

A Jornada do Analista Moderno: Das Planilhas à Análise Avançada

Como escalar suas habilidades e impacto, combinando a agilidade do Excel com o poder do Python, PySpark e plataformas de BI.

	A	B	C	D	E	F	G	H	I
1									
2	Date	Sales	Region						
3	23/10/21	123.45	Mas						
4	12/12/21	123.45	Wes						
5	12/16/21	123.45	East						
6	12/10/21	123.45	East						
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									
20									
21									

```
import pandas as pd  
  
df = pd.read_csv('data.csv')  
  
df.groupby('region')['sales'].sum()  
df.groupby(sum())  
  
df = pd.read_csv('data.csv')  
df.groupby('region')  
  
from pyspark.sql import SparkSession  
spark = SparkSession.builder...  
spark = SparkSession.builder...sum()  
  
df.write.parquet(".write.parquet")  
df.write.parquet()
```



O Ponto de Partida Universal: O Domínio do Excel

O Excel é a ferramenta de entrada para a maioria dos problemas de dados. É rápido, familiar e incrivelmente eficaz para exploração, relatórios rápidos e comunicação com stakeholders. Para datasets de pequeno a médio porte, muitas vezes é tudo o que você precisa.



Velocidade para Análises Rápidas

Tabelas dinâmicas, PROCV e dashboards básicos resolvem a maioria dos problemas do dia a dia.



Familiaridade

A interface visual é intuitiva e universalmente compreendida no mundo dos negócios.



Compartilhamento Fácil

Arquivos `xlsx` são o padrão para compartilhar análises e relatórios ad-hoc.



Os Sinais de Ruptura: Quando a Planilha Atinge Seus Limites

A ferramenta certa depende da tarefa. O Excel começa a apresentar problemas quando a complexidade e a escala aumentam. Você está enfrentando esses desafios?

-  **Volume de Dados:** O arquivo trava com mais de 100.000 linhas? (Python processa milhões sem esforço).
-  **Tarefas Repetitivas Diárias:** Você gasta horas atualizando os mesmos relatórios manualmente? (Scripts em Python automatizam o processo).
-  **Conexão com APIs:** O Power Query se torna lento ou instável para buscar dados externos? (A biblioteca `requests` do Python é robusta e direta).
-  **Análise Avançada:** Aninhando incontáveis funções `SE` para engenharia de features? (Scikit-learn em Python é feito para isso).
-  **Colaboração e Versionamento:** A equipe se perde em múltiplas versões de arquivos? (Git/GitHub para código oferece controle total).

Métrica de Impacto: "Um analista na Flipkart reduziu o tempo de atualização diária de dashboards de 2 horas para 10 minutos automatizados com um pipeline em Python."

O Próximo Passo na Evolução: Python para Análise de Dados

A transição para o Python não é sobre substituir o Excel, mas sobre adicionar uma ferramenta mais poderosa para automação, escala e análises complexas. **Ele funciona onde o Excel falha.**

Ecossistema Poderoso



Pandas: A "planilha em código" para manipulação de dados estruturados.



PySpark: Para análise distribuída em datasets massivos na nuvem (Databricks).



SQLAlchemy / pyodbc: Para se conectar diretamente a bancos de dados SQL, eliminando exportações manuais.



Matplotlib / Seaborn: Para criar visualizações ricas e customizáveis.

"Em vez de clicar em etapas, você escreve a lógica uma vez e a reutiliza. Isso torna seu trabalho mais rápido, mais confiável e mais fácil de explicar."

Pedra de Roseta: Traduzindo suas Habilidades do Excel para o Python (Pandas)

Tarefa no Excel	Equivalente em Python (Pandas)	Vantagem do Python
=SOMA(B2:B1000)	<code>df['valor'].sum()</code>	Lida com bilhões de linhas; encadeável.
PROCV(A2; D:F; 2; FALSO)	<code>df.merge(outro_df, on='id_cliente')</code>	Múltiplas chaves, joins no estilo SQL.
Tabela Dinâmica (Soma de Valor por Cidade)	<code>df.groupby('cidade')['valor'].sum()</code>	Scriptável, exportável, base para gráficos.
Gráfico de Barras	<code>sns.barplot(x='cidade', y='valor', data=df)</code>	Interativo (com Plotly), pronto para publicação.
Filtro (Cidade = 'São Paulo')	<code>df[df['cidade'] == 'São Paulo']</code>	Indexação booleana, condições complexas.
Formatação Condisional	<code>df.style.background_gradient()</code>	Dinâmico, exportável como HTML.

Estudo de Caso Prático: Unificando Vendas de 12 Meses com PySpark

O Desafio

Você recebe 12 arquivos CSV, um para cada mês de vendas de 2019. Como consolidá-los em um único dataset para análise anual?

Passo 1: Listar e Filtrar os Arquivos Corretos

Usamos `dbutils.fs.ls` para listar todos os arquivos em um diretório. Criamos uma lista que seleciona apenas os arquivos que terminam com `2019.csv`, ignorando arquivos de texto, duplicatas ou de outros anos.

```
# Listar todos os arquivos e pegar apenas o caminho (path)
arquivos = dbutils.fs.ls('/FileStore/tables/')

# Filtrar para manter apenas os arquivos CSV de 2019
arquivos_2019 = [
    arquivo.path for arquivo in arquivos
    if arquivo.path.endswith('2019.csv')
]
```

Passo 2: Carregar os Dados e Unificar os DataFrames

Carregamos o primeiro arquivo para criar o DataFrame inicial. Iteramos sobre os 11 arquivos restantes em um loop. A cada iteração, carregamos o CSV do mês e o unimos ao DataFrame principal usando ``.union()`.

```
# Criar o DataFrame inicial com o primeiro arquivo da lista
df = spark.read.csv(arquivos_2019[0], header=True, inferSchema=True)

# Loop para unir os arquivos restantes
for mes in arquivos_2019[1:]:
    df_temp = spark.read.csv(mes, header=True, inferSchema=True)
    df = df.union(df_temp)
```

"Em poucos segundos, consolidamos **um ano** de vendas em **uma única fonte da verdade, uma tarefa que seria manual e demorada no Excel.**"

Enriquecendo os Dados: Criando Novas Colunas a Partir de Datas

Objetivo: A coluna `Order Date` contém data e hora. Para analisar a sazonalidade, precisamos extrair Mês, Dia e Hora em colunas separadas.

A Solução com `withColumn` e `substring`

O comando `withColumn('nome_nova_coluna', <condição>)` cria ou modifica colunas. A função `substring` extrai partes de um texto com base na posição inicial e no comprimento.

- **Mês:** Extrai 2 caracteres a partir da posição 1.
- **Dia:** Extrai 2 caracteres a partir da posição 4.
- **Hora:** Extrai 2 caracteres a partir da posição -5 (contando do final).

Antes

Order Date
04/12/19 08:35
05/01/20 14:20
06/02/20 09:00

Depois



Order Date	mês	dia	hora
04/12/19 08:35	04	12	08
05/01/20 14:20	05	01	14
06/02/20 09:00	06	02	09

"Importante: DataFrames no Spark são imutáveis. Para 'modificar' um DataFrame, você cria um novo (ex: `df = df.withColumn(...)`), garantindo a reproduzibilidade da análise."

Python e Power BI: Aprimorando sua Ferramenta de BI Favorita

Você não precisa abandonar o Power BI. Use Python para estender suas capacidades, realizando transformações de dados complexas diretamente no Power Query Editor.

Como Funciona

1. Dentro do Power Query, selecione a transformação "Executar script Python".
2. O Power BI passa o estado atual dos dados para o seu script como um DataFrame do Pandas chamado `dataset`.
3. Você aplica qualquer transformação do Pandas (limpeza, extração, etc.).
4. O script retorna um ou mais DataFrames para o Power BI continuar o pipeline de transformações.

Exemplo: Extração de E-mail

Um script Python que usa expressões regulares (`.str.extract()`) para separar uma coluna "cliente" (contendo 'Nome <email@example.com>') em duas novas colunas: `full_name` e `email`.

```
# 'dataset' é o DataFrame de entrada do Power BI

dataset = dataset.assign(
    full_name=dataset["customer"].str.extract(r"^(.*<"),
    email=dataset["customer"].str.extract(r"<(.*?)>")
)
```

Visualizações Sem Limites: Criando Gráficos Personalizados com Python

O Desafio

Os gráficos padrão do Power BI são excelentes, mas às vezes você precisa de uma visualização específica que não está disponível.

A Solução

1. Adicione um 'Visual do Python' ao seu relatório do Power BI.
2. Arraste os campos de dados necessários para a área 'Valores'. O Power BI os disponibilizará como um DataFrame `dataset`.
3. Use bibliotecas como Matplotlib ou Seaborn para criar qualquer tipo de gráfico.
4. Finalize com `plt.show()` para renderizar a imagem no seu dashboard.

```
# 'dataset' is the DataFrame
import matplotlib.pyplot as plt
import seaborn as sns

color_palette = dataset['Color'].unique()
plt.figure(figsize=(10, 6))
sns.barplot(x='Count', y='Color', data=dataset, palette=color_palette)
plt.title('Vendas de Carros por Cor')
plt.xlabel('Quantidade')
plt.ylabel('Cor')
plt.show()
```



Uma Análise Realista: Entendendo as Limitações da Integração

A combinação de Python e Power BI é poderosa, mas não é uma solução mágica. É crucial entender os trade-offs para usá-la de forma eficaz.



Performance (Overhead de Marshaling)

O Power BI se comunica com o Python trocando arquivos CSV temporários. Para datasets muito grandes, essa leitura/escrita pode gerar um gargalo de performance significativo em comparação com as transformações nativas do Power BI.



Visualizações Estáticas

Os gráficos gerados por Python são imagens (PNGs). Eles se atualizam com filtros, mas não são interativos (sem cross-filtering ou tooltips dinâmicos).



Complexidade de Ambiente

Requer que o ambiente Python (com as bibliotecas corretas) esteja configurado corretamente na máquina do usuário, o que pode dificultar o compartilhamento de relatórios.

Recomendação: Use Python no Power BI para transformações complexas que são difíceis ou impossíveis com DAX/Power Query, e para visualizações customizadas que agregam valor analítico único.

Escalando para a Nuvem: Análise de Big Data com Databricks e PySpark

Quando os dados excedem a capacidade de uma única máquina, a análise precisa se mover para a nuvem. Plataformas como o Databricks fornecem um ambiente colaborativo para executar análises distribuídas em larga escala.

Componentes-Chave do Databricks



Clusters: Conjuntos de máquinas virtuais que trabalham juntas para processar dados em paralelo. Você define a potência (memória, CPUs) de acordo com sua necessidade.



Notebooks: Interface interativa, similar ao Jupyter, onde você escreve e executa código Python (PySpark), SQL e documenta sua análise em um único lugar.



Integração com Data Lake: Conecta-se diretamente a fontes de dados massivas (como S3, ADLS) para análise in-loco.

The screenshot shows a Databricks notebook interface with two cells. Cell 1 contains Python code for reading CSV data from S3, grouping by 'Regiao', and summing 'Valor'. Cell 2 displays the resulting DataFrame and a bar chart titled 'Vendas por Região' comparing sales by region.

Cell 1:

```
1 # Carregar e transformar dados em massa
2 df = spark.read.format('csv').option('header', 'true').load('s3://neu-datalake/vendas/')
3 vendas_agrupadas = df.groupBy('Regiao').sum('Valor')
```

Cell 2:

```
1 # Visualizar resultados
2 display(vendas_agrupadas)
```

Regiao	sum(Valor)
Norte	15000
Sul	22000
Região	15000

Vendas por Região

Region	Sales (sum(Valor))
Norte	~15000
Sul	~22000
Região	~15000
Sul	~22000

O nosso estudo de caso de unificação de vendas foi executado neste ambiente, mostrando como a **mesma lógica Python pode escalar** de um arquivo local para terabytes de dados na nuvem.

Aplicando o Poder Analítico: A Transformação Digital do Setor Bancário no Brasil

As ferramentas são apenas o meio. O verdadeiro objetivo é extrair insights que expliquem transformações de mercado. Vamos analisar dados da Pesquisa Anual da FGVCia para entender a revolução digital nos bancos brasileiros.

Contexto da Análise

Fonte: Pesquisa Anual do Uso de TI - 2024, FGVCia.

Foco: Evolução dos canais de transação, crescimento da bancarização e mudança no comportamento do consumidor.

Credibilidade da Fonte



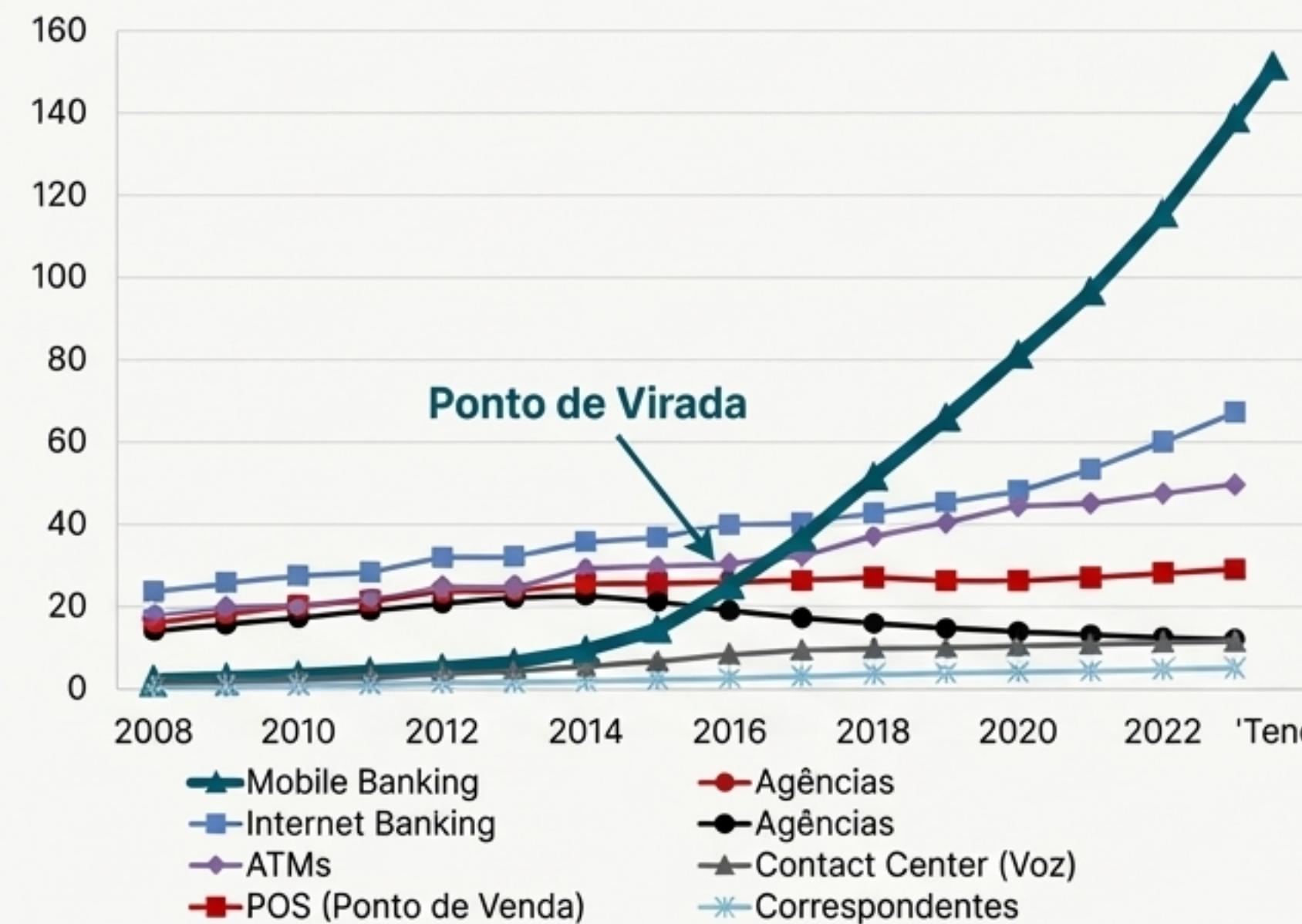
TRIPLE CROWN



A Explosão do Mobile Banking: O Canal Dominante

Volume da Transações por Origem (bilhões)

Fonte: Bacen e Febraban



Estatísticas de Destaque (2023)

60%

de todas as transações bancárias são realizadas via Mobile Banking (incluindo Pix).

90%

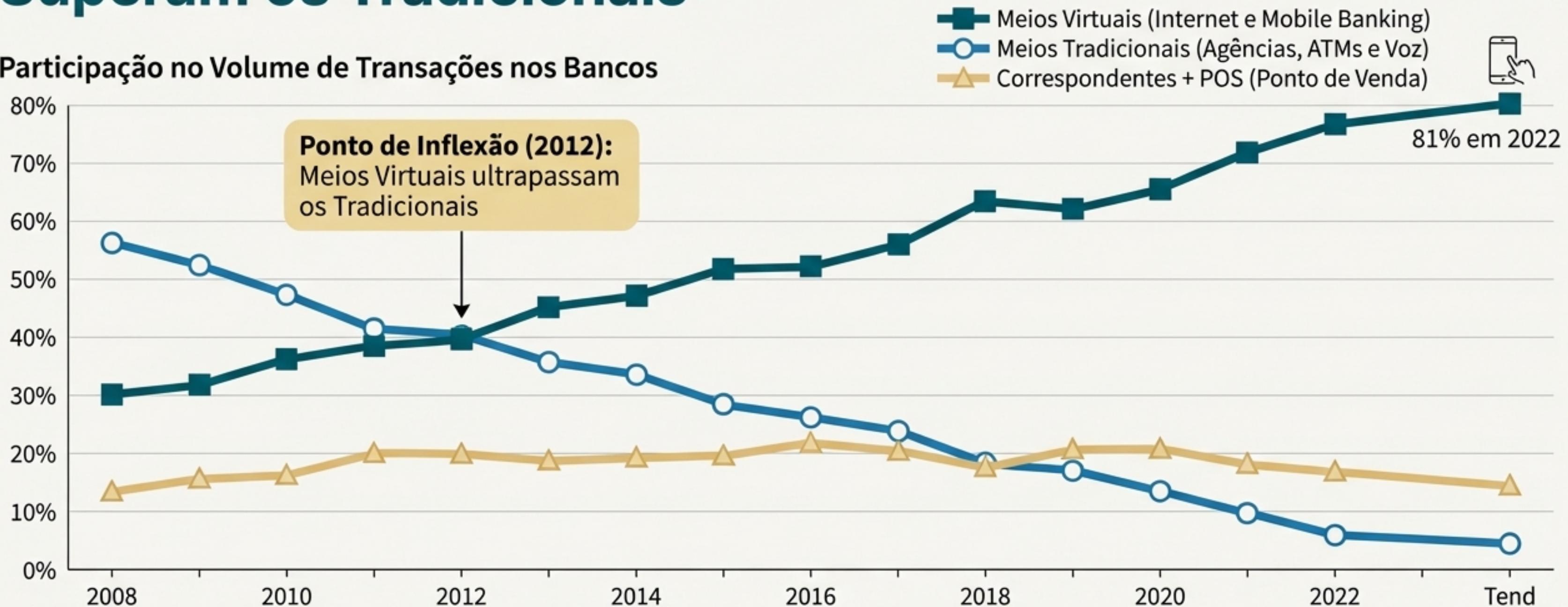
é a projeção da participação de Mobile e Internet Banking somados nos próximos 2 a 3 anos.

Insight

A migração para o mobile não é uma tendência, é um fato consolidado que redefiniu o ponto de contato primário entre bancos e clientes.
Source Sans Pro Regular (#222222).

A Virada de Chave: Canais Virtuais Superam os Tradicionais

Participação no Volume de Transações nos Bancos



O Que Isso Significa

 A estratégia dos bancos mudou de uma presença física para uma experiência digital-first. O foco agora é em conveniência, usabilidade e disponibilidade 24/7 através de aplicativos.

O Novo Papel do Analista: De Gerador de Relatórios a Parceiro Estratégico

Síntese da Jornada

Começamos no **Excel**, ideal para tarefas rápidas. Adotamos o **Python** e **PySpark** para escalar e automatizar. Integramos essas habilidades com ferramentas de **BI** e, finalmente, aplicamos esse arsenal para dissecar uma transformação de mercado real.

A Conclusão Estratégica

A capacidade de analisar a ascensão do Mobile Banking não vem de uma única Tabela Dinâmica. Ela exige a habilidade de unificar dados, enriquecê-los e visualizá-los de forma impactante.



INSIGHT ESTRATÉGICO

O valor do **analista moderno** não está em dominar uma ferramenta, mas em saber navegar por um ecossistema de soluções para transformar dados brutos em insights que direcionam a estratégia de negócios. A sua jornada de aprendizado é a jornada de relevância da sua carreira.