In [ ]:
```
import pandas as pd # type: ignore
import os
import glob
from pathlib import Path
import json
import re
```

Define file paths and load data

In [ ]:
```
# Specify the folder paths
olympics_folder_path = '../datasets/olympics/'
countries_folder_path='../datasets/countries/countries of the world.csv'

olympics_dataframes,countries_dataframes = [], []
# Get a list of all CSV files in the folder
oly_csv_files = glob.glob(os.path.join(olympics_folder_path, '*.csv')) #for
#country_csv_file = glob.glob(os.path.join(countries_folder_path, '*.csv'))

# Read each CSV file into a dataframe and store them in a list

for csv_file in oly_csv_files:
    df = pd.read_csv(csv_file)
    olympics_dataframes.append(df)

#combined dataframe with  all olympics data
olympics_combined_df = pd.concat(olympics_dataframes, ignore_index=True)

#display first 20 values of the combined dataframe
print(olympics_combined_df.head(20))
```

```
    NOC  Gold  Silver  Bronze  Total
0   GER    11      12       6     29
1   USA     9       9       7     25
2   AUT     9       7       7     23
3   RUS     8       6       8     22
4   CAN     7      10       7     24
5   SWE     7       2       5     14
6   KOR     6       3       2     11
7   SUI     5       4       5     14
8   ITA     5       0       6     11
9   FRA     3       2       4      9
10  NED     3       2       4      9
11  EST     3       0       0      3
12  NOR     2       8       9     19
13  CHN     2       4       5     11
14  CZE     1       2       1      4
15  CRO     1       2       0      3
16  AUS     1       0       1      2
17  JPN     1       0       0      1
18  FIN     0       6       3      9
19  POL     0       1       1      2
```

Basic Transformations fo Olympics

In [ ]:
```
# Define a schema: ensuring column names and data types are consistent
olympics_combined_df.columns = [col.strip().lower().replace(' ', '_') for co

# Display the updated schema
olympics_combined_df.dtypes

#saving the output
olympics_combined_df.to_csv('../output_files/combined_olympics.csv', index=F
```

Load Countries data

In [ ]:
```
#for countries
countries_df = pd.read_csv(countries_folder_path)
countries_df.head(10)
```

Out[ ]:

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	I mor b
0	Afghanistan	ASIA (EX. NEAR EAST)	31056997	647500	48,0	0,00	23,06	1
1	Albania	EASTERN EUROPE	3581655	28748	124,6	1,26	-4,93	
2	Algeria	NORTHERN AFRICA	32930091	2381740	13,8	0,04	-0,39	
3	American Samoa	OCEANIA	57794	199	290,4	58,29	-20,71	
4	Andorra	WESTERN EUROPE	71201	468	152,1	0,00	6,6	
5	Angola	SUB-SAHARAN AFRICA	12127071	1246700	9,7	0,13	0	
6	Anguilla	LATIN AMER. & CARIB	13477	102	132,1	59,80	10,76	
7	Antigua & Barbuda	LATIN AMER. & CARIB	69108	443	156,0	34,54	-6,15	
8	Argentina	LATIN AMER. & CARIB	39921833	2766890	14,4	0,18	0,61	
9	Armenia	C.W. OF IND. STATES	2976372	29800	99,9	0,00	-6,47	

Column Preprocessing

In [ ]:
```
# Ensuring column names and data types are consistent with that of Olympics
countries_df.columns = [col.strip().lower().replace(' ', '_') for col in cou
print(countries_df.columns)
# Converting some datatypes

countries_df['population'] = countries_df['population'].apply(lambda x: pd.t
countries_df['gdp_($_per_capita)'] = countries_df['gdp_($_per_capita)'].appl

#method to make sure column names are consistent to avoid any issues while J
def format_country_name(name):
    # Convert to lowercase, remove special characters, and trim whitespace u
    if pd.notna(name):
        name = re.sub(r'[^a-zA-Z\s]', '', name).strip().lower()
    return name

# Display the updated schema
countries_df.dtypes
```

```
Index(['country', 'region', 'population', 'area_(sq._mi.)',
       'pop._density_(per_sq._mi.)', 'coastline_(coast/area_ratio)',
       'net_migration', 'infant_mortality_(per_1000_births)',
       'gdp_($_per_capita)', 'literacy_(%)', 'phones_(per_1000)', 'arable_
(%)',
       'crops_(%)', 'other_(%)', 'climate', 'birthrate', 'deathrate',
       'agriculture', 'industry', 'service'],
      dtype='object')
```

Out[ ]:
```
country                                object
region                                 object
population                              int64
area_(sq._mi.)                          int64
pop._density_(per_sq._mi.)             object
coastline_(coast/area_ratio)           object
net_migration                          object
infant_mortality_(per_1000_births)     object
gdp_($_per_capita)                    float64
literacy_(%)                           object
phones_(per_1000)                      object
arable_(%)                             object
crops_(%)                              object
other_(%)                              object
climate                                object
birthrate                              object
deathrate                              object
agriculture                            object
industry                               object
service                                object
dtype: object
```

Basic Transformations for Countries

In [ ]:
```
# Handle missing values or duplicates
countries_df.dropna(inplace=True)
countries_df.drop_duplicates(inplace=True)

# Add row number to add primary key index to COuntries dataset
countries_df['country_id'] = countries_df.reset_index().index + 1
#reorder the postion of the country_id
countries_df = countries_df[['country_id'] + [col for col in countries_df.co
#saving the output
countries_df.to_csv('../output_files/countries_data.csv', index=False)
```

Combined Transformations

1. Normalized Data

In [ ]:
```
################### 1. NORMALIZATIONS  ######################
#Add an artificial key to combine Olympics and Country data
# Adding a dictionary under datasets that maps each NOC to a country so we c
with open('../datasets/noc_countries.json', 'r') as f:
    country_mapping = json.load(f)

#country_name is the foreign key from Countries dataset that will be used in
olympics_combined_df['country'] = olympics_combined_df['noc'].map(country_ma



olympics_combined_df.to_csv('../output_files/olympics_normalized.csv', index
countries_df.to_csv('../output_files/countries_normalized.csv',index=False)


################################################
```

2.Denormized Data

In [ ]:
```
###################### DENORMALIZATION ###############
# we need to combine the data from the 2 tables into 1 big table to avoid th
# this dataframe gives us the aggregated medals won
olympics_agg_df = olympics_combined_df.groupby('country').agg({
    'gold': 'sum',
    'silver': 'sum',
    'bronze': 'sum',
    'total':'sum'
}).reset_index()


# Since all the countires are unique and we have high cardinality it is a 1-
# to the countries dataframe

olympics_agg_df['country'] = olympics_agg_df['country'].apply(format_country
countries_df['country'] = countries_df['country'].apply(format_country_name)
merged_df = pd.merge(olympics_agg_df, countries_df, on='country', how='right
```

```
print(merged_df.head(10))

merged_df.to_csv('../output_files/denormalized_data.csv',index=False)
```

```
           country  gold  silver  bronze  total  country_id  \
0      afghanistan   0.0     0.0     1.0    1.0           1
1          albania   0.0     0.0     2.0    2.0           2
2          algeria   6.0     3.0     5.0   14.0           3
3         anguilla   NaN     NaN     NaN    NaN           4
4  antigua  barbuda   NaN     NaN     NaN    NaN           5
5        argentina   7.0     8.0    12.0   27.0           6
6          armenia   2.0    10.0     5.0   17.0           7
7            aruba   NaN     NaN     NaN    NaN           8
8        australia  99.0   109.0   123.0  331.0           9
9          austria  44.0    50.0    63.0  157.0          10

                              region  population  area_(sq._mi.)  \
0       ASIA (EX. NEAR EAST)             31056997          647500
1  EASTERN EUROPE                         3581655           28748
2  NORTHERN AFRICA                       32930091         2381740
3             LATIN AMER. & CARIB           13477             102
4             LATIN AMER. & CARIB           69108             443
5             LATIN AMER. & CARIB        39921833         2766890
6                C.W. OF IND. STATES      2976372           29800
7             LATIN AMER. & CARIB           71891             193
8  OCEANIA                               20264082         7686850
9  WESTERN EUROPE                         8192880           83870

  pop._density_(per_sq._mi.)  ... phones_(per_1000) arable_(%) crops_(%)  \
0                      48,0  ...               3,2      12,13      0,22
1                     124,6  ...              71,2      21,09      4,42
2                      13,8  ...              78,1       3,22      0,25
3                     132,1  ...             460,0          0         0
4                     156,0  ...             549,9      18,18      4,55
5                      14,4  ...             220,4      12,31      0,48
6                      99,9  ...             195,7      17,55       2,3
7                     372,5  ...             516,1      10,53         0
8                       2,6  ...             565,5       6,55      0,04
9                      97,7  ...             452,2      16,91      0,86

   other_(%) climate birthrate deathrate agriculture industry service
0      87,65       1      46,6     20,34        0,38     0,24    0,38
1      74,49       3     15,11      5,22       0,232    0,188   0,579
2      96,53       1     17,14      4,61       0,101      0,6   0,298
3        100       2     14,17      5,34        0,04     0,18    0,78
4      77,27       2     16,93      5,37       0,038     0,22   0,743
5      87,21       3     16,73      7,55       0,095    0,358   0,547
6      80,15       4     12,07      8,23       0,239    0,343   0,418
7      89,47       2     11,03      6,68       0,004    0,333   0,663
8      93,41       1     12,14      7,51       0,038    0,262     0,7
9      82,23       3      8,74      9,76       0,018    0,304   0,678

[10 rows x 25 columns]
```