```
import pandas as pd
import numpy as np
from openpyxl import Workbook
from openpyxl.styles import Font, Alignment, PatternFill, Border, Side
from openpyxl.utils.dataframe import dataframe_to_rows

# Load the datasets
tracking_data = pd.read_excel('Tracking_data.xlsx')
pivot_data = pd.read_excel('sustainable_metrics_pivot.xlsx')
levant_data = pd.read_excel('enviroment-levant-all.xlsx', header=1)

# Set correct column names for levant_data
levant_data.columns = ['Country Name', 'Country ISO3', 'Year', 'Indicator Name',
'Indicator Code', 'Value']

# Define expanded KPIs with sub-KPIs
kpi_structure = {
    'Carbon Emissions': [
        'CO2 emissions (metric tons per capita)',
        'CO2 emissions from solid fuel consumption (% of total)',
        'CO2 emissions from liquid fuel consumption (% of total)',
        'CO2 emissions from gaseous fuel consumption (% of total)'
    ],
    'Renewable Energy': [
        'Renewable energy consumption (% of total final energy consumption)',
        'Renewable electricity output (% of total electricity output)',
        'Alternative and nuclear energy (% of total energy use)'
    ],
    'Energy Access': [
        'Access to electricity (% of population)',
        'Access to clean fuels and technologies for cooking (% of population)'
    ],
    'Air Quality': [
        'PM2.5 air pollution, mean annual exposure (micrograms per cubic meter)',
        'PM2.5 air pollution, population exposed to levels exceeding WHO guideline
value (% of total)'
    ],
    'Water Access': [
        'People using at least basic drinking water services (% of population)',
        'People using safely managed drinking water services (% of population)'
    ],
    'Waste Management': [
        'Municipal solid waste collection coverage, (% of population)',
        'Recycling rate of municipal waste (%)'
    ]
}

# Create a function to get the most recent value and calculate trend
def get_recent_value_and_trend(indicator, data_sources):
    for source in data_sources:
        if indicator in source.columns:
            recent_value = source[indicator].dropna().iloc[-1] if not
source[indicator].empty else np.nan
            trend = calculate_trend(source[indicator])
            return recent_value, trend, source.name
    return np.nan, 'Data Not Available', 'N/A'

# Function to calculate trend
def calculate_trend(data):
    if len(data.dropna()) < 2:
        return 'Insufficient Data'
    last_value = data.dropna().iloc[-1]
    previous_value = data.dropna().iloc[-2]
    if last_value > previous_value:
        return 'Increasing'
    elif last_value < previous_value:
        return 'Decreasing'
    else:
        return 'Stable'

# Prepare data sources
tracking_data.name = 'Tracking Data'
pivot_data.name = 'Pivot Data'
levant_data_pivot = levant_data.pivot(index='Year', columns='Indicator Name',
values='Value')
levant_data_pivot.name = 'Levant Data'

data_sources = [tracking_data, pivot_data, levant_data_pivot]

# Create expanded KPI table
kpi_table = []

for main_kpi, sub_kpis in kpi_structure.items():
    for sub_kpi in sub_kpis:
        current_value, trend, data_source = get_recent_value_and_trend(sub_kpi,
data_sources)

        # Set target value (for demonstration, we'll set it to 10% improvement)
        target_value = current_value * 1.1 if 'emissions' not in sub_kpi.lower()
else current_value * 0.9

        # Extract unit from indicator name
        unit = sub_kpi.split('(')[-1].strip(')') if '(' in sub_kpi else 'No Unit'

        # Set estimated timeframe (for demonstration, we'll set it to 2 years from
now)
        current_year = pd.Timestamp.now().year
        estimated_timeframe = f"{current_year + 2}"

        kpi_table.append({
            'Main KPI': main_kpi,
            'Sub KPI': sub_kpi,
            'Measurement': unit,
            'Current Value': round(current_value, 2) if not pd.isna(current_value)
else 'N/A',
            'Target Value': round(target_value, 2) if not pd.isna(target_value)
else 'N/A',
            'Current Stage': 'In Progress',  # This could be updated based on
specific criteria
            'Trend': trend,
            'Estimated Timeframe': estimated_timeframe,
            'Data Source': data_source
        })

kpi_df = pd.DataFrame(kpi_table)

# Create a new workbook and select the active sheet
wb = Workbook()
ws = wb.active
ws.title = "Enhanced KPI Table"

# Write the DataFrame to the worksheet
for r in dataframe_to_rows(kpi_df, index=False, header=True):
    ws.append(r)

# Style the header
header_fill = PatternFill(start_color="4F81BD", end_color="4F81BD",
fill_type="solid")
header_font = Font(color="FFFFFF", bold=True)
for cell in ws[1]:
    cell.fill = header_fill
    cell.font = header_font

# Style the table
thin_border = Border(left=Side(style='thin'), right=Side(style='thin'),
top=Side(style='thin'), bottom=Side(style='thin'))
for row in ws.iter_rows(min_row=2, max_row=ws.max_row, min_col=1,
max_col=ws.max_column):
    for cell in row:
        cell.border = thin_border

# Auto-adjust column widths
for column in ws.columns:
    max_length = 0
    column_letter = column[0].column_letter
    for cell in column:
        try:
            if len(str(cell.value)) > max_length:
                max_length = len(cell.value)
        except:
            pass
    adjusted_width = (max_length + 2) * 1.2
    ws.column_dimensions[column_letter].width = adjusted_width

# Add additional information
ws.append([])
ws.append(["Additional Metrics"])
ws.append(["Number of countries in analysis", levant_data['Country
Name'].nunique()])
ws.append(["Date range of data", f"{levant_data['Year'].min()} -
{levant_data['Year'].max()}"])
ws.append(["Total number of indicators available", levant_data['Indicator
Name'].nunique()])

# Add top performing countries for renewable energy
ws.append([])
ws.append(["Top 3 Countries for Renewable Energy Consumption"])
top_renewable = pivot_data.sort_values('Renewable energy consumption (% of total
final energy consumption)', ascending=False).head(3)
for _, row in top_renewable[['Country Name', 'Renewable energy consumption (% of
total final energy consumption)']].iterrows():
    ws.append(row.tolist())

# Save the workbook
filename = 'Enhanced_Environmental_KPI_Table.xlsx'
wb.save(filename)

print(f"Enhanced KPI table has been saved as {filename}")

# Display the first few rows of the KPI table
print("\
First few rows of the Enhanced KPI table:")
print(kpi_df.head().to_string(index=False))
```