```
import pandas as pd

# Load dataset with correct encoding
births_df = pd.read_csv('/content/bt3-2023-births-time-series-csv.csv', encoding='ISO-8859-1')

# Extract and set headers from the correct row (row index 2)
births_df.columns = births_df.iloc[2]

# Keep relevant data starting from the next row (row index 3)
births_df_cleaned = births_df.iloc[3:, :14].reset_index(drop=True)

# Rename columns for clarity
births_df_cleaned.columns = ['Year', 'Total', 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

# Remove the 'Total' column (optional)
births_df_cleaned.drop(columns=['Total'], inplace=True)

# Reshape dataset from wide to long format
births_long = births_df_cleaned.melt(id_vars=['Year'], var_name='Month', value_name='Births')

# Clean numeric values by removing commas and converting to numeric
births_long['Births'] = births_long['Births'].str.replace(',', '').astype(float)

# Combine Year and Month to create a proper datetime column
births_long['Date'] = pd.to_datetime(births_long['Year'] + '-' + births_long['Month'], format='%Y-%b', errors='coerce')

# Drop rows with invalid dates
births_long.dropna(subset=['Date'], inplace=True)

# Final clean dataset
births_final_cleaned = births_long[['Date', 'Births']].sort_values(by='Date').reset_index(drop=True)

# Save the cleaned dataset to CSV (optional)
births_final_cleaned.to_csv('cleaned_birth_rates.csv', index=False)

# Display the cleaned data
print(births_final_cleaned.head())
```

```
        Date  Births
0 1954-01-01  7716.0
1 1954-02-01  7161.0
2 1954-03-01  8579.0
3 1954-04-01  8084.0
4 1954-05-01  8304.0
```

```
import pandas as pd

# Load the dataset with correct encoding
lms_df = pd.read_csv('/content/lms.csv', encoding='ISO-8859-1', low_memory=False)

# Check the first 30 column names to find the correct ones
print(lms_df.columns[:20].tolist())
# Display first 15 rows to find potential date columns clearly
print(lms_df.iloc[:15, :5])
```

```
10                                                NaN
11                                                NaN
12                                                NaN
13                                                NaN
14                                                NaN

   AWE: Whole Economy Real Terms Level (Â£): Seasonally Adjusted Regular Pay  \
0                                                A2FC
1                                                  Â£
2                                                 NaN
3                                          17-07-2025
4                                       12 August 2025
5                                                 NaN
6                                                 NaN
7                                                 NaN
8                                                 NaN
9                                                 NaN
10                                                NaN
11                                                NaN
12                                                NaN
13                                                NaN
14                                                NaN

   AWE: Whole Economy Real Terms Year on Year Single Month Growth (%): Seasonally Adjusted Total Pay
0                                                A3WV
1                                                 NaN
2                                                 NaN
3                                          17-07-2025
4                                       12 August 2025
5                                                 NaN
6                                                 NaN
7                                                 NaN
8                                                 NaN
9                                                 NaN
10                                                NaN
11                                                NaN
12                                                NaN
13                                                NaN
14                                                NaN
```

```
import pandas as pd

# Load the dataset with correct encoding
lms_df = pd.read_csv('/content/lms.csv', encoding='ISO-8859-1', low_memory=False)

# Use 'Title' column as the time indicator (contains years)
# Extract relevant unemployment rate column for UK (e.g. one that actually exists in the file)
lms_df_subset = lms_df[['Title', 'Standardised ILO Unemployment rates - Total EU']]

# Rename columns for clarity
lms_df_subset.columns = ['Year', 'EU_Unemployment_Rate']

# Remove initial metadata rows (e.g. anything that is not a year)
lms_df_subset = lms_df_subset[lms_df_subset['Year'].str.isnumeric()].reset_index(drop=True)

# Convert Year to datetime (assume January of that year)
lms_df_subset['Date'] = pd.to_datetime(lms_df_subset['Year'] + '-01', format='%Y-%m')

# Convert unemployment rate to numeric
lms_df_subset['EU_Unemployment_Rate'] = pd.to_numeric(lms_df_subset['EU_Unemployment_Rate'], errors='coerce')

# Drop rows with missing or invalid values
lms_df_cleaned = lms_df_subset.dropna().reset_index(drop=True)

# Save the cleaned dataset to CSV (optional)
lms_df_cleaned.to_csv('cleaned_eu_unemployment.csv', index=False)

# Display the cleaned data
print(lms_df_cleaned.head())
```

```
   Year  EU_Unemployment_Rate       Date
0  2000                   9.5 2000-01-01
1  2001                   9.3 2001-01-01
2  2002                   9.6 2002-01-01
3  2003                   9.8 2003-01-01
4  2004                   9.9 2004-01-01
```

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Load the dataset
mm_df = pd.read_csv('/content/mm23.csv')

# Step 1: Clean column names
mm_df.columns = mm_df.columns.str.strip().str.replace(" ", "_").str.replace(r"[()]", "", regex=True)

# Step 2: Drop columns with more than 50% missing values
threshold = len(mm_df) * 0.5
mm_df = mm_df.dropna(thresh=threshold, axis=1)

# Step 3: Fill remaining missing values with median (for numerical columns)
for col in mm_df.select_dtypes(include=[np.number]).columns:
    mm_df[col].fillna(mm_df[col].median(), inplace=True)

# Step 4: Remove outliers using IQR
def remove_outliers_iqr(df, columns):
    for col in columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower = Q1 - 1.5 * IQR
        upper = Q3 + 1.5 * IQR
        df = df[(df[col] >= lower) & (df[col] <= upper)]
    return df

numerical_cols = mm_df.select_dtypes(include=[np.number]).columns.tolist()
mm_df = remove_outliers_iqr(mm_df, numerical_cols)

# Step 5: Normalize numeric columns
valid_numeric_cols = [col for col in numerical_cols if mm_df[col].dtype in [np.float64, np.int64] and mm_df[col].notnull().any()]

if valid_numeric_cols:
    scaler = MinMaxScaler()
    mm_df[valid_numeric_cols] = scaler.fit_transform(mm_df[valid_numeric_cols])

# Save cleaned dataset (optional)
mm_df.to_csv('cleaned_mm23.csv', index=False)

# Display cleaned dataset
print(mm_df.head())
```

```
/tmp/ipython-input-3438069199.py:6: DtypeWarning: Columns (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,2
  mm_df = pd.read_csv('/content/mm23.csv')
          Title RPI:Percentage_change_over_12_months_-_Food  \
0          CDID                                       CCYY
1       PreUnit                                        NaN
2          Unit                                          %
3  Release Date                                 16-07-2025
4  Next release                              20 August 2025

  Retail_Prices_Index:_Long_run_series:_1800_to_2024:_Jan_1974=100  \
0                                                  CDKO
1                                                   NaN
2                             Index, base year = 100
3                                         16-07-2025
4                                    20 August 2025

  RPI:_Percentage_change_over_1_month_-_Food  \
0                                       CDKP
1                                        NaN
2                                          %
3                                   16-07-2025
4                             20 August 2025

  RPI_All_Items_Excl_Mortgage_Interest_RPIX:_Percentage_change_over_12_months  \
0                                                  CDKQ
1                                                   NaN
2                                                     %
3                                            16-07-2025
4                                       20 August 2025

  RPI_All_Items:_Percentage_change_over_12_months:_Jan_1987=100  \
0                                                  CZBH
1                                                   NaN
2                                                     %
```

```
3                                        16-07-2025
4                                    20 August 2025

  RPI:Percentage_change_over_12_months_-_all_items_excluding_housing  \
0                                                CZBI
1                                                 NaN
2                                                   %
3                                        16-07-2025
4                                    20 August 2025

  RPI:Percentage_change_over_12_months_-_Food_and_catering  \
0                                                CZBJ
1                                                 NaN
2                                                   %
3                                        16-07-2025
4                                    20 August 2025

  RPI:Percentage_change_over_12_months_-_Seasonal_food  \
0                                                CZBP
1                                                 NaN
2                                                   %
3                                        16-07-2025
4                                    20 August 2025
```

```
import pandas as pd

# Load the Scotland holidays dataset
holidays_df = pd.read_csv('scotland_holidays.csv')

# Step 1: Convert 'date' to datetime
holidays_df['date'] = pd.to_datetime(holidays_df['date'], errors='coerce')

# Step 2: Fill missing notes with 'None'
holidays_df['notes'] = holidays_df['notes'].fillna('None')

# Step 3: One-hot encode holidays for use in time series models (optional)
holidays_df['is_holiday'] = True

# Step 4: Create a cleaned dataset with only essential fields for merging
cleaned_holidays = holidays_df[['date', 'title', 'is_holiday']]

# Save cleaned holidays to file (optional)
cleaned_holidays.to_csv('cleaned_scotland_holidays.csv', index=False)

# Display result
print(cleaned_holidays.head())
```

```
        date                   title  is_holiday
0 2019-01-01          New Year’s Day        True
1 2019-01-02             2nd January        True
2 2019-04-19             Good Friday        True
3 2019-05-06  Early May bank holiday        True
4 2019-05-27     Spring bank holiday        True
```

Start coding or generate with AI.

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Load the Scotland unemployment rate dataset
series_df = pd.read_csv('series-280725.csv')

# Step 1: Rename columns for clarity
series_df.columns = ['Year', 'Unemployment_Rate']

# Step 2: Remove metadata rows (non-numeric years)
series_df = series_df[series_df['Year'].str.isnumeric()].copy()

# Step 3: Convert columns to appropriate types
series_df['Year'] = pd.to_datetime(series_df['Year'] + '-01', format='%Y-%m')
series_df['Unemployment_Rate'] = pd.to_numeric(series_df['Unemployment_Rate'], errors='coerce')

# Step 4: Handle missing values (drop rows with missing unemployment rate)
series_df.dropna(subset=['Unemployment_Rate'], inplace=True)
```

```
# Step 5: Normalize unemployment rate
scaler = MinMaxScaler()
series_df['Unemployment_Rate_Norm'] = scaler.fit_transform(series_df[['Unemployment_Rate']])

# Step 6: Save cleaned dataset (optional)
series_df.to_csv('cleaned_scotland_unemployment.csv', index=False)

# Display cleaned dataset
print(series_df.head())
```

```
          Year  Unemployment_Rate  Unemployment_Rate_Norm
7  1992-01-01                9.6                0.925373
8  1993-01-01               10.1                1.000000
9  1994-01-01                9.6                0.925373
10 1995-01-01                8.9                0.820896
11 1996-01-01                8.7                0.791045
```

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Step 1: Load dataset with correct header row
maternal_df = pd.read_csv('/content/table-1-(2014-15).csv', skiprows=12, dtype=str)

# Step 2: Clean column names
maternal_df.columns = maternal_df.columns.str.strip().str.replace(' ', '_').str.replace('[()]', '', regex=True)

# Step 3: Inspect and locate the actual provider name column
print("Available Columns:", maternal_df.columns.tolist())

# Step 4: Drop rows where likely provider name column is missing
if 'HE_provider' in maternal_df.columns:
    maternal_df.dropna(subset=['HE_provider'], inplace=True)
else:
    maternal_df.dropna(subset=[maternal_df.columns[0]], inplace=True)

# Step 5: Convert enrollment count column to numeric and drop NaNs
maternal_df['Number'] = pd.to_numeric(maternal_df.iloc[:, -1], errors='coerce')
maternal_df.dropna(subset=['Number'], inplace=True)

# Step 6: Remove outliers using IQR
Q1 = maternal_df['Number'].quantile(0.25)
Q3 = maternal_df['Number'].quantile(0.75)
IQR = Q3 - Q1
maternal_df = maternal_df[(maternal_df['Number'] >= Q1 - 1.5 * IQR) & (maternal_df['Number'] <= Q3 + 1.5 * IQR)]

# Step 7: Normalize 'Number' column
scaler = MinMaxScaler()
maternal_df['Number_Norm'] = scaler.fit_transform(maternal_df[['Number']])

# Step 8: Save and display cleaned dataset
maternal_df.to_csv('cleaned_maternal_enrollment.csv', index=False)
print(maternal_df.head())
```

```
Available Columns: ['Unnamed:_0', 'Unnamed:_1', 'Unnamed:_2', 'Unnamed:_3', 'Unnamed:_4', 'Unnamed:_5', 'Unnamed:_6', 'Unnamed:_7',
/tmp/ipython-input-221680273.py:32: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  maternal_df['Number_Norm'] = scaler.fit_transform(maternal_df[['Number']])
   Unnamed:_0                  Unnamed:_1 Unnamed:_2 Unnamed:_3 Unnamed:_4  \
3    10007783  The University of Aberdeen        All        All        All
6    10007783  The University of Aberdeen        All        All        All
7    10007783  The University of Aberdeen        All        All        All
8    10007783  The University of Aberdeen        All        All        All
13   10007783  The University of Aberdeen        All        All        All

   Unnamed:_5 Unnamed:_6 Unnamed:_7         Unnamed:_8        Unnamed:_9  \
3         All        All    2014/15                Sex           Unknown
6         All        All    2014/15  Permanent address             Wales
7         All        All    2014/15  Permanent address  Northern Ireland
8         All        All    2014/15  Permanent address          Other UK
13        All        All    2014/15  Permanent address        Not known

   Unnamed:_10  Number  Number_Norm
3            0     0.0     0.000000
```

```
6           35    35.0     0.041916
7          230   230.0     0.275449
8            5     5.0     0.005988
13           0     0.0     0.000000
```

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Step 1: Load dataset with correct header row for 2015-16
maternal_df = pd.read_csv('/content/table-1-(2015-16).csv', skiprows=12, dtype=str)

# Step 2: Clean column names
maternal_df.columns = maternal_df.columns.str.strip().str.replace(' ', '_').str.replace('[()]', '', regex=True)

# Step 3: Display available columns for clarity
print("Available Columns:", maternal_df.columns.tolist())

# Step 4: Drop rows with missing provider name (first column fallback if needed)
provider_column = maternal_df.columns[0]
maternal_df.dropna(subset=[provider_column], inplace=True)

# Step 5: Convert enrollment count column to numeric and drop missing values
count_column = maternal_df.columns[-1]  # Assume last column contains enrollment numbers
maternal_df['Number'] = pd.to_numeric(maternal_df[count_column], errors='coerce')
maternal_df.dropna(subset=['Number'], inplace=True)

# Step 6: Remove outliers using IQR
Q1 = maternal_df['Number'].quantile(0.25)
Q3 = maternal_df['Number'].quantile(0.75)
IQR = Q3 - Q1
maternal_df = maternal_df[(maternal_df['Number'] >= Q1 - 1.5 * IQR) & (maternal_df['Number'] <= Q3 + 1.5 * IQR)]

# Step 7: Normalize the 'Number' column
scaler = MinMaxScaler()
maternal_df['Number_Norm'] = scaler.fit_transform(maternal_df[['Number']])

# Step 8: Save and display the cleaned dataset
maternal_df.to_csv('cleaned_maternal_enrollment_2015_16.csv', index=False)
print(maternal_df.head())
```

```
Available Columns: ['Unnamed:_0', 'Unnamed:_1', 'Unnamed:_2', 'Unnamed:_3', 'Unnamed:_4', 'Unnamed:_5', 'Unnamed:_6', 'Unnamed:_7',
/tmp/ipython-input-3859735901.py:31: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  maternal_df['Number_Norm'] = scaler.fit_transform(maternal_df[['Number']])
  Unnamed:_0                 Unnamed:_1 Unnamed:_2 Unnamed:_3 Unnamed:_4  \
1   10008071  AA School of Architecture        All        All        All
2   10008071  AA School of Architecture        All        All        All
3   10008071  AA School of Architecture        All        All        All
4   10008071  AA School of Architecture        All        All        All
5   10008071  AA School of Architecture        All        All        All

  Unnamed:_5 Unnamed:_6 Unnamed:_7         Unnamed:_8 Unnamed:_9 Unnamed:_10  \
1        All        All    2015/16                Sex     Female         240
2        All        All    2015/16                Sex       Male         230
3        All        All    2015/16                Sex    Unknown           0
4        All        All    2015/16  Permanent address    England          70
5        All        All    2015/16  Permanent address   Scotland           0

   Number  Number_Norm
1   240.0     0.363636
2   230.0     0.348485
3     0.0     0.000000
4    70.0     0.106061
5     0.0     0.000000
```

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Step 1: Load dataset with correct header row for 2018-19
maternal_df = pd.read_csv('/content/table-1-(2018-19).csv', skiprows=12, dtype=str)
```

```
# Step 2: Clean column names
maternal_df.columns = maternal_df.columns.str.strip().str.replace(' ', '_').str.replace('[()]', '', regex=True)

# Step 3: Display available columns for clarity
print("Available Columns:", maternal_df.columns.tolist())

# Step 4: Drop rows with missing provider name (first column fallback if needed)
provider_column = maternal_df.columns[0]
maternal_df.dropna(subset=[provider_column], inplace=True)

# Step 5: Convert enrollment count column to numeric and drop missing values
count_column = maternal_df.columns[-1]  # Assume last column contains enrollment numbers
maternal_df['Number'] = pd.to_numeric(maternal_df[count_column], errors='coerce')
maternal_df.dropna(subset=['Number'], inplace=True)

# Step 6: Remove outliers using IQR
Q1 = maternal_df['Number'].quantile(0.25)
Q3 = maternal_df['Number'].quantile(0.75)
IQR = Q3 - Q1
maternal_df = maternal_df[(maternal_df['Number'] >= Q1 - 1.5 * IQR) & (maternal_df['Number'] <= Q3 + 1.5 * IQR)]

# Step 7: Normalize the 'Number' column
scaler = MinMaxScaler()
maternal_df['Number_Norm'] = scaler.fit_transform(maternal_df[['Number']])

# Step 8: Save and display the cleaned dataset
maternal_df.to_csv('cleaned_maternal_enrollment_2018_19.csv', index=False)
print(maternal_df.head())
```

```
Available Columns: ['Unnamed:_0', 'Unnamed:_1', 'Unnamed:_2', 'Unnamed:_3', 'Unnamed:_4', 'Unnamed:_5', 'Unnamed:_6', 'Unnamed:_7',
/tmp/ipython-input-2977011558.py:31: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  maternal_df['Number_Norm'] = scaler.fit_transform(maternal_df[['Number']])
  Unnamed:_0                Unnamed:_1 Unnamed:_2 Unnamed:_3 Unnamed:_4  \
1   10008071  AA School of Architecture        All        All        All
2   10008071  AA School of Architecture        All        All        All
3   10008071  AA School of Architecture        All        All        All
4   10008071  AA School of Architecture        All        All        All
5   10008071  AA School of Architecture        All        All        All

  Unnamed:_5 Unnamed:_6 Unnamed:_7         Unnamed:_8 Unnamed:_9 Unnamed:_10  \
1        All        All    2018/19                Sex     Female         435
2        All        All    2018/19                Sex       Male         340
3        All        All    2018/19                Sex    Unknown           0
4        All        All    2018/19  Permanent address    England          90
5        All        All    2018/19  Permanent address   Scotland           0

   Number  Number_Norm
1   435.0     0.790909
2   340.0     0.618182
3     0.0     0.000000
4    90.0     0.163636
5     0.0     0.000000
```

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Step 1: Load dataset with correct header row for 2019-20
maternal_df = pd.read_csv('/content/table-1-(2019-20).csv', skiprows=12, dtype=str)

# Step 2: Clean column names
maternal_df.columns = maternal_df.columns.str.strip().str.replace(' ', '_').str.replace('[()]', '', regex=True)

# Step 3: Display available columns for clarity
print("Available Columns:", maternal_df.columns.tolist())

# Step 4: Drop rows with missing provider name (first column fallback if needed)
provider_column = maternal_df.columns[0]
maternal_df.dropna(subset=[provider_column], inplace=True)

# Step 5: Convert enrollment count column to numeric and drop missing values
count_column = maternal_df.columns[-1]  # Assume last column contains enrollment numbers
maternal_df['Number'] = pd.to_numeric(maternal_df[count_column], errors='coerce')
```

```
maternal_df.dropna(subset=['Number'], inplace=True)

# Step 6: Remove outliers using IQR
Q1 = maternal_df['Number'].quantile(0.25)
Q3 = maternal_df['Number'].quantile(0.75)
IQR = Q3 - Q1
maternal_df = maternal_df[(maternal_df['Number'] >= Q1 - 1.5 * IQR) & (maternal_df['Number'] <= Q3 + 1.5 * IQR)]

# Step 7: Normalize the 'Number' column
scaler = MinMaxScaler()
maternal_df['Number_Norm'] = scaler.fit_transform(maternal_df[['Number']])

# Step 8: Save and display the cleaned dataset
maternal_df.to_csv('cleaned_maternal_enrollment_2019_20.csv', index=False)
print(maternal_df.head())
```

```
Available Columns: ['Unnamed:_0', 'Unnamed:_1', 'Unnamed:_2', 'Unnamed:_3', 'Unnamed:_4', 'Unnamed:_5', 'Unnamed:_6', 'Unnamed:_7',
/tmp/ipython-input-348385628.py:31: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  maternal_df['Number_Norm'] = scaler.fit_transform(maternal_df[['Number']])
  Unnamed:_0                Unnamed:_1 Unnamed:_2 Unnamed:_3 Unnamed:_4  \
1   10008071  AA School of Architecture        All        All        All
2   10008071  AA School of Architecture        All        All        All
3   10008071  AA School of Architecture        All        All        All
4   10008071  AA School of Architecture        All        All        All
5   10008071  AA School of Architecture        All        All        All

  Unnamed:_5 Unnamed:_6 Unnamed:_7         Unnamed:_8 Unnamed:_9 Unnamed:_10  \
1        All        All    2019/20                Sex     Female         485
2        All        All    2019/20                Sex       Male         345
3        All        All    2019/20                Sex    Unknown           0
4        All        All    2019/20  Permanent address    England          85
5        All        All    2019/20  Permanent address   Scotland           0

   Number  Number_Norm
1   485.0     0.906542
2   345.0     0.644860
3     0.0     0.000000
4    85.0     0.158879
5     0.0     0.000000
```

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Step 1: Load dataset with correct header row for 2020-21
maternal_df = pd.read_csv('/content/table-1-(2020-21).csv', skiprows=12, dtype=str)

# Step 2: Clean column names
maternal_df.columns = maternal_df.columns.str.strip().str.replace(' ', '_').str.replace('[()]', '', regex=True)

# Step 3: Display available columns for clarity
print("Available Columns:", maternal_df.columns.tolist())

# Step 4: Drop rows with missing provider name (first column fallback if needed)
provider_column = maternal_df.columns[0]
maternal_df.dropna(subset=[provider_column], inplace=True)

# Step 5: Convert enrollment count column to numeric and drop missing values
count_column = maternal_df.columns[-1]  # Assume last column contains enrollment numbers
maternal_df['Number'] = pd.to_numeric(maternal_df[count_column], errors='coerce')
maternal_df.dropna(subset=['Number'], inplace=True)

# Step 6: Remove outliers using IQR
Q1 = maternal_df['Number'].quantile(0.25)
Q3 = maternal_df['Number'].quantile(0.75)
IQR = Q3 - Q1
maternal_df = maternal_df[(maternal_df['Number'] >= Q1 - 1.5 * IQR) & (maternal_df['Number'] <= Q3 + 1.5 * IQR)]

# Step 7: Normalize the 'Number' column
scaler = MinMaxScaler()
maternal_df['Number_Norm'] = scaler.fit_transform(maternal_df[['Number']])

# Step 8: Save and display the cleaned dataset
maternal_df.to_csv('cleaned_maternal_enrollment_2020_21.csv', index=False)
```

```
maternal_df.to_csv('cleaned_maternal_enrollment_2020_21.csv', index=False)
print(maternal_df.head())
```

```
Available Columns: ['Unnamed:_0', 'Unnamed:_1', 'Unnamed:_2', 'Unnamed:_3', 'Unnamed:_4', 'Unnamed:_5', 'Unnamed:_6', 'Unnamed:_7',
  Unnamed:_0                Unnamed:_1 Unnamed:_2 Unnamed:_3 Unnamed:_4  \
2   10008071  AA School of Architecture        All        All        All
3   10008071  AA School of Architecture        All        All        All
4   10008071  AA School of Architecture        All        All        All
5   10008071  AA School of Architecture        All        All        All
6   10008071  AA School of Architecture        All        All        All

  Unnamed:_5 Unnamed:_6 Unnamed:_7        Unnamed:_8 Unnamed:_9 Unnamed:_10  \
2        All        All    2020/21               Sex       Male         390
3        All        All    2020/21               Sex    Unknown           0
4        All        All    2020/21  Permanent address    England          90
5        All        All    2020/21  Permanent address   Scotland           0
6        All        All    2020/21  Permanent address      Wales           5

   Number  Number_Norm
2   390.0     0.742857
3     0.0     0.000000
4    90.0     0.171429
5     0.0     0.000000
6     5.0     0.009524
```

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Step 1: Load dataset with correct header row for 2020-21
maternal_df = pd.read_csv('/content/table-1-(2021-22).csv', skiprows=12, dtype=str)

# Step 2: Clean column names
maternal_df.columns = maternal_df.columns.str.strip().str.replace(' ', '_').str.replace('[()]', '', regex=True)

# Step 3: Display available columns for clarity
print("Available Columns:", maternal_df.columns.tolist())

# Step 4: Drop rows with missing provider name (first column fallback if needed)
provider_column = maternal_df.columns[0]
maternal_df.dropna(subset=[provider_column], inplace=True)

# Step 5: Convert enrollment count column to numeric and drop missing values
count_column = maternal_df.columns[-1]  # Assume last column contains enrollment numbers
maternal_df['Number'] = pd.to_numeric(maternal_df[count_column], errors='coerce')
maternal_df.dropna(subset=['Number'], inplace=True)

# Step 6: Remove outliers using IQR
Q1 = maternal_df['Number'].quantile(0.25)
Q3 = maternal_df['Number'].quantile(0.75)
IQR = Q3 - Q1
maternal_df = maternal_df[(maternal_df['Number'] >= Q1 - 1.5 * IQR) & (maternal_df['Number'] <= Q3 + 1.5 * IQR)]

# Step 7: Normalize the 'Number' column
scaler = MinMaxScaler()
maternal_df['Number_Norm'] = scaler.fit_transform(maternal_df[['Number']])

# Step 8: Save and display the cleaned dataset
maternal_df.to_csv('cleaned_maternal_enrollment_2020_21.csv', index=False)
print(maternal_df.head())
```

```
Available Columns: ['Unnamed:_0', 'Unnamed:_1', 'Unnamed:_2', 'Unnamed:_3', 'Unnamed:_4', 'Unnamed:_5', 'Unnamed:_6', 'Unnamed:_7',
/tmp/ipython-input-3168791248.py:31: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  maternal_df['Number_Norm'] = scaler.fit_transform(maternal_df[['Number']])
  Unnamed:_0                Unnamed:_1 Unnamed:_2 Unnamed:_3 Unnamed:_4  \
2   10008071  AA School of Architecture        All        All        All
3   10008071  AA School of Architecture        All        All        All
4   10008071  AA School of Architecture        All        All        All
5   10008071  AA School of Architecture        All        All        All
6   10008071  AA School of Architecture        All        All        All

  Unnamed:_5 Unnamed:_6 Unnamed:_7        Unnamed:_8 Unnamed:_9 Unnamed:_10  \
2        All        All    2021/22               Sex       Male         410
```

```
3        All        All    2021/22               Sex    Unknown           0
4        All        All    2021/22  Permanent address    England          95
5        All        All    2021/22  Permanent address   Scotland           0
6        All        All    2021/22  Permanent address      Wales           0

   Number  Number_Norm
2   410.0     0.803922
3     0.0     0.000000
4    95.0     0.186275
5     0.0     0.000000
6     0.0     0.000000
```

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Step 1: Load dataset with correct header row for 2020-21
maternal_df = pd.read_csv('/content/table-1-(2022-23).csv', skiprows=12, dtype=str)

# Step 2: Clean column names
maternal_df.columns = maternal_df.columns.str.strip().str.replace(' ', '_').str.replace('[()]', '', regex=True)

# Step 3: Display available columns for clarity
print("Available Columns:", maternal_df.columns.tolist())

# Step 4: Drop rows with missing provider name (first column fallback if needed)
provider_column = maternal_df.columns[0]
maternal_df.dropna(subset=[provider_column], inplace=True)

# Step 5: Convert enrollment count column to numeric and drop missing values
count_column = maternal_df.columns[-1]  # Assume last column contains enrollment numbers
maternal_df['Number'] = pd.to_numeric(maternal_df[count_column], errors='coerce')
maternal_df.dropna(subset=['Number'], inplace=True)

# Step 6: Remove outliers using IQR
Q1 = maternal_df['Number'].quantile(0.25)
Q3 = maternal_df['Number'].quantile(0.75)
IQR = Q3 - Q1
maternal_df = maternal_df[(maternal_df['Number'] >= Q1 - 1.5 * IQR) & (maternal_df['Number'] <= Q3 + 1.5 * IQR)]

# Step 7: Normalize the 'Number' column
scaler = MinMaxScaler()
maternal_df['Number_Norm'] = scaler.fit_transform(maternal_df[['Number']])

# Step 8: Save and display the cleaned dataset
maternal_df.to_csv('cleaned_maternal_enrollment_2020_21.csv', index=False)
print(maternal_df.head())
```

```
Available Columns: ['Unnamed:_0', 'Unnamed:_1', 'Unnamed:_2', 'Unnamed:_3', 'Unnamed:_4', 'Unnamed:_5', 'Unnamed:_6', 'Unnamed:_7',
/tmp/ipython-input-3941627757.py:31: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  maternal_df['Number_Norm'] = scaler.fit_transform(maternal_df[['Number']])
  Unnamed:_0                Unnamed:_1 Unnamed:_2 Unnamed:_3 Unnamed:_4  \
2   10008071  AA School of Architecture        All        All        All
3   10008071  AA School of Architecture        All        All        All
4   10008071  AA School of Architecture        All        All        All
5   10008071  AA School of Architecture        All        All        All
6   10008071  AA School of Architecture        All        All        All

  Unnamed:_5 Unnamed:_6 Unnamed:_7         Unnamed:_8 Unnamed:_9 Unnamed:_10  \
2        All        All    2022/23               Sex       Male         410
3        All        All    2022/23               Sex    Unknown           0
4        All        All    2022/23  Permanent address    England          75
5        All        All    2022/23  Permanent address   Scotland           0
6        All        All    2022/23  Permanent address      Wales           0

   Number  Number_Norm
2   410.0         0.82
3     0.0         0.00
4    75.0         0.15
5     0.0         0.00
6     0.0         0.00
```

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Step 1: Load dataset with correct header row for 2020-21
maternal_df = pd.read_csv('/content/table-1-(2023-24).csv', skiprows=12, dtype=str)

# Step 2: Clean column names
maternal_df.columns = maternal_df.columns.str.strip().str.replace(' ', '_').str.replace('[()]', '', regex=True)

# Step 3: Display available columns for clarity
print("Available Columns:", maternal_df.columns.tolist())

# Step 4: Drop rows with missing provider name (first column fallback if needed)
provider_column = maternal_df.columns[0]
maternal_df.dropna(subset=[provider_column], inplace=True)

# Step 5: Convert enrollment count column to numeric and drop missing values
count_column = maternal_df.columns[-1]  # Assume last column contains enrollment numbers
maternal_df['Number'] = pd.to_numeric(maternal_df[count_column], errors='coerce')
maternal_df.dropna(subset=['Number'], inplace=True)

# Step 6: Remove outliers using IQR
Q1 = maternal_df['Number'].quantile(0.25)
Q3 = maternal_df['Number'].quantile(0.75)
IQR = Q3 - Q1
maternal_df = maternal_df[(maternal_df['Number'] >= Q1 - 1.5 * IQR) & (maternal_df['Number'] <= Q3 + 1.5 * IQR)]

# Step 7: Normalize the 'Number' column
scaler = MinMaxScaler()
maternal_df['Number_Norm'] = scaler.fit_transform(maternal_df[['Number']])

# Step 8: Save and display the cleaned dataset
maternal_df.to_csv('cleaned_maternal_enrollment_2020_21.csv', index=False)
print(maternal_df.head())
```

```
Available Columns: ['Unnamed:_0', 'Unnamed:_1', 'Unnamed:_2', 'Unnamed:_3', 'Unnamed:_4', 'Unnamed:_5', 'Unnamed:_6', 'Unnamed:_7',
/tmp/ipython-input-2687974491.py:31: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  maternal_df['Number_Norm'] = scaler.fit_transform(maternal_df[['Number']])
  Unnamed:_0                 Unnamed:_1 Unnamed:_2 Unnamed:_3 Unnamed:_4  \
2   10008071  AA School of Architecture        All        All        All
3   10008071  AA School of Architecture        All        All        All
4   10008071  AA School of Architecture        All        All        All
5   10008071  AA School of Architecture        All        All        All
6   10008071  AA School of Architecture        All        All        All

  Unnamed:_5 Unnamed:_6 Unnamed:_7         Unnamed:_8 Unnamed:_9 Unnamed:_10  \
2        All        All    2023/24                Sex       Male         360
3        All        All    2023/24                Sex    Unknown           0
4        All        All    2023/24  Permanent address    England          70
5        All        All    2023/24  Permanent address   Scotland           0
6        All        All    2023/24  Permanent address      Wales           0

   Number  Number_Norm
2   360.0     0.800000
3     0.0     0.000000
4    70.0     0.155556
5     0.0     0.000000
6     0.0     0.000000
```

```
import pandas as pd

# List of cleaned maternal enrollment files
file_paths = [
    '/content/cleaned_maternal_enrollment_2015_16.csv',
    '/content/cleaned_maternal_enrollment_2018_19.csv',
    '/content/cleaned_maternal_enrollment_2019_20.csv',
    '/content/cleaned_maternal_enrollment_2020_21.csv',

]

# Load and append 'Year' column based on filename
dataframes = []
```

```
    for file in file_paths:
        df = pd.read_csv(file)
        year = file.split('_')[-1].replace('.csv', '')  # Extract year portion
        df['Year'] = year
        dataframes.append(df)

    # Merge all into one DataFrame
    merged_df = pd.concat(dataframes, ignore_index=True)

    # Save merged DataFrame
    merged_df.to_csv('final_maternal_enrollment.csv', index=False)

    print("✅ Merged maternal enrollment data saved as 'final_maternal_enrollment.csv'")
```

```
✅ Merged maternal enrollment data saved as 'final_maternal_enrollment.csv'
```

```
    import pandas as pd

    # Load and inspect each dataset
    births = pd.read_csv("/content/cleaned_birth_rates.csv")
    print("Births Columns:", births.columns.tolist())
    print(births.head())

    unemployment = pd.read_csv("/content/cleaned_scotland_unemployment.csv")
    print("\nUnemployment Columns:", unemployment.columns.tolist())
    print(unemployment.head())

    cpi = pd.read_csv("/content/cleaned_mm23.csv")
    print("\nCPI Columns:", cpi.columns.tolist())
    print(cpi.head())
```

```
  RPI:all_items_ex._mortgage_interest_payments_&_council_tax_Jan_87=100
0                                                DQAD
1                                                 NaN
2                             Index, base year = 100
3                                         16-07-2025
4                                     20 August 2025

[5 rows x 53 columns]
```

```
import pandas as pd

# Load all 3 datasets carefully
births_df = pd.read_csv("/content/cleaned_birth_rates.csv")
unemp_df = pd.read_csv("/content/cleaned_scotland_unemployment.csv")
cpi_df = pd.read_csv("/content/cleaned_mm23.csv", skiprows=5)

# Print basic summaries and column names
births_summary = births_df.head()
unemp_summary = unemp_df.head()
cpi_summary = cpi_df.head()

births_columns = births_df.columns.tolist()
unemp_columns = unemp_df.columns.tolist()
cpi_columns = cpi_df.columns.tolist()

births_shape = births_df.shape
unemp_shape = unemp_df.shape
cpi_shape = cpi_df.shape

(births_summary, births_columns, births_shape), (unemp_summary, unemp_columns, unemp_shape), (cpi_summary, cpi_columns, cpi_shape)
```

```
           '20 August 2025.43',
           '20 August 2025.44',
           '20 August 2025.45',
           '20 August 2025.46',
           '20 August 2025.47',
           '20 August 2025.48',
           '20 August 2025.49',
           '20 August 2025.50',
           '20 August 2025.51'],
          (1472, 53)))


    # Try skipping 6 rows to reach the real column headers
    cpi_df = pd.read_csv("/content/cleaned_mm23.csv", skiprows=6)

    # Check the first row now for real columns
    cpi_df.columns = cpi_df.iloc[0]  # First row as header
    cpi_df = cpi_df[1:]              # Drop that header row from data

    # Remove unnamed or entirely null columns
    cpi_df.dropna(axis=1, how="all", inplace=True)

    # Fix column names
    cpi_df.columns.name = None
    cpi_df.rename(columns=lambda x: str(x).strip(), inplace=True)

    # Show preview of correct structure
    # import ace_tools as tools; tools.display_dataframe_to_user(name="Corrected CPI Raw Preview", dataframe=cpi_df.head(10))
    print(cpi_df.head(10))
    print(cpi_df.columns[:10].tolist())

    ➪      1800  nan  13.5  nan  nan  nan  nan  nan  nan  nan  ...  nan  nan  nan  \
        1   1801  NaN  15.1  NaN  NaN  NaN  NaN  NaN  NaN  NaN  ...  NaN  NaN  NaN
        2   1802  NaN  11.6  NaN  NaN  NaN  NaN  NaN  NaN  NaN  ...  NaN  NaN  NaN
        3   1803  NaN  11.0  NaN  NaN  NaN  NaN  NaN  NaN  NaN  ...  NaN  NaN  NaN
        4   1804  NaN  11.3  NaN  NaN  NaN  NaN  NaN  NaN  NaN  ...  NaN  NaN  NaN
        5   1805  NaN  13.1  NaN  NaN  NaN  NaN  NaN  NaN  NaN  ...  NaN  NaN  NaN
        6   1806  NaN  12.6  NaN  NaN  NaN  NaN  NaN  NaN  NaN  ...  NaN  NaN  NaN
        7   1807  NaN  12.3  NaN  NaN  NaN  NaN  NaN  NaN  NaN  ...  NaN  NaN  NaN
        8   1808  NaN  12.8  NaN  NaN  NaN  NaN  NaN  NaN  NaN  ...  NaN  NaN  NaN
        9   1809  NaN  14.0  NaN  NaN  NaN  NaN  NaN  NaN  NaN  ...  NaN  NaN  NaN
        10  1810  NaN  14.4  NaN  NaN  NaN  NaN  NaN  NaN  NaN  ...  NaN  NaN  NaN

            nan  nan  nan  nan  nan  nan  nan
        1   NaN  NaN  NaN  NaN  NaN  NaN  NaN
        2   NaN  NaN  NaN  NaN  NaN  NaN  NaN
        3   NaN  NaN  NaN  NaN  NaN  NaN  NaN
        4   NaN  NaN  NaN  NaN  NaN  NaN  NaN
        5   NaN  NaN  NaN  NaN  NaN  NaN  NaN
        6   NaN  NaN  NaN  NaN  NaN  NaN  NaN
        7   NaN  NaN  NaN  NaN  NaN  NaN  NaN
        8   NaN  NaN  NaN  NaN  NaN  NaN  NaN
        9   NaN  NaN  NaN  NaN  NaN  NaN  NaN
        10  NaN  NaN  NaN  NaN  NaN  NaN  NaN

        [10 rows x 53 columns]
        ['1800', 'nan', '13.5', 'nan', 'nan', 'nan', 'nan', 'nan', 'nan', 'nan']


    import pandas as pd
    from sklearn.preprocessing import StandardScaler

    # Load datasets
    births = pd.read_csv("/content/cleaned_birth_rates.csv")
    unemp = pd.read_csv("/content/cleaned_scotland_unemployment.csv")
    cpi_raw = pd.read_csv("/content/cleaned_mm23.csv", skiprows=5)

    # --------------------------
    # Step 1: Convert date columns
    # --------------------------
    births.rename(columns={"Date": "date"}, inplace=True)
    births["date"] = pd.to_datetime(births["date"])

    unemp.rename(columns={"Year": "date"}, inplace=True)
    unemp["date"] = pd.to_datetime(unemp["date"])

    cpi_raw.rename(columns={"Title": "date"}, inplace=True)
    cpi_raw["date"] = pd.to_datetime(cpi_raw["date"], errors="coerce")

    # We'll use: RPI:Percentage_change_over_12_months_-_Food
    cpi = cpi_raw[["date", "RPI:Percentage_change_over_12_months_-_Food"]].copy()
```

```
cpi.columns = ["date", "cpi_food"]

# Remove NaN and invalid dates
cpi.dropna(inplace=True)
births_yearly = births.set_index("date").resample("YS").sum().reset_index()


merged = births_yearly.merge(unemp, on="date", how="inner").merge(cpi, on="date", how="inner")

# Drop rows with any missing values
merged.dropna(inplace=True)

# --------------------------
# Step 4: Feature Engineering
# --------------------------
# 1. Differencing to remove trends
merged_diff = merged.copy()
merged_diff["births_diff"] = merged_diff["Births"].diff()
merged_diff["unemp_diff"] = merged_diff["Unemployment_Rate_Norm"].diff()
merged_diff["cpi_diff"] = merged_diff["cpi_food"].diff()

# Drop first row with NaNs from differencing
merged_diff.dropna(inplace=True)

# 2. Standard scaling
scaler = StandardScaler()
scaled_features = scaler.fit_transform(merged_diff[["births_diff", "unemp_diff", "cpi_diff"]])
scaled_df = pd.DataFrame(scaled_features, columns=["births_scaled", "unemp_scaled", "cpi_scaled"])
scaled_df["date"] = merged_diff["date"].values
scaled_df.set_index("date", inplace=True)

# Display final feature engineered data
import ace_tools as tools; tools.display_dataframe_to_user(name="Feature Engineered Dataset", dataframe=scaled_df)
scaled_df.head()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.11/dist-packages/pandas/core/indexes/base.py in get_loc(self, key)
   3804         try:
-> 3805             return self._engine.get_loc(casted_key)
   3806         except KeyError as err:

index.pyx in pandas._libs.index.IndexEngine.get_loc()

index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'date'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
                                    2 frames
/usr/local/lib/python3.11/dist-packages/pandas/core/indexes/base.py in get_loc(self, key)
   3810             ):
   3811                 raise InvalidIndexError(key)
-> 3812             raise KeyError(key) from err
   3813         except TypeError:
   3814             # If we have a listlike key, _check_indexing_error will raise

KeyError: 'date'
```

Next steps: Explain error

```
# Let's inspect the actual columns of the CPI file after skipping 6 rows
cpi_raw = pd.read_csv("/content/cleaned_mm23.csv", skiprows=6)

# Show all column names
cpi_raw.columns.tolist()
```

```
['Important Notes',
 'Unnamed: 1',
 'Unnamed: 2',
 'Unnamed: 3',
 'Unnamed: 4',
 'Unnamed: 5',
 'Unnamed: 6',
 'Unnamed: 7',
 'Unnamed: 8',
 'Unnamed: 9',
 'Unnamed: 10',
 'Unnamed: 11',
 'Unnamed: 12',
 'Unnamed: 13',
 'Unnamed: 14',
 'Unnamed: 15',
 'Unnamed: 16',
 'Unnamed: 17',
 'Unnamed: 18',
 'Unnamed: 19',
 'Unnamed: 20',
 'Unnamed: 21',
 'Unnamed: 22',
 'Unnamed: 23',
 'Unnamed: 24',
 'Unnamed: 25',
 'Unnamed: 26',
 'Unnamed: 27',
 'Unnamed: 28',
 'Unnamed: 29',
 'Unnamed: 30',
 'Unnamed: 31',
 'Unnamed: 32',
 'Unnamed: 33',
 'Unnamed: 34',
 'Unnamed: 35',
 'Unnamed: 36',
 'Unnamed: 37',
 'Unnamed: 38',
 'Unnamed: 39',
 'Unnamed: 40',
 'Unnamed: 41',
 'Unnamed: 42',
 'Unnamed: 43',
 'Unnamed: 44',
 'Unnamed: 45',
 'Unnamed: 46',
 'Unnamed: 47',
 'Unnamed: 48',
 'Unnamed: 49',
 'Unnamed: 50',
 'Unnamed: 51',
 'Unnamed: 52']
```

```
# Read the file as raw text to locate the true data header
with open("/content/cleaned_mm23.csv", "r", encoding="utf-8") as f:
    raw_lines = f.readlines()

# Display the first 20 lines so we can locate the correct header row index
raw_lines[:20]
```

```
['Title,RPI:Percentage_change_over_12_months_-
_Food,Retail_Prices_Index:_Long_run_series:_1800_to_2024:_Jan_1974=100,RPI:_Percentage_change_over_1_month_-
_Food,RPI_All_Items_Excl_Mortgage_Interest_RPIX:_Percentage_change_over_12_months,RPI_All_Items:_Percentage_change_over_12_months
_all_items_excluding_housing,RPI:Percentage_change_over_12_months_-_Food_and_catering,RPI:Percentage_change_over_12_months_-
_Seasonal_food,RPI:Percentage_change_over_12_months_-_Food_excluding_seasonal,RPI:Percentage_change_over_12_months_-
_Housing,RPI:Percentage_change_over_12_months_-_Household_goods,RPI:Percentage_change_over_12_months_-
_Clothing_and_footwear,RPI:Percentage_change_over_1_month_-_All_items,RPI:Percentage_change_over_1_month_-
_Seasonal_food,RPI:Percentage_change_over_1_month_-_Food_excluding_seasonal,RPI:Percentage_change_over_1_month_-
_All_items_excluding_housing,RPI:Percentage_change_over_1_month_-_Housing,RPI:Percentage_change_over_1_month_-
_Clothing_and_footwear,"RPI:_Ave_price_-_Ultra_low_sulphur_diesel,_per_litre_Derv_prior_to_Feb_2000","RPI:_Ave_price_-
_Bananas,_per_Kg,","RPI:_Ave_price_-_Oranges,_each_per_Kg_prior_to_Feb_1988","RPI:_Ave_price_-
_Pears,_dessert,_per_Kg","RPI:_Ave_price_-_Apples,_dessert,_per_Kg","RPI:_Ave_price_-_Apples,_cooking,_per_Kg","RPI:_Ave_price_-
_Mushrooms,_per_Kg","RPI:_Ave_price_-_Onions,_per_Kg","RPI:_Ave_price_-_Carrots,_per_Kg","RPI:_Ave_price_-
_Cauliflower,_each","RPI:_Ave_price_-_Cabbage,_hearts,_per_Kg","RPI:_Ave_price_-_Tomatoes,_per_Kg","RPI:_Ave_price_-
_Sugar:_Granulated,_per_Kg","RPI:_Ave_price_-_Coffee:_pure,_instant,_per_100g","RPI_:Ave_price_-
_Milk:_Pasteurised,_per_pint","RPI:_Ave_price_-_Eggs:_size_4_55-60g,_per_dozen","RPI:_Ave_price_-
_Cheese:_cheddar_type,_per_Kg","RPI:_Ave_price_-_Flour:_self_raising,_per_1.5_Kg","RPI:_Ave_price_-
_Bread:_white_loaf,_unwrapped,_800g","RPI:_Ave_price_-_Bread:_white_loaf,_sliced,_800g","RPI:_Ave_price_-
_White_fish_fillets,_per_Kg_cod_prior_Feb_02","RPI:_Ave_price_-_Chicken:_roasting,_ovn_ready,_frsh/chilld,_Kg","RPI:_Ave_price_-
_Sausages:_pork,_per_Kg","RPI:_Ave_price_-_Ham_not_shoulder,_per_113g","RPI:_Ave_price_-
```

```
_Bacon:_Gammon,_per_Kg","RPI:_Ave_price_-_Pork:_home-killed,_loin_with_bone,_per_Kg","RPI:_Ave_price_-
_Lamb:_imported_frozen,_leg_inc_bone,_per_Kg","RPI:_Ave_price_-_Lamb:_home-killed,_shoulder_inc_bone,_Kg","RPI:_Ave_price_-
_Lamb:_home-killed,_loin_with_bone,_per_Kg","RPI:_Ave_price_-_Beef:_home-
killed,_rump_steak,_per_Kg",RPI:Percentage_change_over_12_months_-_All_items_exc._MIPS_+_Council_Tax,RPI:_Ave_price_-
_Butter:_imported_per_250g,RPI:Percentage_points_change_over_previous_month_12_month_rate-
_All_items,RPI:all_items_ex._mortgage_interest_payments_&_council_tax_Jan_87=100\n',

'CDID,CCYY,CDKO,CDKP,CDKQ,CZBH,CZBI,CZBJ,CZBP,CZBQ,CZCP,CZDC,CZDO,CZEQ,CZEX,CZEY,CZFB,CZFG,CZFK,CZML,CZMV,CZMW,CZMX,CZMY,CZMZ,CZN
 'PreUnit,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\n',
 'Unit,%,"Index, base year =
100",%,%,%,%,%,%,%,%,%,%,%,%,%,%,%,%,Pence,Pence,Pence,Pence,Pence,Pence,Pence,Pence,Pence,Pence,Pence,Pence,Pence,Pence,Pence,Pe
base year = 100"\n',
 'Release Date,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-
07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-
2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-
2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-
2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025,16-07-2025\n',
 'Next release,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20
August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August
2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20
August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August
2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20
August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August 2025,20 August
2025,20 August 2025,20 August 2025\n',
 'Important Notes,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\n',
 '1800,,13.5,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\n',
 '1801,,15.1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\n',
 '1802,,11.6,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\n',
 '1803,,11.0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\n',
 '1804,,11.3,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\n',
 '1805,,13.1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\n',
 '1806,,12.6,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\n',
 '1807,,12.3,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\n',
 '1808,,12.8,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\n',
 '1809,,14.0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\n',
 '1810,,14.4,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\n',
 '1811,,14.0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,\n',
```

```python
# Correctly load the CPI data with the first row as the header
cpi_raw = pd.read_csv("/content/cleaned_mm23.csv")

# Now we rename the 'Title' column to 'date' and parse dates
cpi_raw.rename(columns={"Title": "date"}, inplace=True)
cpi_raw["date"] = pd.to_datetime(cpi_raw["date"], errors="coerce")

# Filter the required column
cpi = cpi_raw[["date", "RPI:Percentage_change_over_12_months_-_Food"]].copy()
cpi.columns = ["date", "cpi_food"]
cpi.dropna(inplace=True)

# Display clean CPI data
# import ace_tools as tools; tools.display_dataframe_to_user(name="CPI Cleaned (Proper Header)", dataframe=cpi)
print(cpi.head())
```

```
          date cpi_food
155 1949-01-01      5.5
156 1950-01-01      7.5
157 1951-01-01     11.1
158 1952-01-01     15.8
159 1953-01-01      5.6
/tmp/ipython-input-1232109739.py:6: UserWarning: Could not infer format, so each element will be parsed individually, falling back
  cpi_raw["date"] = pd.to_datetime(cpi_raw["date"], errors="coerce")
```

```python
# Step 1: Reload files and examine true column names
births = pd.read_csv("/content/cleaned_birth_rates.csv")
unemp = pd.read_csv("/content/cleaned_scotland_unemployment.csv")
cpi = pd.read_csv("/content/cleaned_mm23.csv", skiprows=5)

# Print column names to verify for correct renaming
births_cols = births.columns.tolist()
unemp_cols = unemp.columns.tolist()
cpi_cols = cpi.columns.tolist()

births_cols, unemp_cols, cpi_cols
```

```
(['Date', 'Births'],
 ['Year', 'Unemployment_Rate', 'Unemployment_Rate_Norm'],
 ['Next release',
```

```
          '20 August 2025',
          '20 August 2025.1',
          '20 August 2025.2',
          '20 August 2025.3',
          '20 August 2025.4',
          '20 August 2025.5',
          '20 August 2025.6',
          '20 August 2025.7',
          '20 August 2025.8',
          '20 August 2025.9',
          '20 August 2025.10',
          '20 August 2025.11',
          '20 August 2025.12',
          '20 August 2025.13',
          '20 August 2025.14',
          '20 August 2025.15',
          '20 August 2025.16',
          '20 August 2025.17',
          '20 August 2025.18',
          '20 August 2025.19',
          '20 August 2025.20',
          '20 August 2025.21',
          '20 August 2025.22',
          '20 August 2025.23',
          '20 August 2025.24',
          '20 August 2025.25',
          '20 August 2025.26',
          '20 August 2025.27',
          '20 August 2025.28',
          '20 August 2025.29',
          '20 August 2025.30',
          '20 August 2025.31',
          '20 August 2025.32',
          '20 August 2025.33',
          '20 August 2025.34',
          '20 August 2025.35',
          '20 August 2025.36',
          '20 August 2025.37',
          '20 August 2025.38',
          '20 August 2025.39',
          '20 August 2025.40',
          '20 August 2025.41',
          '20 August 2025.42',
          '20 August 2025.43',
          '20 August 2025.44',
          '20 August 2025.45',
          '20 August 2025.46',
          '20 August 2025.47',
          '20 August 2025.48',
          '20 August 2025.49',
          '20 August 2025.50',
          '20 August 2025.51'])


# Display actual column names from CPI after cleaning header row
cpi.columns.tolist()
```

```
['Next release',
 '20 August 2025',
 '20 August 2025.1',
 '20 August 2025.2',
 '20 August 2025.3',
 '20 August 2025.4',
 '20 August 2025.5',
 '20 August 2025.6',
 '20 August 2025.7',
 '20 August 2025.8',
 '20 August 2025.9',
 '20 August 2025.10',
 '20 August 2025.11',
 '20 August 2025.12',
 '20 August 2025.13',
 '20 August 2025.14',
 '20 August 2025.15',
 '20 August 2025.16',
 '20 August 2025.17',
 '20 August 2025.18',
 '20 August 2025.19',
 '20 August 2025.20',
 '20 August 2025.21',
 '20 August 2025.22',
 '20 August 2025.23',
 '20 August 2025.24',
 '20 August 2025.25',
```

```
        '20 August 2025.26',
        '20 August 2025.27',
        '20 August 2025.28',
        '20 August 2025.29',
        '20 August 2025.30',
        '20 August 2025.31',
        '20 August 2025.32',
        '20 August 2025.33',
        '20 August 2025.34',
        '20 August 2025.35',
        '20 August 2025.36',
        '20 August 2025.37',
        '20 August 2025.38',
        '20 August 2025.39',
        '20 August 2025.40',
        '20 August 2025.41',
        '20 August 2025.42',
        '20 August 2025.43',
        '20 August 2025.44',
        '20 August 2025.45',
        '20 August 2025.46',
        '20 August 2025.47',
        '20 August 2025.48',
        '20 August 2025.49',
        '20 August 2025.50',
        '20 August 2025.51']


# Rename correctly mapped columns
cpi.rename(columns={cpi.columns[0]: "year", cpi.columns[1]: "cpi"}, inplace=True)

# Keep only the required columns
cpi_cleaned = cpi[["year", "cpi"]]

# Convert to numeric
cpi_cleaned["year"] = pd.to_numeric(cpi_cleaned["year"], errors="coerce")
cpi_cleaned["cpi"] = pd.to_numeric(cpi_cleaned["cpi"], errors="coerce")

# Drop missing rows
cpi_cleaned.dropna(inplace=True)

# Display cleaned CPI
# import ace_tools as tools; tools.display_dataframe_to_user(name="Final Cleaned CPI (Year vs CPI)", dataframe=cpi_cleaned)
print(cpi_cleaned.head())
```

```
        year   cpi
150  1949.0   5.5
151  1950.0   7.5
152  1951.0  11.1
153  1952.0  15.8
154  1953.0   5.6
/tmp/ipython-input-635596258.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  cpi_cleaned["year"] = pd.to_numeric(cpi_cleaned["year"], errors="coerce")
/tmp/ipython-input-635596258.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  cpi_cleaned["cpi"] = pd.to_numeric(cpi_cleaned["cpi"], errors="coerce")
/tmp/ipython-input-635596258.py:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  cpi_cleaned.dropna(inplace=True)
```

```
# Check actual column names after skipping metadata
cpi_cleaned = pd.read_csv("/content/mm23.csv", skiprows=6)
cpi_cleaned.columns.tolist()
```

```
        'Unnamed: 951',
        'Unnamed: 952',
        'Unnamed: 953',
        'Unnamed: 954',
        'Unnamed: 955',
        'Unnamed: 956',
        'Unnamed: 957',
        'Unnamed: 958',
        'Unnamed: 959',
        'Unnamed: 960',
        'Unnamed: 961',
        'Unnamed: 962',
        'Unnamed: 963',
        'Unnamed: 964',
        'Unnamed: 965',
        'Unnamed: 966',
        'Unnamed: 967',
        'Unnamed: 968',
        'Unnamed: 969',
        'Unnamed: 970',
        'Unnamed: 971',
        'Unnamed: 972',
        'Unnamed: 973',
        'Unnamed: 974',
        'Unnamed: 975',
        'Unnamed: 976',
        'Unnamed: 977',
        'Unnamed: 978',
        'Unnamed: 979',
        'Unnamed: 980',
        'Unnamed: 981',
        'Unnamed: 982',
        'Unnamed: 983',
        'Unnamed: 984',
        'Unnamed: 985',
        'Unnamed: 986',
        'Unnamed: 987',
        'Unnamed: 988',
        'Unnamed: 989',
        'Unnamed: 990',
        'Unnamed: 991',
        'Unnamed: 992',
        'Unnamed: 993',
        'Unnamed: 994',
        'Unnamed: 995',
        'Unnamed: 996',
        'Unnamed: 997',
        'Unnamed: 998',
        'Unnamed: 999',
        ...]
```

```
# Check actual column names after skipping metadata
cpi_cleaned = pd.read_csv("/content/cleaned_mm23.csv", skiprows=6)
cpi_cleaned.columns.tolist()
```

```
['Important Notes',
 'Unnamed: 1',
 'Unnamed: 2',
 'Unnamed: 3',
 'Unnamed: 4',
 'Unnamed: 5',
 'Unnamed: 6',
 'Unnamed: 7',
 'Unnamed: 8',
 'Unnamed: 9',
 'Unnamed: 10',
 'Unnamed: 11',
 'Unnamed: 12',
 'Unnamed: 13',
 'Unnamed: 14',
 'Unnamed: 15',
 'Unnamed: 16',
 'Unnamed: 17',
 'Unnamed: 18',
 'Unnamed: 19',
 'Unnamed: 20',
 'Unnamed: 21',
 'Unnamed: 22',
 'Unnamed: 23',
 'Unnamed: 24',
 'Unnamed: 25',
 'Unnamed: 26',
```

```
        'Unnamed: 27',
        'Unnamed: 28',
        'Unnamed: 29',
        'Unnamed: 30',
        'Unnamed: 31',
        'Unnamed: 32',
        'Unnamed: 33',
        'Unnamed: 34',
        'Unnamed: 35',
        'Unnamed: 36',
        'Unnamed: 37',
        'Unnamed: 38',
        'Unnamed: 39',
        'Unnamed: 40',
        'Unnamed: 41',
        'Unnamed: 42',
        'Unnamed: 43',
        'Unnamed: 44',
        'Unnamed: 45',
        'Unnamed: 46',
        'Unnamed: 47',
        'Unnamed: 48',
        'Unnamed: 49',
        'Unnamed: 50',
        'Unnamed: 51',
        'Unnamed: 52']
```

```
# Inspect first few rows of 'Important Notes' column to find date-like values
cpi_cleaned["Important Notes"].head(20)
```

	Important Notes
0	1800
1	1801
2	1802
3	1803
4	1804
5	1805
6	1806
7	1807
8	1808
9	1809
10	1810
11	1811
12	1812
13	1813
14	1814
15	1815
16	1816
17	1817
18	1818
19	1819

dtype: object

```
import pandas as pd

cpi_cleaned.rename(columns={"Important Notes": "year", "Unnamed: 1": "cpi"}, inplace=True)

# --- Step 3: Convert Columns to Numeric ---
cpi_cleaned["year"] = pd.to_numeric(cpi_cleaned["year"], errors="coerce")
cpi_cleaned["cpi"] = pd.to_numeric(cpi_cleaned["cpi"], errors="coerce")

# --- Step 4: Drop Rows with Invalid Data ---
```

```
cpi_cleaned = cpi_cleaned.dropna(subset=["year", "cpi"]).astype({"year": int})

# --- Step 5: Filter for Relevant Years ---
cpi_filtered = cpi_cleaned[cpi_cleaned["year"] >= 2000]

# --- Step 6: Show Cleaned CPI Data ---
print("✅ Cleaned CPI Data:")
print(cpi_filtered.head(10))  # Display first 10 rows
```

```
✅ Cleaned CPI Data:
      year  cpi  Unnamed: 2  Unnamed: 3  Unnamed: 4  Unnamed: 5  Unnamed: 6  \
200  2000 -0.3       671.8        -0.3         2.1         3.0         1.5
201  2001  3.3       683.7         3.3         2.1         1.8         1.5
202  2002  0.7       695.1         0.7         2.2         1.7         1.4
203  2003  1.3       715.2         1.3         2.8         2.9         1.7
204  2004  0.6       736.5         0.6         2.2         3.0         1.2
205  2005  1.2       757.3         1.2         2.3         2.8         1.6
206  2006  2.1       781.5         2.1         2.9         3.2         2.6
207  2007  4.6       815.0         4.6         3.2         4.3         2.7
208  2008  9.3       847.5         9.3         4.3         4.0         4.4
209  2009  5.3       843.0         5.3         2.0        -0.5         2.6

     Unnamed: 7  Unnamed: 8  Unnamed: 9  ...  Unnamed: 43  Unnamed: 44  \
200         0.8        -0.2        -0.3  ...        544.0        446.0
201         3.5        11.1         1.9  ...        575.0        454.0
202         1.6        -0.7         1.1  ...        570.0        479.0
203         1.9         2.3         1.1  ...        575.0        481.0
204         1.3        -2.2         1.2  ...        586.0        506.0
205         1.7         2.6         1.0  ...        585.0        502.0
206         2.3         2.1         2.2  ...        624.0        518.0
207         4.2         7.3         4.0  ...        649.0        519.0
208         7.7         8.9         9.3  ...        669.0        553.0
209         4.6         4.8         5.4  ...        701.0        593.0

     Unnamed: 45  Unnamed: 46  Unnamed: 47  Unnamed: 48  Unnamed: 49  \
200        411.0        340.0        855.0        866.0          2.0
201        440.0        354.0        893.0        871.0          2.0
202        455.0        379.0        970.0        869.0          2.0
203        500.0        426.0       1010.0        895.0          2.5
204        514.0        424.0       1025.0        897.0          2.0
205        529.0        402.0        986.0        882.0          2.2
206        510.0        423.0       1069.0        941.0          2.9
207        518.0        422.0       1098.0        966.0          3.2
208        522.0        488.0       1166.0       1100.0          4.3
209        596.0        546.0       1251.0       1153.0          1.9

     Unnamed: 50  Unnamed: 51  Unnamed: 52
200         87.9          NaN        166.9
201         88.1          NaN        170.2
202         88.2          NaN        173.6
203         89.2          NaN        178.0
204         91.6          NaN        181.5
205         91.4          NaN        185.5
206         90.2          NaN        190.8
207         99.2          NaN        196.9
208        124.3          NaN        205.3
209        125.0          NaN        209.3

[10 rows x 53 columns]
```

```
import pandas as pd

# Load cleaned datasets
births = pd.read_csv("/content/cleaned_birth_rates.csv", parse_dates=["Date"])
births.rename(columns={"Date": "date"}, inplace=True)

unemp = pd.read_csv("/content/cleaned_scotland_unemployment.csv", parse_dates=["Year"])
unemp.rename(columns={"Year": "date"}, inplace=True)

# Step 1: Trim to shared date range
births['date'] = births['date'].dt.to_period('M').dt.to_timestamp()
unemp['date'] = unemp['date'].dt.to_period('M').dt.to_timestamp()
cpi['date'] = cpi['date'].dt.to_period('M').dt.to_timestamp()

# Step 2: Merge all 3 datasets on 'date'
df_merged = births.merge(unemp, on="date", how="inner")
df_merged = df_merged.merge(cpi, on="date", how="inner")

# Step 3: Drop any missing values for now
```

```
df_merged.dropna(inplace=True)

import ace_tools as tools; tools.display_dataframe_to_user(name="Final Merged Dataset", dataframe=df_merged)
df_merged.head()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.11/dist-packages/pandas/core/indexes/base.py in get_loc(self, key)
   3804         try:
-> 3805             return self._engine.get_loc(casted_key)
   3806         except KeyError as err:

index.pyx in pandas._libs.index.IndexEngine.get_loc()

index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'date'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
                                          2 frames
/usr/local/lib/python3.11/dist-packages/pandas/core/indexes/base.py in get_loc(self, key)
   3810             ):
   3811                 raise InvalidIndexError(key)
-> 3812             raise KeyError(key) from err
   3813         except TypeError:
   3814             # If we have a listlike key, _check_indexing_error will raise

KeyError: 'date'
```

Next steps: Explain error

Start coding or generate with AI.