ETAPA 1 - Análise Descritiva da Exposição de IA Generativa com ILO Index e PNADc

PREPARAÇÃO DOS DADOS

Dissertação: Inteligência Artificial Generativa e o Mercado de Trabalho Brasileiro: Uma Análise de Exposição Ocupacional e seus Efeitos Distributivos.

Aluno: Manoel Brasil Orlandi

Contextualização

A rápida difusão de modelos de IA generativa (LLMs, geradores de imagem/código) levanta questões centrais sobre seus impactos no mercado de trabalho. Para mensurar esse potencial de impacto, Gmyrek, Berg & Cappelli (2024, 2025) desenvolveram, no âmbito da Organização Internacional do Trabalho (OIT), um índice de exposição ocupacional à IA generativa, publicado como *Working Paper* 140 (WP140). O índice atribui scores de exposição a cada ocupação da classificação ISCO-08, com base na avaliação de suas tarefas constituintes por modelos de linguagem e validação humana.

Este notebook constrói a base analítica que une os microdados da **PNAD Contínua** (Pesquisa Nacional por Amostra de Domicílios Contínua, IBGE, 3º trimestre de 2025) ao **índice de exposição à IA generativa da OIT**, permitindo caracterizar a exposição do mercado de trabalho brasileiro a essa tecnologia.

Objetivo

Construir a base analítica que une PNAD Contínua e o índice de exposição à IA (ILO), com ocupações em COD e ISCO-08.

Entradas: Microdados PNAD (BigQuery), planilha ILO (Gmyrek et al., 2025), estrutura COD.
Saída principal: `data/output/pnad_ilo_merged.csv`

Referências principais

- Gmyrek, P., Berg, J. & Cappelli, D. (2025). *Generative AI and Jobs: An updated global assessment of potential effects on job quantity and quality.* ILO Working Paper 140.
- IBGE. *Pesquisa Nacional por Amostra de Domicílios Contínua* (PNADc), 3º trimestre de 2025.

1. Configuração do ambiente

Definir caminhos, importar bibliotecas e configurar logs.

```
# Instalar dependências no kernel atual (executar apenas uma vez)
%pip install pandas numpy pyarrow openpyxl basedosdados --quiet
```

```
[notice] A new release of pip is available: 24.2 -> 26.0.1
[notice] To update, run: python3.10 -m pip install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

```
# Etapa 1.1 - Preparação de Dados - Configuração do ambiente

import warnings
import pandas as pd
import numpy as np
from pathlib import Path
import re

warnings.filterwarnings("ignore", category=FutureWarning)

# ------------------------------------------------------------------------
# Caminhos (relativos ao diretório do notebook)
# ------------------------------------------------------------------------
DATA_INPUT     = Path("data/input")
DATA_RAW       = Path("data/raw")
DATA_PROCESSED = Path("data/processed")
DATA_OUTPUT    = Path("data/output")

for d in [DATA_RAW, DATA_PROCESSED, DATA_OUTPUT]:
    d.mkdir(parents=True, exist_ok=True)

# ------------------------------------------------------------------------
```

```python
# Parâmetros PNAD / GCP
# ------------------------------------------------------------------------
GCP_PROJECT_ID  = "mestrado-pnad-2026"
PNAD_ANO        = 2025
PNAD_TRIMESTRE  = 3
SALARIO_MINIMO  = 1518  # Valor vigente em Q3/2025 (R$)

# ------------------------------------------------------------------------
# Arquivo ILO (já copiado para data/input)
# ------------------------------------------------------------------------
ILO_FILE = DATA_INPUT / "Final_Scores_ISCO08_Gmyrek_et_al_2025.xlsx"

# ------------------------------------------------------------------------
# Mapeamentos
# ------------------------------------------------------------------------
REGIAO_MAP = {
    'RO': 'Norte', 'AC': 'Norte', 'AM': 'Norte', 'RR': 'Norte',
    'PA': 'Norte', 'AP': 'Norte', 'TO': 'Norte',
    'MA': 'Nordeste', 'PI': 'Nordeste', 'CE': 'Nordeste', 'RN': 'Nordeste',
    'PB': 'Nordeste', 'PE': 'Nordeste', 'AL': 'Nordeste', 'SE': 'Nordeste', 'BA': 'Nordeste'
    'MG': 'Sudeste', 'ES': 'Sudeste', 'RJ': 'Sudeste', 'SP': 'Sudeste',
    'PR': 'Sul', 'SC': 'Sul', 'RS': 'Sul',
    'MS': 'Centro-Oeste', 'MT': 'Centro-Oeste', 'GO': 'Centro-Oeste', 'DF': 'Centro-Oeste',
}

GRANDES_GRUPOS = {
    '1': 'Dirigentes e gerentes',
    '2': 'Profissionais das ciências',
    '3': 'Técnicos nível médio',
    '4': 'Apoio administrativo',
    '5': 'Serviços e vendedores',
    '6': 'Agropecuária qualificada',
    '7': 'Indústria qualificada',
    '8': 'Operadores de máquinas',
    '9': 'Ocupações elementares',
}

RACA_AGREGADA_MAP = {
    '1': 'Branca',
    '2': 'Negra',   # Preta
    '4': 'Negra',   # Parda
    '3': 'Outras',  # Amarela
```

```
    '5': 'Outras',  # Indígena
    '9': 'Outras',  # Sem declaração
}

POSICAO_FORMAL = ['1', '3', '5']  # Empregado c/ carteira, Militar, Empregador

IDADE_BINS   = [0, 25, 35, 45, 55, 100]
IDADE_LABELS = ['18-24', '25-34', '35-44', '45-54', '55+']

# ------------------------------------------------------------------------
# Mapeamento CNAE Domiciliar 2.0 → Setor agregado
# Seções A-T conforme classificação oficial IBGE.
# Fonte: IBGE, Classificação Nacional de Atividades Econômicas (CNAE 2.0)
# ------------------------------------------------------------------------
CNAE_SETOR_MAP = {
    # A - Agropecuária
    '01': 'Agropecuária', '02': 'Agropecuária', '03': 'Agropecuária',
    # B - Indústria Extrativa
    '05': 'Ind. Extrativa', '06': 'Ind. Extrativa', '07': 'Ind. Extrativa',
    '08': 'Ind. Extrativa', '09': 'Ind. Extrativa',
    # C - Indústria de Transformação
    '10': 'Ind. Transformação', '11': 'Ind. Transformação', '12': 'Ind. Transformação',
    '13': 'Ind. Transformação', '14': 'Ind. Transformação', '15': 'Ind. Transformação',
    '16': 'Ind. Transformação', '17': 'Ind. Transformação', '18': 'Ind. Transformação',
    '19': 'Ind. Transformação', '20': 'Ind. Transformação', '21': 'Ind. Transformação',
    '22': 'Ind. Transformação', '23': 'Ind. Transformação', '24': 'Ind. Transformação',
    '25': 'Ind. Transformação', '26': 'Ind. Transformação', '27': 'Ind. Transformação',
    '28': 'Ind. Transformação', '29': 'Ind. Transformação', '30': 'Ind. Transformação',
    '31': 'Ind. Transformação', '32': 'Ind. Transformação', '33': 'Ind. Transformação',
    # D+E - Utilidades (Eletricidade, Gás, Água, Esgoto, Resíduos)
    '35': 'Utilidades', '36': 'Utilidades', '37': 'Utilidades',
    '38': 'Utilidades', '39': 'Utilidades',
    # F - Construção
    '41': 'Construção', '42': 'Construção', '43': 'Construção',
    # G - Comércio
    '45': 'Comércio', '46': 'Comércio', '47': 'Comércio',
    # H - Transporte, Armazenagem e Correio
    '49': 'Transporte', '50': 'Transporte', '51': 'Transporte',
    '52': 'Transporte', '53': 'Transporte',
    # I - Alojamento e Alimentação
    '55': 'Alojamento e Alimentação', '56': 'Alojamento e Alimentação',
    # J - Informação e Comunicação
```

```
    '58': 'Informação e Comunicação', '59': 'Informação e Comunicação',
    '60': 'Informação e Comunicação', '61': 'Informação e Comunicação',
    '62': 'Informação e Comunicação', '63': 'Informação e Comunicação',
    # K - Atividades Financeiras
    '64': 'Finanças e Seguros', '65': 'Finanças e Seguros', '66': 'Finanças e Seguros',
    # L - Atividades Imobiliárias
    '68': 'Atividades Imobiliárias',
    # M - Atividades Profissionais, Científicas e Técnicas
    '69': 'Serviços Profissionais', '70': 'Serviços Profissionais',
    '71': 'Serviços Profissionais', '72': 'Serviços Profissionais',
    '73': 'Serviços Profissionais', '74': 'Serviços Profissionais',
    '75': 'Serviços Profissionais',
    # N - Atividades Administrativas e Serviços Complementares
    '77': 'Serviços Administrativos', '78': 'Serviços Administrativos',
    '79': 'Serviços Administrativos', '80': 'Serviços Administrativos',
    '81': 'Serviços Administrativos', '82': 'Serviços Administrativos',
    # O - Administração Pública
    '84': 'Administração Pública',
    # P - Educação
    '85': 'Educação',
    # Q - Saúde Humana e Serviços Sociais
    '86': 'Saúde', '87': 'Saúde', '88': 'Saúde',
    # R - Artes, Cultura, Esporte e Recreação
    '90': 'Artes e Cultura', '91': 'Artes e Cultura',
    '92': 'Artes e Cultura', '93': 'Artes e Cultura',
    # S - Outras Atividades de Serviços
    '94': 'Outros Serviços', '95': 'Outros Serviços', '96': 'Outros Serviços',
    # T - Serviços Domésticos
    '97': 'Serviços Domésticos',
}

# Setores com maior proporção de tarefas expostas à IA generativa
# Ref: Gmyrek et al. (2024); Eloundou et al. (2023)
SETORES_CRITICOS_IA = [
    'Informação e Comunicação',
    'Finanças e Seguros',
    'Serviços Profissionais',
]

# ---------------------------------------------------------------------------
# Funções utilitárias - estatísticas ponderadas
# ---------------------------------------------------------------------------
```

```python
def weighted_mean(values, weights):
    """Média ponderada (ignora NaN)."""
    mask = ~(pd.isna(values) | pd.isna(weights))
    if mask.sum() == 0:
        return np.nan
    return np.average(values[mask], weights=weights[mask])

def weighted_std(values, weights):
    """Desvio-padrão ponderado (ignora NaN)."""
    mask = ~(pd.isna(values) | pd.isna(weights))
    if mask.sum() == 0:
        return np.nan
    avg = np.average(values[mask], weights=weights[mask])
    variance = np.average((values[mask] - avg) ** 2, weights=weights[mask])
    return np.sqrt(variance)

def weighted_quantile(values, weights, quantile):
    """Quantil ponderado por pesos amostrais (ignora NaN).
    Fonte: adaptado de etapa1_ia_generativa/src/utils/weighted_stats.py
    """
    mask = ~(pd.isna(values) | pd.isna(weights))
    if mask.sum() == 0:
        return np.nan
    sorted_idx = np.argsort(values[mask])
    sorted_values = values[mask].iloc[sorted_idx]
    sorted_weights = weights[mask].iloc[sorted_idx]
    cumsum = np.cumsum(sorted_weights)
    cutoff = quantile * cumsum.iloc[-1]
    return sorted_values.iloc[np.searchsorted(cumsum, cutoff)]

def weighted_qcut(values, weights, q, labels=None):
    """Classificação em quantis ponderados por peso amostral.

    Diferente de pd.qcut (que divide por contagem de linhas), esta função
    calcula os breakpoints de modo que cada faixa represente ~1/q da
    POPULAÇÃO (soma dos pesos), não da amostra.

    Parâmetros:
        values  : pd.Series com os valores a classificar
        weights : pd.Series com os pesos amostrais
        q       : int, número de quantis (5 = quintis, 10 = decis)
        labels  : lista de labels (len == q), ou None para retornar inteiros 1..q
```

```
    Retorna:
        pd.Series (Categorical) com os labels atribuídos
    """
    mask = values.notna() & weights.notna()
    breakpoints = [values[mask].min() - 1e-10]  # incluir mínimo
    for i in range(1, q):
        bp = weighted_quantile(values[mask], weights[mask], i / q)
        breakpoints.append(bp)
    breakpoints.append(values[mask].max() + 1e-10)  # incluir máximo

    # Remover duplicatas mantendo ordem (pode acontecer com valores concentrados)
    breakpoints = sorted(set(breakpoints))

    if labels is not None and len(labels) != len(breakpoints) - 1:
        labels = None  # fallback se breakpoints colapsaram

    result = pd.cut(values, bins=breakpoints, labels=labels, include_lowest=True)
    return result

print("Configuração carregada com sucesso.")
print(f"  PNAD: {PNAD_ANO} Q{PNAD_TRIMESTRE}")
print(f"  Projeto GCP: {GCP_PROJECT_ID}")
print(f"  Salário mínimo: R$ {SALARIO_MINIMO}")
print(f"  ILO file: {ILO_FILE} (existe: {ILO_FILE.exists()})")
print(f"  Setores CNAE mapeados: {len(set(CNAE_SETOR_MAP.values()))} categorias")
print(f"  Setores críticos IA: {SETORES_CRITICOS_IA}")
```

```
Configuração carregada com sucesso.
  PNAD: 2025 Q3
  Projeto GCP: mestrado-pnad-2026
  Salário mínimo: R$ 1518
  ILO file: data/input/Final_Scores_ISCO08_Gmyrek_et_al_2025.xlsx (existe: True)
  Setores CNAE mapeados: 19 categorias
  Setores críticos IA: ['Informação e Comunicação', 'Finanças e Seguros', 'Serviços Profissi
```

2a. Download dos microdados PNAD

Extrair da PNAD Contínua (BigQuery) as variáveis necessárias para o trimestre/ano definido.
Saída: `data/raw/pnad_*.parquet`

Ficha técnica dos dados

Item	Descrição
Fonte	PNAD Contínua (PNADc), IBGE
Período	3º trimestre de 2025
Acesso	Base dos Dados (BigQuery)
Peso amostral	V1028 (projeção de população para dados trimestrais)
Universo	População ocupada com código de ocupação válido

Variáveis selecionadas

Variável IBGE	Nome no dataset	Descrição
V2007	`sexo`	Sexo biológico
V2009	`idade`	Idade em anos
V2010	`raca_cor`	Cor ou raça (autoclassificação)
VD3004	`nivel_instrucao`	Nível de instrução mais elevado alcançado
V4010	`cod_ocupacao`	Código de ocupação (COD, 4 dígitos)
V4013	`grupamento_atividade`	Grupamento de atividade (CNAE Domiciliar 2.0)
VD4009	`posicao_ocupacao`	Posição na ocupação
VD4016	`rendimento_habitual`	Rendimento mensal habitual do trabalho principal
VD4020	`rendimento_efetivo`	Rendimento mensal efetivo do trabalho principal
VD4031	`horas_habituais`	Horas habitualmente trabalhadas (todos os trabalhos)
VD4035	`horas_efetivas`	Horas efetivamente trabalhadas na semana de referência
V1028	`peso`	Peso amostral (projeção de população)

Nota metodológica — Variáveis de renda: O rendimento mensal habitual (`VD4016`) é a variável primária para análises estruturais de exposição ocupacional, por ser menos volátil que o rendimento efetivo (`VD4020`), que captura flutuações mensais por horas extras, bônus, etc. (Cf. IBGE, Notas Metodológicas PNAD Contínua, 2023). Ambas são mantidas na base.

Nota metodológica — Horas trabalhadas: Utiliza-se `VD4031` (horas habitualmente trabalhadas em todos os trabalhos, variável derivada IBGE) como variável principal de jornada, por ter cobertura superior à variável bruta `V4019` (~30% de preenchimento na versão anterior). `VD4035` (horas efetivamente trabalhadas na semana de referência) é incluída para análises de sazonalidade e produtividade.

Nota metodológica — Inclusão de todos os ocupados: A query inclui **todos os ocupados com código de ocupação válido**, independentemente de terem renda declarada. O filtro de renda é aplicado via flag `tem_renda` na etapa de limpeza (4a).

Nota metodológica — Variáveis indisponíveis: As variáveis `V4040` (tempo no emprego atual) e `V4018` (porte da empresa) não estão populadas na fonte utilizada (Base dos Dados/BigQuery) para o período analisado (Q3/2025), sendo portanto excluídas desta análise.

```
# Etapa 1.2a - Preparação de Dados - Download dos microdados PNAD
# Lógica: se o parquet já existe em data/raw/, carrega direto; senão, baixa do BigQuery.

pnad_files = sorted(DATA_RAW.glob("pnad_*.parquet"))

if pnad_files:
    # --- Caminho rápido: arquivo local já disponível ---
    pnad_path = pnad_files[-1]  # mais recente
    print(f"Arquivo PNAD encontrado localmente: {pnad_path.name}")
    df_pnad_raw = pd.read_parquet(pnad_path)
    print(f"Carregado: {len(df_pnad_raw):,} observações")

    # Validar que o arquivo corresponde à configuração
    match = re.search(r"pnad_(\d{4})q(\d)", pnad_path.name)
    if match:
        ano_arquivo, trim_arquivo = int(match.group(1)), int(match.group(2))
        if ano_arquivo != PNAD_ANO or trim_arquivo != PNAD_TRIMESTRE:
            print(f"  WARNING: Arquivo é {ano_arquivo} Q{trim_arquivo}, "
                  f"mas config diz {PNAD_ANO} Q{PNAD_TRIMESTRE}!")
            print(f"  Atualizando variáveis de config para corresponder aos dados.")
            PNAD_ANO = ano_arquivo
```

```
            PNAD_TRIMESTRE = trim_arquivo
        else:
            print(f"  OK: Arquivo corresponde à configuração ({PNAD_ANO} Q{PNAD_TRIMESTRE})")

else:
    # --- Caminho completo: download via BigQuery ---
    print("Nenhum arquivo PNAD local encontrado. Iniciando download do BigQuery...")
    import basedosdados as bd

    # Verificar trimestres disponíveis
    query_check = """
    SELECT DISTINCT ano, trimestre, COUNT(*) as n_obs
    FROM `basedosdados.br_ibge_pnadc.microdados`
    WHERE ano >= 2024
    GROUP BY ano, trimestre
    ORDER BY ano DESC, trimestre DESC
    LIMIT 5
    """
    df_check = bd.read_sql(query_check, billing_project_id=GCP_PROJECT_ID)
    print(f"Trimestres disponíveis:\n{df_check}")

    trimestre_existe = len(
        df_check[(df_check['ano'] == PNAD_ANO) & (df_check['trimestre'] == PNAD_TRIMESTRE)]
    ) > 0

    if trimestre_existe:
        ano_usar, trim_usar = PNAD_ANO, PNAD_TRIMESTRE
    else:
        ano_usar = int(df_check.iloc[0]['ano'])
        trim_usar = int(df_check.iloc[0]['trimestre'])
        print(f"AVISO: {PNAD_ANO} Q{PNAD_TRIMESTRE} indisponível. Usando {ano_usar} Q{trim_us
        PNAD_ANO = ano_usar
        PNAD_TRIMESTRE = trim_usar

    query = f"""
    SELECT
        ano,
        trimestre,
        sigla_uf,
        v2007  AS sexo,
        v2009  AS idade,
        v2010  AS raca_cor,
```

```
        vd3004 AS nivel_instrucao,
        v4010  AS cod_ocupacao,
        v4013  AS grupamento_atividade,
        vd4009 AS posicao_ocupacao,
        vd4016 AS rendimento_habitual,
        vd4020 AS rendimento_efetivo,
        vd4031 AS horas_habituais,
        vd4035 AS horas_efetivas,
        v1028  AS peso
    FROM `basedosdados.br_ibge_pnadc.microdados`
    WHERE ano = {ano_usar}
      AND trimestre = {trim_usar}
      AND v4010 IS NOT NULL
    """

    print(f"Executando query para {ano_usar} Q{trim_usar} (pode demorar 2-5 min)...")
    df_pnad_raw = bd.read_sql(query, billing_project_id=GCP_PROJECT_ID)

    # Salvar parquet
    ano_real = int(df_pnad_raw['ano'].iloc[0])
    trim_real = int(df_pnad_raw['trimestre'].iloc[0])
    output_path = DATA_RAW / f"pnad_{ano_real}q{trim_real}.parquet"
    df_pnad_raw.to_parquet(output_path, index=False)
    print(f"Salvo em: {output_path}")

print(f"\ndf_pnad_raw: {df_pnad_raw.shape[0]:,} linhas x {df_pnad_raw.shape[1]} colunas")
print(f"Período: {PNAD_ANO} Q{PNAD_TRIMESTRE}")
```

```
Arquivo PNAD encontrado localmente: pnad_2025q3.parquet
Carregado: 220,091 observações
  OK: Arquivo corresponde à configuração (2025 Q3)

df_pnad_raw: 220,091 linhas x 15 colunas
Período: 2025 Q3
```

2b. Verificar dados microdados PNAD (CHECKPOINT)

Verificar dados gerados

```
# Etapa 1.2b - Preparação de Dados - Verificar dados microdados PNAD
```

```python
print("=" * 60)
print("CHECKPOINT - Microdados PNAD")
print("=" * 60)

print(f"\nShape: {df_pnad_raw.shape}")
print(f"Colunas: {list(df_pnad_raw.columns)}")

# UFs
n_ufs = df_pnad_raw['sigla_uf'].nunique()
print(f"\nUFs presentes: {n_ufs}")
if n_ufs != 27:
    print(f"  WARNING: Esperado 27 UFs, encontrado {n_ufs}")

# População
pop_milhoes = df_pnad_raw['peso'].sum() / 1e6
print(f"População representada: {pop_milhoes:.1f} milhões")

# Linhas
if len(df_pnad_raw) < 100_000:
    print(f"  WARNING: Apenas {len(df_pnad_raw):,} linhas (esperado > 100.000)")

# Verificar preenchimento das variáveis-chave
print(f"\nPreenchimento das variáveis:")
for col in df_pnad_raw.columns:
    n_valid = df_pnad_raw[col].notna().sum()
    pct = n_valid / len(df_pnad_raw) * 100
    flag = "  " if pct > 80 else "  WARNING -" if pct > 50 else "  CRITICO -"
    print(f"{flag} {col}: {n_valid:,} ({pct:.1f}%)")

# Tipos
print(f"\nDtypes:\n{df_pnad_raw.dtypes}")

# Amostra
print("\nPrimeiras linhas:")
df_pnad_raw.head()
```

```
============================================================
CHECKPOINT - Microdados PNAD
============================================================

Shape: (220091, 15)
Colunas: ['ano', 'trimestre', 'sigla_uf', 'sexo', 'idade', 'raca_cor', 'nivel_instrucao', 'co
```

```
UFs presentes: 27
População representada: 102.4 milhões

Preenchimento das variáveis:
   ano: 220,091 (100.0%)
   trimestre: 220,091 (100.0%)
   sigla_uf: 220,091 (100.0%)
   sexo: 220,091 (100.0%)
   idade: 220,091 (100.0%)
   raca_cor: 220,091 (100.0%)
   nivel_instrucao: 220,091 (100.0%)
   cod_ocupacao: 220,091 (100.0%)
   grupamento_atividade: 220,091 (100.0%)
   posicao_ocupacao: 220,091 (100.0%)
   rendimento_habitual: 215,370 (97.9%)
   rendimento_efetivo: 215,405 (97.9%)
   horas_habituais: 220,091 (100.0%)
   horas_efetivas: 220,091 (100.0%)
   peso: 220,091 (100.0%)

Dtypes:
ano                      Int64
trimestre                Int64
sigla_uf                object
sexo                    object
idade                    Int64
raca_cor                object
nivel_instrucao         object
cod_ocupacao            object
grupamento_atividade    object
posicao_ocupacao        object
rendimento_habitual    float64
rendimento_efetivo     float64
horas_habituais          Int64
horas_efetivas           Int64
peso                   float64
dtype: object

Primeiras linhas:
```

	ano	trimestre	sigla_uf	sexo	idade	raca_cor	nivel_instrucao	cod_ocupacao	grupamento_ativ
0	2025	3	RR	1	47	4	4	8322	49030
1	2025	3	DF	2	35	1	5	4120	78000
2	2025	3	SE	1	62	4	2	5414	85012
3	2025	3	SE	2	40	4	5	5221	56011
4	2025	3	SE	2	34	4	5	5212	56020

3a. Processar índice de exposição ILO

Lê a planilha ILO com scores de exposição por ISCO-08, padroniza e gera níveis

```
# Etapa 1.3a - Preparação de Dados - Processar índice de exposição ILO

print(f"Lendo arquivo ILO: {ILO_FILE}")
df_ilo_raw = pd.read_excel(ILO_FILE)
print(f"Linhas raw (tarefas): {len(df_ilo_raw):,}")
print(f"Colunas disponíveis: {list(df_ilo_raw.columns)}")

# Mapeamento de colunas
col_mapping = {
    'ISCO_08': 'isco_08',
    'Title': 'occupation_title',
    'mean_score_2025': 'exposure_score',
    'SD_2025': 'exposure_sd',
    'potential25': 'exposure_gradient',
}

available_cols = [c for c in col_mapping.keys() if c in df_ilo_raw.columns]
print(f"Colunas mapeadas: {available_cols}")

df_ilo_renamed = df_ilo_raw.rename(
    columns={k: v for k, v in col_mapping.items() if k in df_ilo_raw.columns}
)

# Agregar por ocupação (arquivo original tem múltiplas tarefas por ocupação)
df_ilo = df_ilo_renamed.groupby('isco_08').agg({
    'occupation_title': 'first',
    'exposure_score': 'mean',
    'exposure_sd': 'mean',
    'exposure_gradient': 'first',
}).reset_index()
```

```
# Garantir formato string com 4 dígitos
df_ilo['isco_08_str'] = df_ilo['isco_08'].astype(str).str.zfill(4)

print(f"\nOcupações únicas: {len(df_ilo):,}")
print(f"Score médio: {df_ilo['exposure_score'].mean():.3f}")
print(f"Score range: [{df_ilo['exposure_score'].min():.3f}, {df_ilo['exposure_score'].max():

# Salvar processado
ilo_output = DATA_PROCESSED / "ilo_exposure_clean.csv"
df_ilo.to_csv(ilo_output, index=False)
print(f"\nSalvo em: {ilo_output}")
```

```
Lendo arquivo ILO: data/input/Final_Scores_ISCO08_Gmyrek_et_al_2025.xlsx
Linhas raw (tarefas): 3,265
Colunas disponíveis: ['label4d', 'label1d', 'ISCO_08', 'Title', 'taskID', 'Task_ISCO', 'score
Colunas mapeadas: ['ISCO_08', 'Title', 'mean_score_2025', 'SD_2025', 'potential25']

Ocupações únicas: 427
Score médio: 0.297
Score range: [0.090, 0.700]

Salvo em: data/processed/ilo_exposure_clean.csv
```

3b. Verificar índice de exposição ILO

Verificar: número de ocupações, coluna de score, distribuição por gradiente

```
# Etapa 1.3b - Preparação de Dados - Verificar índice de exposição ILO

print("=" * 60)
print("CHECKPOINT - Índice ILO")
print("=" * 60)

# Número de ocupações
n_ocup = len(df_ilo)
print(f"\nOcupações: {n_ocup}")
if n_ocup < 400:
    print(f"  WARNING: Poucas ocupações ({n_ocup}). Esperado ~427.")

# Range de scores
```

```python
score_min = df_ilo['exposure_score'].min()
score_max = df_ilo['exposure_score'].max()
print(f"Score range: [{score_min:.3f}, {score_max:.3f}]")
if score_min < 0 or score_max > 1:
    print(f"  WARNING: Scores fora do intervalo [0, 1]")

# Distribuição por gradiente
print("\nDistribuição por gradiente:")
for grad, count in df_ilo['exposure_gradient'].value_counts().items():
    print(f"  {grad}: {count} ocupações")

# Amostra
print("\nAmostra (5 maiores scores):")
df_ilo.nlargest(5, 'exposure_score')[['isco_08_str', 'occupation_title', 'exposure_score']]
```

```
============================================================
CHECKPOINT - Índice ILO
============================================================

Ocupações: 427
Score range: [0.090, 0.700]

Distribuição por gradiente:
  Not Exposed: 231 ocupações
  Minimal Exposure: 84 ocupações
  Exposed: Gradient 2: 44 ocupações
  Exposed: Gradient 3: 38 ocupações
  Exposed: Gradient 1: 17 ocupações
  Exposed: Gradient 4: 13 ocupações

Amostra (5 maiores scores):
```

	isco_08_str	occupation_title	exposure_score
207	4132	Data Entry Clerks	0.70
206	4131	Typists and Word Processing Operators	0.65
220	4311	Accounting and Bookkeeping Clerks	0.64
221	4312	Statistical, Finance and Insurance Clerks	0.64
164	3311	Securities and Finance Dealers and Brokers	0.63

Notas sobre terminologia

Sobre sexo: A PNADc coleta a variável V2007 (sexo biológico: masculino/feminino). Esta pesquisa não coleta identidade de gênero. Utilizamos o termo "sexo" ao longo desta análise, em conformidade com a terminologia do IBGE.

Sobre raça/cor: Utilizamos a variável V2010 (autoclassificação de cor ou raça) com as cinco categorias do IBGE: Branca, Preta, Parda, Amarela e Indígena. Quando apresentamos resultados agregados em "Negros" (Pretos + Pardos), seguimos a convenção amplamente adotada na sociologia e economia do trabalho brasileira (Osorio, 2003; Soares, 2008). Resultados desagregados estão disponíveis nos apêndices.

Sobre "exposição": O índice da OIT mede o potencial de que tarefas ocupacionais sejam afetadas pela IA generativa — seja por automação, seja por complementação/aumento de produtividade. "Exposição" não é sinônimo de "risco de desemprego" ou "ameaça". Ocupações altamente expostas podem tanto perder tarefas quanto ganhar produtividade, dependendo do contexto institucional, regulatório e organizacional.

4a. Limpeza e variáveis derivadas – PNAD

Filtra população de interesse, cria variáveis derivadas (região, grandes grupos COD, faixas de renda, etc.) e padroniza códigos de ocupação.

Entrada: `data/raw/pnad_*.parquet`.
Saída: `data/processed/pnad_clean.csv`

Nota metodológica — Inclusão de todos os ocupados: A análise de exposição inclui todos os ocupados com código de ocupação válido, **independentemente de terem renda declarada**. A variável `tem_renda` sinaliza trabalhadores com rendimento habitual positivo. Para análises de rendimento (tabelas salariais, faixas de renda), filtrar por `tem_renda == 1`.

Nota metodológica — Faixas de renda em salários mínimos: Optamos por classificar a renda em faixas de salários mínimos (até 1 SM, 1-2 SM, 2-3 SM, 3-5 SM, 5+ SM) em vez de quintis populacionais. Esta escolha se justifica por: (1) a elevada concentração de rendimentos em torno de 1 SM no Brasil gera empates que distorcem os quintis (Q1 absorveria ~31% da população); (2) faixas em SM são mais interpretáveis e amplamente utilizadas na literatura brasileira de economia do trabalho.

Nota metodológica — Winsorização: Aplicamos winsorização nos percentis 1 e 99 da distribuição de rendimento habitual, calculados com pesos amostrais (V1028), para limitar a influência de valores extremos preservando o tamanho amostral. Esta

técnica é preferível ao trimming (que descarta observações) e é prática padrão em análises de renda com dados de survey.

```
# Etapa 1.4a - Preparação de Dados - Limpeza e variáveis derivadas

df_pnad = df_pnad_raw.copy()
n_inicial = len(df_pnad)
print(f"Observações iniciais: {n_inicial:,}")

# ------------------------------------------------------------------------
# LIMPEZA - Conversão de tipos
# ------------------------------------------------------------------------
df_pnad['cod_ocupacao'] = df_pnad['cod_ocupacao'].astype(str).str.zfill(4)
df_pnad['idade'] = pd.to_numeric(df_pnad['idade'], errors='coerce')
df_pnad['rendimento_habitual'] = pd.to_numeric(df_pnad['rendimento_habitual'], errors='coerce
df_pnad['rendimento_efetivo'] = pd.to_numeric(df_pnad['rendimento_efetivo'], errors='coerce'
df_pnad['horas_habituais'] = pd.to_numeric(df_pnad['horas_habituais'], errors='coerce')
df_pnad['horas_efetivas'] = pd.to_numeric(df_pnad['horas_efetivas'], errors='coerce')
df_pnad['peso'] = pd.to_numeric(df_pnad['peso'], errors='coerce')

# ------------------------------------------------------------------------
# LIMPEZA - Filtros
# ------------------------------------------------------------------------

# Remover missings críticos (ocupação, idade, peso - NÃO renda)
df_pnad = df_pnad.dropna(subset=['cod_ocupacao', 'idade', 'peso'])
print(f"Após remover missings críticos: {len(df_pnad):,} ({len(df_pnad)/n_inicial:.1%})")

# Filtrar faixa etária (18-65)
df_pnad = df_pnad[(df_pnad['idade'] >= 18) & (df_pnad['idade'] <= 65)]
print(f"Após filtrar 18-65 anos: {len(df_pnad):,} ({len(df_pnad)/n_inicial:.1%})")

# Remover ocupações inválidas
df_pnad = df_pnad[~df_pnad['cod_ocupacao'].isin(['0000', '9999'])]
print(f"Após remover ocupações inválidas: {len(df_pnad):,}")

# ------------------------------------------------------------------------
# VARIÁVEIS DERIVADAS
# ------------------------------------------------------------------------

# Flag de renda (em vez de excluir sem renda)
df_pnad['tem_renda'] = (df_pnad['rendimento_habitual'].notna() & (df_pnad['rendimento_habitua
n_sem_renda = (df_pnad['tem_renda'] == 0).sum()
```

```
pop_sem_renda = df_pnad.loc[df_pnad['tem_renda'] == 0, 'peso'].sum() / 1e6
print(f"\nTrabalhadores sem renda declarada: {n_sem_renda:,} obs ({pop_sem_renda:.1f} milhões

# Formalidade
df_pnad['formal'] = df_pnad['posicao_ocupacao'].astype(str).isin(POSICAO_FORMAL).astype(int)
print(f"Taxa de formalidade: {df_pnad['formal'].mean():.1%}")

# Faixas etárias
df_pnad['faixa_etaria'] = pd.cut(
    df_pnad['idade'], bins=IDADE_BINS, labels=IDADE_LABELS
)

# Região
df_pnad['regiao'] = df_pnad['sigla_uf'].map(REGIAO_MAP)

# Raça agregada
df_pnad['raca_agregada'] = df_pnad['raca_cor'].astype(str).map(RACA_AGREGADA_MAP)

# Grande grupo ocupacional
df_pnad['grande_grupo'] = df_pnad['cod_ocupacao'].str[0].map(GRANDES_GRUPOS)

# Sexo como texto
df_pnad['sexo_texto'] = df_pnad['sexo'].map({1: 'Homem', 2: 'Mulher', '1': 'Homem', '2': 'Mul

# Winsorização de renda (percentis ponderados 1 e 99) - APENAS para quem tem renda
mask_renda = df_pnad['tem_renda'] == 1
p01 = weighted_quantile(
    df_pnad.loc[mask_renda, 'rendimento_habitual'],
    df_pnad.loc[mask_renda, 'peso'], 0.01
)
p99 = weighted_quantile(
    df_pnad.loc[mask_renda, 'rendimento_habitual'],
    df_pnad.loc[mask_renda, 'peso'], 0.99
)
df_pnad['rendimento_winsor'] = df_pnad['rendimento_habitual'].clip(lower=p01, upper=p99)
print(f"Winsorização ponderada: P1 = R$ {p01:,.0f}, P99 = R$ {p99:,.0f}")

# Faixas de renda em salários mínimos
df_pnad['faixa_renda_sm'] = pd.cut(
    df_pnad['rendimento_habitual'] / SALARIO_MINIMO,
    bins=[0, 1, 2, 3, 5, float('inf')],
    labels=['Até 1 SM', '1-2 SM', '2-3 SM', '3-5 SM', '5+ SM'],
```

```
    right=True,
    include_lowest=True,
)
print(f"\nDistribuição por faixa de renda (SM = R$ {SALARIO_MINIMO}):")
for faixa, peso in df_pnad[df_pnad['tem_renda'] == 1].groupby('faixa_renda_sm')['peso'].sum()
    pct = peso / df_pnad.loc[mask_renda, 'peso'].sum() * 100
    print(f"  {faixa}: {peso/1e6:.1f} milhões ({pct:.1f}%)")

# Verificar preenchimento de horas
n_horas_hab = df_pnad['horas_habituais'].notna().sum()
n_horas_efe = df_pnad['horas_efetivas'].notna().sum()
print(f"\nHoras habituais: {n_horas_hab:,} ({n_horas_hab/len(df_pnad):.1%})")
print(f"Horas efetivas:  {n_horas_efe:,} ({n_horas_efe/len(df_pnad):.1%})")

# ------------------------------------------------------------------------
# SALVAR
# ------------------------------------------------------------------------
pnad_clean_path = DATA_PROCESSED / "pnad_clean.csv"
df_pnad.to_csv(pnad_clean_path, index=False)
print(f"\nSalvo em: {pnad_clean_path}")
print(f"df_pnad: {df_pnad.shape[0]:,} linhas x {df_pnad.shape[1]} colunas")
```

```
Observações iniciais: 220,091
Após remover missings críticos: 220,091 (100.0%)
Após filtrar 18-65 anos: 207,919 (94.5%)
Após remover ocupações inválidas: 207,901

Trabalhadores sem renda declarada: 3,759 obs (1.1 milhões)
Taxa de formalidade: 37.0%
Winsorização ponderada: P1 = R$ 200, P99 = R$ 21,000

Distribuição por faixa de renda (SM = R$ 1518):
  Até 1 SM: 30.3 milhões (31.3%)
  1-2 SM: 39.0 milhões (40.4%)
  2-3 SM: 10.1 milhões (10.5%)
  3-5 SM: 9.5 milhões (9.8%)
  5+ SM: 7.8 milhões (8.0%)

Horas habituais: 207,901 (100.0%)
Horas efetivas:  207,901 (100.0%)

Salvo em: data/processed/pnad_clean.csv
```

```
df_pnad: 207,901 linhas x 24 colunas
```

4b. Verificar Limpeza e variáveis derivadas – PNAD

Verificar: número de linhas, colunas criadas, valores faltantes em COD.

```python
# Etapa 1.4b - Preparação de Dados - Verificar Limpeza e variáveis derivadas

print("=" * 60)
print("CHECKPOINT - Limpeza PNAD")
print("=" * 60)

# Perda de observações
pct_perda = 1 - len(df_pnad) / n_inicial
print(f"\nObservações: {n_inicial:,} -> {len(df_pnad):,} (perda: {pct_perda:.1%})")
if pct_perda > 0.20:
    print(f"  WARNING: Perda de {pct_perda:.1%} das observações (> 20%)")

# Missings em variáveis derivadas
for col in ['regiao', 'raca_agregada', 'grande_grupo', 'faixa_etaria', 'sexo_texto']:
    n_miss = df_pnad[col].isna().sum()
    if n_miss > 0:
        print(f"  WARNING: {col} tem {n_miss:,} valores faltantes")

print(f"\nOcupações únicas (COD): {df_pnad['cod_ocupacao'].nunique()}")
print(f"UFs: {df_pnad['sigla_uf'].nunique()}")
print(f"População representada: {df_pnad['peso'].sum()/1e6:.1f} milhões")

print("\nDistribuição por sexo:")
for sexo, peso in df_pnad.groupby('sexo_texto')['peso'].sum().items():
    print(f"  {sexo}: {peso/1e6:.1f} milhões")

print("\nDistribuição por região:")
for regiao, peso in df_pnad.groupby('regiao')['peso'].sum().sort_values(ascending=False).ite
    print(f"  {regiao}: {peso/1e6:.1f} milhões")

print("\nDistribuição por faixa etária:")
print(df_pnad['faixa_etaria'].value_counts().sort_index())

print("\nDistribuição por faixa de renda (SM):")
print(df_pnad['faixa_renda_sm'].value_counts().sort_index())
```

```
============================================================
CHECKPOINT - Limpeza PNAD
============================================================

Observações: 220,091 -> 207,901 (perda: 5.5%)
  WARNING: grande_grupo tem 1,671 valores faltantes

Ocupações únicas (COD): 428
UFs: 27
População representada: 97.8 milhões

Distribuição por sexo:
  Homem: 55.0 milhões
  Mulher: 42.8 milhões

Distribuição por região:
  Sudeste: 43.4 milhões
  Nordeste: 22.3 milhões
  Sul: 15.7 milhões
  Centro-Oeste: 8.5 milhões
  Norte: 7.8 milhões

Distribuição por faixa etária:
faixa_etaria
18-24    29595
25-34    48629
35-44    55588
45-54    46136
55+      27953
Name: count, dtype: int64

Distribuição por faixa de renda (SM):
faixa_renda_sm
Até 1 SM    74904
1-2 SM      77010
2-3 SM      19520
3-5 SM      18120
5+ SM       14588
Name: count, dtype: int64
```

5a. Crosswalk COD → ISCO-08

Mapear códigos de ocupação COD (PNAD) para ISCO-08 para permitir o merge com o índice ILO.

Estratégia de correspondência

A COD (Classificação de Ocupações para Pesquisas Domiciliares) do IBGE é derivada diretamente da ISCO-08 da OIT. Os códigos compartilham a mesma estrutura hierárquica de 4 dígitos, com o primeiro dígito representando os mesmos 9 grandes grupos ocupacionais (Fonte: IBGE, Nota Técnica COD 2010). Isso permite um match direto de string entre COD e ISCO-08 na maioria dos casos.

Adotamos uma estratégia de correspondência hierárquica para maximizar a cobertura:

1. **4 dígitos (exato):** match direto COD ISCO-08. Cobre ~98% das observações.
2. **3 dígitos (subgrupo):** para códigos COD sem equivalente exato na ISCO-08, atribui-se a média do subgrupo (3 primeiros dígitos). Cobre ~1-2% adicional.
3. **2 dígitos (grupo menor):** fallback para o grupo de 2 dígitos.
4. **1 dígito (grande grupo):** fallback final para o grande grupo ocupacional.

Limitação: Não há validação semântica título-a-título; possíveis "falsos cognatos numéricos" são mitigados pelos sanity checks por grande grupo (verificação de que a ordenação de exposição por grande grupo é coerente com a literatura).

```
# Etapa 1.5a - Preparação de Dados - Crosswalk COD → ISCO-08

# Garantir formatos string
df_ilo['isco_08_str'] = df_ilo['isco_08_str'].astype(str).str.zfill(4)
df_pnad['cod_ocupacao'] = df_pnad['cod_ocupacao'].astype(str).str.zfill(4)

print(f"PNAD: {len(df_pnad):,} observações")
print(f"ILO:  {len(df_ilo):,} ocupações ISCO-08")

# ------------------------------------------------------------------------
# Criar dicionários de lookup em cada nível hierárquico
# ------------------------------------------------------------------------
ilo_4d = df_ilo.groupby('isco_08_str')['exposure_score'].mean().to_dict()
ilo_3d = df_ilo.groupby(df_ilo['isco_08_str'].str[:3])['exposure_score'].mean().to_dict()
ilo_2d = df_ilo.groupby(df_ilo['isco_08_str'].str[:2])['exposure_score'].mean().to_dict()
ilo_1d = df_ilo.groupby(df_ilo['isco_08_str'].str[:1])['exposure_score'].mean().to_dict()

# Lookup do gradiente oficial ILO (potential25) - apenas para match 4-digit
```

```
ilo_gradient_4d = df_ilo.groupby('isco_08_str')['exposure_gradient'].first().to_dict()

print(f"\nCódigos ILO: 4d={len(ilo_4d)}, 3d={len(ilo_3d)}, 2d={len(ilo_2d)}, 1d={len(ilo_1d)}

# ------------------------------------------------------------------------
# Crosswalk hierárquico (4 → 3 → 2 → 1 dígito)
# ------------------------------------------------------------------------
df_crosswalked = df_pnad.copy()
df_crosswalked['exposure_score'] = np.nan
df_crosswalked['exposure_gradient'] = None
df_crosswalked['match_level'] = None

# Nível 4-digit
mask_4d = df_crosswalked['cod_ocupacao'].isin(ilo_4d.keys())
df_crosswalked.loc[mask_4d, 'exposure_score'] = df_crosswalked.loc[mask_4d, 'cod_ocupacao'].m
df_crosswalked.loc[mask_4d, 'exposure_gradient'] = df_crosswalked.loc[mask_4d, 'cod_ocupacao
df_crosswalked.loc[mask_4d, 'match_level'] = '4-digit'
print(f"\nMatch 4-digit: {mask_4d.sum():,} ({mask_4d.mean():.1%})")

# Nível 3-digit
mask_missing = df_crosswalked['exposure_score'].isna()
cod_3d = df_crosswalked.loc[mask_missing, 'cod_ocupacao'].str[:3]
mask_3d = cod_3d.isin(ilo_3d.keys())
idx_3d = mask_missing[mask_missing].index[mask_3d.values]
df_crosswalked.loc[idx_3d, 'exposure_score'] = cod_3d[mask_3d].map(ilo_3d).values
df_crosswalked.loc[idx_3d, 'exposure_gradient'] = 'Sem classificação'
df_crosswalked.loc[idx_3d, 'match_level'] = '3-digit'
print(f"Match 3-digit: {len(idx_3d):,} ({len(idx_3d)/len(df_crosswalked):.1%})")

# Nível 2-digit
mask_missing = df_crosswalked['exposure_score'].isna()
cod_2d = df_crosswalked.loc[mask_missing, 'cod_ocupacao'].str[:2]
mask_2d = cod_2d.isin(ilo_2d.keys())
idx_2d = mask_missing[mask_missing].index[mask_2d.values]
df_crosswalked.loc[idx_2d, 'exposure_score'] = cod_2d[mask_2d].map(ilo_2d).values
df_crosswalked.loc[idx_2d, 'exposure_gradient'] = 'Sem classificação'
df_crosswalked.loc[idx_2d, 'match_level'] = '2-digit'
print(f"Match 2-digit: {len(idx_2d):,} ({len(idx_2d)/len(df_crosswalked):.1%})")

# Nível 1-digit
mask_missing = df_crosswalked['exposure_score'].isna()
cod_1d = df_crosswalked.loc[mask_missing, 'cod_ocupacao'].str[:1]
```

```
mask_1d = cod_1d.isin(ilo_1d.keys())
idx_1d = mask_missing[mask_missing].index[mask_1d.values]
df_crosswalked.loc[idx_1d, 'exposure_score'] = cod_1d[mask_1d].map(ilo_1d).values
df_crosswalked.loc[idx_1d, 'exposure_gradient'] = 'Sem classificação'
df_crosswalked.loc[idx_1d, 'match_level'] = '1-digit'
print(f"Match 1-digit: {len(idx_1d):,} ({len(idx_1d)/len(df_crosswalked):.1%})")

# Sem match
n_sem_match = df_crosswalked['exposure_score'].isna().sum()
df_crosswalked.loc[df_crosswalked['exposure_score'].isna(), 'exposure_gradient'] = 'Sem class
print(f"Sem match:     {n_sem_match:,} ({n_sem_match/len(df_crosswalked):.1%})")
```

```
PNAD: 207,901 observações
ILO:  427 ocupações ISCO-08

Códigos ILO: 4d=427, 3d=127, 2d=40, 1d=9

Match 4-digit: 203,617 (97.9%)
Match 3-digit: 2,613 (1.3%)
Match 2-digit: 0 (0.0%)
Match 1-digit: 0 (0.0%)
Sem match:     1,671 (0.8%)
```

5a. Verificar Crosswalk COD → ISCO-08

Verificar: cobertura do crosswalk (percentual de linhas com ISCO preenchido).

```
# Etapa 1.5b - Preparação de Dados - Verificar Crosswalk COD → ISCO-08

print("=" * 60)
print("CHECKPOINT - Crosswalk COD → ISCO-08")
print("=" * 60)

# Cobertura total
coverage = df_crosswalked['exposure_score'].notna().mean()
print(f"\nCobertura total: {coverage:.1%}")
if coverage < 0.90:
    print(f"  WARNING: Cobertura {coverage:.1%} abaixo de 90%")

# Distribuição por nível de match
print("\nDistribuição por nível de match:")
```

```
for level, count in df_crosswalked['match_level'].value_counts().items():
    pct = count / len(df_crosswalked) * 100
    print(f"  {level}: {count:,} ({pct:.1f}%)")

# FIX 3: Verificar concentração em match genérico (fallback)
n_total = len(df_crosswalked)
n_generic = df_crosswalked['match_level'].isin(['1-digit', '2-digit']).sum()
pct_generic = n_generic / n_total * 100
print(f"\nMatch genérico (1-digit + 2-digit): {n_generic:,} ({pct_generic:.1f}%)")
if pct_generic > 5:
    print(f"  WARNING: {pct_generic:.1f}% caiu em match genérico (>5%). "
          "Scores podem não refletir a ocupação real.")
else:
    print(f"  OK: Apenas {pct_generic:.1f}% em match genérico.")

# Estatísticas de score
print(f"\nEstatísticas do exposure_score:")
print(f"  Média:  {df_crosswalked['exposure_score'].mean():.3f}")
print(f"  Std:    {df_crosswalked['exposure_score'].std():.3f}")
print(f"  Min:    {df_crosswalked['exposure_score'].min():.3f}")
print(f"  Max:    {df_crosswalked['exposure_score'].max():.3f}")

# Sanity check: exposição por grande grupo
print("\nExposição média por grande grupo (sanity check):")
exp_grupos = df_crosswalked.groupby('grande_grupo').apply(
    lambda x: weighted_mean(x['exposure_score'].dropna(), x.loc[x['exposure_score'].notna(),
).sort_values(ascending=False)

for grupo, score in exp_grupos.items():
    print(f"  {grupo}: {score:.3f}")

# Validações de sanidade
print("\nVALIDAÇÃO DE SANIDADE:")
if 'Profissionais das ciências' in exp_grupos.index:
    val = exp_grupos['Profissionais das ciências']
    if val > 0.30:
        print(f"  OK - Profissionais das ciências com exposição ALTA ({val:.3f})")
    else:
        print(f"  WARNING: Profissionais das ciências com exposição BAIXA ({val:.3f}). Espera

if 'Ocupações elementares' in exp_grupos.index:
    val = exp_grupos['Ocupações elementares']
```

```
    if val < 0.20:
        print(f"  OK - Ocupações elementares com exposição BAIXA ({val:.3f})")
    else:
        print(f"  WARNING: Ocupações elementares com exposição ALTA ({val:.3f}). Esperado < 0
```

```
============================================================
CHECKPOINT - Crosswalk COD → ISCO-08
============================================================

Cobertura total: 99.2%

Distribuição por nível de match:
  4-digit: 203,617 (97.9%)
  3-digit: 2,613 (1.3%)

Match genérico (1-digit + 2-digit): 0 (0.0%)
  OK: Apenas 0.0% em match genérico.

Estatísticas do exposure_score:
  Média:  0.265
  Std:    0.144
  Min:    0.090
  Max:    0.700

Exposição média por grande grupo (sanity check):
  Apoio administrativo: 0.554
  Dirigentes e gerentes: 0.400
  Profissionais das ciências: 0.353
  Técnicos nível médio: 0.345
  Serviços e vendedores: 0.305
  Operadores de máquinas: 0.223
  Agropecuária qualificada: 0.174
  Indústria qualificada: 0.151
  Ocupações elementares: 0.130

VALIDAÇÃO DE SANIDADE:
  OK - Profissionais das ciências com exposição ALTA (0.353)
  OK - Ocupações elementares com exposição BAIXA (0.130)
```

6. Merge final – PNAD + índice ILO

Juntar a base PNAD (com ISCO-08) ao índice ILO por código de ocupação. Gera a base analítica final da Etapa 1. **Saída:** `data/output/pnad_ilo_merged.csv`

Classificação de exposição

Utilizamos a classificação oficial do WP140 da OIT (Tabela 5), que categoriza ocupações em 6 níveis de exposição com base em critérios bivariados — a média ( ) e o desvio-padrão ( ) dos scores de tarefa:

Categoria	Descrição
Not Exposed	Exposição negligenciável
Minimal Exposure	Exposição mínima
Gradient 1	Exposição baixa (alto potencial de aumento de produtividade)
Gradient 2	Exposição moderada-baixa
Gradient 3	Exposição moderada-alta
Gradient 4	Exposição alta (maior potencial de automação)

Esta classificação vem pré-computada na coluna `potential25` do dataset publicado pela OIT (Gmyrek, Berg & Cappelli, 2025). Verificamos a consistência reproduzindo a lógica bivariada da Tabela 5, obtendo 99,1% de concordância (423/427 ocupações). Os 4 mismatches são casos de fronteira.

Para ocupações com match hierárquico (3 dígitos), onde não há classificação individual disponível, atribuímos a categoria "Sem classificação". O score numérico (`exposure_score`) permanece disponível para essas ocupações.

```
# Etapa 1.6 - Preparação de Dados - Merge final PNAD + ILO

df_final = df_crosswalked.copy()

# ------------------------------------------------------------------------
# Checkpoint de qualidade do merge
# ------------------------------------------------------------------------
n_com_score = df_final['exposure_score'].notna().sum()
n_sem_score = df_final['exposure_score'].isna().sum()
pct_pop_perdida = df_final.loc[df_final['exposure_score'].isna(), 'peso'].sum() / df_final['

print("=" * 60)
```

```
print("CHECKPOINT - Qualidade do Merge")
print("=" * 60)
print(f"Observações totais:     {len(df_final):,}")
print(f"Com score de exposição: {n_com_score:,}")
print(f"Sem score (NaN):        {n_sem_score:,}")
print(f"% pop. sem score:       {pct_pop_perdida:.1f}%")

# ------------------------------------------------------------------------
# Distribuição por gradiente ILO (potential25)
# ------------------------------------------------------------------------
print("\nDistribuição por gradiente ILO (potential25):")
for grad, peso in df_final.groupby('exposure_gradient')['peso'].sum().sort_values(ascending=
    print(f"  {grad}: {peso/1e6:.1f} milhões")

# ------------------------------------------------------------------------
# Quintis e decis de EXPOSIÇÃO - ponderados por peso amostral
# Usa weighted_qcut (definida na célula 1) para que cada faixa represente
# ~20% (quintis) ou ~10% (decis) da POPULAÇÃO, não da amostra.
# ------------------------------------------------------------------------
mask_valid = df_final['exposure_score'].notna()

df_final.loc[mask_valid, 'quintil_exposure'] = weighted_qcut(
    df_final.loc[mask_valid, 'exposure_score'],
    df_final.loc[mask_valid, 'peso'],
    q=5,
    labels=['Q1 (Baixa)', 'Q2', 'Q3', 'Q4', 'Q5 (Alta)'],
)

df_final.loc[mask_valid, 'decil_exposure'] = weighted_qcut(
    df_final.loc[mask_valid, 'exposure_score'],
    df_final.loc[mask_valid, 'peso'],
    q=10,
    labels=[f'D{i}' for i in range(1, 11)],
)

# Verificar distribuição populacional dos quintis de exposição
print("\nPopulação por quintil de exposição (deve ser ~20% cada):")
for q, peso in df_final.groupby('quintil_exposure')['peso'].sum().items():
    pct = peso / df_final.loc[mask_valid, 'peso'].sum() * 100
    print(f"  {q}: {peso/1e6:.1f} milhões ({pct:.1f}%)")

# ------------------------------------------------------------------------
```

```
# Agregação setorial - CNAE Domiciliar 2.0, Seções A-T (IBGE)
# Usa CNAE_SETOR_MAP definido na célula de configuração.
# ------------------------------------------------------------------------
df_final['cnae_2d'] = df_final['grupamento_atividade'].astype(str).str[:2]
df_final['setor_agregado'] = df_final['cnae_2d'].map(CNAE_SETOR_MAP).fillna('Outros Serviços

# Flag de setores críticos para IA
# Ref: Gmyrek et al. (2024); Eloundou et al. (2023)
df_final['setor_critico_ia'] = df_final['setor_agregado'].isin(SETORES_CRITICOS_IA).astype(in

print(f"\nSetores: {df_final['setor_agregado'].nunique()} categorias")
print(f"Trabalhadores em setores críticos IA: {df_final.loc[df_final['setor_critico_ia']==1,

print("\nDistribuição por setor:")
for setor, peso in df_final.groupby('setor_agregado')['peso'].sum().sort_values(ascending=Fal
    flag = " *" if setor in SETORES_CRITICOS_IA else ""
    print(f"  {setor}: {peso/1e6:.1f} milhões{flag}")

# ------------------------------------------------------------------------
# Selecionar colunas finais e salvar
# ------------------------------------------------------------------------
cols_output = [
    'ano', 'trimestre', 'sigla_uf', 'regiao',
    'sexo', 'sexo_texto', 'idade', 'faixa_etaria',
    'raca_cor', 'raca_agregada', 'nivel_instrucao',
    'cod_ocupacao', 'grande_grupo',
    'grupamento_atividade', 'setor_agregado', 'setor_critico_ia',
    'posicao_ocupacao', 'formal', 'tem_renda',
    'rendimento_habitual', 'rendimento_winsor', 'rendimento_efetivo',
    'horas_habituais', 'horas_efetivas',
    'faixa_renda_sm',
    'peso',
    'exposure_score', 'exposure_gradient', 'match_level',
    'quintil_exposure', 'decil_exposure',
]

df_final = df_final[[c for c in cols_output if c in df_final.columns]]

output_path = DATA_OUTPUT / "pnad_ilo_merged.csv"
df_final.to_csv(output_path, index=False)

# ------------------------------------------------------------------------
```

```python
# Resumo final
# ------------------------------------------------------------------------
print(f"\n{'=' * 60}")
print("BASE FINAL CONSOLIDADA")
print(f"{'=' * 60}")
print(f"Observações:       {len(df_final):,}")
print(f"Com score:         {df_final['exposure_score'].notna().sum():,}")
print(f"Cobertura:         {df_final['exposure_score'].notna().mean():.1%}")
print(f"Colunas:           {df_final.shape[1]}")
print(f"População total:   {df_final['peso'].sum()/1e6:.1f} milhões")
print(f"  com renda:       {df_final.loc[df_final['tem_renda']==1, 'peso'].sum()/1e6:.1f} mi
print(f"  sem renda:       {df_final.loc[df_final['tem_renda']==0, 'peso'].sum()/1e6:.1f} mi
print(f"Setores:           {df_final['setor_agregado'].nunique()} categorias")
print(f"Setor crítico IA:  {df_final['setor_critico_ia'].sum():,} obs")
print(f"Salvo em:          {output_path}")
print(f"Tamanho em disco:  {output_path.stat().st_size / 1e6:.1f} MB")

df_final.info()
```

```
============================================================
CHECKPOINT - Qualidade do Merge
============================================================
Observações totais:     207,901
Com score de exposição: 206,230
Sem score (NaN):        1,671
% pop. sem score:       0.8%

Distribuição por gradiente ILO (potential25):
  Not Exposed: 52.2 milhões
  Minimal Exposure: 15.2 milhões
  Exposed: Gradient 2: 10.0 milhões
  Exposed: Gradient 1: 8.7 milhões
  Exposed: Gradient 4: 5.0 milhões
  Exposed: Gradient 3: 4.8 milhões
  Sem classificação: 1.8 milhões

População por quintil de exposição (deve ser ~20% cada):
  Q1 (Baixa): 20.8 milhões (21.5%)
  Q2: 18.1 milhões (18.7%)
  Q3: 21.4 milhões (22.0%)
  Q4: 17.4 milhões (17.9%)
  Q5 (Alta): 19.3 milhões (19.9%)
```

```
Setores: 17 categorias
Trabalhadores em setores críticos IA: 7.8 milhões

Distribuição por setor:
  Outros Serviços: 19.1 milhões
  Ind. Transformação: 18.7 milhões
  Construção: 7.2 milhões
  Educação: 7.2 milhões
  Saúde: 6.2 milhões
  Transporte: 5.7 milhões
  Serviços Domésticos: 5.3 milhões
  Alojamento e Alimentação: 5.1 milhões
  Administração Pública: 5.0 milhões
  Serviços Administrativos: 4.6 milhões
  Serviços Profissionais: 4.3 milhões *
  Comércio: 3.2 milhões
  Informação e Comunicação: 1.9 milhões *
  Finanças e Seguros: 1.6 milhões *
  Artes e Cultura: 1.2 milhões
  Utilidades: 0.7 milhões
  Atividades Imobiliárias: 0.7 milhões

============================================================
BASE FINAL CONSOLIDADA
============================================================
Observações:       207,901
Com score:         206,230
Cobertura:         99.2%
Colunas:           31
População total:   97.8 milhões
  com renda:       96.7 milhões
  sem renda:       1.1 milhões
Setores:           17 categorias
Setor crítico IA:  13,427 obs
Salvo em:          data/output/pnad_ilo_merged.csv
Tamanho em disco:  40.0 MB
<class 'pandas.core.frame.DataFrame'>
Index: 207901 entries, 0 to 220090
Data columns (total 31 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   ano                  207901 non-null  Int64
```

```
 1   trimestre             207901 non-null  Int64
 2   sigla_uf              207901 non-null  object
 3   regiao                207901 non-null  object
 4   sexo                  207901 non-null  object
 5   sexo_texto            207901 non-null  object
 6   idade                 207901 non-null  Int64
 7   faixa_etaria          207901 non-null  category
 8   raca_cor              207901 non-null  object
 9   raca_agregada         207901 non-null  object
 10  nivel_instrucao       207901 non-null  object
 11  cod_ocupacao          207901 non-null  object
 12  grande_grupo          206230 non-null  object
 13  grupamento_atividade  207901 non-null  object
 14  setor_agregado        207901 non-null  object
 15  setor_critico_ia      207901 non-null  int64
 16  posicao_ocupacao      207901 non-null  object
 17  formal                207901 non-null  int64
 18  tem_renda             207901 non-null  int64
 19  rendimento_habitual   204142 non-null  float64
 20  rendimento_winsor     204142 non-null  float64
 21  rendimento_efetivo    204171 non-null  float64
 22  horas_habituais       207901 non-null  Int64
 23  horas_efetivas        207901 non-null  Int64
 24  faixa_renda_sm        204142 non-null  category
 25  peso                  207901 non-null  float64
 26  exposure_score        206230 non-null  float64
 27  exposure_gradient     207901 non-null  object
 28  match_level           206230 non-null  object
 29  quintil_exposure      206230 non-null  category
 30  decil_exposure        206230 non-null  category
dtypes: Int64(5), category(4), float64(5), int64(3), object(14)
memory usage: 54.3+ MB
```

Limitações desta etapa

1. **Crosswalk hierárquico:** O match a 3 dígitos (subgrupo) suaviza diferenças entre ocupações dentro do mesmo subgrupo, afetando ~1,3% das observações. O score atribuído é a média do subgrupo, não o score específico da ocupação.
2. **Trimestre único:** Os dados referem-se a um único trimestre (Q3/2025). Resultados podem variar sazonalmente, especialmente em setores com forte sazonalidade (agropecuária, comércio).

3. **Variáveis indisponíveis:** As variáveis de tempo no emprego (`V4040`) e porte da empresa (`V4018`) não estão populadas na fonte utilizada (Base dos Dados/BigQuery) para o período analisado, limitando análises de estabilidade ocupacional e adoção de IA por tamanho de empresa.

4. **Índice global aplicado ao Brasil:** O índice da OIT foi desenvolvido com foco global e pode não capturar especificidades do mercado de trabalho brasileiro, como a elevada informalidade (~40% da força de trabalho) e diferenças na adoção tecnológica entre setores formais e informais.

5. **Exposição impacto:** O índice mede potencial de exposição das tarefas à IA generativa, não o impacto efetivo. A materialização do impacto depende de fatores como velocidade de adoção tecnológica, regulação, custos de implementação e respostas institucionais.