Bilenkin540_Term_Project_Milestone_4

May 14, 2025

1 DSC 540 - Project Milestone 4: Cleaning and Formatting Data from API

Loading Data from WHO GHO API (Life Expectancy)

[21]:
```
import requests
import pandas as pd

# Loading data from WHO GHO API (Life Expectancy Indicator)
url = "https://ghoapi.azureedge.net/api/WHOSIS_000001"
headers = {"User-Agent": "Mozilla/5.0"}

response = requests.get(url, headers=headers)

if response.status_code == 200:
    data = response.json()
    df = pd.DataFrame(data['value'])
    print(df.head())
else:
    print(f"Failed to fetch data. Status code: {response.status_code}")
```

```
     Id  IndicatorCode SpatialDimType SpatialDim ParentLocationCode  \
0   830  WHOSIS_000001        COUNTRY        SOM                EMR
1  1012  WHOSIS_000001        COUNTRY        BTN               SEAR
2  3432  WHOSIS_000001        COUNTRY        BHR                EMR
3  6575  WHOSIS_000001        COUNTRY        SAU                EMR
4  6823  WHOSIS_000001        COUNTRY        CYP                EUR

  TimeDimType         ParentLocation Dim1Type  TimeDim      Dim1  … \
0        YEAR  Eastern Mediterranean      SEX     2008   SEX_MLE  …
1        YEAR        South-East Asia      SEX     2002  SEX_BTSX  …
2        YEAR  Eastern Mediterranean      SEX     2011  SEX_FMLE  …
3        YEAR  Eastern Mediterranean      SEX     2005  SEX_FMLE  …
4        YEAR                Europe      SEX     2003   SEX_MLE  …

  DataSourceDim             Value NumericValue       Low      High Comments  \
0          None  48.0 [46.7-49.6]     48.03754  46.71678  49.62846     None
1          None  67.8 [67.1-68.6]     67.84567  67.08312  68.55874     None
```

```
2          None  75.2 [75.1-75.4]     75.20536  75.05048  75.38518     None
3          None  73.1 [72.8-73.5]     73.12227  72.80645  73.47895     None
4          None  76.9 [76.7-77.2]     76.86013  76.67124  77.16348     None

                           Date  TimeDimensionValue  \
0  2024-08-02T09:43:39.193+02:00                2008
1  2024-08-02T09:43:39.193+02:00                2002
2  2024-08-02T09:43:39.193+02:00                2011
3  2024-08-02T09:43:39.193+02:00                2005
4  2024-08-02T09:43:39.193+02:00                2003

         TimeDimensionBegin           TimeDimensionEnd
0  2008-01-01T00:00:00+01:00  2008-12-31T00:00:00+01:00
1  2002-01-01T00:00:00+01:00  2002-12-31T00:00:00+01:00
2  2011-01-01T00:00:00+01:00  2011-12-31T00:00:00+01:00
3  2005-01-01T00:00:00+01:00  2005-12-31T00:00:00+01:00
4  2003-01-01T00:00:00+01:00  2003-12-31T00:00:00+01:00

[5 rows x 25 columns]
```

1.0.1 Step 1: Drop Irrelevant or Redundant Columns

I simplified the dataset by removing columns that were either redundant, consistently null, or not useful for analysis (e.g., 'Id', 'Date', 'TimeDimType'). This makes the data more efficient and streamlines the dataset for further transformations.

[22]:
```python
# Dropping irrelevant or redundant columns
columns_to_drop = [
    'Id', 'Date', 'TimeDim', 'TimeDimType',
    'TimeDimensionBegin', 'TimeDimensionEnd',
    'Comments', 'DataSourceDim'
]

df_cleaned = df.drop(columns=columns_to_drop)
print("Step 1 complete: Dropped irrelevant columns. Current columns are:")
print(df_cleaned.columns)
```

```
Step 1 complete: Dropped irrelevant columns. Current columns are:
Index(['IndicatorCode', 'SpatialDimType', 'SpatialDim', 'ParentLocationCode',
       'ParentLocation', 'Dim1Type', 'Dim1', 'Dim2Type', 'Dim2', 'Dim3Type',
       'Dim3', 'DataSourceDimType', 'Value', 'NumericValue', 'Low', 'High',
       'TimeDimensionValue'],
      dtype='object')
```

1.0.2 Step 2: Fix Inconsistent Casing in Categorical Columns

To standardize the data and ensure consistency, I converted the values in categorical columns like 'SpatialDimType', 'ParentLocation', and 'Dim1Type' to title case. This helps prevent issues with grouping or filtering later on.

[23]:
```
# Fixing inconsistent casing by converting selected columns to title case
columns_to_title_case = ['SpatialDimType', 'ParentLocation', 'Dim1Type']
for col in columns_to_title_case:
    df[col] = df[col].str.title()

# Displaying sample to verify the transformation
print(df[columns_to_title_case].drop_duplicates().head())
```

```
   SpatialDimType         ParentLocation Dim1Type
0         Country  Eastern Mediterranean      Sex
1         Country        South-East Asia      Sex
4         Country                 Europe      Sex
6         Country        Western Pacific      Sex
10        Country               Americas      Sex
```

1.0.3 Step 3: Convert NumericValue Column to Proper Numeric Type and Round Values

I converted the NumericValue column from object to float and rounded the values to two decimal places. This standardization ensures numeric consistency for analysis and easier visual interpretation of key figures such as life expectancy.

[24]:
```
# Converting NumericValue to float and round to 2 decimal places
df['NumericValue'] = pd.to_numeric(df['NumericValue'], errors='coerce').round(2)

# Checking if conversion was successful
print(df[['NumericValue']].head())
```

```
   NumericValue
0         48.04
1         67.85
2         75.21
3         73.12
4         76.86
```

1.0.4 Step 4: Handle Missing Values (Nulls) in Key Columns

I handled missing values by filling critical columns such as ParentLocation and ParentLocationCode with the placeholder value 'Unknown' to maintain data integrity. Additionally, other columns with missing values, like Dim2Type, Dim2, Dim3Type, and Dim3, were also filled with appropriate placeholders. After these adjustments, the dataset is now complete and contains no missing values.

[25]:
```
# Filling missing values in categorical columns with a placeholder 'Unknown'
df['ParentLocationCode'] = df['ParentLocationCode'].fillna('Unknown')
df['ParentLocation'] = df['ParentLocation'].fillna('Unknown')
df['Dim2Type'] = df['Dim2Type'].fillna('Unknown')
df['Dim2'] = df['Dim2'].fillna('Unknown')
df['Dim3Type'] = df['Dim3Type'].fillna('Unknown')
df['Dim3'] = df['Dim3'].fillna('Unknown')
```

```
df['DataSourceDimType'] = df['DataSourceDimType'].fillna('Unknown')
df['DataSourceDim'] = df['DataSourceDim'].fillna('Unknown')
df['Comments'] = df['Comments'].fillna('No Comment')

# If the columns are numerical, using the methods like mean or median for␣
 ↪filling
df['Low'] = df['Low'].fillna(df['Low'].mean())  # Filling with mean value
df['High'] = df['High'].fillna(df['High'].mean())  # Filling with mean value

# Verifying that the missing data is handled
print(df.isnull().sum())
```

```
Id                    0
IndicatorCode         0
SpatialDimType        0
SpatialDim            0
ParentLocationCode    0
TimeDimType           0
ParentLocation        0
Dim1Type              0
TimeDim               0
Dim1                  0
Dim2Type              0
Dim2                  0
Dim3Type              0
Dim3                  0
DataSourceDimType     0
DataSourceDim         0
Value                 0
NumericValue          0
Low                   0
High                  0
Comments              0
Date                  0
TimeDimensionValue    0
TimeDimensionBegin    0
TimeDimensionEnd      0
dtype: int64
```

1.0.5 Step 5: Standardize Numerical Data and Encode Categorical Columns

I standardized the NumericValue column to ensure all numerical data is on the same scale, making it easier to work with for modeling. I also applied one-hot encoding to categorical columns like SpatialDimType, ParentLocation, and Dim1Type to transform them into a format suitable for machine learning algorithms.

[26]:
```
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
```

```
from sklearn.pipeline import Pipeline

# Standardizing/Normalizing Numerical Data
scaler = StandardScaler()

# Applying the scaler to the 'NumericValue' column
df['NumericValue'] = scaler.fit_transform(df[['NumericValue']])

# Using one-hot encoding on categorical columns like 'SpatialDimType',␣
 ↪'ParentLocation', etc.
df = pd.get_dummies(df, columns=['SpatialDimType', 'ParentLocation',␣
 ↪'Dim1Type', 'Dim1'], drop_first=True)

# Verifying the changes
print(df.head())
```

```
     Id  IndicatorCode SpatialDim ParentLocationCode TimeDimType  TimeDim  \
0   830  WHOSIS_000001        SOM                EMR        YEAR     2008
1  1012  WHOSIS_000001        BTN               SEAR        YEAR     2002
2  3432  WHOSIS_000001        BHR                EMR        YEAR     2011
3  6575  WHOSIS_000001        SAU                EMR        YEAR     2005
4  6823  WHOSIS_000001        CYP                EUR        YEAR     2003

  Dim2Type     Dim2 Dim3Type     Dim3  … SpatialDimType_Region  \
0  Unknown  Unknown  Unknown  Unknown  …                 False
1  Unknown  Unknown  Unknown  Unknown  …                 False
2  Unknown  Unknown  Unknown  Unknown  …                 False
3  Unknown  Unknown  Unknown  Unknown  …                 False
4  Unknown  Unknown  Unknown  Unknown  …                 False

  SpatialDimType_Worldbankincomegroup ParentLocation_Americas  \
0                               False                   False
1                               False                   False
2                               False                   False
3                               False                   False
4                               False                   False

   ParentLocation_Eastern Mediterranean  ParentLocation_Europe  \
0                                  True                  False
1                                 False                  False
2                                  True                  False
3                                  True                  False
4                                 False                   True

   ParentLocation_South-East Asia ParentLocation_Unknown  \
0                           False                  False
1                            True                  False
```

```
2                            False                 False
3                            False                 False
4                            False                 False

  ParentLocation_Western Pacific Dim1_SEX_FMLE Dim1_SEX_MLE
0                          False         False         True
1                          False         False        False
2                          False          True        False
3                          False          True        False
4                          False         False         True

[5 rows x 32 columns]
```

1.0.6 Step 6: Identifying and Handling Duplicates

I checked for and removed any duplicate rows from the dataset to ensure that the data is unique and consistent for analysis. This helps prevent biased results caused by repeated entries. After removal, there were no duplicates found, and the dataset shape remains unchanged.

[27]:
```
# Checking for duplicates in the dataset
duplicates = df.duplicated()

# Printing the number of duplicate rows
print(f'Number of duplicate rows: {duplicates.sum()}')

# Removing duplicates if any
df = df.drop_duplicates()

# Verifying that duplicates are removed
print(f'Dataset shape after removing duplicates: {df.shape}')
```

```
Number of duplicate rows: 0
Dataset shape after removing duplicates: (12936, 32)
```

1.0.7 Step 7: Finalizing the Dataset for Analysis

In this step, I verified that all necessary transformations were applied correctly, ensuring the dataset is ready for further analysis or modeling tasks. This included confirming that there are no remaining missing values, the data types are correct, and the dataset is clean and consistent.

[28]:
```
# Verifying the data types and ensuring the dataset is ready for further␣
 ↪analysis
print(df.dtypes)

# Checking for any remaining missing values
print("Remaining missing values:", df.isnull().sum().sum())

# Displaying the first five rows of the cleaned dataset
print(df.head())
```

```
Id                                      int64
IndicatorCode                          object
SpatialDim                             object
ParentLocationCode                     object
TimeDimType                            object
TimeDim                                 int64
Dim2Type                               object
Dim2                                   object
Dim3Type                               object
Dim3                                   object
DataSourceDimType                      object
DataSourceDim                          object
Value                                  object
NumericValue                          float64
Low                                   float64
High                                  float64
Comments                               object
Date                                   object
TimeDimensionValue                     object
TimeDimensionBegin                     object
TimeDimensionEnd                       object
SpatialDimType_Global                    bool
SpatialDimType_Region                    bool
SpatialDimType_Worldbankincomegroup      bool
ParentLocation_Americas                  bool
ParentLocation_Eastern Mediterranean     bool
ParentLocation_Europe                    bool
ParentLocation_South-East Asia           bool
ParentLocation_Unknown                   bool
ParentLocation_Western Pacific           bool
Dim1_SEX_FMLE                            bool
Dim1_SEX_MLE                             bool
dtype: object
Remaining missing values: 0
     Id  IndicatorCode SpatialDim ParentLocationCode TimeDimType  TimeDim  \
0   830  WHOSIS_000001        SOM                EMR        YEAR     2008
1  1012  WHOSIS_000001        BTN               SEAR        YEAR     2002
2  3432  WHOSIS_000001        BHR                EMR        YEAR     2011
3  6575  WHOSIS_000001        SAU                EMR        YEAR     2005
4  6823  WHOSIS_000001        CYP                EUR        YEAR     2003

  Dim2Type     Dim2 Dim3Type     Dim3  … SpatialDimType_Region  \
0  Unknown  Unknown  Unknown  Unknown  …                 False
1  Unknown  Unknown  Unknown  Unknown  …                 False
2  Unknown  Unknown  Unknown  Unknown  …                 False
3  Unknown  Unknown  Unknown  Unknown  …                 False
4  Unknown  Unknown  Unknown  Unknown  …                 False
```

```
  SpatialDimType_Worldbankincomegroup ParentLocation_Americas  \
0                               False                   False
1                               False                   False
2                               False                   False
3                               False                   False
4                               False                   False

   ParentLocation_Eastern Mediterranean  ParentLocation_Europe  \
0                                  True                  False
1                                 False                  False
2                                  True                  False
3                                  True                  False
4                                 False                   True

   ParentLocation_South-East Asia ParentLocation_Unknown  \
0                           False                  False
1                            True                  False
2                           False                  False
3                           False                  False
4                           False                  False

  ParentLocation_Western Pacific Dim1_SEX_FMLE Dim1_SEX_MLE
0                          False         False         True
1                          False         False        False
2                          False          True        False
3                          False          True        False
4                          False         False         True

[5 rows x 32 columns]
```

1.0.8 Ethical implications of data wrangling

In this data wrangling process, I addressed missing values in key categorical columns such as ParentLocation and ParentLocationCode by replacing them with "Unknown" to preserve data integrity and prevent loss of useful records. For numerical columns like Low and High, I imputed missing values using the column mean to maintain the completeness of the dataset for analysis.

The dataset was sourced from the WHO Global Health Observatory API, which provides publicly accessible health-related data. While this is health-related data, it does not contain any personal or identifiable information. Therefore, regulatory frameworks such as the Health Insurance Portability and Accountability Act (HIPAA), which governs the protection of personal health information in the U.S., do not apply in this context. However, broader regulations like the General Data Protection Regulation (GDPR) must still be considered in general when handling health data. Since this dataset is anonymized and aggregated, no legal or regulatory violations apply in this case.

Some ethical risks include the potential loss of valuable information during cleaning and transformation steps. For example, filling missing values with "Unknown" might obscure meaningful patterns, and imputing with the mean can reduce variability and potentially mask outliers. I made a few assumptions during transformation—for instance, assuming that missing location codes could

safely be categorized as “Unknown” and that the mean is a suitable proxy for missing numerical data.

The WHO is a globally recognized and credible data source, and the data was obtained ethically through their open-access API. To mitigate potential ethical concerns, I ensured all transformation steps were well-documented to maintain transparency. In future analyses, a deeper investigation into the reasons for missing data and consideration of more sophisticated imputation techniques may help minimize bias.