ADOT

May 9, 2025

Traffic data extraction code

```
[16]: # Import required library
      import pandas as pd
      import os
```

This code will check the data to ensure the correct header

This code will extract traffic data for the year 2007-2009

```
[19]: # Load the dataset
      data = pd.read_excel('/Users/kishupatel/Desktop/DesFert/ADOT/
       ↪2007-2009-AADT-PUBLICATION.xlsx')

      # Specify the list of CNTLOCID values you want to extract
      cntlocid_values = [100022, 100023, 100085, 100086, 100088, 100100, 100101,␣
       ↪100102, 100112, 100113, 100117, 100348, 100677,
                         100678, 100679, 100918, 100980, 100982, 101240, 101385, 101597,␣
       ↪101598, 101599, 101600, 101890, 101891,
                         101902, 101903, 101904]  # Replace with your specific CNTLOCID␣
       ↪values

      # Filter the data for the specified CNTLOCID values
      filtered_data = data[data['CNTLOCID'].isin(cntlocid_values)]

      # Select only the required columns
      columns_to_extract = ['CNTLOCID', 'ROUTE', 'START', 'END', 'AADT 2007', 'AADT␣
       ↪2008', 'AADT 2009']
      filtered_data = filtered_data[columns_to_extract]

      # Create the new file name by appending 'filtered' to the original name
      new_file = '/Users/kishupatel/Desktop/2007-2009_filtered.xlsx'

      # Save the filtered data to an Excel file
      filtered_data.to_excel(new_file, index=False)

      print(f"Filtered data saved to {new_file}")
```

```
Filtered data saved to /Users/kishupatel/Desktop/2007-2009_filtered.xlsx
```

This code will extract traffic data for the year 2011

```
[22]: # Load the dataset
file_path = ('/Users/kishupatel/Desktop/DesFert/ADOT/2011-AADT-PUBLICATION.xls')
data = pd.read_excel(file_path, sheet_name='SHS Traffic Log 2011')
# Specify the list of CNTLOCID values you want to extract
cntlocid_values = [100022, 100023, 100085, 100086, 100088, 100100, 100101,␣
 ↪100102, 100112, 100113, 100117, 100348, 100677,
              100678, 100679, 100918, 100980, 100982, 101240, 101385, 101597,␣
 ↪101598, 101599, 101600, 101890, 101891,
              101902, 101903, 101904]

# Filter the data for the specified CNTLOCID values
filtered_data = data[data['CNTLOCID'].isin(cntlocid_values)]

# Select only the required columns
columns_to_extract = ['CNTLOCID', 'ROUTE', 'START', 'END', 'AADT 2011']
filtered_data = filtered_data[columns_to_extract]

# Create the new file name by appending 'filtered' to the original name
new_file = '/Users/kishupatel/Desktop/2011_filtered.xlsx'

# Save the filtered data to an Excel file
filtered_data.to_excel(new_file, index=False)

print(f"Filtered data saved to {new_file}")
```

```
Filtered data saved to /Users/kishupatel/Desktop/2011_filtered.xlsx
```

This code will extract traffic data for the year 2014

```
[35]: # Load the dataset
file_path = ('/Users/kishupatel/Desktop/DesFert/ADOT/2014-AADT.xlsx')
data = pd.read_excel(file_path, sheet_name='Table 1')
# Specify the list of CNTLOCID values you want to extract
cntlocid_values = [100022, 100023, 100085, 100086, 100088, 100100, 100101,␣
 ↪100102, 100112, 100113, 100117, 100348, 100677,
              100678, 100679, 100918, 100980, 100982, 101240, 101385, 101597,␣
 ↪101598, 101599, 101600, 101890, 101891,
              101902, 101903, 101904]

# Filter the data for the specified CNTLOCID values
filtered_data = data[data['CNTLOCID'].isin(cntlocid_values)]

# Select only the required columns
columns_to_extract = ['CNTLOCID', 'ROUTE', 'START', 'END', 'AADT 2014']
filtered_data = filtered_data[columns_to_extract]
```

```python
# Create the new file name by appending 'filtered' to the original name
new_file = '/Users/kishupatel/Desktop/2014_filtered.xlsx'

# Save the filtered data to an Excel file
filtered_data.to_excel(new_file, index=False)

print(f"Filtered data saved to {new_file}")
```

```
Filtered data saved to /Users/kishupatel/Desktop/2014_filtered.xlsx
```

This code will extract traffic data for the year 2015

[38]:
```python
# Read the Excel file
file_path = '/Users/kishupatel/Desktop/DesFert/ADOT/2015-AADT-Publication_0.
 ↪xlsx'
dfs = pd.read_excel(file_path, sheet_name=None, header=None, dtype=str)  # Read␣
 ↪all sheets without assuming a header

cleaned_sheets = {}

for sheet_name, df in dfs.items():
    print(f"\n  Processing sheet: {sheet_name}")
    # Drop fully empty rows (entirely NaN)
    df.dropna(how="all", inplace=True)
    # Find the first row with valid data (assumes it has at least half non-null␣
 ↪values)
    for i in range(len(df)):
        non_null_count = df.iloc[i].notna().sum()
        if non_null_count > len(df.columns) / 2:
            valid_header_row = i
            break

    # Set detected row as header
    df.columns = df.iloc[valid_header_row].astype(str).str.strip()  # Remove␣
 ↪spaces from column names
    # Drop all rows above the detected header row
    df = df.iloc[valid_header_row + 1:].reset_index(drop=True)
    # Remove duplicate header rows appearing later
    df = df[df['Loc ID'] != 'Loc ID']
    # Print detected column names for debugging
    print(f"  Finalized columns: {df.columns.tolist()}")
    # Ensure 'Loc ID' exists before proceeding
    if 'Loc ID' in df.columns:
        df = df[df['Loc ID'].notna()]
    else:
        print(f"  Warning: 'Loc ID' not found in sheet '{sheet_name}'. Skipping␣
 ↪this sheet.")
        continue  # Skip if 'Loc ID' is missing
    # Convert 'Loc ID' to numeric
    df['Loc ID'] = pd.to_numeric(df['Loc ID'], errors='coerce')
    # Store cleaned sheet
    cleaned_sheets[sheet_name] = df
# --------------------------- #
#   Now Filter and Combine Data #
# --------------------------- #
# Specify the Loc IDs to retrieve
given_ids = [100022, 100023, 100085, 100086, 100088, 100100, 100101, 100102,␣
 ↪100112, 100113, 100117, 100348, 100677,
             100678, 100679, 100918, 100980, 100982, 101240, 101385, 101597,␣
 ↪101598, 101599, 101600, 101890, 101891,
             101902, 101903, 101904]
# Specify the columns to retrieve
columns_to_select = ['Loc ID', 'Route', 'Start', 'End', 'AADT 2015']
all_filtered_data = []

# Process cleaned sheets
for sheet_name, df in cleaned_sheets.items():
    print(f" Filtering data from sheet: {sheet_name}")
    # Ensure 'Loc ID' exists before filtering
    if 'Loc ID' in df.columns:
        filtered_data = df[df['Loc ID'].isin(given_ids)][columns_to_select]
        if not filtered_data.empty:
            all_filtered_data.append(filtered_data)
    else:
        print(f" Warning: 'Loc ID' not found in sheet '{sheet_name}'. Skipping␣
 ↪this sheet.")

# Combine all filtered data
if all_filtered_data:
    combined_data = pd.concat(all_filtered_data, ignore_index=True).
 ↪drop_duplicates()
    # Get the first 4 letters of the original file name (without extension)
    file_name = os.path.splitext(os.path.basename(file_path))[0][:4]
    # Create output file path with the first 4 letters + filtered data
    output_file_path = f'/Users/kishupatel/Desktop/{file_name}_filtered_data.
 ↪xlsx'
    # Save final output
    with pd.ExcelWriter(output_file_path, engine='openpyxl') as writer:
        combined_data.to_excel(writer, sheet_name='CombinedData', index=False)
    print(f"Filtered data saved to: {output_file_path}")
else:
    print("\n No matching data found for the specified Loc IDs.")
```

```
Processing sheet: State Routes
```

```
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2015', 'AADT Derivation Code', 'K Factor',
'D Factor', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2035
AADT']

 Processing sheet: US Routes
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2015', 'AADT Derivation Code', 'K Factor',
'D Factor', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2035
AADT']

 Processing sheet: Interstates
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2015', 'AADT Derivation Code', 'K Factor',
'D Factor', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2035
AADT']

 Processing sheet: AADT 2015
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2015', 'AADT Derivation Code', 'K Factor',
'D Factor', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2035
AADT']
 Filtering data from sheet: State Routes
 Filtering data from sheet: US Routes
 Filtering data from sheet: Interstates
 Filtering data from sheet: AADT 2015
Filtered data saved to: /Users/kishupatel/Desktop/2015_filtered_data.xlsx
```

This code will extract traffic data for the year 2016

```
[41]:
# Read the Excel file
file_path = '/Users/kishupatel/Desktop/DesFert/ADOT/2016-aadt-Publication_1.
 ↪xlsx'
dfs = pd.read_excel(file_path, sheet_name=None, header=None, dtype=str)  # Read␣
 ↪all sheets without assuming a header

cleaned_sheets = {}

for sheet_name, df in dfs.items():
    print(f"\n  Processing sheet: {sheet_name}")
    # Drop fully empty rows (entirely NaN)
    df.dropna(how="all", inplace=True)
    # Find the first row with valid data (assumes it has at least half non-null␣
 ↪values)
    for i in range(len(df)):
        non_null_count = df.iloc[i].notna().sum()
        if non_null_count > len(df.columns) / 2:
            valid_header_row = i
            break

    # Set detected row as header
    df.columns = df.iloc[valid_header_row].astype(str).str.strip()  # Remove␣
↪spaces from column names
    # Drop all rows above the detected header row
    df = df.iloc[valid_header_row + 1:].reset_index(drop=True)
    # Remove duplicate header rows appearing later
    df = df[df['Loc ID'] != 'Loc ID']
    # Print detected column names for debugging
    print(f" Finalized columns: {df.columns.tolist()}")
    # Ensure 'Loc ID' exists before proceeding
    if 'Loc ID' in df.columns:
        df = df[df['Loc ID'].notna()]
    else:
        print(f" Warning: 'Loc ID' not found in sheet '{sheet_name}'. Skipping␣
↪this sheet.")
        continue  # Skip if 'Loc ID' is missing
    # Convert 'Loc ID' to numeric
    df['Loc ID'] = pd.to_numeric(df['Loc ID'], errors='coerce')
    # Store cleaned sheet
    cleaned_sheets[sheet_name] = df
# -------------------------- #
#   Now Filter and Combine Data #
# -------------------------- #
# Specify the Loc IDs to retrieve
given_ids = [100022, 100023, 100085, 100086, 100088, 100100, 100101, 100102,␣
↪100112, 100113, 100117, 100348, 100677,
             100678, 100679, 100918, 100980, 100982, 101240, 101385, 101597,␣
↪101598, 101599, 101600, 101890, 101891,
             101902, 101903, 101904]
# Specify the columns to retrieve
columns_to_select = ['Loc ID', 'Route', 'Start', 'End', 'AADT 2016']
all_filtered_data = []

# Process cleaned sheets
for sheet_name, df in cleaned_sheets.items():
    print(f" Filtering data from sheet: {sheet_name}")
    # Ensure 'Loc ID' exists before filtering
    if 'Loc ID' in df.columns:
        filtered_data = df[df['Loc ID'].isin(given_ids)][columns_to_select]
        if not filtered_data.empty:
            all_filtered_data.append(filtered_data)
    else:
        print(f" Warning: 'Loc ID' not found in sheet '{sheet_name}'. Skipping␣
↪this sheet.")
```

```python
# Combine all filtered data
if all_filtered_data:
    combined_data = pd.concat(all_filtered_data, ignore_index=True).
 ↪drop_duplicates()
    # Get the first 4 letters of the original file name (without extension)
    file_name = os.path.splitext(os.path.basename(file_path))[0][:4]
    # Create output file path with the first 4 letters + filtered data
    output_file_path = f'/Users/kishupatel/Desktop/{file_name}_filtered_data.
 ↪xlsx'
    # Save final output
    with pd.ExcelWriter(output_file_path, engine='openpyxl') as writer:
        combined_data.to_excel(writer, sheet_name='CombinedData', index=False)
    print(f"Filtered data saved to: {output_file_path}")
else:
    print("\n  No matching data found for the specified Loc IDs.")
```

```
 Processing sheet: AADT 2016
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2016', 'AADT Derivation Code', 'K Factor',
'D Factor', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2035
AADT']

 Processing sheet: INTERSTATES
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2016', 'AADT Derivation Code', 'K Factor',
'D Factor', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2035
AADT']

 Processing sheet: US ROUTES
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2016', 'AADT Derivation Code', 'K Factor',
'D Factor', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2035
AADT']

 Processing sheet: STATE ROUTES
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2016', 'AADT Derivation Code', 'K Factor',
'D Factor', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', 'Future
AADT']
 Filtering data from sheet: AADT 2016
 Filtering data from sheet: INTERSTATES
 Filtering data from sheet: US ROUTES
 Filtering data from sheet: STATE ROUTES
Filtered data saved to: /Users/kishupatel/Desktop/2016_filtered_data.xlsx
```

This code will extract traffic data for the year 2017

```
[44]: # Read the Excel file
file_path = '/Users/kishupatel/Desktop/DesFert/ADOT/2017-aadt-Publication.xlsm'
dfs = pd.read_excel(file_path, sheet_name=None, header=None, dtype=str)  # Read␣
 ↪all sheets without assuming a header

cleaned_sheets = {}

for sheet_name, df in dfs.items():
    print(f"\n  Processing sheet: {sheet_name}")
    # Drop fully empty rows (entirely NaN)
    df.dropna(how="all", inplace=True)
    # Find the first row with valid data (assumes it has at least half non-null␣
 ↪values)
    for i in range(len(df)):
        non_null_count = df.iloc[i].notna().sum()
        if non_null_count > len(df.columns) / 2:
            valid_header_row = i
            break

    # Set detected row as header
    df.columns = df.iloc[valid_header_row].astype(str).str.strip()  # Remove␣
 ↪spaces from column names
    # Drop all rows above the detected header row
    df = df.iloc[valid_header_row + 1:].reset_index(drop=True)
    # Remove duplicate header rows appearing later
    df = df[df['Loc ID'] != 'Loc ID']
    # Print detected column names for debugging
    print(f"  Finalized columns: {df.columns.tolist()}")
    # Ensure 'Loc ID' exists before proceeding
    if 'Loc ID' in df.columns:
        df = df[df['Loc ID'].notna()]
    else:
        print(f"  Warning: 'Loc ID' not found in sheet '{sheet_name}'. Skipping␣
 ↪this sheet.")
        continue  # Skip if 'Loc ID' is missing
    # Convert 'Loc ID' to numeric
    df['Loc ID'] = pd.to_numeric(df['Loc ID'], errors='coerce')
    # Store cleaned sheet
    cleaned_sheets[sheet_name] = df
# --------------------------- #
#   Now Filter and Combine Data #
# --------------------------- #
# Specify the Loc IDs to retrieve
given_ids = [100022, 100023, 100085, 100086, 100088, 100100, 100101, 100102,␣
 ↪100112, 100113, 100117, 100348, 100677,
             100678, 100679, 100918, 100980, 100982, 101240, 101385, 101597,␣
 ↪101598, 101599, 101600, 101890, 101891,
             101902, 101903, 101904]
# Specify the columns to retrieve
columns_to_select = ['Loc ID', 'Route', 'Start', 'End', 'AADT 2017']
all_filtered_data = []

# Process cleaned sheets
for sheet_name, df in cleaned_sheets.items():
    print(f"  Filtering data from sheet: {sheet_name}")
    # Ensure 'Loc ID' exists before filtering
    if 'Loc ID' in df.columns:
        filtered_data = df[df['Loc ID'].isin(given_ids)][columns_to_select]
        if not filtered_data.empty:
            all_filtered_data.append(filtered_data)
    else:
        print(f"  Warning: 'Loc ID' not found in sheet '{sheet_name}'. Skipping␣
 ↪this sheet.")

# Combine all filtered data
if all_filtered_data:
    combined_data = pd.concat(all_filtered_data, ignore_index=True).
 ↪drop_duplicates()
    # Get the first 4 letters of the original file name (without extension)
    file_name = os.path.splitext(os.path.basename(file_path))[0][:4]
    # Create output file path with the first 4 letters + filtered data
    output_file_path = f'/Users/kishupatel/Desktop/{file_name}_filtered_data.
 ↪xlsx'
    # Save final output
    with pd.ExcelWriter(output_file_path, engine='openpyxl') as writer:
        combined_data.to_excel(writer, sheet_name='CombinedData', index=False)
    print(f"Filtered data saved to: {output_file_path}")
else:
    print("\n  No matching data found for the specified Loc IDs.")
```

```
  Processing sheet: 2017 AADTS
  Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2017', 'AADT Derivation Code', 'K Factor
%', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2035
Future AADT']

  Processing sheet: STATE ROUTES
  Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2017', 'AADT Derivation Code', 'K Factor
%', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2035
Future AADT']

  Processing sheet: US ROUTES
```

```
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2017', 'AADT Derivation Code', 'K Factor
%', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2035
Future AADT']

 Processing sheet: INTERSTATES
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2017', 'AADT Derivation Code', 'K Factor
%', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2035
Future AADT']
 Filtering data from sheet: 2017 AADTS
 Filtering data from sheet: STATE ROUTES
 Filtering data from sheet: US ROUTES
 Filtering data from sheet: INTERSTATES
Filtered data saved to: /Users/kishupatel/Desktop/2017_filtered_data.xlsx
```

This code will extract traffic data for the year 2018

[47]:
```python
# Read the Excel file
file_path = '/Users/kishupatel/Desktop/DesFert/ADOT/2018-AADT-PUBLICATION.xlsx'
dfs = pd.read_excel(file_path, sheet_name=None, header=None, dtype=str)  # Read␣
 ↪all sheets without assuming a header

cleaned_sheets = {}

for sheet_name, df in dfs.items():
    print(f"\n  Processing sheet: {sheet_name}")
    # Drop fully empty rows (entirely NaN)
    df.dropna(how="all", inplace=True)
    # Find the first row with valid data (assumes it has at least half non-null␣
 ↪values)
    for i in range(len(df)):
        non_null_count = df.iloc[i].notna().sum()
        if non_null_count > len(df.columns) / 2:
            valid_header_row = i
            break

    # Set detected row as header
    df.columns = df.iloc[valid_header_row].astype(str).str.strip()  # Remove␣
 ↪spaces from column names
    # Drop all rows above the detected header row
    df = df.iloc[valid_header_row + 1:].reset_index(drop=True)
    # Remove duplicate header rows appearing later
    df = df[df['Loc ID'] != 'Loc ID']
    # Print detected column names for debugging
    print(f"  Finalized columns: {df.columns.tolist()}")
    # Ensure 'Loc ID' exists before proceeding
    if 'Loc ID' in df.columns:
        df = df[df['Loc ID'].notna()]
    else:
        print(f" Warning: 'Loc ID' not found in sheet '{sheet_name}'. Skipping␣
 ↪this sheet.")
        continue  # Skip if 'Loc ID' is missing
    # Convert 'Loc ID' to numeric
    df['Loc ID'] = pd.to_numeric(df['Loc ID'], errors='coerce')
    # Store cleaned sheet
    cleaned_sheets[sheet_name] = df
# -------------------------- #
#  Now Filter and Combine Data #
# -------------------------- #
# Specify the Loc IDs to retrieve
given_ids = [100022, 100023, 100085, 100086, 100088, 100100, 100101, 100102,␣
 ↪100112, 100113, 100117, 100348, 100677,
             100678, 100679, 100918, 100980, 100982, 101240, 101385, 101597,␣
 ↪101598, 101599, 101600, 101890, 101891,
             101902, 101903, 101904]
# Specify the columns to retrieve
columns_to_select = ['Loc ID', 'Route', 'Start', 'End', 'AADT 2018']
all_filtered_data = []

# Process cleaned sheets
for sheet_name, df in cleaned_sheets.items():
    print(f" Filtering data from sheet: {sheet_name}")
    # Ensure 'Loc ID' exists before filtering
    if 'Loc ID' in df.columns:
        filtered_data = df[df['Loc ID'].isin(given_ids)][columns_to_select]
        if not filtered_data.empty:
            all_filtered_data.append(filtered_data)
    else:
        print(f" Warning: 'Loc ID' not found in sheet '{sheet_name}'. Skipping␣
 ↪this sheet.")

# Combine all filtered data
if all_filtered_data:
    combined_data = pd.concat(all_filtered_data, ignore_index=True).
 ↪drop_duplicates()
    # Get the first 4 letters of the original file name (without extension)
    file_name = os.path.splitext(os.path.basename(file_path))[0][:4]
    # Create output file path with the first 4 letters + filtered data
    output_file_path = f'/Users/kishupatel/Desktop/{file_name}_filtered_data.
 ↪xlsx'
    # Save final output
    with pd.ExcelWriter(output_file_path, engine='openpyxl') as writer:
        combined_data.to_excel(writer, sheet_name='CombinedData', index=False)
    print(f"Filtered data saved to: {output_file_path}")
else:
    print("\n No matching data found for the specified Loc IDs.")
```

```
 Processing sheet: Arizona Interstate Sections
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2018', 'AADT Derivation Code', 'K Factor
%', 'D Factor %', 'AADT Motorcyles', 'AADT Passenger Cars', 'AADT Light Trucks',
'AADT Buses', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2040
Future AADT']

 Processing sheet: Arizona State Routes
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2018', 'AADT Derivation Code', 'K Factor
%', 'D Factor %', 'AADT Motorcyles', 'AADT Passenger Cars', 'AADT Light Trucks',
'AADT Buses', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2040
Future AADT']

 Processing sheet: ALL MAINLINE SECTIONS
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2018', 'AADT Derivation Code', 'K Factor
%', 'D Factor %', 'AADT Motorcyles', 'AADT Passenger Cars', 'AADT Light Trucks',
'AADT Buses', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2040
Future AADT']

 Processing sheet: Arizona US Routes
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2018', 'AADT Derivation Code', 'K Factor
%', 'D Factor %', 'AADT Motorcyles', 'AADT Passenger Cars', 'AADT Light Trucks',
'AADT Buses', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2040
Future AADT']
 Filtering data from sheet: Arizona Interstate Sections
 Filtering data from sheet: Arizona State Routes
 Filtering data from sheet: ALL MAINLINE SECTIONS
 Filtering data from sheet: Arizona US Routes
Filtered data saved to: /Users/kishupatel/Desktop/2018_filtered_data.xlsx
```

This code will extract traffic data for the year 2019

```
[50]: # Read the Excel file
file_path = '/Users/kishupatel/Desktop/DesFert/ADOT/2019-AADT-PUBLICATION.xlsx'
dfs = pd.read_excel(file_path, sheet_name=None, header=None, dtype=str)  # Read␣
 ↪all sheets without assuming a header

cleaned_sheets = {}

for sheet_name, df in dfs.items():
    print(f"\n  Processing sheet: {sheet_name}")
    # Drop fully empty rows (entirely NaN)
    df.dropna(how="all", inplace=True)
    # Find the first row with valid data (assumes it has at least half non-null␣
 ↪values)
    for i in range(len(df)):
        non_null_count = df.iloc[i].notna().sum()
        if non_null_count > len(df.columns) / 2:
            valid_header_row = i
            break

    # Set detected row as header
    df.columns = df.iloc[valid_header_row].astype(str).str.strip()  # Remove␣
 ↪spaces from column names
    # Drop all rows above the detected header row
    df = df.iloc[valid_header_row + 1:].reset_index(drop=True)
    # Remove duplicate header rows appearing later
    df = df[df['Loc ID'] != 'Loc ID']
    # Print detected column names for debugging
    print(f"  Finalized columns: {df.columns.tolist()}")
    # Ensure 'Loc ID' exists before proceeding
    if 'Loc ID' in df.columns:
        df = df[df['Loc ID'].notna()]
    else:
        print(f"  Warning: 'Loc ID' not found in sheet '{sheet_name}'. Skipping␣
 ↪this sheet.")
        continue  # Skip if 'Loc ID' is missing
    # Convert 'Loc ID' to numeric
    df['Loc ID'] = pd.to_numeric(df['Loc ID'], errors='coerce')
    # Store cleaned sheet
    cleaned_sheets[sheet_name] = df
# -------------------------- #
#   Now Filter and Combine Data #
# -------------------------- #
# Specify the Loc IDs to retrieve
given_ids = [100022, 100023, 100085, 100086, 100088, 100100, 100101, 100102,␣
 ↪100112, 100113, 100117, 100348, 100677,
             100678, 100679, 100918, 100980, 100982, 101240, 101385, 101597,␣
 ↪101598, 101599, 101600, 101890, 101891,
             101902, 101903, 101904]
# Specify the columns to retrieve
columns_to_select = ['Loc ID', 'Route', 'Start', 'End', 'AADT 2019']
all_filtered_data = []

# Process cleaned sheets
for sheet_name, df in cleaned_sheets.items():
    print(f"  Filtering data from sheet: {sheet_name}")
    # Ensure 'Loc ID' exists before filtering
    if 'Loc ID' in df.columns:
        filtered_data = df[df['Loc ID'].isin(given_ids)][columns_to_select]
        if not filtered_data.empty:
            all_filtered_data.append(filtered_data)
    else:
        print(f" Warning: 'Loc ID' not found in sheet '{sheet_name}'. Skipping␣
 ↪this sheet.")

# Combine all filtered data
if all_filtered_data:
    combined_data = pd.concat(all_filtered_data, ignore_index=True).
 ↪drop_duplicates()
    # Get the first 4 letters of the original file name (without extension)
    file_name = os.path.splitext(os.path.basename(file_path))[0][:4]
    # Create output file path with the first 4 letters + filtered data
    output_file_path = f'/Users/kishupatel/Desktop/{file_name}_filtered_data.
 ↪xlsx'
    # Save final output
    with pd.ExcelWriter(output_file_path, engine='openpyxl') as writer:
        combined_data.to_excel(writer, sheet_name='CombinedData', index=False)
    print(f"Filtered data saved to: {output_file_path}")
else:
    print("\n No matching data found for the specified Loc IDs.")
```

```
 Processing sheet: ALL MAINLINE SECTIONS
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2019', 'AADT Derivation Code', 'K Factor
%', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2040
Future AADT']

 Processing sheet: Arizona Interstate Sections
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2019', 'AADT Derivation Code', 'K Factor
%', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2040
Future AADT']

 Processing sheet: Arizona State Routes
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2019', 'AADT Derivation Code', 'K Factor
%', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2040
Future AADT']

 Processing sheet: Arizona US Routes
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2019', 'AADT Derivation Code', 'K Factor
%', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2040
Future AADT']
 Filtering data from sheet: ALL MAINLINE SECTIONS
 Filtering data from sheet: Arizona Interstate Sections
 Filtering data from sheet: Arizona State Routes
 Filtering data from sheet: Arizona US Routes
Filtered data saved to: /Users/kishupatel/Desktop/2019_filtered_data.xlsx
```

This code will extract traffic data for the year 2020

[53]:
```
# Read the Excel file
file_path = '/Users/kishupatel/Desktop/DesFert/ADOT/2020-AADT-PUBLICATION.xlsx'
dfs = pd.read_excel(file_path, sheet_name=None, header=None, dtype=str)  # Read␣
↪all sheets without assuming a header

cleaned_sheets = {}

for sheet_name, df in dfs.items():
    print(f"\n Processing sheet: {sheet_name}")
    # Drop fully empty rows (entirely NaN)
    df.dropna(how="all", inplace=True)
    # Find the first row with valid data (assumes it has at least half non-null␣
↪values)
    for i in range(len(df)):
        non_null_count = df.iloc[i].notna().sum()
        if non_null_count > len(df.columns) / 2:
            valid_header_row = i
            break

    # Set detected row as header
    df.columns = df.iloc[valid_header_row].astype(str).str.strip()  # Remove␣
↪spaces from column names
    # Drop all rows above the detected header row
    df = df.iloc[valid_header_row + 1:].reset_index(drop=True)
    # Remove duplicate header rows appearing later
    df = df[df['Loc ID'] != 'Loc ID']
    # Print detected column names for debugging
    print(f" Finalized columns: {df.columns.tolist()}")
    # Ensure 'Loc ID' exists before proceeding
    if 'Loc ID' in df.columns:
        df = df[df['Loc ID'].notna()]
    else:
        print(f" Warning: 'Loc ID' not found in sheet '{sheet_name}'. Skipping␣
↪this sheet.")
        continue  # Skip if 'Loc ID' is missing
    # Convert 'Loc ID' to numeric
    df['Loc ID'] = pd.to_numeric(df['Loc ID'], errors='coerce')
    # Store cleaned sheet
    cleaned_sheets[sheet_name] = df
# --------------------------- #
#  Now Filter and Combine Data #
# --------------------------- #
# Specify the Loc IDs to retrieve
given_ids = [100022, 100023, 100085, 100086, 100088, 100100, 100101, 100102,␣
 ↪100112, 100113, 100117, 100348, 100677,
             100678, 100679, 100918, 100980, 100982, 101240, 101385, 101597,␣
 ↪101598, 101599, 101600, 101890, 101891,
             101902, 101903, 101904]
# Specify the columns to retrieve
columns_to_select = ['Loc ID', 'Route', 'Start', 'End', 'AADT 2020']
all_filtered_data = []

# Process cleaned sheets
for sheet_name, df in cleaned_sheets.items():
    print(f" Filtering data from sheet: {sheet_name}")
    # Ensure 'Loc ID' exists before filtering
    if 'Loc ID' in df.columns:
        filtered_data = df[df['Loc ID'].isin(given_ids)][columns_to_select]
        if not filtered_data.empty:
            all_filtered_data.append(filtered_data)
    else:
        print(f" Warning: 'Loc ID' not found in sheet '{sheet_name}'. Skipping␣
 ↪this sheet.")

# Combine all filtered data
if all_filtered_data:
    combined_data = pd.concat(all_filtered_data, ignore_index=True).
 ↪drop_duplicates()
    # Get the first 4 letters of the original file name (without extension)
    file_name = os.path.splitext(os.path.basename(file_path))[0][:4]
    # Create output file path with the first 4 letters + filtered data
    output_file_path = f'/Users/kishupatel/Desktop/{file_name}_filtered_data.
 ↪xlsx'
    # Save final output
    with pd.ExcelWriter(output_file_path, engine='openpyxl') as writer:
        combined_data.to_excel(writer, sheet_name='CombinedData', index=False)
    print(f"Filtered data saved to: {output_file_path}")
else:
    print("\n No matching data found for the specified Loc IDs.")
```

```
 Processing sheet: ALL MAINLINE SECTIONS
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2020', 'AADT Derivation Code', 'K Factor
%', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2040
Future AADT']

  Processing sheet: Arizona Interstate Sections
  Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2020', 'AADT Derivation Code', 'K Factor
%', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2040
Future AADT']

  Processing sheet: Arizona State Routes
  Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2020', 'AADT Derivation Code', 'K Factor
%', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2040
Future AADT']

  Processing sheet: Arizona US Routes
  Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2020', 'AADT Derivation Code', 'K Factor
%', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2040
Future AADT']
  Filtering data from sheet: ALL MAINLINE SECTIONS
  Filtering data from sheet: Arizona Interstate Sections
  Filtering data from sheet: Arizona State Routes
  Filtering data from sheet: Arizona US Routes
Filtered data saved to: /Users/kishupatel/Desktop/2020_filtered_data.xlsx
```

This code will extract traffic data for the year 2021

[56]:
```python
# Read the Excel file
file_path = '/Users/kishupatel/Desktop/DesFert/ADOT/
 ↪2021-AADT-PUBLICATION_20220615.xlsx'
dfs = pd.read_excel(file_path, sheet_name=None, header=None, dtype=str)  # Read␣
 ↪all sheets without assuming a header

cleaned_sheets = {}

for sheet_name, df in dfs.items():
    print(f"\n  Processing sheet: {sheet_name}")
    # Drop fully empty rows (entirely NaN)
    df.dropna(how="all", inplace=True)
    # Find the first row with valid data (assumes it has at least half non-null␣
 ↪values)
    for i in range(len(df)):
        non_null_count = df.iloc[i].notna().sum()
        if non_null_count > len(df.columns) / 2:
            valid_header_row = i
            break

    # Set detected row as header
    df.columns = df.iloc[valid_header_row].astype(str).str.strip()  # Remove␣
  ↪spaces from column names
    # Drop all rows above the detected header row
    df = df.iloc[valid_header_row + 1:].reset_index(drop=True)
    # Remove duplicate header rows appearing later
    df = df[df['Loc ID'] != 'Loc ID']
    # Print detected column names for debugging
    print(f" Finalized columns: {df.columns.tolist()}")
    # Ensure 'Loc ID' exists before proceeding
    if 'Loc ID' in df.columns:
        df = df[df['Loc ID'].notna()]
    else:
        print(f" Warning: 'Loc ID' not found in sheet '{sheet_name}'. Skipping␣
  ↪this sheet.")
        continue  # Skip if 'Loc ID' is missing
    # Convert 'Loc ID' to numeric
    df['Loc ID'] = pd.to_numeric(df['Loc ID'], errors='coerce')
    # Store cleaned sheet
    cleaned_sheets[sheet_name] = df
# -------------------------- #
#   Now Filter and Combine Data #
# -------------------------- #
# Specify the Loc IDs to retrieve
given_ids = [100022, 100023, 100085, 100086, 100088, 100100, 100101, 100102,␣
  ↪100112, 100113, 100117, 100348, 100677,
             100678, 100679, 100918, 100980, 100982, 101240, 101385, 101597,␣
  ↪101598, 101599, 101600, 101890, 101891,
             101902, 101903, 101904]
# Specify the columns to retrieve
columns_to_select = ['Loc ID', 'Route', 'Start', 'End', 'AADT 2021']
all_filtered_data = []

# Process cleaned sheets
for sheet_name, df in cleaned_sheets.items():
    print(f" Filtering data from sheet: {sheet_name}")
    # Ensure 'Loc ID' exists before filtering
    if 'Loc ID' in df.columns:
        filtered_data = df[df['Loc ID'].isin(given_ids)][columns_to_select]
        if not filtered_data.empty:
            all_filtered_data.append(filtered_data)
    else:
        print(f" Warning: 'Loc ID' not found in sheet '{sheet_name}'. Skipping␣
  ↪this sheet.")

# Combine all filtered data
if all_filtered_data:
    combined_data = pd.concat(all_filtered_data, ignore_index=True).
 ↪drop_duplicates()
    # Get the first 4 letters of the original file name (without extension)
    file_name = os.path.splitext(os.path.basename(file_path))[0][:4]
    # Create output file path with the first 4 letters + filtered data
    output_file_path = f'/Users/kishupatel/Desktop/{file_name}_filtered_data.
 ↪xlsx'
    # Save final output
    with pd.ExcelWriter(output_file_path, engine='openpyxl') as writer:
        combined_data.to_excel(writer, sheet_name='CombinedData', index=False)
    print(f"Filtered data saved to: {output_file_path}")
else:
    print("\n  No matching data found for the specified Loc IDs.")
```

```
 Processing sheet: ALL MAINLINE SECTIONS
 Finalized columns: ['Index', 'Loc ID', 'Route', 'BMP', 'Start', 'TCS MP',
'EMP', 'End', 'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2021', 'AADT Source Code',
'K Factor %', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor
%', '2040 Future AADT']

 Processing sheet: Arizona Interstate Sections
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2021', 'AADT Souce Code', 'K Factor %', 'D
Factor %', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2040 Future
AADT']

 Processing sheet: Arizona State Routes
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2021', 'AADT Souce Code', 'K Factor %', 'D
Factor %', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2040 Future
AADT']

 Processing sheet: Arizona US Routes
 Finalized columns: ['Loc ID', 'Route', 'BMP', 'Start', 'TCS MP', 'EMP', 'End',
'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2021', 'AADT Souce Code', 'K Factor %', 'D
Factor %', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2040 Future
AADT']
 Filtering data from sheet: ALL MAINLINE SECTIONS
 Filtering data from sheet: Arizona Interstate Sections
 Filtering data from sheet: Arizona State Routes
 Filtering data from sheet: Arizona US Routes
Filtered data saved to: /Users/kishupatel/Desktop/2021_filtered_data.xlsx
```

This code will extract traffic data for the year 2022

[59]:
```
# Read the Excel file
```

```
file_path = '/Users/kishupatel/Desktop/DesFert/ADOT/
↪2022-AADT-PUBLICATION_20230825.xlsx'
dfs = pd.read_excel(file_path, sheet_name=None, header=None, dtype=str)  # Read␣
↪all sheets without assuming a header

cleaned_sheets = {}

for sheet_name, df in dfs.items():
    print(f"\n  Processing sheet: {sheet_name}")
    # Drop fully empty rows (entirely NaN)
    df.dropna(how="all", inplace=True)
    # Find the first row with valid data (assumes it has at least half non-null␣
↪values)
    for i in range(len(df)):
        non_null_count = df.iloc[i].notna().sum()
        if non_null_count > len(df.columns) / 2:
            valid_header_row = i
            break

    # Set detected row as header
    df.columns = df.iloc[valid_header_row].astype(str).str.strip()  # Remove␣
↪spaces from column names
    # Drop all rows above the detected header row
    df = df.iloc[valid_header_row + 1:].reset_index(drop=True)
    # Remove duplicate header rows appearing later
    df = df[df['Loc ID'] != 'Loc ID']
    # Print detected column names for debugging
    print(f"  Finalized columns: {df.columns.tolist()}")
    # Ensure 'Loc ID' exists before proceeding
    if 'Loc ID' in df.columns:
        df = df[df['Loc ID'].notna()]
    else:
        print(f"  Warning: 'Loc ID' not found in sheet '{sheet_name}'. Skipping␣
↪this sheet.")
        continue  # Skip if 'Loc ID' is missing
    # Convert 'Loc ID' to numeric
    df['Loc ID'] = pd.to_numeric(df['Loc ID'], errors='coerce')
    # Store cleaned sheet
    cleaned_sheets[sheet_name] = df
# -------------------------- #
#   Now Filter and Combine Data #
# -------------------------- #
# Specify the Loc IDs to retrieve
given_ids = [100022, 100023, 100085, 100086, 100088, 100100, 100101, 100102,␣
↪100112, 100113, 100117, 100348, 100677,
             100678, 100679, 100918, 100980, 100982, 101240, 101385, 101597,␣
↪101598, 101599, 101600, 101890, 101891,
             101902, 101903, 101904]
# Specify the columns to retrieve
columns_to_select = ['Loc ID', 'Road', 'FromRoad', 'ToRoad', 'AADT 2022']
all_filtered_data = []

# Process cleaned sheets
for sheet_name, df in cleaned_sheets.items():
    print(f" Filtering data from sheet: {sheet_name}")
    # Ensure 'Loc ID' exists before filtering
    if 'Loc ID' in df.columns:
        filtered_data = df[df['Loc ID'].isin(given_ids)][columns_to_select]
        if not filtered_data.empty:
            all_filtered_data.append(filtered_data)
    else:
        print(f" Warning: 'Loc ID' not found in sheet '{sheet_name}'. Skipping␣
↪this sheet.")

# Combine all filtered data
if all_filtered_data:
    combined_data = pd.concat(all_filtered_data, ignore_index=True).
↪drop_duplicates()
    # Get the first 4 letters of the original file name (without extension)
    file_name = os.path.splitext(os.path.basename(file_path))[0][:4]
    # Create output file path with the first 4 letters + filtered data
    output_file_path = f'/Users/kishupatel/Desktop/{file_name}_filtered_data.
↪xlsx'
    # Save final output
    with pd.ExcelWriter(output_file_path, engine='openpyxl') as writer:
        combined_data.to_excel(writer, sheet_name='CombinedData', index=False)
    print(f"Filtered data saved to: {output_file_path}")
else:
    print("\n No matching data found for the specified Loc IDs.")
```

```
 Processing sheet: 2022 ALL ADOT SECTIONS
 Finalized columns: ['SectionJoinID', 'Reference', 'Loc ID',
'TrafficSectionType', 'RouteId', 'Miles', 'Road', 'FromMeasure', 'BMP',
'FromRoad', 'TCS MP', 'ToMeasure', 'EMP', 'ToRoad', 'Pos Dir AADT', 'Neg Dir
AADT', 'AADT 2022', 'AADT Source Dataset', 'K Factor %', 'D Factor %', 'AADT
Single Trucks', 'AADT Combo Trucks', 'T Factor %', '2042 Future AADT', 'Index',
'nan', '2035', 'nan', 'CheckMP']

 Processing sheet: Arizona Interstate Sections
 Finalized columns: ['SectionJoinID', 'Reference', 'Loc ID',
'TrafficSectionType', 'RouteId', 'Miles', 'Road', 'BMP', 'FromRoad', 'TCS MP',
'EMP', 'ToRoad', 'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2022', 'AADT Source
Dataset', 'K Factor %', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks',
'T Factor %', '2042 Future AADT', 'nan', '2031']

 Processing sheet: Arizona State Routes
 Finalized columns: ['SectionJoinID', 'Reference', 'Loc ID',
'TrafficSectionType', 'RouteId', 'Miles', 'Road', 'BMP', 'FromRoad', 'TCS MP',
'EMP', 'ToRoad', 'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2022', 'AADT Source
Dataset', 'K Factor %', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks',
'T Factor %', '2042 Future AADT', 'Index', 'nan', '2042']

 Processing sheet: Arizona US Routes
 Finalized columns: ['SectionJoinID', 'Reference', 'Loc ID',
'TrafficSectionType', 'RouteId', 'Miles', 'Road', 'BMP', 'FromRoad', 'TCS MP',
'EMP', 'ToRoad', 'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2022', 'AADT Source
Dataset', 'K Factor %', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks',
'T Factor %', '2042 Future AADT', 'Index', 'nan', '2035']

 Processing sheet: Ramps and Frontage Roads
 Finalized columns: ['SectionJoinID', 'Reference', 'Loc ID',
'TrafficSectionType', 'RouteId', 'Miles', 'Road', 'FromMeasure', 'FromRoad',
'ToMeasure', 'ToRoad', 'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2022', 'AADT Source
Dataset', 'K Factor %', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks',
'T Factor %', '2042 Future AADT', 'nan', '2042']
 Filtering data from sheet: 2022 ALL ADOT SECTIONS
 Filtering data from sheet: Arizona Interstate Sections
 Filtering data from sheet: Arizona State Routes
 Filtering data from sheet: Arizona US Routes
 Filtering data from sheet: Ramps and Frontage Roads
Filtered data saved to: /Users/kishupatel/Desktop/2022_filtered_data.xlsx
```

This code will extract traffic data for the year 2023

[61]:
```python
# Read the Excel file
file_path = '/Users/kishupatel/Desktop/DesFert/ADOT/
 ↪2023-AADT-PUBLICATION_20240704.xlsx'
dfs = pd.read_excel(file_path, sheet_name=None, header=None, dtype=str)  # Read␣
 ↪all sheets without assuming a header

cleaned_sheets = {}

for sheet_name, df in dfs.items():
    print(f"\n  Processing sheet: {sheet_name}")
    # Drop fully empty rows (entirely NaN)
    df.dropna(how="all", inplace=True)
    # Find the first row with valid data (assumes it has at least half non-null␣
 ↪values)
    for i in range(len(df)):
        non_null_count = df.iloc[i].notna().sum()
        if non_null_count > len(df.columns) / 2:
            valid_header_row = i
            break

    # Set detected row as header
    df.columns = df.iloc[valid_header_row].astype(str).str.strip()  # Remove␣
↪spaces from column names
    # Drop all rows above the detected header row
    df = df.iloc[valid_header_row + 1:].reset_index(drop=True)
    # Remove duplicate header rows appearing later
    df = df[df['Loc ID'] != 'Loc ID']
    # Print detected column names for debugging
    print(f" Finalized columns: {df.columns.tolist()}")
    # Ensure 'Loc ID' exists before proceeding
    if 'Loc ID' in df.columns:
        df = df[df['Loc ID'].notna()]
    else:
        print(f" Warning: 'Loc ID' not found in sheet '{sheet_name}'. Skipping␣
↪this sheet.")
        continue  # Skip if 'Loc ID' is missing
    # Convert 'Loc ID' to numeric
    df['Loc ID'] = pd.to_numeric(df['Loc ID'], errors='coerce')
    # Store cleaned sheet
    cleaned_sheets[sheet_name] = df
# -------------------------- #
#  Now Filter and Combine Data #
# -------------------------- #
# Specify the Loc IDs to retrieve
given_ids = [100022, 100023, 100085, 100086, 100088, 100100, 100101, 100102,␣
↪100112, 100113, 100117, 100348, 100677,
             100678, 100679, 100918, 100980, 100982, 101240, 101385, 101597,␣
↪101598, 101599, 101600, 101890, 101891,
             101902, 101903, 101904]
# Specify the columns to retrieve
columns_to_select = ['Loc ID', 'Road', 'FromRoad', 'ToRoad', 'AADT 2023']
all_filtered_data = []

# Process cleaned sheets
for sheet_name, df in cleaned_sheets.items():
    print(f" Filtering data from sheet: {sheet_name}")
    # Ensure 'Loc ID' exists before filtering
    if 'Loc ID' in df.columns:
        filtered_data = df[df['Loc ID'].isin(given_ids)][columns_to_select]
        if not filtered_data.empty:
            all_filtered_data.append(filtered_data)
    else:
        print(f"  Warning: 'Loc ID' not found in sheet '{sheet_name}'. Skipping␣
↪this sheet.")

# Combine all filtered data
if all_filtered_data:
    combined_data = pd.concat(all_filtered_data, ignore_index=True).
↪drop_duplicates()
    # Get the first 4 letters of the original file name (without extension)
    file_name = os.path.splitext(os.path.basename(file_path))[0][:4]
    # Create output file path with the first 4 letters + filtered data
    output_file_path = f'/Users/kishupatel/Desktop/{file_name}_filtered_data.
↪xlsx'
    # Save final output
    with pd.ExcelWriter(output_file_path, engine='openpyxl') as writer:
        combined_data.to_excel(writer, sheet_name='CombinedData', index=False)
    print(f"Filtered data saved to: {output_file_path}")
else:
    print("\n  No matching data found for the specified Loc IDs.")
```

```
 Processing sheet: 2023 ALL ADOT SECTIONS
 Finalized columns: ['Section JoinID', 'Reference', 'Loc ID', 'Traffic Section
Type', 'RouteId', 'Miles', 'Road', 'From Measure', 'BMP', 'FromRoad', 'TCS MP',
'To Measure', 'EMP', 'ToRoad', 'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2023',
'AADT Source Dataset', 'K Factor %', 'D Factor %', 'AADT Single Trucks', 'AADT
Combo Trucks', 'T Factor %', '2043 Future AADT', 'Type', 'Index']

 Processing sheet: Arizona Interstate Sections
 Finalized columns: ['SectionJoinID', 'Reference', 'Loc ID',
'TrafficSectionType', 'RouteId', 'Miles', 'Road', 'BMP', 'FromRoad', 'TCS MP',
'EMP', 'ToRoad', 'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2023', 'AADT Source
Dataset', 'K Factor %', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks',
'T Factor %', '2043 Future AADT', 'Index']

 Processing sheet: Arizona State Routes
 Finalized columns: ['SectionJoinID', 'Reference', 'Loc ID',
'TrafficSectionType', 'RouteId', 'Miles', 'Road', 'BMP', 'FromRoad', 'TCS MP',
'EMP', 'ToRoad', 'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2023', 'AADT Source
Dataset', 'K Factor %', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks',
'T Factor %', '2043 Future AADT', 'nan']

 Processing sheet: Arizona US Routes
 Finalized columns: ['SectionJoinID', 'Reference', 'Loc ID',
'TrafficSectionType', 'RouteId', 'Miles', 'Road', 'BMP', 'FromRoad', 'TCS MP',
'EMP', 'ToRoad', 'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2023', 'AADT Source
Dataset', 'K Factor %', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks',
'T Factor %', '2043 Future AADT', 'nan']
```

```
  Processing sheet: Ramps and Frontage Roads
  Finalized columns: ['SectionJoinID', 'Reference', 'Loc ID',
'TrafficSectionType', 'RouteId', 'Miles', 'Road', 'From Measure', 'FromRoad',
'To Measure', 'ToRoad', 'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2023', 'AADT
Source Dataset', 'K Factor %', 'D Factor %', 'AADT Single Trucks', 'AADT Combo
Trucks', 'T Factor %', '2043 Future AADT', 'nan', 'Index']

  Processing sheet: 2023 LOCAL SECTIONS
  Finalized columns: ['SectionJoinID', 'Reference', 'Loc ID', 'Traffic Section
Type', 'RouteId', 'Miles', 'Road', 'From Measure', 'FromRoad', 'To Measure',
'ToRoad', 'Pos Dir AADT', 'Neg Dir AADT', 'AADT 2023', 'AADT Source Dataset', 'K
Factor %', 'D Factor %', 'AADT Single Trucks', 'AADT Combo Trucks', 'T Factor
%', '2043 Future AADT']
  Filtering data from sheet: 2023 ALL ADOT SECTIONS
  Filtering data from sheet: Arizona Interstate Sections
  Filtering data from sheet: Arizona State Routes
  Filtering data from sheet: Arizona US Routes
  Filtering data from sheet: Ramps and Frontage Roads
  Filtering data from sheet: 2023 LOCAL SECTIONS
Filtered data saved to: /Users/kishupatel/Desktop/2023_filtered_data.xlsx
```