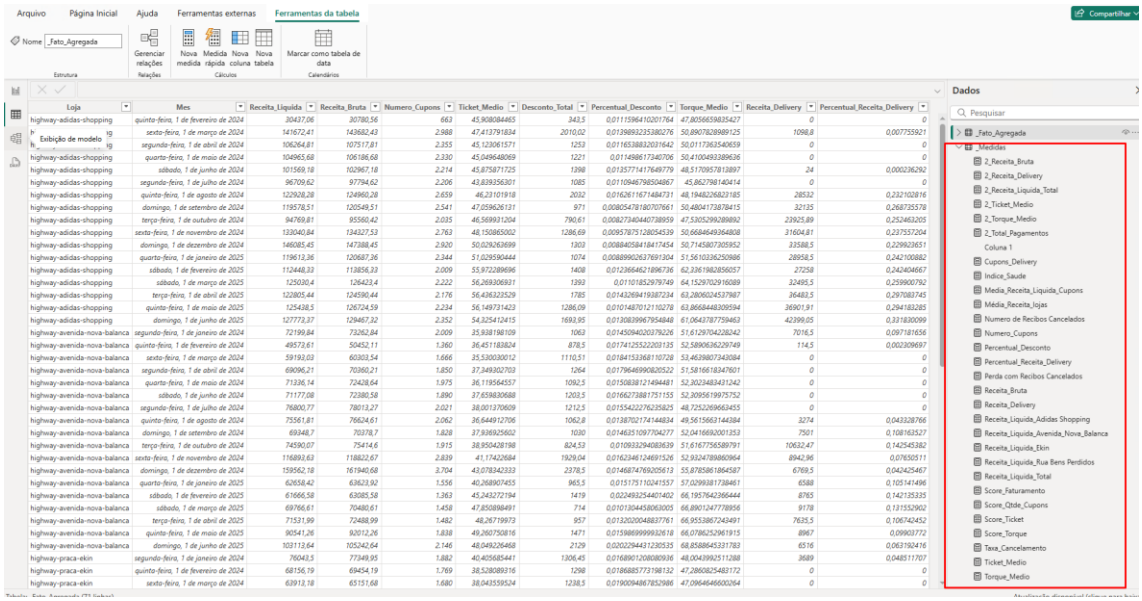
Indicadores específicos:



Comparativo de Cupons:



Medidas de apoio cridas:



Indicadores para a criação dos botões:

Principais fórmulas DAX que utilizei (em linguagem natural):

- Receita_Liquida_Total → soma o valor total de cada recibo mais taxa/fee.
- Receita_Bruta → soma o total dos itens sem taxas/descontos.
- Receita_Delivery → calcula a receita líquida apenas dos recibos marcados como Delivery.
- Numero_Cupons → conta de forma distinta os identificadores de recibos (quantos cupons/atendimentos).
- Torque_Medio → média do torque líquido.
- Total_Descontos → soma dos descontos aplicados.
- Score_Faturamento, Score_Qtde_Cupons, Score_Ticket e Score_Torque → cada um divide o valor da loja pela média de todas as lojas (função AVERAGEX(ALL())) para gerar um índice comparativo. Assim sei se cada loja está acima (>1) ou abaixo (<1) da média da rede.

Funções DAX básicas usadas nas suas medidas

- **SUM(tabela[coluna])**
  Soma todos os valores de uma coluna.
  Ex.: SUM(Items[total_value]) → soma todos os valores de venda bruta de itens.

- **SUMX(tabela, expressão)**
  Percorre linha a linha de uma tabela e avalia a expressão em cada linha, somando os resultados.
  Ex.: SUMX(Receipts, Receipts[total_value] + Receipts[fee]) → para cada recibo soma valor + taxa, e no final soma tudo.

- **CALCULATE(expressão, filtros)**
  Avalia uma expressão no contexto modificado por filtros.
  Ex.: CALCULATE([Receita_Liquida_Total], Receipts[delivery] = "true") → calcula a receita líquida apenas dos recibos marcados como delivery.

- **DISTINCTCOUNT(tabela[coluna])**
  Conta quantos valores distintos existem em uma coluna (não conta duplicados).
  Ex.: DISTINCTCOUNT(Receipts[identifier]) → quantos cupons únicos foram emitidos.

- **AVERAGE(tabela[coluna])**
  Retorna a média simples dos valores de uma coluna.
  Ex.: AVERAGE(Torque[net_torque]) → média do torque.

- **AVERAGEX(tabela, expressão)**
  Avalia uma expressão para cada linha da tabela e depois calcula a média dos resultados.
  Ex.: AVERAGEX(ALL('Dim_Lojas'), [Receita_Liquida_Total]) → média de receita líquida considerando todas as lojas.

- **ALL(tabela)**
  Remove filtros aplicados àquela tabela/coluna.
  Ex.: usado dentro de AVERAGEX(ALL('Dim_Lojas'), ...) para calcular a média de todas as lojas sem considerar filtros do relatório.

- **DIVIDE(numerador, denominador, alternativo)**
  Divide numerador por denominador e retorna um valor alternativo (geralmente 0) caso o denominador seja 0 ou vazio.
  Ex.: DIVIDE([Receita_Liquida_Total], média de todas as lojas, 0) → calcula o score de faturamento com segurança.

Script em linguagem M da dimensão de calendário que utilizo:

```
let
  DataMin = List.Min(receipts[operation_date]), // Alterar de acordo com o nome da coluna e da base//
  DataMax = List.Max(receipts[operation_date]),  // Alterar de acordo com o nome da
  QtdDias = Duration.Days(DataMax - DataMin) + 1,
  ListaCriada = List.Dates(DataMin, QtdDias, #duration(1, 0, 0, 0)),
  Convertido_Tabela = Table.RenameColumns(
   Table.TransformColumnTypes(
    Table.FromList(ListaCriada, Splitter.SplitByNothing(), null, null, ExtraValues.Error),
    {{"Column1", type date}}
   ),
   {{"Column1", "Data"}}
  ),
  Dia_Analises = Table.TransformColumnTypes(
   Table.AddColumn(
    Table.AddColumn(
     Table.AddColumn(
      Table.AddColumn(Convertido_Tabela, "Dia", each Date.Day([Data]), Int64.Type),
      "Dia_da_Semana",
      each
       if Date.DayOfWeek([Data], Day.Sunday) = 0 then
         1
       else
         Date.DayOfWeek([Data], Day.Sunday) + 1,
      Int64.Type
     ),
     "Dia_Nome",
```

```
    each
     let
      dayName = Date.DayOfWeekName([Data]),
      capitalizedDayName = Text.Upper(Text.Start(dayName, 1))
       & Text.Lower(Text.Middle(dayName, 1))
     in
      capitalizedDayName,
    type text
   ),
   "Tipo_Dia",
   each
    if [Dia_da_Semana] = 1 then
     "Fim de semana"
    else if [Dia_da_Semana] = 7 then
     "Fim de semana"
    else
     "Dia útil"
  ),
  {{"Tipo_Dia", type text}}
),
Semana_Analises = Table.AddColumn(
  Dia_Analises,
  "Semana_do_Mes",
  each "S" & Text.From(Date.WeekOfMonth([Data])),
  type text
),
Mes_Analises = Table.AddColumn(
  Table.AddColumn(
```

```
    Table.AddColumn(Semana_Analises, "Mes_Num", each Date.Month([Data]),
Int64.Type),
    "Mes_Nome",
    each
     let
       monthName = Date.MonthName([Data]),
       formattedMonthName = Text.Upper(Text.Start(monthName, 1))
        & Text.Lower(Text.Middle(monthName, 1))
     in
       formattedMonthName,
    type text
  ),
  "Mes_Abreviado",
  each
   let
     monthName = Date.MonthName([Data]),
     shortMonthName = Text.Upper(Text.Start(monthName, 1))
       & Text.Lower(Text.Middle(monthName, 1, 2))
   in
     shortMonthName,
  type text
 ),
 Ano_Analises = Table.TransformColumnTypes(
  Table.AddColumn(
   Table.AddColumn(Mes_Analises, "Ano", each Date.Year([Data]), Int64.Type),
   "Semestre",
   each if [Mes_Num] <= 6 then "1º Semestre" else "2º Semestre"
  ),
```

{{"Semestre", type text}}

)

in

Ano_Analises

Explicação:

1. Definição do intervalo de datas

DataMin: Encontra a data **mais antiga** da coluna operation_date da tabela receipts.

DataMax: Encontra a data **mais recente** da mesma coluna.

QtdDias: Calcula a quantidade total de dias entre DataMin e DataMax, somando 1 para incluir ambos os extremos.

2. Criação da lista de datas

Cria uma **lista contínua de datas** começando em DataMin e indo até DataMax, com incremento de 1 dia (#duration(1, 0, 0, 0)).

3. Conversão da lista em tabela

Transforma a lista de datas em uma tabela.

Altera o tipo da coluna para date.

Renomeia a coluna de "Column1" para "Data".

4. Adição de colunas para análise diária

- "Dia": Extrai o número do dia do mês.
- "Dia_da_Semana": Converte o dia da semana para número de 1 (domingo) a 7 (sábado).
- "Dia_Nome": Nome do dia (ex: Segunda-feira), com a primeira letra maiúscula.
- "Tipo_Dia":
- "Fim de semana" se o dia for domingo (1) ou sábado (7),
- "Dia útil" caso contrário.

5. Adição da semana do mês

Cria a coluna **"Semana_do_Mes"** com o número da semana dentro do mês, prefixado com "S" (ex: S1, S2…).

6. Adição de colunas relacionadas ao mês

- "Mes_Num": Número do mês (1 a 12).
- "Mes_Nome": Nome completo do mês com a primeira letra em maiúscula.
- "Mes_Abreviado": Abreviação com 3 letras (ex: "Jan", "Fev").

Resultado final

Coluna	**Descrição**
Data	Data completa
Dia	Dia do mês (1–31)
Dia_da_Semana	Número do dia da semana (1=Domingo)
Dia_Nome	Nome do dia da semana
Tipo_Dia	"Dia útil" ou "Fim de semana"
Semana_do_Mes	Semana dentro do mês (S1, S2, etc.)
Mes_Num	Número do mês (1–12)
Mes_Nome	Nome do mês
Mes_Abreviado	Abreviação do mês (3 letras)
Ano	Ano da data
Semestre	1º ou 2º semestre