```
# STEP 1: Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# STEP 2: Extract ZIP
import zipfile, os

zip_path = "/content/drive/MyDrive/Fitabase Data.zip"
extract_root = "/content/fitabase_cleaned"

with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_root)

print("ZIP extracted to:", extract_root)
```

```
ZIP extracted to: /content/fitabase_cleaned
```

```
# STEP 3: Detect subfolder
subdirs = [d for d in os.listdir(extract_root) if
os.path.isdir(os.path.join(extract_root, d))]
data_subfolder = os.path.join(extract_root, subdirs[0])  # should be
'Fitabase Data 4.12.16-5.12.16'
print("□ Using subfolder:", data_subfolder)
```

```
□ Using subfolder: /content/fitabase_cleaned/Fitabase Data 4.12.16-
5.12.16
```

```
# □ STEP 4: File list for cleaning
confirmed_files = [
    'dailyActivity_merged.csv',
    'dailyCalories_merged.csv',
    'dailyIntensities_merged.csv',
    'dailySteps_merged.csv',
    'sleepDay_merged.csv',
    'weightLogInfo_merged.csv',
    'minuteCaloriesNarrow_merged.csv',
    'minuteStepsNarrow_merged.csv',
    'minuteIntensitiesNarrow_merged.csv',
    'minuteCaloriesWide_merged.csv',
    'minuteIntensitiesWide_merged.csv',
    'minuteStepsWide_merged.csv',
    'hourlyCalories_merged.csv',
    'hourlySteps_merged.csv',
    'hourlyIntensities_merged.csv'
]

# □ STEP 5: Cleaning function
import pandas as pd
```

```
output_dir = "/content/cleaned_bellabeat"
os.makedirs(output_dir, exist_ok=True)

def clean_file(df):
    df.columns = df.columns.str.strip().str.replace(' ',
'').str.replace('-', '').str.replace(':', '')
    if 'Id' in df.columns:
        df.rename(columns={'Id': 'user_id'}, inplace=True)
    for col in df.columns:
        if any(x in col for x in ['Date', 'Time', 'Minute', 'Hour']):
            try:
                df[col] = pd.to_datetime(df[col], errors='coerce')
            except:
                pass
    return df

# □ STEP 6: Load, clean, and save
for file in confirmed_files:
    full_path = os.path.join(data_subfolder, file)
    if os.path.exists(full_path):
        df = pd.read_csv(full_path)
        df_clean = clean_file(df)
        cleaned_path = os.path.join(output_dir, f'cleaned_{file}')
        df_clean.to_csv(cleaned_path, index=False)
        print(f"□ Cleaned & saved: {file}")
    else:
        print(f"□ File not found: {file}")
```

```
□ Cleaned & saved: dailyActivity_merged.csv
□ Cleaned & saved: dailyCalories_merged.csv
□ Cleaned & saved: dailyIntensities_merged.csv
□ Cleaned & saved: dailySteps_merged.csv
□ Cleaned & saved: sleepDay_merged.csv
□ Cleaned & saved: weightLogInfo_merged.csv

<ipython-input-5-4c4ab8cfdd32>:14: UserWarning: Could not infer
format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  df[col] = pd.to_datetime(df[col], errors='coerce')
<ipython-input-5-4c4ab8cfdd32>:14: UserWarning: Could not infer
format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  df[col] = pd.to_datetime(df[col], errors='coerce')

□ Cleaned & saved: minuteCaloriesNarrow_merged.csv

<ipython-input-5-4c4ab8cfdd32>:14: UserWarning: Could not infer
format, so each element will be parsed individually, falling back to
```

```
`dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  df[col] = pd.to_datetime(df[col], errors='coerce')

 Cleaned & saved: minuteStepsNarrow_merged.csv

<ipython-input-5-4c4ab8cfdd32>:14: UserWarning: Could not infer
format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  df[col] = pd.to_datetime(df[col], errors='coerce')

 Cleaned & saved: minuteIntensitiesNarrow_merged.csv

<ipython-input-5-4c4ab8cfdd32>:14: UserWarning: Could not infer
format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  df[col] = pd.to_datetime(df[col], errors='coerce')

 Cleaned & saved: minuteCaloriesWide_merged.csv

<ipython-input-5-4c4ab8cfdd32>:14: UserWarning: Could not infer
format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  df[col] = pd.to_datetime(df[col], errors='coerce')

 Cleaned & saved: minuteIntensitiesWide_merged.csv

<ipython-input-5-4c4ab8cfdd32>:14: UserWarning: Could not infer
format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  df[col] = pd.to_datetime(df[col], errors='coerce')

 Cleaned & saved: minuteStepsWide_merged.csv

<ipython-input-5-4c4ab8cfdd32>:14: UserWarning: Could not infer
format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  df[col] = pd.to_datetime(df[col], errors='coerce')

 Cleaned & saved: hourlyCalories_merged.csv

<ipython-input-5-4c4ab8cfdd32>:14: UserWarning: Could not infer
format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  df[col] = pd.to_datetime(df[col], errors='coerce')
```

```
 Cleaned & saved: hourlySteps_merged.csv

<ipython-input-5-4c4ab8cfdd32>:14: UserWarning: Could not infer
format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  df[col] = pd.to_datetime(df[col], errors='coerce')

 Cleaned & saved: hourlyIntensities_merged.csv
```

```python
# Zip the cleaned files
import zipfile
import os

zip_filename = 'cleaned_data.zip'
zip_path = os.path.join('/content', zip_filename)

with zipfile.ZipFile(zip_path, 'w') as zipf:
    for root, dirs, filenames in os.walk(output_dir):
        for filename in filenames:
            zipf.write(os.path.join(root, filename),
os.path.relpath(os.path.join(root, filename), output_dir))

print(f"\n Created zip file: {zip_filename}")
```

```
 Created zip file: cleaned_data.zip
```

```python
from google.colab import files

# This will prompt a download of the zip file to your computer
files.download('cleaned_data.zip')
```

```
<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>
```

```python
#  STEP 7: Unzip cleaned ZIP
import zipfile
import os
import pandas as pd

zip_file_path = "/content/cleaned_data.zip"
unzip_dir = "/content/cleaned_bellabeat_merged"

os.makedirs(unzip_dir, exist_ok=True)

with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
```

```
    zip_ref.extractall(unzip_dir)

print("✅ Cleaned ZIP extracted to:", unzip_dir)
```

```
✅ Cleaned ZIP extracted to: /content/cleaned_bellabeat_merged
```

```
import glob

# Grab all CSV files in the extracted folder
csv_files = glob.glob(os.path.join(unzip_dir, "*.csv"))

# Try to merge only files that share the same structure
merged_df = pd.DataFrame()

for file in csv_files:
    df = pd.read_csv(file)

    # Only merge if columns match the current merged_df or if it's the
first file
    if merged_df.empty or list(df.columns) == list(merged_df.columns):
        merged_df = pd.concat([merged_df, df], ignore_index=True)
    else:
        print(f"⚠️ Skipping merge for file due to mismatched columns:
{os.path.basename(file)}")

# Save the merged file
merged_output_path = "/content/bellabeat_merged_data.csv"
merged_df.to_csv(merged_output_path, index=False)

print("✅ Merged file saved to:", merged_output_path)
```

```
⚠️ Skipping merge for file due to mismatched columns:
cleaned_minuteIntensitiesNarrow_merged.csv
⚠️ Skipping merge for file due to mismatched columns:
cleaned_sleepDay_merged.csv
⚠️ Skipping merge for file due to mismatched columns:
cleaned_hourlySteps_merged.csv
⚠️ Skipping merge for file due to mismatched columns:
cleaned_minuteStepsNarrow_merged.csv
⚠️ Skipping merge for file due to mismatched columns:
cleaned_dailyCalories_merged.csv
⚠️ Skipping merge for file due to mismatched columns:
cleaned_dailySteps_merged.csv
⚠️ Skipping merge for file due to mismatched columns:
cleaned_dailyActivity_merged.csv
⚠️ Skipping merge for file due to mismatched columns:
cleaned_hourlyIntensities_merged.csv
⚠️ Skipping merge for file due to mismatched columns:
cleaned_weightLogInfo_merged.csv
⚠️ Skipping merge for file due to mismatched columns:
cleaned_minuteStepsWide_merged.csv
```

```
⚠ Skipping merge for file due to mismatched columns:
cleaned_minuteCaloriesNarrow_merged.csv
⚠ Skipping merge for file due to mismatched columns:
cleaned_hourlyCalories_merged.csv
⚠ Skipping merge for file due to mismatched columns:
cleaned_minuteCaloriesWide_merged.csv
⚠ Skipping merge for file due to mismatched columns:
cleaned_dailyIntensities_merged.csv
 Merged file saved to: /content/bellabeat_merged_data.csv
```

```
from google.colab import files
files.download(merged_output_path)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
```