D:\NHRI\AMR\amr-main\datasets\clean_SOURCE_1.py

```python
import pandas as pd
import numpy as np
import argparse
import re
import sys
from cod_prep.claude.configurator import Configurator
from cod_prep.downloaders import get_ages, get_map_version, pretty_print
from cod_prep.utils import (report_if_merge_fail, print_log_message,
                            clean_icd_codes)
from cod_prep.utils.formatting import ages
from mcod_prep.mcod_mapping import MCoDMapper
from amr_prep.utils.amr_io import save_amr_data

CONF = Configurator()
SOURCE = "SOURCE_NAME"
L_DIR = "FILEPATH"


def create_age_group_id(df):

    df.loc[df.age < 0, "age"] = np.NaN

    df['age_unit'] = 'year'
    df.rename(columns={'Age': 'age'}, inplace=True)
    age_formatter = ages.PointAgeFormatter()
    df = age_formatter.run(df)

    assert df['age_group_id'].notnull().values.all()

    return df


def create_sex_id(df):
    df['sex_id'] = df['sex'].apply(
        lambda x: 1 if x == 'male' else (2 if x == 'female' else 9)
    )
    print(
        f"{(len(df.loc[df['sex_id'] == 9]) / len(df)) * 100}% of rows are missing sex")
    report_if_merge_fail(df, 'sex_id', 'sex')
    return df


def mark_admissions_deaths(df):

    discharge_map = pd.read_csv("FILEPATH")
    discharge_dict = discharge_map.set_index("discharge_disp")[
        "Died"].to_dict()

    df["admissions"] = 1

    df["deaths"] = df["discharge_disp"].map(discharge_dict)
```

```
    df.loc[(df.deaths.isnull()) & (df.discharge_disp.str.contains(
        "hospice", flags=re.IGNORECASE)), "deaths"] = 1
    df.loc[df.discharge_disp.isin(["nan", "UNKNOWN", "-"]), "deaths"] = np.nan
    return df


def apply_pathogen_severity_map(df, named_cols):

    pathogen_severity = pd.read_excel("FILEPATH")
    pathogen_dict = pathogen_severity.set_index("pathogen_name")["rank"].to_dict()

    df[named_cols] = df[named_cols].apply(lambda x: x.map(
        pathogen_dict)).fillna(100).astype(int)

    df["severe_pathogen_int"] = df[named_cols].apply(lambda x: x.min(), axis=1)

    reverse_dict = dict([value, key] for key, value in pathogen_dict.items())
    df["severe_pathogen"] = df["severe_pathogen_int"].map(
        reverse_dict)

    df[named_cols] = df[named_cols].apply(lambda x: x.map(
        reverse_dict))

    df["severe_pathogen_int"] = df["severe_pathogen_int"].replace(100, np.NaN)

    return df


def fix_multiple_deaths(df):
    date_map = pd.read_csv("FILEPATH")
    df = mark_admissions_deaths(df)

    date_map.rename(columns={"date_id": "discharge_date_id",
                             "date": "discharge_date"}, inplace=True)
    df = df.merge(date_map, on="discharge_date_id", how="left")
    df["discharge_date"] = pd.to_datetime(df.discharge_date)
    report_if_merge_fail(df, "discharge_date_id", "date")
    null = df.loc[df['patient_id'].isna(), ]
    notnull = df.loc[~df['patient_id'].isna(), ]
    notnull["max"] = notnull.groupby("patient_id", as_index=False)
["discharge_date"].transform("max")
    notnull["deaths"] = np.where(notnull["discharge_date"] == notnull["max"],
notnull["deaths"], 0)
    df = pd.concat([null, notnull])

    return df


def map_specimen_source(df):
    specimen_detail = pd.read_excel("FILEPATH")
    specimen_detail.rename(columns={"specimen_source_group": "specimen_source"},
inplace=True)

```

```
    df = df.merge(specimen_detail, on=["site_id", "specimen_id", "specimen_source"], how="left")
    df.loc[df.specimen_source_detail.isnull(), "specimen_source_detail"] = df["specimen_source"]
    report_if_merge_fail(df, "specimen_source_detail", "specimen_source")

    df.rename(columns={'specimen_source_detail': 'raw_specimen'}, inplace=True)

    return df


def apply_syndrome_severity_map(df):
    priority_dict = {"peritonitis": 1,
                     "lri": 2,
                     "uti": 3,
                     "sepsis/bacteremia": 4,
                     "cellulitis": 5,
                     "other": 6}

    df['syndrome_num'] = df.groupby('admission_id')['specimen_syndrome'].transform(
        lambda syndromes: [
            f"specimen_syndrome_{i}" for i in range(len(syndromes))]
    )
    syndrome_df = df.pivot(
        index='admission_id', columns='syndrome_num', values='specimen_syndrome'
    ).reset_index()

    syndrome_cols = [x for x in list(syndrome_df) if "specimen_syndrome" in x]
    syndrome_df[syndrome_cols] = syndrome_df[syndrome_cols].apply(lambda x: x.map(
        priority_dict)).fillna(100).astype(int)
    syndrome_df = syndrome_df.assign(severe_syndrome_int=lambda x: x.min(axis=1))

    reverse_dict = dict([value, key] for key, value in priority_dict.items())
    syndrome_df["severe_syndrome"] = syndrome_df["severe_syndrome_int"].map(
        reverse_dict)

    df = df.merge(
        syndrome_df[["admission_id", "severe_syndrome"]], on="admission_id", how="left")
    drop_synds = df[df["specimen_syndrome"] != df["severe_syndrome"]].index
    df.drop(drop_synds, inplace=True)

    df.drop(columns="severe_syndrome", axis=1, inplace=True)

    return df


def create_pathogen_cols(df):
    df.rename(columns={'org_name': 'raw_pathogen'}, inplace=True)
    df['raw_pathogen'] = df['raw_pathogen'].str.strip()

    return df


def mark_estimated_pathogens(df, cols):
```

```
    df["pathogen"] = np.where(df[cols].isnull().all(axis=1), "none", "other")
    df["pathogen"] = np.where(df["severe_pathogen"].notnull(),
                              df["severe_pathogen"], df["pathogen"])

    return df


def map_cause_cols(df):
    print_log_message("creating pathogens cols")
    df = create_pathogen_cols(df)
    diag_df = prep_diagnosis()

    df = df.merge(diag_df, on="admission_id", how="left")
    df.rename(columns={"primary_diagnosis_code": "cause"}, inplace=True)

    print_log_message("mapping specimen source col")
    df = map_specimen_source(df)

    print_log_message("fixings patients with multiple 'deaths'")
    df = fix_multiple_deaths(df)

    for col in [x for x in list(df) if "cause" in x]:
        df[col] = df[col].fillna("none")
    df['cause'] = clean_icd_codes(df['cause'], remove_decimal=True)
    df.loc[~df.cause.str.match(r"[A-Z]\d{2,5}"), 'cause'] = "none"

    return df


def prep_diagnosis():
    df = pd.read_csv("FILEPATH")
    df['diagnosis_code'] = clean_icd_codes(df['diagnosis_code'], remove_decimal=True)
    df.loc[~df.diagnosis_code.str.match(r"[A-Z]\d{2,5}"), 'diagnosis_code'] = "none"
    df['cause_num'] = df.groupby('admission_id')['diagnosis_code'].transform(
        lambda codes: [f"multiple_cause_{i}" for i in range(len(codes))]
    )
    df.loc[df['primary_diagnosis'], 'cause_num'] = 'cause'
    df = df.pivot(
        index='admission_id', columns='cause_num', values='diagnosis_code'
    ).reset_index().fillna('none')
    df.drop(columns="cause", inplace=True)
    return df


def map_hosp(df):
    date_map = pd.read_csv("FILEPATH")

    admit_date_map = date_map.rename(columns={"date_id": "admit_date_id",
                                              "date": "admit_date"})
    collected_date_map = date_map.rename(columns={"date_id": "collected_date_id",
                                                  "date": "collected_date"})

    df = df.merge(admit_date_map, on='admit_date_id', how='left')
    df = df.merge(collected_date_map, on='collected_date_id', how='left')
```

```
    df['date_diff'] = (pd.to_datetime(df['collected_date']) -
                       pd.to_datetime(df['admit_date'])).dt.days

    df['hosp'] = 'hospital'
    df.loc[df['date_diff'] <= 2, 'hosp'] = 'community'
    df.loc[df['date_diff'].isna(), 'hosp'] = 'unknown'

    return df


def clean_SOURCE():

    df = pd.read_csv("FILEPATH")
    cols = [col for col in df.columns if 'result_id' not in col]
    df = df.drop_duplicates(cols)
    df = create_age_group_id(df)
    df = create_sex_id(df)
    df = map_cause_cols(df)
    df = map_hosp(df)
    df.rename(columns={'drug_name': 'raw_antibiotic'}, inplace=True)
    df['resistance'] = df.interpretation.map({
        'Resistant': 'resistant',
        'Susceptible': 'sensitive',
        'Intermediate': 'resistant',
        'Positive': 'resistant',
        'Non Suceptible': 'resistant',
        'Negative': 'sensitive'
    }).fillna('unknown')

    df["location_id"] = 102
    df["nid"] = 410446
    df["year_id"] = df["discharge_date"].apply(lambda x: x.year)
    df['sample_id'] = (df['patient_id'].astype(str) + df['specimen_id'].astype(str))
    df['cases'] = 1

    df['hospital_type'] = 'unknown'
    df['hospital_name'] = 'unknown'

    cause_columns = [x for x in list(df) if "multiple_cause" in x] + ['cause']
    demo_cols = ['nid', 'location_id', 'year_id', 'age_group_id', 'sex_id']
    biology = ['raw_specimen', 'raw_pathogen', 'raw_antibiotic', 'resistance']
    other_values = ['hosp', 'hospital_name', 'hospital_type', 'deaths', 'sample_id',
'cases']
    lowercase_cols = ['raw_specimen', 'raw_pathogen', 'raw_antibiotic']

    df = df[demo_cols + cause_columns + biology + other_values]

    for col in lowercase_cols:
        df[col] = df[col].str.lower().str.strip()
    assert df.sample_id.notnull().values.all()

    df = df.drop_duplicates()

    return df
```

```
261
262
263 if __name__ == "__main__":
264     df = clean_SOURCE()
265     save_amr_data(df, phase='unmapped', ldir=L_DIR)
266
```