1.0 PySpark - Instalando a biblioteca PySpark no Google Colab

```
! pip install pyspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.11/dist-packages (3.5.1)
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.11/dist-packages (from pyspark) (0.10.9.7)
```

```
! pip install findspark
```

```
Collecting findspark
  Downloading findspark-2.0.1-py2.py3-none-any.whl.metadata (352 bytes)
Downloading findspark-2.0.1-py2.py3-none-any.whl (4.4 kB)
Installing collected packages: findspark
Successfully installed findspark-2.0.1
```

1.1 Importanto as bibliotecas:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import findspark
findspark.init()

from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local[*]").getOrCreate()
from pyspark.sql.functions import concat, col, to_date, lpad, when, sum as _sum, round, avg
```

```
df = spark.read.csv('/content/PNAD_COVID_consolidado.csv', sep = ',', inferSchema = True, header = True)
# header para falar que tem cabeçalho
# inferSchema para forçar a rodar os dados importados

print('df.count : ', df.count()) # para mostrar o total de linhas
print('df. col ct : ', len(df.columns)) # para mostrar o total de colunas
print('df.columns : ', df.columns) # para listar todas as colunas
```

```
df.count :  1154279
df. col ct :  146
df.columns :  ['Ano', 'UF', 'CAPITAL', 'RM_RIDE', 'V1008', 'V1012', 'V1013', 'V1016', 'Estrato', 'UPA', 'V1022', 'V1023', 'V1030',
```

2.0 Usando SQL no PySpark:

```
df.describe().show() # para mostrar as estatísticas do DataFrame
```

```
df.printSchema()
```

2.1 Remover as duplicatas:

```
df = df.dropDuplicates()
print('Total de linhas:', df.count())
print('Total de colunas: ', df.columns)
```

...

2.1 Renomeando as colunas:

```
df1 = df \
    .withColumnRenamed('UF', 'unidade_federacao') \
    .withColumnRenamed('CAPITAL', 'capital_estado') \
    .withColumnRenamed('RM_RIDE', 'regiao_metropolitana') \
    .withColumnRenamed('V1008', 'numero_domicilio') \
    .withColumnRenamed('V1012', 'semana_no_mes') \
    .withColumnRenamed('V1013', 'mes_pesquisa') \
    .withColumnRenamed('V1016', 'numero_entrevista_domicilio') \
    .withColumnRenamed('Estrato', 'estrato_amostral') \
    .withColumnRenamed('UPA', 'unidade_primaria_amostragem') \
    .withColumnRenamed('V1023', 'tipo_area') \
    .withColumnRenamed('V1030', 'projecao_populacao') \
    .withColumnRenamed('V1031', 'peso_sem_pos_estratificacao') \
    .withColumnRenamed('V1032', 'peso_com_pos_estratificacao') \
    .withColumnRenamed('posest', 'dominio_projecao') \
    .withColumnRenamed('A001', 'numero_ordem_morador') \
    .withColumnRenamed('A001A', 'condicao_no_domicilio') \
    .withColumnRenamed('A001B1', 'dia_nascimento') \
    .withColumnRenamed('A001B2', 'mes_nascimento') \
    .withColumnRenamed('A001B3', 'ano_nascimento') \
    .withColumnRenamed('A002', 'idade_morador') \
    .withColumnRenamed('A003', 'sexo') \
    .withColumnRenamed('A004', 'cor_ou_raca') \
    .withColumnRenamed('A005', 'escolaridade') \
    .withColumnRenamed('B0011', 'semana_passada_febre') \
```

```
    .withColumnRenamed('B0012', 'semana_passada_tosse') \
    .withColumnRenamed('B0013', 'semana_passada_dor_garganta') \
    .withColumnRenamed('B0014', 'semana_passada_dificuldade_respirar') \
    .withColumnRenamed('B0015', 'semana_passada_dor_cabeca') \
    .withColumnRenamed('B0016', 'semana_passada_dor_peito') \
    .withColumnRenamed('B0017', 'semana_passada_nausea') \
    .withColumnRenamed('B0018', 'semana_passada_nariz_entupido_escorrendo') \
    .withColumnRenamed('B0019', 'semana_passada_fadiga') \
    .withColumnRenamed('B00110', 'semana_passada_dor_olhos') \
    .withColumnRenamed('B00111', 'semana_passada_perda_cheiro_sabor') \
    .withColumnRenamed('B00112', 'semana_passada_dor_muscular') \
    .withColumnRenamed('B002', 'procurou_estab_saude') \
    .withColumnRenamed('B0031', 'recup_sintomas_ficou_casa') \
    .withColumnRenamed('B0032', 'recup_sintomas_ligou_prof_saude') \
    .withColumnRenamed('B0033', 'recup_sintomas_tomou_remedio_conta') \
    .withColumnRenamed('B0034', 'recup_sintomas_tomou_remedio_medico') \
    .withColumnRenamed('B0035', 'recup_sintomas_visita_prof_sus') \
    .withColumnRenamed('B0036', 'recup_sintomas_visita_prof_particular') \
    .withColumnRenamed('B0041', 'atendimento_posto_ubs') \
    .withColumnRenamed('B0042', 'atendimento_pronto_socorro_sus') \
    .withColumnRenamed('B0043', 'atendimento_hospital_sus') \
    .withColumnRenamed('B0044', 'atendimento_ambulatorio_privado') \
    .withColumnRenamed('B0045', 'atendimento_pronto_socorro_privado') \
    .withColumnRenamed('B0046', 'atendimento_hospital_privado') \
    .withColumnRenamed('B005', 'internado_um_dia_ou_mais') \
    .withColumnRenamed('B006', 'sedado_entubado_ventilado') \
    .withColumnRenamed('B007', 'tem_plano_saude') \
    .withColumnRenamed('C001', 'trabalhou_ou_bico_semana') \
    .withColumnRenamed('C002', 'afastado_trabalho') \
    .withColumnRenamed('C003', 'motivo_afastamento') \
    .withColumnRenamed('C004', 'continuou_remunerado') \
    .withColumnRenamed('C005', 'tempo_afastado') \
    .withColumnRenamed('C0051', 'meses_afastado_menos_um_ano') \
    .withColumnRenamed('C0052', 'meses_afastado_um_a_dois_anos') \
    .withColumnRenamed('C0053', 'anos_afastado_dois_ou_mais') \
    .withColumnRenamed('C006', 'mais_de_um_trabalho') \
    .withColumnRenamed('C007', 'tipo_ocupacao') \
    .withColumnRenamed('C007A', 'area_trabalho') \
    .withColumnRenamed('C007B', 'carteira_assinada_ou_estatutario') \
    .withColumnRenamed('C007C', 'cargo_funcao_trabalho') \
    .withColumnRenamed('C007D', 'atividade_empresa') \
    .withColumnRenamed('C007E', 'numero_empregados_negocio') \
    .withColumnRenamed('C008', 'horas_trabalhadas_habitual') \
    .withColumnRenamed('C009', 'horas_trabalhadas_semana') \
    .withColumnRenamed('C010', 'recebia_habitualmente') \
    .withColumnRenamed('C0101', 'recebia_em_dinheiro') \
    .withColumnRenamed('C01011', 'faixa_rendimento_dinheiro') \
    .withColumnRenamed('C01012', 'valor_rendimento_dinheiro') \
    .withColumnRenamed('C0103', 'recebia_beneficios') \
    .withColumnRenamed('C0104', 'nao_remunerado') \
    .withColumnRenamed('C011A', 'recebia_efetivamente') \
    .withColumnRenamed('C011A1', 'recebia_efetivamente_dinheiro') \
    .withColumnRenamed('C011A11', 'faixa_rendimento_efetivo_dinheiro') \
    .withColumnRenamed('C011A12', 'valor_rendimento_efetivo_dinheiro') \
    .withColumnRenamed('C012', 'trabalho_mesmo_local') \
    .withColumnRenamed('C013', 'trabalho_remoto') \
    .withColumnRenamed('C014', 'contribui_inss') \
    .withColumnRenamed('C015', 'procurou_trabalho') \
    .withColumnRenamed('C016', 'motivo_nao_procurou_trabalho') \
    .withColumnRenamed('C017A', 'gostaria_trabalhar') \
    .withColumnRenamed('D0011', 'recebeu_aposentadoria_pensao') \
    .withColumnRenamed('D0013', 'valor_aposentadoria_pensao') \
    .withColumnRenamed('D0021', 'recebeu_pensao_doacao_mesada') \
    .withColumnRenamed('D0023', 'valor_pensao_doacao_mesada') \
    .withColumnRenamed('D0031', 'recebeu_bolsa_familia') \
    .withColumnRenamed('D0033', 'valor_bolsa_familia') \
    .withColumnRenamed('D0041', 'recebeu_bpc_loas') \
    .withColumnRenamed('D0043', 'valor_bpc_loas') \
    .withColumnRenamed('D0051', 'recebeu_auxilio_emergencial') \
    .withColumnRenamed('D0053', 'valor_auxilio_emergencial') \
    .withColumnRenamed('D0061', 'recebeu_seguro_desemprego') \
    .withColumnRenamed('D0063', 'valor_seguro_desemprego') \
    .withColumnRenamed('D0071', 'recebeu_outros_rendimentos') \
    .withColumnRenamed('D0073', 'valor_outros_rendimentos') \
    .withColumnRenamed('F001', 'condicao_domicilio') \
    .withColumnRenamed('F0021', 'valor_aluguel_pago') \
    .withColumnRenamed('F0022', 'faixa_valor_aluguel') \
    .withColumnRenamed('F0061', 'quem_respondeu_questionario') \
```

```
    .withColumnRenamed('F006', 'numero_ordem_informante')
df1.show(10)
```

...

2.3 Agrupando coluna dia_nascimento, mes_nascimento e ano_nascimento para colocar apenas a data de nacimento:

```
# vamor formatar o número para que ele sempre tenha dois dígitos:
df1 = df1.withColumn('dia_formatado', lpad(col('dia_nascimento').cast('string'), 2, '0'))
df1 = df1.withColumn('mes_formatado', lpad(col('mes_nascimento').cast('string'), 2, '0'))
```

```
# agora, agrupando em uma única coluna:
df2 = df1.withColumn('data_nascimento', concat(col('dia_formatado'), col('mes_formatado'), col('ano_nascimento')))
```

```
df2.show(10)
```

...

```
df3 = df2.withColumn('data_nasc', to_date(col('data_nascimento'), 'ddMMyyyy'))
```

```
df3.show(10)
```

...

2.4 Apagando as colunas:

- dia_nascimento
- mes_nascimento
- ano_nascimento

- dia_formatado
- mes_formatado

```
# vamos verificar o númeoro de colunas antes:
print('df3. col ct : ', len(df3.columns))
```

```
# apagando as colunas:
df4 = df3.drop(*['dia_nascimento', 'mes_nascimento', 'ano_nascimento', 'dia_formatado', 'mes_formatado'])
df4.show(5)
```

```
# apagando a coluna data_nascimento:
df5 = df4.drop(*['data_nascimento','numero_formatado'])
df5.show(5)
```

```
# verificando o número de colunas depois:
print('df5. col ct : ', len(df5.columns))
```

2.5 Selecionando algumas colunas:

2.5.1 Contando o número de pesquisa realizada em cada mês:

```
# Agrupar pelo mês da pesquisa e contar o número de entrevistas
numero_pesquisa_meses = df5.groupBy('mes_pesquisa').count().orderBy('mes_pesquisa')
numero_pesquisa_meses.show()
```

```
# contando o número total de pessoas:
total_pessoas = df5.count()
print(f"Total de pessoas: {total_pessoas}")
```

2.5.2 Agrupando por cor/raça:

```
# criando uma coluna com o nome da cor/raçadf_raca = df.withColumn(
df_raca = df5.withColumn(
    'cor_ou_raca_nome',
    when(col('cor_ou_raca') == '1', 'Branca')
```

```
    .when(col('cor_ou_raca') == '2', 'Preta')
    .when(col('cor_ou_raca') == '3', 'Amarela')
    .when(col('cor_ou_raca') == '4', 'Parda')
    .when(col('cor_ou_raca') == '5', 'Indígena')
    .otherwise('Ignorado')
)

# Contar o número de pessoas por cor/raça
tabela_raca = df_raca.groupBy('cor_ou_raca_nome').count().orderBy('count', ascending=False)
tabela_raca.show()
```

...

2.5.3 Agrupando por escolaridade:

```
# criando uma coluna descritiva de escolaridade
df_escolaridade = df5.withColumn(
    'escolaridade_nome',
    when(col('escolaridade') == '1', 'Sem instrução')
    .when(col('escolaridade') == '2', 'Fundamental incompleto')
    .when(col('escolaridade') == '3', 'Fundamental completo')
    .when(col('escolaridade') == '4', 'Médio incompleto')
    .when(col('escolaridade') == '5', 'Médio completo')
    .when(col('escolaridade') == '6', 'Superior incompleto')
    .when(col('escolaridade') == '7', 'Superior completo')
    .when(col('escolaridade') == '8', 'Pós-graduação')
    .otherwise('Ignorado')
)

# Conta o número de pessoas por escolaridade
tabela_escolaridade = df_escolaridade.groupBy('escolaridade_nome').count().orderBy('count', ascending=False)
tabela_escolaridade.show()
```

...

2.5.4 Agrupando por estado:

```
# criando uma coluna com o nome do estado
df_estado = df5.withColumn(
    'estado_nome',
    when(col('unidade_federacao') == '11', 'Rondônia')
    .when(col('unidade_federacao') == '12', 'Acre')
    .when(col('unidade_federacao') == '13', 'Amazonas')
    .when(col('unidade_federacao') == '14', 'Roraima')
    .when(col('unidade_federacao') == '15', 'Pará')
    .when(col('unidade_federacao') == '16', 'Amapá')
    .when(col('unidade_federacao') == '17', 'Tocantins')
    .when(col('unidade_federacao') == '21', 'Maranhão')
    .when(col('unidade_federacao') == '22', 'Piauí')
    .when(col('unidade_federacao') == '23', 'Ceará')
    .when(col('unidade_federacao') == '24', 'Rio Grande do Norte')
    .when(col('unidade_federacao') == '25', 'Paraíba')
    .when(col('unidade_federacao') == '26', 'Pernambuco')
    .when(col('unidade_federacao') == '27', 'Alagoas')
    .when(col('unidade_federacao') == '28', 'Sergipe')
    .when(col('unidade_federacao') == '29', 'Bahia')
```

```
    .when(col('unidade_federacao') == '31', 'Minas Gerais')
    .when(col('unidade_federacao') == '32', 'Espírito Santo')
    .when(col('unidade_federacao') == '33', 'Rio de Janeiro')
    .when(col('unidade_federacao') == '35', 'São Paulo')
    .when(col('unidade_federacao') == '41', 'Paraná')
    .when(col('unidade_federacao') == '42', 'Santa Catarina')
    .when(col('unidade_federacao') == '43', 'Rio Grande do Sul')
    .when(col('unidade_federacao') == '50', 'Mato Grosso do Sul')
    .when(col('unidade_federacao') == '51', 'Mato Grosso')
    .when(col('unidade_federacao') == '52', 'Goiás')
    .when(col('unidade_federacao') == '53', 'Distrito Federal')
)

# Contar o número de pessoas por estado
tabela_estado = df_estado.groupBy('estado_nome').count().orderBy('count', ascending=False)
tabela_estado.show(27)
```

...

2.5.5 Agora, vamos agrupar por sintomas:

```
# Lista de sintomas e seus nomes descritivos
sintomas = [
    ('semana_passada_febre', 'Febre'),
    ('semana_passada_tosse', 'Tosse'),
    ('semana_passada_dor_garganta', 'Dor de garganta'),
    ('semana_passada_dificuldade_respirar', 'Dificuldade para respirar'),
    ('semana_passada_dor_cabeca', 'Dor de cabeça'),
    ('semana_passada_dor_peito', 'Dor no peito'),
    ('semana_passada_nausea', 'Náusea'),
    ('semana_passada_nariz_entupido_escorrendo', 'Nariz entupido ou escorrendo'),
    ('semana_passada_fadiga', 'Fadiga'),
    ('semana_passada_dor_olhos', 'Dor nos olhos'),
    ('semana_passada_perda_cheiro_sabor', 'Perda de cheiro ou sabor'),
    ('semana_passada_dor_muscular', 'Dor muscular')
]

# Número total de pessoas (para porcentagem)
total_pessoas

# Lista para armazenar resultados
resultados = []

for codigo, nome in sintomas:
    # Conta quantas pessoas responderam '1' (Sim) para o sintoma
    count_sim = df5.filter(col(codigo) == '1').count()
    perc_sim = __builtins__.round((count_sim / total_pessoas) * 100, 2)
    resultados.append((nome, count_sim, float(perc_sim)))

# Cria um DataFrame com os resultados
```

```
result_df = spark.createDataFrame(resultados, ['Sintoma', 'Número absoluto', 'Porcentagem (%)'])
result_df.show(truncate=False)
```

...

```
from pyspark.sql.functions import col

# Lista de sintomas e seus nomes descritivos
sintomas = [
    ('semana_passada_febre', 'Febre'),
    ('semana_passada_tosse', 'Tosse'),
    ('semana_passada_dor_garganta', 'Dor de garganta'),
    ('semana_passada_dificuldade_respirar', 'Dificuldade para respirar'),
    ('semana_passada_dor_cabeca', 'Dor de cabeça'),
    ('semana_passada_dor_peito', 'Dor no peito'),
    ('semana_passada_nausea', 'Náusea'),
    ('semana_passada_nariz_entupido_escorrendo', 'Nariz entupido ou escorrendo'),
    ('semana_passada_fadiga', 'Fadiga'),
    ('semana_passada_dor_olhos', 'Dor nos olhos'),
    ('semana_passada_perda_cheiro_sabor', 'Perda de cheiro ou sabor'),
    ('semana_passada_dor_muscular', 'Dor muscular')
]

# Número total de pessoas (para porcentagem)
total_pessoas

# Lista para armazenar resultados
resultados = []

for codigo, nome in sintomas:
    # Conta quantas pessoas responderam '1' (Sim) para o sintoma
    count_sim = df5.filter(col(codigo) == '1').count()
    perc_sim = __builtins__.round((count_sim / total_pessoas) * 100, 2)

    # Conta quantas pessoas responderam '2' (Não) para o sintoma
    count_nao = df5.filter(col(codigo) == '2').count()
    perc_nao = __builtins__.round((count_nao / total_pessoas) * 100, 2)

    resultados.append((nome, count_sim, perc_sim, count_nao, perc_nao))

# Cria um DataFrame com os resultados
result_df = spark.createDataFrame(resultados, [
    'Sintoma',
    'Número (Sim)', 'Porcentagem (Sim) %',
    'Número (Não)', 'Porcentagem (Não) %'
])
result_df.show(truncate=False)
```

...

3.0 Construindo alguns gráficos:

3.1 Gráfico do número de pesquisa realizada em cada mês:

```
# Converta o DataFrame Spark para pandas
numero_pesquisa_meses_pd = numero_pesquisa_meses.toPandas()

# Gera o gráfico de barras
plt.figure(figsize=(10, 5))
plt.bar(numero_pesquisa_meses_pd['mes_pesquisa'], numero_pesquisa_meses_pd['count'], color='blue', width=0.3)

plt.xlabel('Mês da Pesquisa')
plt.ylabel('Número de Entrevistas')
plt.title('Número de Entrevistas por Mês')
plt.grid(True, linestyle='--', color='black', linewidth=0.8)
plt.gca().set_facecolor('lightgray')
plt.tight_layout()
plt.show()
```

...

Clique duas vezes (ou pressione "Enter") para editar

3.2 Gráfico por cor/raça:

```
# Convertendo para pandas para plotar o gráfico
tabela_raca_pd = tabela_raca.toPandas()

# Calcular porcentagem
total = tabela_raca_pd['count'].sum()
tabela_raca_pd['porcentagem'] = (tabela_raca_pd['count'] / total) * 100

# Plotar o gráfico
plt.figure(figsize=(12, 6))
plt.bar(tabela_raca_pd['cor_ou_raca_nome'], tabela_raca_pd['porcentagem'], color='blue',width=0.3)
plt.xlabel('Cor ou Raça')
plt.ylabel('Porcentagem (%)')
plt.title('Distribuição por Cor ou Raça')
plt.xticks(rotation=45, ha='right')
plt.grid(True, linestyle='--', color='black', linewidth=0.8)
plt.gca().set_facecolor('lightgray')  # Fundo cinza
plt.tight_layout()
plt.show()
```

...

3.3 Gráfico por escolaridade:

```
# Converter o DataFrame Spark para pandas
tabela_escolaridade_pd = tabela_escolaridade.toPandas()

# Gráfico de barras com barras finas
plt.figure(figsize=(12, 6))
plt.bar(tabela_escolaridade_pd['escolaridade_nome'], tabela_escolaridade_pd['count'], color='blue', width=0.3)
plt.xlabel('Escolaridade')
plt.ylabel('Número de Pessoas')
plt.title('Distribuição por Escolaridade')
plt.xticks(rotation=45, ha='right')
plt.grid(True, linestyle='--', color='black', linewidth=0.8)
plt.gca().set_facecolor('lightgray')
plt.tight_layout()
plt.show()
```

...

3.4 Gráfico por estado:

```
import matplotlib.pyplot as plt

# Converter o DataFrame Spark para pandas
tabela_estado_pd = tabela_estado.toPandas()
```

```
5
6 # Gráfico de barras com barras finas
7 plt.figure(figsize=(14, 6))
8 plt.bar(tabela_estado_pd['estado_nome'], tabela_estado_pd['count'], color='blue')
9 plt.xlabel('Estado')
10 plt.ylabel('Número de Pessoas')
11 plt.title('Distribuição de Pessoas por Estado')
12 plt.xticks(rotation=45, ha='right')
13 plt.grid(True, linestyle='--', color='black', linewidth=0.8)
14 plt.gca().set_facecolor('lightgray')  # Fundo cinza
15 plt.tight_layout()
16 plt.show()
```

•••

Gráfico por sintomas:

```
1 # Converta o DataFrame Spark para pandas
2 result_pd = result_df.toPandas()
3
4 # Crie o gráfico de barras para a porcentagem de pessoas que tiveram o sintoma
5 plt.figure(figsize=(12, 6))
6 plt.bar(result_pd['Sintoma'], result_pd['Porcentagem (Sim) %'], color='blue', width=0.3)
7 plt.xticks(rotation=45, ha='right')
8 plt.xlabel('Sintoma')
9 plt.ylabel('Porcentagem (%)')
10 plt.title('Porcentagem de Pessoas com Sintomas na Semana Passada')
11 plt.grid(True, linestyle='--', color='gray', linewidth=0.8)
12 plt.gca().set_facecolor('lightgray')
13 plt.tight_layout()
14 plt.show()
15
```

•••

```
import numpy as np

x = np.arange(len(result_pd['Sintoma']))
width = 0.35

fig, ax = plt.subplots(figsize=(14, 6))
bars1 = ax.bar(x - width/2, result_pd['Porcentagem (Sim) %'], width, label='Sim', color='blue')
bars2 = ax.bar(x + width/2, result_pd['Porcentagem (Não) %'], width, label='Não', color='green')

ax.set_xlabel('Sintoma')
ax.set_ylabel('Porcentagem (%)')
ax.set_title('Porcentagem de Pessoas com e sem Sintomas na Semana Passada')
ax.set_xticks(x)
ax.set_xticklabels(result_pd['Sintoma'], rotation=45, ha='right')
ax.legend()
plt.tight_layout()
plt.grid(True, linestyle='--', color='gray', linewidth=0.8)
plt.gca().set_facecolor('lightgray')
plt.show()

```

...

4.0 Analisando os dados:

4.1 Analisando se os sintomas tem relação com escolaridade e/ou renda:

```
# Lista das colunas de sintomas (conforme o dicionário PNAD COVID-19)
colunas_sintomas = [
    'semana_passada_febre', 'semana_passada_tosse', 'semana_passada_dor_garganta',
    'semana_passada_dificuldade_respirar', 'semana_passada_dor_cabeca', 'semana_passada_dor_peito',
    'semana_passada_nausea', 'semana_passada_nariz_entupido_escorrendo', 'semana_passada_fadiga',
    'semana_passada_dor_olhos', 'semana_passada_perda_cheiro_sabor', 'semana_passada_dor_muscular'
]

# Expressão para somar sintomas com valor 1 (Sim)
expr_sintomas = sum([when(col(c) == 1, 1).otherwise(0) for c in colunas_sintomas])

# Adiciona a coluna descritiva de escolaridade e o número de sintomas
df_novo = df5.withColumn(
    'escolaridade_nome',
    when(col('escolaridade') == 1, 'Sem instrução')
    .when(col('escolaridade') == 2, 'Fundamental incompleto')
    .when(col('escolaridade') == 3, 'Fundamental completo')
```

```
    .when(col('escolaridade') == 4, 'Médio incompleto')
    .when(col('escolaridade') == 5, 'Médio completo')
    .when(col('escolaridade') == 6, 'Superior incompleto')
    .when(col('escolaridade') == 7, 'Superior completo')
    .when(col('escolaridade') == 8, 'Pós-graduação')
    .otherwise('Ignorado')
).withColumn(
    'numero_sintomas', expr_sintomas
)

# Seleciona as colunas desejadas
df_final = df_novo.select('idade_morador', 'escolaridade_nome', 'numero_sintomas')

df_final.show(10)
```

...

```
Comece a programar ou gere código com IA.
```