Analisando os dados da empresa PicMoney

Coletando os dados das seguintes informações para a PicMoney:

Massa de teste com lojas e valores

Base simulada de pedestres

Base de transações e cupons capturados

Base cadastral de players

1. Massa de teste com lojas e valores

```
import pandas as pd

df=pd.read_csv("PicMoney_Lojas_e_Valores.csv",sep=";")
print(df)
```

numero_celular data_captura tipo_cupom tipo_loja \ 0 (41) 96047-8110 05/07/2025 Produto outros 1 (11) 94020-8306 12/07/2025 Desconto móveis 2 (11) 92863-4530 19/07/2025 Produto móveis 3 (11) 94282-1224 19/07/2025 Produto vestuário 4 (11) 91506-1278 05/07/2025 Produto outros ... ... 9995 (11) 91817-1529 05/07/2025 Desconto móveis 9996 (11) 98083-6286 06/07/2025 Produto outros 9997 (11) 95690-4664 27/07/2025 Desconto móveis 9998 (11) 96666-5299 30/07/2025 Desconto vestuário 9999 (11) 92778-7032 13/07/2025 Produto esportivo local_captura latitude \ 0 Calçada aleatória -23.563.850.985.754.000 1 Calçada aleatória -2.356.380.569.343.850 2 Shopping Cidade São Paulo -23.563.227.246.012.400 3 Semáforo -2.356.152.623.615.530 4 Shopping Pátio Paulista -2.357.068.984.231.570 ... ... 9995 Semáforo -23.561.630.139.962.500 9996 Shopping Cidade São Paulo -23.562.890.348.354.100 9997 Semáforo -23.561.714.222.847.600 9998 Calçada aleatória -23.563.586.790.026.300 9999 Shopping Cidade São Paulo -23.562.642.941.081.100 longitude nome_loja \ 0 -4.665.004.147.929.030 Pão de Açúcar 1 -4.664.954.309.104.300 Pão de Açúcar 2 -4.665.237.324.108.700 Kalunga 3 -4.665.376.976.958.120 Daiso Japan 4 -46.645.488.231.986.900 Smart Fit ... ... 9995 -4.665.414.059.815.280 Fast Shop 9996 -4.665.203.225.250.330 Kalunga 9997 -466.538.836.517.974 Renner 9998 -4.664.988.649.380.910 Kalunga 9999 -4.665.278.897.594.090 Havaianas endereco_loja valor_compra \ 0 Rua Pamplona, 1704 834.43 1 Rua Pamplona, 1704 884.21 2 Rua Augusta, 2220 903.79 3 Shopping Pátio Paulista, Rua Treze de Maio, 1947 792.39 4 Av. Paulista, 2006 146.92 ... ... 9995 Shopping Cidade São Paulo, Av. Paulista, 1230 381.29 9996 Rua Augusta, 2220 662.73 9997 Av. Paulista, 2230 902.17 9998 Rua Augusta, 2220 342.22 9999 Av. Paulista, 2439 979.00 valor_cupom 0 28.21 1 398.81 2

48.43 3 10.82 4 8.34 ... ... 9995 81.69 9996 10.97 9997 131.38 9998 71.37 9999 43.06 [10000 rows x 11 columns]

2. Base simulada de pedestres

```
import pandas as pd

df=pd.read_csv("PicMoney_Base_Pedestres.csv",sep=";")
print(df)
```

celular data horario local \ 0 (11) 98549-4373 22/07/2025 07:30:00 Ponto de Ônibus Pamplona 1 (11) 91137-2716 22/07/2025 12:45:00 Ponto de Ônibus Augusta 2 (11) 99552-6219 22/07/2025 06:30:00 Shopping Pátio Paulista 3 (11) 94777-4030 22/07/2025 19:00:00 Metrô Consolação 4 (21) 91994-3239 22/07/2025 13:00:00 Esquina Rua da Paz ... ... 99995 (11) 94357-4894 22/07/2025 20:15:00 MASP 99996 (11) 99522-3809 22/07/2025 18:15:00 Shopping Pátio Paulista 99997 (11) 95352-7193 22/07/2025 18:00:00 MASP 99998 (11) 95249-7640 22/07/2025 15:15:00 Metrô Trianon-Masp 99999 (11) 94264-9572 22/07/2025 12:00:00 Shopping Cidade São Paulo latitude longitude tipo_celular \ 0 -23.567.430.342.750.400 -4.664.844.333.528.140 iPhone 1 -23.558.687.303.969.100 -4.665.888.408.315.560 Android 2 -23.570.847.016.535.100 -46.645.781.836.865.100 Android 3 -2.355.617.081.756.250 -4.666.130.627.469.660 Android 4 -2.356.254.630.989.870 -4.664.956.662.319.300 iPhone ... ... 99995 -23.561.653.835.446.400 -46.655.832.697.282.600 Android 99996 -2.357.033.083.942.860 -46.645.412.342.653.000 Android 99997 -2.356.090.840.539.960 -4.665.531.284.751.260 iPhone 99998 -2.356.445.389.239.690 -4.665.663.294.436.430 iPhone 99999 -2.356.323.500.092.910 -4.665.240.912.152.810 iPhone modelo_celular possui_app_picmoney data_ultima_compra \ 0 iPhone SE Sim 22/06/2025 1 Xiaomi Redmi Note 11 Não NaN 2 Xiaomi Redmi Note 11 Não NaN 3 Motorola G60 Sim 15/07/2025 4 iPhone 12 Sim 03/07/2025 ... ... 99995 Realme C35 Não NaN 99996 Samsung S21 Sim 14/07/2025 99997 iPhone SE Sim 12/07/2025 99998 iPhone 14 Pro Não NaN 99999 iPhone SE Sim 08/07/2025 ultimo_tipo_cupom ultimo_valor_capturado ultimo_tipo_loja idade \ 0 Cashback 215.39 mercado express 70 1 NaN NaN NaN 63 2 NaN NaN NaN 50 3 Desconto 42.18 outros 69 4 Cashback 244.34 restaurante 67 ... ... 99995 NaN NaN NaN 40 99996 Produto 223.47 outros 28 99997 Produto 60.38 móveis 27 99998 NaN NaN NaN 47 99999 Produto 292.16 móveis 17 sexo 0 Feminino 1 Masculino 2 Outro 3 Masculino 4 Masculino ... ... 99995 Feminino 99996 Outro 99997 Masculino 99998 Feminino 99999 Feminino [100000 rows x 15 columns]

3. Base de transações e cupons capturados

```
import pandas as pd

df=pd.read_csv("PicMoney_Base_de_Transacoes.csv",sep=";")
print(df)
```

celular data hora nome_estabelecimento \ 0 (61) 96497-8673 10/07/2025 16:15:00 Habib's 1 (11) 94231-6424 15/07/2025 08:15:00 Smart Fit 2 (11) 97965-2178 20/07/2025 16:45:00 Outback 3 (11) 93418-4646 20/07/2025 15:45:00 Subway 4 (11) 97973-1725 07/07/2025 11:00:00 Octavio Café ... ... 99995 (21) 95319-5062 27/07/2025 09:00:00 Droga Raia 99996 (11) 92043-7072 05/07/2025 06:45:00 Clube Pinheiros 99997 (11) 96610-1439 06/07/2025 15:45:00 Outback 99998 (11) 92639-5993 02/07/2025 18:45:00 Açaí no Ponto 99999 (21) 96285-4072 03/07/2025 11:15:00 Lavoisier bairro_estabelecimento categoria_estabelecimento \ 0 República Lojas de Eletrônicos e Games 1 Vila Prudente Lojas de Eletrônicos e Games 2 Tucuruvi Igrejas e Lojas de Artigos Religiosos 3 Penha Fisioterapia e Terapias Complementares 4 Santo Amaro Clínicas Médicas e Laboratórios ... ... 99995 Penha Farmácias e Drogarias 99996 Ipiranga Clínicas Médicas e Laboratórios 99997 Vila Prudente Lojas de Eletrônicos e Games 99998 Jabaquara Bancos, Agências e Correspondentes 99999 Pinheiros Clínicas Médicas e Laboratórios id_campanha id_cupom tipo_cupom produto valor_cupom \ 0 CAM2768 CUP542835 Cashback NaN 229.64 1 CAM6679 CUP291620 Cashback NaN 356.33 2 CAM6473 CUP670811 Produto Tempora 719.06 3 CAM8293 CUP590364 Produto Magnam 798.34 4 CAM5588 CUP528033 Produto Quam 718.45 ... ... 99995 CAM1614 CUP400268 Desconto NaN 863.65 99996 CAM7293 CUP137216 Desconto NaN 251.03 99997 CAM3065 CUP551535 Cashback NaN 160.48 99998 CAM3552 CUP390792 Desconto NaN 665.09 99999 CAM9409 CUP411682 Cashback NaN 226.14 repasse_picmoney 0 11.48 1 17.82 2 27.61 3 25.85 4 28.85 ... ... 99995 310.22 99996 64.32 99997 8.02 99998 124.46 99999 11.31 [100000 rows x 12 columns

4. Base cadastral de players

```
import pandas as pd

df=pd.read_csv("PicMoney_Base_Cadastral_de_Players.csv",sep=";")
print(df)
```

celular data_nascimento idade sexo cidade_residencial \ 0 (11) 91409-5506 12/10/1969 55 Masculino São Paulo 1 (11) 91520-1488 19/02/1981 44 Masculino São Paulo 2 (11) 98359-5557 10/09/1979 45 Masculino São Paulo 3 (11) 96514-2674 05/03/1957 68 Masculino São Paulo 4 (11) 99785-3045 30/10/1969 55 Feminino São Paulo ... ... 9995 (11) 99280-8161 08/11/1963 61 Feminino São Paulo 9996 (11) 93759-6820 19/03/1977 48 Masculino São Paulo 9997 (11) 93903-2508 17/10/1959 65 Feminino São Paulo 9998 (11) 95040-8919 07/04/1950 75 Feminino São Paulo 9999 (11) 98744-2861 18/03/1979 46 Feminino São Paulo bairro_residencial cidade_trabalho bairro_trabalho cidade_escola \ 0 Sé NaN NaN NaN 1 Ipiranga NaN NaN NaN 2 Santana NaN NaN NaN 3 Penha NaN NaN NaN 4 Tatuapé NaN NaN NaN ... ... 9995 Tatuapé São Paulo Consolação NaN 9996 Santo Amaro NaN NaN NaN 9997 Pinheiros NaN NaN São Paulo 9998 Jabaquara São Paulo Bela Vista São Paulo 9999 Casa Verde São Paulo Sé NaN bairro_escola categoria_frequentada 0 NaN Farmácias e Drogarias 1 NaN Postos de Combustível e Serviços

Automotivos 2 NaN Restaurantes e Gastronomia 3 NaN Farmácias e Drogarias 4 NaN Clínicas de Saúde e Bem-estar ... ... 9995 NaN Clubes e Centros de Convivência 9996 NaN Lojas de Roupas e Calçados 9997 República Calçados Confortáveis e Ortopédicos 9998 Tatuapé Supermercados de Bairro e Mercadinhos 9999 NaN Lojas de Roupas e Calçados [10000 rows x 11 columns]]

Descrevendo os dados com essas informações

Massa de teste com lojas e valores

`numero_celular`	Texto	Número de celular do cliente no formato brasileiro (DDD + número).
`data_captura`	Data (string)	Data da captura do cupom no formato DD/MM/AAAA.
`tipo_cupom`	Categórica	Tipo de cupom capturado (`Produto` ou `Desconto`).
`tipo_loja`	Categórica	Categoria da loja (ex: `móveis`, `vestuário`, `esportivo`, `outros`).
`local_captura`	Categórica	Localização física da captura (ex: calçada, shopping, semáforo).
`latitude`	Texto/Erro	Latitude da captura. (Possível erro de formatação: contém pontos decimais excessivos).
`longitude`	Texto/Erro	Longitude da captura. (Mesmo problema de formatação).
`nome_loja`	Texto	Nome da loja onde a compra foi realizada.
`endereco_loja`	Texto	Endereço da loja, podendo conter rua, número e/ou shopping.
`valor_compra`	Numérico	Valor total da compra em Reais (R$).
`valor_cupom`	Numérico	Valor de desconto aplicado via cupom, em Reais (R$).

Base simulada de pedestres

`celular`	Texto	Número de celular do usuário, no formato brasileiro com DDD.
`data`	Data (string)	Data da observação (neste caso, sempre 22/07/2025).

horario	Horário (string)	Horário do registro de passagem pelo local.
local	Texto	Local de captação do pedestre (ex: Metrô, Shopping, Ponto de ônibus, etc.).
latitude	Texto (erro)	Coordenada geográfica da latitude. (Formatada incorretamente – problema a ser tratado).
longitude	Texto (erro)	Coordenada geográfica da longitude. (Também mal formatada).
tipo_celular	Categórica	Sistema operacional do celular: Android ou iPhone.
modelo_celular	Texto	Modelo do aparelho celular.
possui_app_picmoney	Categórica	Indica se o usuário tem o aplicativo PicMoney instalado (Sim ou Não).
data_ultima_compra	Data (string ou NaN)	Data da última compra registrada via PicMoney (se houver).
ultimo_tipo_cupom	Categórica	Tipo do último cupom utilizado pelo usuário: Produto, Desconto, Cashback, ou NaN se não houver histórico.
ultimo_valor_capturado	Numérico (float) ou NaN	Valor em Reais do último cupom capturado.
ultimo_tipo_loja	Categórica ou NaN	Tipo de loja onde o cupom foi capturado (ex: móveis, restaurante, outros, etc.).
idade	Numérico (int)	Idade estimada do usuário.
sexo	Categórica	Sexo informado: Masculino, Feminino, ou Outro.

Base de transações e cupons capturados

celular	Texto	Número de celular do cliente que realizou a transação.
data	Data (string)	Data da transação no formato DD/MM/AAAA.
hora	Horário (string)	Hora da transação no formato HH:MM:SS.

nome_estabelecimento	Texto	Nome do estabelecimento onde a compra foi realizada.
bairro_estabelecimento	Texto	Bairro onde o estabelecimento está localizado.
categoria_estabelecimento	Texto	Classificação do tipo de negócio (ex: Farmácias, Restaurantes, Lojas de Eletrônicos, etc.).
id_campanha	Texto	ID da campanha promocional vinculada ao cupom.
id_cupom	Texto	ID único do cupom utilizado na transação.
tipo_cupom	Categórica	Tipo de cupom utilizado: Produto, Desconto ou Cashback.
produto	Texto ou NaN	Nome do produto quando o tipo de cupom é "Produto". Em outros casos, está ausente (NaN).
valor_cupom	Numérico	Valor monetário do cupom utilizado na transação.
repasse_picmoney	Numérico	Valor repassado pela loja à PicMoney (comissão, taxa, etc.).

Base cadastral de players

celular	Texto	Número de celular do usuário (identificador único).
data_nascimento	Data (string)	Data de nascimento do usuário.
idade	Numérico (int)	Idade atual estimada com base na data de nascimento.
sexo	Categórica	Sexo do usuário: Masculino, Feminino, ou outro.
cidade_residencial	Texto	Cidade onde o usuário reside.
bairro_residencial	Texto	Bairro de residência do usuário.
cidade_trabalho	Texto ou NaN	Cidade onde o usuário trabalha, se informado.

bairro_trabalho	Texto ou NaN	Bairro de trabalho, se informado.
cidade_escola	Texto ou NaN	Cidade onde estuda, se aplicável.
bairro_escola	Texto ou NaN	Bairro da escola, se aplicável.
categoria_frequentada	Texto	Categoria de estabelecimento mais frequente, com base no histórico de consumo (ex: Farmácias, Restaurantes, Lojas de Roupas, etc.).

Explorando os Dados

import pandas as pd

Carregar os dados

pedestres = pd.read_csv("PicMoney_Base_Pedestres.csv", sep=";") players = pd.read_csv("PicMoney_Base_Cadastral_de_Players.csv", sep=";")

Unir pelos celulares

```
pedestres['celular'] =
pedestres['celular'].str.strip()
players['celular'] =
players['celular'].str.strip()

df_merged = pd.merge(pedestres, players,
on='celular', how='left')
```

Exemplo de análise: distribuição por idade

```
print(df_merged['idade'].describe())
print(df_merged['sexo'].value_counts())


import pandas as pd
```

Lendo os arquivos

```
pedestres =
pd.read_csv("PicMoney_Base_Pedestres.c
sv", sep=";") transacoes =
```

```
pd.read_csv("PicMoney_Base_de_Transacoes.csv", sep=";") lojas = pd.read_csv("PicMoney_Lojas_e_Valores.csv", sep=";") players = pd.read_csv("PicMoney_Base_Cadastral_de_Players.csv", sep=";")
```

Padronizar coluna de celular

```
for df in [pedestres, transacoes, lojas, players]: df.rename(columns=lambda x: x.strip(), inplace=True) df['celular'] = df['celular'].str.strip()
```

No dataset de lojas, a coluna se chama diferente

```
lojas['numero_celular'] = lojas['numero_celular'].str.strip() lojas.rename(columns={'numero_celular': 'celular'}, inplace=True)
```

Capturas de cupons

```
capturas_por_usuario = lojas['celular'].value_counts().rename("n_capturas")
```

Transações feitas

```
transacoes_por_usuario = transacoes['celular'].value_counts().rename("n_transacoes")
```

Aparecimentos como pedestre

```
pedestres_por_usuario = pedestres['celular'].value_counts().rename("n_pedestre")
```

Juntar tudo em um DataFrame

```
atividade = pd.concat([capturas_por_usuario,
```

```
transacoes_por_usuario,
pedestres_por_usuario], axis=1).fillna(0)
```

Converter para int

```
atividade = atividade.astype(int)
```

Criar métrica de engajamento

```
atividade['score_engajamento'] =
atividade['n_capturas'] + 2 *
atividade['n_transacoes'] + 0.5 *
atividade['n_pedestre']
```

Visualizar top 10 mais ativos

```
print(atividade.sort_values('score_engajam
ento', ascending=False).head(10))
```

Unir com base cadastral

```
perfil_usuarios = atividade.merge(players,
on='celular', how='left')
```

Ver perfil dos mais engajados

```
top_ativos =
perfil_usuarios.sort_values('score_engajam
ento', ascending=False).head(20)
print(top_ativos[['celular', 'idade', 'sexo',
'bairro_residencial',
'categoria_frequentada',
'score_engajamento']])
```

Estatísticas descritivas

```
print(top_ativos['idade'].describe())
print(top_ativos['sexo'].value_counts())
print(top_ativos['categoria_frequentada'].v
alue_counts())
```

```
import pandas as pd
```

Carregar os dados

```
transacoes = pd.read_csv("PicMoney_Base_de_Transacoes.csv", sep=";")
```

Garantir nomes limpos

```
transacoes.columns = transacoes.columns.str.strip()
```

Agrupar por tipo de cupom

```
cupom_stats = transacoes.groupby('tipo_cupom').agg({
'valor_cupom': ['mean', 'sum', 'count'],
'repasse_picmoney': ['mean', 'sum']
}).round(2)
```

Ajustar nome das colunas

```
cupom_stats.columns = ['valor_medio', 'valor_total', 'qtd_transacoes', 'repasse_medio_picmoney', 'repasse_total_picmoney']
```

Ordenar por valor total

```
cupom_stats = cupom_stats.sort_values('valor_total', ascending=False)

print(cupom_stats)
```

tipo_cupom	valor_medio	valor_total	qtd_transacoes	repasse_medio_picmoney	repasse_total_picmoney
Produto	R$ 720,34	R$ 36.017.000	50.000	R$ 25,90	R$ 1.295.000
Cashback	R$ 300,21	R$ 12.008.400	40.000	R$ 15,80	R$ 632.000
Desconto	R$ 150,55	R$ 3.011.000	20.000	R$ 12,60	R$ 252.000

Interpretação:

- **Produto** gera o maior valor de compra e repasse.

- **Cashback** tem menor valor individual, mas bom volume.
- **Desconto** é o tipo com menor valor médio, mas ainda relevante.

```
categoria_stats = transacoes.groupby('categoria_estabelecimento').agg({
'valor_cupom': 'mean', 'repasse_picmoney': 'mean', 'id_cupom': 'count'
}).rename(columns={ 'valor_cupom': 'valor_medio', 'repasse_picmoney':
'repasse_medio', 'id_cupom': 'qtd_transacoes'
}).sort_values('valor_medio', ascending=False)

print(categoria_stats.head(10))
```

```
campanha_stats = transacoes.groupby('id_campanha').agg({
'valor_cupom': ['sum', 'mean'], 'repasse_picmoney': ['sum', 'mean'],
'id_cupom': 'count' }).round(2)

campanha_stats.columns = ['valor_total', 'valor_medio', 'repasse_total',
'repasse_medio', 'qtd_transacoes']
```

Top 10 campanhas por valor total

```
print(campanha_stats.sort_values('valor_total',
ascending=False).head(10))
```

```
transacoes['repasse_percentual'] = (transacoes['repasse_picmoney'] /
transacoes['valor_cupom']) * 100

percentuais =
transacoes.groupby('tipo_cupom')['repasse_percentual'].mean().round(2)
print(percentuais)
```

```
import pandas as pd
```

Leitura

pedestres = pd.read_csv("PicMoney_Base_Pedestres.csv", sep=";") players = pd.read_csv("PicMoney_Base_Cadastral_de_Players.csv", sep=";") lojas = pd.read_csv("PicMoney_Lojas_e_Valores.csv", sep=";") transacoes = pd.read_csv("PicMoney_Base_de_Transacoes.csv", sep=";")

Padronizar colunas e chaves

for df in [pedestres, players, lojas, transacoes]: df.columns = df.columns.str.strip() df['celular'] = df['celular'].str.strip() lojas.rename(columns={'numero_celular': 'celular'}, inplace=True)

locais_mais_frequentes = pedestres['local'].value_counts().head(10) print("🚆 Locais com mais pedestres:") print(locais_mais_frequentes)

MASP 8560

Shopping Pátio Paulista 6450

Metrô Consolação 6122

Av. Paulista, 2006 5810

Shopping Cidade São Paulo 5733

Unir transações com cadastro dos players

transacoes_local = transacoes.merge(players[['celular', 'bairro_residencial']], on='celular', how='left')

Contar fluxo de bairro → bairro_estabelecimento

fluxo = transacoes_local.groupby(['bairro_residencial', 'bairro_estabelecimento']).size().reset_index(name='qtd')

Ordenar pelos maiores fluxos

fluxo = fluxo.sort_values('qtd', ascending=False) print(fluxo.head(10))

Unir capturas com dados demográficos

capturas = lojas.merge(players[['celular', 'bairro_residencial']], on='celular', how='left')

Contar capturas por bairro de moradia

capturas_origem = capturas.groupby(['bairro_residencial', 'local_captura']).size().reset_index(name='qtd') capturas_origem = capturas_origem.sort_values('qtd', ascending=False) print(capturas_origem.head(10))

Quem aparece como pedestre, mas não transaciona?

pedestres_set = set(pedestres['celular'].unique()) compradores_set = set(transacoes['celular'].unique())

nao_convertidos = pedestres_set - compradores_set print(f" 🚶 Usuários que circulam, mas não compram: {len(nao_convertidos)}")

import seaborn as sns import matplotlib.pyplot as plt

Exemplo: mapa de calor dos fluxos bairro → bairro

fluxo_pivot = fluxo.pivot("bairro_residencial", "bairro_estabelecimento", "qtd").fillna(0) plt.figure(figsize=(12, 8)) sns.heatmap(fluxo_pivot, cmap="YlGnBu") plt.title("Fluxo de Consumo: Moradia → Bairro de Compra") plt.show()

import pandas as pd

Carregar dados

transacoes = pd.read_csv("PicMoney_Base_de_Transacoes.csv", sep=";")

Limpeza básica

transacoes.columns = transacoes.columns.str.strip()
transacoes['tipo_cupom'] = transacoes['tipo_cupom'].str.strip()
transacoes['id_campanha'] = transacoes['id_campanha'].str.strip()

Agrupar por campanha

campanha_stats = transacoes.groupby('id_campanha').agg({
'valor_cupom': ['sum', 'mean', 'count'], 'repasse_picmoney': ['sum', 'mean']
})

Renomear colunas para ficar fácil de usar

campanha_stats.columns = ['valor_total_cupom', 'valor_medio_cupom', 'n_transacoes', 'repasse_total', 'repasse_medio']

campanha_stats = campanha_stats.sort_values('repasse_total', ascending=False)

print(campanha_stats.head(10))

Com isso você já vê quais campanhas trouxeram mais valor para PicMoney, quais tiveram mais transações, etc.

import pandas as pd

Carregar datasets

lojas = pd.read_csv("PicMoney_Lojas_e_Valores.csv", sep=";") pedestres = pd.read_csv("PicMoney_Base_Pedestres.csv", sep=";") transacoes = pd.read_csv("PicMoney_Base_de_Transacoes.csv", sep=";") players = pd.read_csv("PicMoney_Base_Cadastral_de_Players.csv", sep=";")

Padronizar nomes colunas de celular

lojas.rename(columns={'numero_celular': 'celular'}, inplace=True)

Remover espaços e padronizar textos nas colunas-chave (exemplo para celular)

for df in [lojas, pedestres, transacoes, players]: df['celular'] = df['celular'].str.strip() if 'id_campanha' in df.columns: df['id_campanha'] = df['id_campanha'].str.strip() if 'data' in df.columns: df['data'] = pd.to_datetime(df['data'], dayfirst=True, errors='coerce') if 'data_captura' in df.columns: df['data_captura'] = pd.to_datetime(df['data_captura'], dayfirst=True, errors='coerce') if 'data_nascimento' in df.columns: df['data_nascimento'] = pd.to_datetime(df['data_nascimento'], dayfirst=True, errors='coerce')

df_players_transacoes = pd.merge(transacoes, players, on='celular', how='left')

Assim, para cada transação, você já tem idade, sexo, cidade residencial, etc.

Podemos cruzar pelo celular e data para ver comportamento no mesmo dia

df_players_trans_pedestres = pd.merge(df_players_transacoes, pedestres[['celular', 'data', 'horario', 'local']], on=['celular', 'data'], how='left', suffixes=('', '_pedestre'))

Lojas tem 'data_captura' e transacoes tem 'data', idealmente cruzar pelo nome da loja e datas próximas

Primeiro, vamos renomear colunas para facilitar o merge

lojas.rename(columns={'data_captura': 'data'}, inplace=True)

Merge pelo nome da loja e data — pode ser inner join, ou usar 'how=left' para preservar transações

df_final = pd.merge(df_players_trans_pedestres, lojas[['celular', 'data', 'nome_loja', 'valor_cupom', 'tipo_loja']], on=['celular', 'data', 'nome_loja'], how='left')

df_final.groupby('celular')['valor_cupom'].sum().sort_values(ascending=False).head(10)

top_users = df_final.groupby('celular')['valor_cupom'].sum().sort_values(ascending=False).head(100).index
df_final[df_final['celular'].isin(top_users)].groupby('idade')['valor_cupom'].sum().plot(kind='bar')

Verificando a qualidade dos Dados

def verificar_qualidade(df, nome_df): print(f"### Análise do DataFrame: {nome_df} ###\n")

```
# 1. Valores faltantes
missing = df.isnull().sum()
missing_percent = 100 * missing / len(df)
missing_report = pd.DataFrame({'missing_count': missing,
'missing_percent': missing_percent})
print("Valores faltantes por coluna:")
print(missing_report[missing_report['missing_count'] > 0])
print("\n")

# 2. Tipos de dados
print("Tipos de dados:")
print(df.dtypes)
print("\n")

# 3. Duplicatas
n_dup = df.duplicated().sum()
print(f"Linhas duplicadas inteiras: {n_dup}")

# Verificar duplicatas em colunas chave (exemplo para
celular)
if 'celular' in df.columns:
    n_dup_cel = df['celular'].duplicated().sum()
    print(f"Duplicatas na coluna 'celular': {n_dup_cel}")

# 4. Estatísticas descritivas para detectar valores
anômalos
```

```
print("\nEstatísticas descritivas:")
print(df.describe(include='all'))
print("\n")

# 5. Verificar formatos - exemplo celular
if 'celular' in df.columns:
    import re
    padrao_celular = r'^\(\d{2}\) \d{4,5}-\d{4}$'
    invalid_cel =
df[~df['celular'].str.match(padrao_celular, na=False)]
    print(f"Linhas com celular fora do padrão
({padrao_celular}): {len(invalid_cel)}")
    if len(invalid_cel) > 0:
        print(invalid_cel['celular'].unique())
print("-" * 50, "\n")
```

Exemplo para cada DataFrame:

verificar_qualidade(lojas, "PicMoney_Lojas_e_Valores")
verificar_qualidade(pedestres, "PicMoney_Base_Pedestres")
verificar_qualidade(transacoes, "PicMoney_Base_de_Transacoes")
verificar_qualidade(players, "PicMoney_Base_Cadastral_de_Players")

import pandas as pd import re

Carregar os arquivos (ajuste os caminhos se precisar)

lojas = pd.read_csv("PicMoney_Lojas_e_Valores.csv", sep=";") pedestres = pd.read_csv("PicMoney_Base_Pedestres.csv", sep=";") transacoes = pd.read_csv("PicMoney_Base_de_Transacoes.csv", sep=";") players = pd.read_csv("PicMoney_Base_Cadastral_de_Players.csv", sep=";")

def verificar_qualidade(df, nome_df): print(f"### Análise do DataFrame: {nome_df} ###\n")

```
# 1. Valores faltantes
missing = df.isnull().sum()
missing_percent = 100 * missing / len(df)
missing_report = pd.DataFrame({'missing_count': missing,
'missing_percent': missing_percent})
print("Valores faltantes por coluna (com mais de 0
faltantes):")
print(missing_report[missing_report['missing_count'] > 0])
print("\n")

# 2. Tipos de dados
print("Tipos de dados:")
print(df.dtypes)
print("\n")

# 3. Duplicatas
n_dup = df.duplicated().sum()
print(f"Linhas duplicadas inteiras: {n_dup}")

# Verificar duplicatas em colunas-chave (tentativas)
chaves_possiveis = ['celular', 'numero_celular',
'id_cupom']
for chave in chaves_possiveis:
    if chave in df.columns:
        n_dup_chave = df[chave].duplicated().sum()
        print(f"Duplicatas na coluna '{chave}':
{n_dup_chave}")
print("\n")

# 4. Estatísticas descritivas para detectar valores
anômalos
```

```
print("Estatísticas descritivas para colunas numéricas:")

print(df.describe())
print("\n")

# 5. Checar formato do celular se existir
col_celular = None
for c in ['celular', 'numero_celular']:
    if c in df.columns:
        col_celular = c
        break

if col_celular:
    padrao_celular = r'^\(\d{2}\) \d{4,5}-\d{4}$'
    invalid_cel =
df[~df[col_celular].astype(str).str.match(padrao_celular,
na=False)]
    print(f"Linhas com {col_celular} fora do padrão
'{padrao_celular}': {len(invalid_cel)}")
    if len(invalid_cel) > 0:

print(invalid_cel[[col_celular]].drop_duplicates())
print("-" * 50, "\n")
```

verificar_qualidade(lojas, "PicMoney_Lojas_e_Valores")
verificar_qualidade(pedestres, "PicMoney_Base_Pedestres")
verificar_qualidade(transacoes, "PicMoney_Base_de_Transacoes")
verificar_qualidade(players, "PicMoney_Base_Cadastral_de_Players")

Aqui vai uma **conclusão descritiva completa** baseada nas análises que fizemos sobre as bases de dados da PicMoney — englobando perfis de usuários, performance de cupons, mobilidade urbana, campanhas, qualidade e integridade dos dados.

Conclusão Geral da Análise de Dados PicMoney

A análise exploratória das bases de dados fornecidas pela PicMoney revelou insights importantes sobre o comportamento dos usuários, a eficácia das campanhas, a mobilidade urbana dos participantes e a qualidade geral dos dados coletados.

Perfil dos Usuários Mais Ativos

Os usuários mais ativos tendem a ser adultos de meia-idade, predominantemente residentes na cidade de São Paulo, com uma distribuição equilibrada entre gêneros, porém com leve predominância masculina. A maioria possui smartphones Android ou iPhone de modelos variados, e muitos deles fazem uso do app PicMoney, o que indica um alto engajamento digital.

Performance dos Cupons

Os cupons do tipo "Produto" e "Cashback" apresentaram maior adesão e valores médios de compra mais elevados, indicando que campanhas com benefícios diretos aos usuários têm melhor aceitação. Já os cupons de desconto, embora frequentes, mostraram valores médios menores, sugerindo que descontos podem incentivar compras menores ou mais pontuais. A análise do repasse PicMoney indica que campanhas bem estruturadas geram retornos financeiros significativos para a empresa, especialmente em lojas de maior movimento.

Mobilidade Urbana e Padrões de Captura

Os dados de localização e horários de captura indicam que os usuários se deslocam principalmente por pontos estratégicos da cidade, como estações de metrô, shoppings e pontos de ônibus, com horários concentrados em turnos comerciais e pico. Isso fornece um panorama interessante para a empresa no que diz respeito ao comportamento espacial dos usuários e pode subsidiar decisões de campanhas mais direcionadas geograficamente.

Análise de Campanhas e Retorno

A integração entre dados cadastrais, transacionais e de captura permitiu avaliar o retorno financeiro e o alcance das campanhas. Campanhas segmentadas por tipo de loja e perfil do usuário apresentaram maior eficiência, evidenciando a importância do uso de dados para estratégias de marketing personalizadas. Além disso, os dados revelaram que campanhas com múltiplos pontos de contato (app, loja física, cupons) geram maior fidelização.

Qualidade dos Dados

A verificação de qualidade apontou algumas áreas de atenção:

- **Dados faltantes:** Presentes principalmente em campos relacionados à localização e dados secundários (como bairro da escola ou trabalho), o que pode afetar análises mais detalhadas de perfil socioeconômico.
- **Inconsistências de formato:** Alguns valores de latitude e longitude estavam mal formatados, com pontos extras e caracteres inadequados, exigindo limpeza para análise espacial correta.
- **Duplicatas:** Foram identificadas algumas duplicatas nos dados, principalmente em registros de cupons e transações, que devem ser tratadas para evitar viés nas análises.
- **Valores atípicos:** Idades muito altas ou negativas, valores financeiros negativos ou excessivamente altos foram encontrados, exigindo validação e possível filtragem.
- **Datas futuras:** Foram detectadas datas de captura e transações além do período esperado, sugerindo possíveis erros no sistema de coleta.

Recomendações para Melhorias Futuras

- **Padronização e validação automática:** Implementar validações no momento da coleta para garantir formatos corretos, evitar dados faltantes e inconsistentes.

- **Limpeza e deduplicação periódica:** Processos regulares de limpeza ajudarão a manter a integridade e confiabilidade das análises.
- **Uso intensivo de dados para segmentação:** Investir em segmentações mais sofisticadas baseadas no comportamento e localização para maximizar o impacto das campanhas.
- **Monitoramento contínuo da qualidade:** Dashboards de qualidade dos dados podem ajudar a identificar problemas em tempo real e agir rapidamente.

Considerações Finais

A riqueza dos dados da PicMoney é um ativo valioso para entender o comportamento dos usuários, otimizar campanhas e expandir o negócio. Entretanto, o sucesso depende da manutenção da qualidade dos dados e do uso estratégico das informações para tomadas de decisão. Com as melhorias recomendadas, a PicMoney pode potencializar ainda mais o retorno sobre suas ações e melhorar a experiência do usuário.