D:\NHRI\AMR\amr-main\explore\pathogen_paper\mcod_by_pathogen.py

For gbd adjustments

```python
import pandas as pd
import numpy as np
import dask.dataframe as dd

from cod_prep.downloaders import get_current_cause_hierarchy, add_location_metadata,
get_current_location_hierarchy, prep_child_to_available_parent_map, get_all_child_cause_ids,
add_age_metadata, prep_child_to_available_parent_loc
from mcod_prep.utils.mcause_io import *
from mcod_prep.utils.nids import add_nid_metadata
from mcod_prep.utils.causes import (
    get_all_related_syndromes,
    get_child_to_available_parent_syndrome,
    get_infsyn_hierarchy,
)
from amr_prep.utils.amr_io import *
from db_tools import ezfuncs, query_tools
from db_queries import get_outputs as go

CONF = Configurator()
ch = get_current_cause_hierarchy()
lh = get_current_location_hierarchy()

ph = pd.read_csv("FILE_PATH")

all_u5 = [2,3,42,1,388,34,238,389,5,4]
gbd2019_u5 = [2,3,4,5]
o5 = list(range(6,21)) + [30,31,32,235]

npartitions = 4


def add_pathogen_metadata(df, path_meta):
    assert "pathogen" in df.columns, "no pathogen information to add names to"
    assert "pathogen_name" not in df.columns, "pathogen name is already here"

    df = df.merge(path_meta[["pathogen", "pathogen_name_long"]], how="left", on="pathogen",
validate="many_to_one")
    df["pathogen_name_long"] = df["pathogen_name_long"].fillna(df["pathogen"])
    df.drop("pathogen", axis=1, inplace=True)
    df.rename(columns={"pathogen_name_long":"pathogen"}, inplace=True)

    return df


def fix_ages(df, gbd_2019=False, make_all=True):
    if gbd_2019==False:
        df.loc[df["age_group_id"].isin(all_u5), "age_group_id"] = 1
        df.loc[~df["age_group_id"].isin(all_u5+[22]), "age_group_id"] = 192
    else:
        df.loc[df["age_group_id"].isin(gbd2019_u5), "age_group_id"] = 1
        df.loc[df["age_group_id"].isin(o5), "age_group_id"] = 192

    df_u5 = df.loc[df["age_group_id"] == 1]
    df_o5 = df.loc[df["age_group_id"] == 192]

    if make_all == True:
        assert df["age_group_id"].isin([1,192]).values.all()
        df["age_group_id"] = 22
    else:
        df = df.loc[df["age_group_id"] == 22]

    df = pd.concat([df, df_u5, df_o5])
    df = add_age_metadata(df, "age_group_name")

    return df


years = list(range(1980, 2022))
locs = [1]

path_assignment_fix = {
       "actinomycosis": "other",
        "aerobacter_aerogenes": "klebsiella_spp",
        'bacillus_anthracis': 'contaminant',
        'bacteroides_fragilis': 'anaerobe',
        'bartonella': 'contaminant',
        'borrelia_recurrentis': 'other',
        'brucella':"other",
        'coronaviruses':"virus",
        'corynebacterium_diphtheriae':"contaminant",
        'cyclosporiasis':'parasite',
        'erysipelothrix_rhusiopathiae': "other",
        'francisella_tularensis':"other",
        'giardiasis':'parasite',
        'haemophilus_ducreyi': 'contaminant',
         'isosporiasis':'parasite',
        'klebsiella_granulomatis':'klebsiella_spp',
        'parahaemolyticus': "other",
        'parainfluenza_viruses':"virus",
        'rhinoviruses':"virus",
        'rickettsias_spp':'other',
        'salmonella_enterica':"non_typhoidal_salmonellae",
        'treponema_carateum':"other",
        'treponema_pallidum':"other",
        'yersinia_enterocolitica':"other",
        'yersinia_pestis':"other",
        'gram_negative_other': 'other',
        'anaerobe': 'other',
        'pseudomonas_spp': 'other',
        'streptococcus_unsp': 'other',
        'clostridium_perfringens': 'other',
        'mycobacterium_non_tb': 'other',
        'burkholderia_spp':'other',
        'clostridium_spp':'other',
        'helicobacter':'other',
        "treponema_pallidum": "other",
        "burkholderia_pseudomallei": "other",
        "clostridium_botulinum": "other"
    }

untracked = list(path_assignment_fix.keys())

coi = ["hiv",
       "tb",
      "malaria",
      "hepatitis_b",
      "hepatitic_c",
      "whooping",
      "std_syphilis",
      "measles",
      "ntd_dengue",
      "hepatitis_a",
      "tetanus",
      "ntd_schisto",
      "ntd_lf",
      "ntd_cysticer",
      "ntd_oncho",
      "ntd_nema_hook",
      "varicella",
      "ntd_rabies",
      "ntd_foodborne",
      "ntd_nema_ascar",
      "ntd_leish_visc",
       "diptheria",
      "ntd_leish_cut",
      "ntd_yellowfever",
      "std_tricho",
      "ntd_chagas",
      "std_herpes",
      "ntd_nema_trichur",
      "hepatitis_e",
      "ntd_ebola",
      "ntd_trachoma",
      "ntd_echino",
      "ntd_afrtryp",
      "leprosy",
      "ntd_zika",
      "ntd_guinea",
      "neo_cervical",
      "cirrhosis_hepb",
       "cirrhosis_hepc",
       "neo_liver_hepb",
       "neo_liver_hepc",
       "_infect",
       "ntd_other",
      ]


coi_id = ch.loc[ch["acause"].isin(coi), "cause_id"].unique().tolist()

```

```python
included_causes = []
for cause in coi_id:
    children = get_all_child_cause_ids(cause, ch)
    included_causes = included_causes + children

child_causes = [x for x in included_causes if x not in coi_id]
get_parent_map = prep_child_to_available_parent_map(ch.loc[ch["cause_id"].isin(included_causes)], coi_id, as_dict=True)

available_causes = [295, 687, 409]
ncode = [896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 894, 914, 915, 916, 917, 918, 919, 920, 294, 887, 994, 889, 860, 861, 862, 863, 864, 865, 866, 867, 740, 868, 869, 743, 744, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 888, 891, 892, 893, 890, 895]
ch = ch.loc[~ch["cause_id"].isin(ncode)]
cause_map = prep_child_to_available_parent_map(ch, available_causes,as_dict=True)


by_ucod_all = []
by_dis_all = []
for year in years:
    print(f"Working on year {year}")
    print("Reading data")
    mcod = get_mcause_data(
    phase="format_map",
    project_id=1,
    is_active=True,
    year_id = year,
    sub_dirs="pathogen",
    )

    mcod = add_nid_metadata(mcod, "source")
    print(mcod["source"].unique().tolist())
    mcod = mcod.loc[mcod["age_group_id"] != 283]

    ID_path = ["cause_id", "year_id", "deaths", "age_group_id"]
    path_value_vars = [x for x in mcod.columns if "multiple" in x and "etiology" in x]

    cause_etiology_cols = [x for x in mcod.columns if "cause" in x and "etiology" in x]
    print(f"Fixing etiologies in {cause_etiology_cols}")
    for col in cause_etiology_cols:
        mcod.loc[mcod[col].isin(untracked), col] = mcod.loc[mcod[col].isin(untracked), col].map(path_assignment_fix)

    ID_dis = ["cause_id", "year_id", "deaths", "age_group_id"]
    dis_value_vars = [x for x in mcod.columns if "multiple" in x and "disease" in x]

    cause_disease_cols = [x for x in mcod.columns if "cause" in x and "disease" in x]
    print(f"Fixing child causes in {cause_disease_cols}")
    for col in cause_disease_cols:
        mcod.loc[(mcod[col].notnull()) & (mcod[col] != "0000"),col] = mcod.loc[(mcod[col].notnull()) & (mcod[col] != "0000"),col].astype(float).astype(int)
        mcod.loc[mcod[col].isin(child_causes), col] = mcod.loc[mcod[col].isin(child_causes), col].map(get_parent_map)

    print("Convert to dask dataframe!")
    mcod = dd.from_pandas(mcod, npartitions=npartitions)

    print(f"Melting pathogens")
    path = dd.melt(mcod, id_vars=ID_path, value_vars = path_value_vars, var_name = "chain_position",
                      value_name="pathogen")
    path = path.compute()
    path["pathogen"].fillna("0000", inplace=True)

    path.loc[path["pathogen"].isin(untracked), "pathogen"] = path.loc[path["pathogen"].isin(untracked), "pathogen"].map(path_assignment_fix)
    path = path.loc[~path["pathogen"].isin(["0000", 0, "contaminant", "parasite"])]

    print(f"Melting diseases")
    dis = dd.melt(mcod, id_vars=ID_dis, value_vars = dis_value_vars, var_name = "chain_position",
                      value_name="chain_cause").compute()
    dis["chain_cause"].fillna("0000", inplace=True)
    dis = dis.loc[~dis["chain_cause"].isin(["0000", 0])]

    print(f"Fixing ages & Grouping")
    by_ucod = path.copy()
    by_ucod["cause_id"] = path["cause_id"].map(cause_map)
    by_ucod = fix_ages(by_ucod)
    by_ucod = by_ucod.groupby(["cause_id","age_group_name", "pathogen"], as_index=False)["deaths"].sum()

    by_dis = dis.copy()
    by_dis = dis.loc[dis["chain_cause"].isin(coi_id)]
    by_dis = fix_ages(by_dis)
    by_dis["cause_id"] = by_dis["cause_id"].map(cause_map)
    by_dis = by_dis.groupby(["cause_id", "age_group_name", "chain_cause"], as_index=False)["deaths"].sum()

    print(f"Adding metadata")
    by_ucod = by_ucod.merge(ch[["cause_id", "cause_name"]], how="left", on="cause_id", validate="many_to_one")
    by_ucod = add_pathogen_metadata(by_ucod, ph)

    by_dis = by_dis.merge(ch[["cause_id", "cause_name"]], how="left", on="cause_id", validate="many_to_one")
    mcod_map = dict(list(zip(ch["cause_id"], ch["cause_name"])))
    by_dis["pathogen"] = by_dis["chain_cause"].map(mcod_map)

    print(f"Adding UCoD")
    mcod = mcod.compute()
    path_ucod_count = mcod.loc[~mcod["cause_etiology"].isin(["0000", 0, "contaminant", "parasite"])]

    path_ucod_count = fix_ages(path_ucod_count)

    path_ucod_count = path_ucod_count.groupby(["cause_etiology", "age_group_name"], as_index=False)["deaths"].sum()
    path_ucod_count["cause_name"] = "Deaths where UCoD"
    path_ucod_count.rename(columns={"cause_etiology":"pathogen"}, inplace=True)
    path_ucod_count = add_pathogen_metadata(path_ucod_count, ph)

    dis_ucod_count = mcod.loc[mcod["cause_disease"].isin(coi_id)]
    dis_ucod_count = fix_ages(dis_ucod_count)

    dis_ucod_count = dis_ucod_count.groupby(["cause_disease", "age_group_name"], as_index=False)["deaths"].sum()
    dis_ucod_count["cause_name"] = "Deaths where UCoD"
    dis_ucod_count["pathogen"] = dis_ucod_count["cause_disease"].map(mcod_map)

    print(f"Merging on UCoD to original df")
    by_ucod = pd.concat([by_ucod, path_ucod_count])
    by_dis = pd.concat([by_dis, dis_ucod_count])

    by_ucod_all.append(by_ucod)
    by_dis_all.append(by_dis)

print("Concatenating & Pivoting")
by_ucod = pd.concat(by_ucod_all)
by_ucod = by_ucod.groupby(["age_group_name", "cause_name", "pathogen"], as_index=False)["deaths"].sum()
by_dis = pd.concat(by_dis_all)

hepb = ["Liver cancer due to hepatitis B",
        "Acute hepatitis B",
       "Cirrhosis and other chronic liver diseases due to hepatitis B"]
hepc = ["Liver cancer due to hepatitis C",
        "Acute hepatitis C",
        "Cirrhosis and other chronic liver diseases due to hepatitis C"]

by_dis.loc[by_dis["pathogen"].isin(hepb), "pathogen"] = "Hepatitis B"
by_dis.loc[by_dis["pathogen"].isin(hepc), "pathogen"] = "Hepatitis C"
by_dis = by_dis.groupby(["age_group_name", "cause_name", "pathogen"], as_index=False)["deaths"].sum()

by_ucod_long = by_ucod.pivot(index=["pathogen", "age_group_name"], columns="cause_name", values="deaths")
by_ucod_long = by_ucod_long.reset_index()

by_dis_long = by_dis.pivot(index=["pathogen", "age_group_name"], columns="cause_name", values="deaths")
by_dis_long = by_dis_long.reset_index()

print(f"Fixing names")
if "Mycobacterium tuberculosis" in by_ucod_long["pathogen"].unique().tolist():
    by_ucod_long = by_ucod_long.loc[by_ucod_long["pathogen"] != "Mycobacterium tuberculosis"]
if 'bordetella_pertussis' in by_ucod_long["pathogen"].unique().tolist():
    by_ucod_long = by_ucod_long.loc[by_ucod_long["pathogen"] != 'bordetella_pertussis']
if 'clostridium_tetani' in by_ucod_long["pathogen"].unique().tolist():
    by_ucod_long = by_ucod_long.loc[by_ucod_long["pathogen"] != 'clostridium_tetani']

```

```python
print(f"Concat and rename")
mcod_final = pd.concat([by_ucod_long, by_dis_long])
mcod_final.fillna(0, inplace=True)

mcod_final["Deaths in Chain"] = mcod_final["Communicable, maternal, neonatal, and nutritional diseases"] + mcod_final["Injuries"] + mcod_final["Non-communicable diseases"]
mcod_final.rename(columns={"Communicable, maternal, neonatal, and nutritional diseases": "Deaths with CMNND UCoDs",
                           "Injuries": "Deaths with injury UCoDs",
                           "Non-communicable diseases": "Deaths with NCD UCoDs"}, inplace=True)
mcod_final.to_csv("FILE_PATH", index=False)
```