AI ASSISTANT CODING - ASSIGNMENT-03

Name: Valaboju Harshavardhanu

Roll No: 2303A52354 **Batch:** 41

Task 1: AI-Generated Logic for Reading Consumer Details Scenario

Prompt:

#Python code which collect the data, reads it accurately (Previous units, current units and type of customer (domestic, commercial or industrial consumers)) and calculates the units consumed and displays it, no functions used, create 6 customers details in code itself with columns, serial number, name, customer id, previous units, current units and type of customer.

Code:

```
import pandas as pd

# Create a dictionary with customer details
customer_data = {
    'Serial Number': [1, 2, 3, 4, 5, 6],
    'Name': ['Alice Smith', 'Bob Johnson', 'Charlie Brown', 'Diana Prince', 'Eve Adams', 'Frank White'],
    'Customer ID': ['CUST001', 'CUST002', 'CUST003', 'CUST004', 'CUST005', 'CUST006'],
    'Previous Units': [100, 500, 2000, 150, 800, 3000],
    'Current Units': [150, 750, 2500, 220, 1000, 3800],
    'Type of Customer': ['Domestic', 'Commercial', 'Industrial', 'Domestic', 'Commercial', 'Industrial']
}

# Create a pandas DataFrame from the dictionary
df = pd.DataFrame(customer_data)

# Calculate units consumed
df['Units Consumed'] = df['Current Units'] - df['Previous Units']

# Display the DataFrame row by row for each customer in a non-table format
for index, row in df.iterrows():
    print(f"--- Customer {row['Serial Number']} ---")
    for col, value in row.items():
        print(f"{col}: {value}")
    print("\n") # Add a blank line for better separation
```

Output:

```
--- Customer 1 ---
Serial Number: 1
Name: Alice Smith
Customer ID: CUST001
Previous Units: 100
Current Units: 150
Type of Customer: Domestic
Units Consumed: 50


--- Customer 2 ---
Serial Number: 2
Name: Bob Johnson
Customer ID: CUST002
Previous Units: 500
Current Units: 750
Type of Customer: Commercial
Units Consumed: 250


--- Customer 3 ---
Serial Number: 3
Name: Charlie Brown
Customer ID: CUST003
Previous Units: 2000
Current Units: 2500
Type of Customer: Industrial
Units Consumed: 500
```

```
--- Customer 4 ---
Serial Number: 4
Name: Diana Prince
Customer ID: CUST004
Previous Units: 150
Current Units: 220
Type of Customer: Domestic
Units Consumed: 70


--- Customer 5 ---
Serial Number: 5
Name: Eve Adams
Customer ID: CUST005
Previous Units: 800
Current Units: 1000
Type of Customer: Commercial
Units Consumed: 200


--- Customer 6 ---
Serial Number: 6
Name: Frank White
Customer ID: CUST006
Previous Units: 3000
Current Units: 3800
Type of Customer: Industrial
Units Consumed: 800
```

Justification:

In this task, we are able to read all the data accurately and perform required actions well by using Gemini Ai. I understood that how the Gemini Ai is performing the task and generating the code using the prompt by user.

Task 2: Energy Charges Calculation Based on Units Consumed

Scenario

Prompt:

Generate a optimized python code which calculates the electricity bill based on type of customers using conditional statements

Code:

```
import pandas as pd
import numpy as np

# Define tariff rates (example values - you can adjust these)
tariff_rates = {
    'Domestic': {
        'slab1_units': 100, 'slab1_rate': 5,
        'slab2_units': 200, 'slab2_rate': 7,
        'slab3_rate': 10
    },
    'Commercial': {
        'slab1_units': 500, 'slab1_rate': 8,
        'slab2_rate': 12
    },
    'Industrial': {
        'slab1_units': 1000, 'slab1_rate': 10,
        'slab2_rate': 15
    }
}

conditions = []
choices = []

# Condition for zero units consumed
conditions.append(df['Units Consumed'] == 0)
choices.append(0)
```

```
# Domestic Customer Calculations
domestic_slab1_cond = (df['Type of Customer'] == 'Domestic') & (df['Units Consumed'] <= tariff_rates['Domestic']['slab1_units'])
conditions.append(domestic_slab1_cond)
choices.append(df['Units Consumed'] * tariff_rates['Domestic']['slab1_rate'])

domestic_slab2_cond = (df['Type of Customer'] == 'Domestic') & \
                      (df['Units Consumed'] > tariff_rates['Domestic']['slab1_units']) & \
                      (df['Units Consumed'] <= (tariff_rates['Domestic']['slab1_units'] + tariff_rates['Domestic']['slab2_units']))
conditions.append(domestic_slab2_cond)
choices.append((tariff_rates['Domestic']['slab1_units'] * tariff_rates['Domestic']['slab1_rate']) + \
               ((df['Units Consumed'] - tariff_rates['Domestic']['slab1_units']) * tariff_rates['Domestic']['slab2_rate']))

domestic_slab3_cond = (df['Type of Customer'] == 'Domestic') & \
                      (df['Units Consumed'] > (tariff_rates['Domestic']['slab1_units'] + tariff_rates['Domestic']['slab2_units']))
conditions.append(domestic_slab3_cond)
choices.append((tariff_rates['Domestic']['slab1_units'] * tariff_rates['Domestic']['slab1_rate']) + \
               (tariff_rates['Domestic']['slab2_units'] * tariff_rates['Domestic']['slab2_rate']) + \
               ((df['Units Consumed'] - tariff_rates['Domestic']['slab1_units'] - tariff_rates['Domestic']['slab2_units']) * tariff_rates['Domestic']['slab3_rate']))

# Commercial Customer Calculations
commercial_slab1_cond = (df['Type of Customer'] == 'Commercial') & (df['Units Consumed'] <= tariff_rates['Commercial']['slab1_units'])
conditions.append(commercial_slab1_cond)
choices.append(df['Units Consumed'] * tariff_rates['Commercial']['slab1_rate'])

commercial_slab2_cond = (df['Type of Customer'] == 'Commercial') & (df['Units Consumed'] > tariff_rates['Commercial']['slab1_units'])
conditions.append(commercial_slab2_cond)
choices.append((tariff_rates['Commercial']['slab1_units'] * tariff_rates['Commercial']['slab1_rate']) + \
               ((df['Units Consumed'] - tariff_rates['Commercial']['slab1_units']) * tariff_rates['Commercial']['slab2_rate']))

# Industrial Customer Calculations
industrial_slab1_cond = (df['Type of Customer'] == 'Industrial') & (df['Units Consumed'] <= tariff_rates['Industrial']['slab1_units'])
conditions.append(industrial_slab1_cond)
choices.append(df['Units Consumed'] * tariff_rates['Industrial']['slab1_rate'])

industrial_slab2_cond = (df['Type of Customer'] == 'Industrial') & (df['Units Consumed'] > tariff_rates['Industrial']['slab1_units'])
conditions.append(industrial_slab2_cond)
choices.append((tariff_rates['Industrial']['slab1_units'] * tariff_rates['Industrial']['slab1_rate']) + \
               ((df['Units Consumed'] - tariff_rates['Industrial']['slab1_units']) * tariff_rates['Industrial']['slab2_rate']))
```

```python
# Apply the conditions and choices to calculate 'Bill Amount'
df['Bill Amount'] = np.select(conditions, choices, default=0)

# Calculate 'Rate per unit', handling division by zero
df['Rate per unit'] = np.where(df['Units Consumed'] != 0, df['Bill Amount'] / df['Units Consumed'], 0)

# Display the DataFrame with the calculated bill amounts and requested details
print("\n--- Customer Details with Bill Amount and Rate per Unit ---")
for index, row in df.iterrows():
    print(f"--- Customer {row['Serial Number']} ---")
    print(f"Customer Details: {row['Name']} (ID: {row['Customer ID']})")
    print(f"Customer Type: {row['Type of Customer']}")
    print(f"Units Consumed: {row['Units Consumed']}")
    print(f"Rate per unit: {row['Rate per unit']:.2f}") # Format to 2 decimal places
    print(f"Total Bill: {row['Bill Amount']:.2f}") # Format to 2 decimal places
    print("\n")
```

Output:

```
--- Customer Details with Bill Amount and Rate per Unit ---
--- Customer 1 ---
Customer Details: Alice Smith (ID: CUST001)
Customer Type: Domestic
Units Consumed: 50
Rate per unit: 5.00
Total Bill: 250.00


--- Customer 2 ---
Customer Details: Bob Johnson (ID: CUST002)
Customer Type: Commercial
Units Consumed: 250
Rate per unit: 8.00
Total Bill: 2000.00


--- Customer 3 ---
Customer Details: Charlie Brown (ID: CUST003)
Customer Type: Industrial
Units Consumed: 500
Rate per unit: 10.00
Total Bill: 5000.00


--- Customer 4 ---
Customer Details: Diana Prince (ID: CUST004)
Customer Type: Domestic
Units Consumed: 70
Rate per unit: 5.00
Total Bill: 350.00


--- Customer 5 ---
Customer Details: Eve Adams (ID: CUST005)
Customer Type: Commercial
Units Consumed: 200
Rate per unit: 8.00
Total Bill: 1600.00


--- Customer 6 ---
Customer Details: Frank White (ID: CUST006)
Customer Type: Industrial
Units Consumed: 800
Rate per unit: 10.00
Total Bill: 8000.00
```

Justification:

In this task we are calculating the electricity bill on the basis of type of customer we are having.

Task 3: Modular Design Using AI Assistance (Using Functions)

Scenario

Prompt:

A python code to calculate energy charges and fixed charges using functions and can be reused by multiple customers and return values, also include comments

Code:

```
# Define tariff rates (reusing previous definition for energy charge calculation)
tariff_rates = {
    'Domestic': {
        'slab1_units': 100, 'slab1_rate': 5,
        'slab2_units': 200, 'slab2_rate': 7,
        'slab3_rate': 10
    },
    'Commercial': {
        'slab1_units': 500, 'slab1_rate': 8,
        'slab2_rate': 12
    },
    'Industrial': {
        'slab1_units': 1000, 'slab1_rate': 10,
        'slab2_rate': 15
    }
}

# Define fixed charge rates per customer type (example values)
fixed_charge_rates = {
    'Domestic': 50,
    'Commercial': 150,
    'Industrial': 500
}

def calculate_energy_and_fixed_charges(customer_type, units_consumed):
    """
    Calculates energy charges and fixed charges for a given customer.

    Args:
        customer_type (str): The type of the customer (e.g., 'Domestic', 'Commercial', 'Industrial').
        units_consumed (int): The number of units consumed by the customer.

    Returns:
        tuple: A tuple containing (energy_charge, fixed_charge).
    """
    energy_charge = 0
    fixed_charge = 0

    # Calculate Fixed Charge
    fixed_charge = fixed_charge_rates.get(customer_type, 0) # Get fixed charge, default to 0 if type not found


    # Calculate Energy Charge based on customer type and slab rates
    if customer_type == 'Domestic':
        if units_consumed <= tariff_rates['Domestic']['slab1_units']:
            energy_charge = units_consumed * tariff_rates['Domestic']['slab1_rate']
        elif units_consumed <= (tariff_rates['Domestic']['slab1_units'] + tariff_rates['Domestic']['slab2_units']):
            energy_charge = (tariff_rates['Domestic']['slab1_units'] * tariff_rates['Domestic']['slab1_rate']) + \
                            ((units_consumed - tariff_rates['Domestic']['slab1_units']) * tariff_rates['Domestic']['slab2_rate'])
        else:
            energy_charge = (tariff_rates['Domestic']['slab1_units'] * tariff_rates['Domestic']['slab1_rate']) + \
                            (tariff_rates['Domestic']['slab2_units'] * tariff_rates['Domestic']['slab2_rate']) + \
                            ((units_consumed - tariff_rates['Domestic']['slab1_units'] - tariff_rates['Domestic']['slab2_units']) * tariff_rates['Domestic']['slab3_rate'])

    elif customer_type == 'Commercial':
        if units_consumed <= tariff_rates['Commercial']['slab1_units']:
            energy_charge = units_consumed * tariff_rates['Commercial']['slab1_rate']
        else:
            energy_charge = (tariff_rates['Commercial']['slab1_units'] * tariff_rates['Commercial']['slab1_rate']) + \
                            ((units_consumed - tariff_rates['Commercial']['slab1_units']) * tariff_rates['Commercial']['slab2_rate'])

    elif customer_type == 'Industrial':
        if units_consumed <= tariff_rates['Industrial']['slab1_units']:
            energy_charge = units_consumed * tariff_rates['Industrial']['slab1_rate']
        else:
            energy_charge = (tariff_rates['Industrial']['slab1_units'] * tariff_rates['Industrial']['slab1_rate']) + \
                            ((units_consumed - tariff_rates['Industrial']['slab1_units']) * tariff_rates['Industrial']['slab2_rate'])

    return energy_charge, fixed_charge

# Apply the function to the DataFrame to calculate Energy and Fixed Charges
df[['Energy Charge', 'Fixed Charge']] = df.apply(
    lambda row: calculate_energy_and_fixed_charges(row['Type of Customer'], row['Units Consumed']),
    axis=1,
    result_type='expand'
)
```

```python
# Calculate Total Bill by summing Energy Charge and Fixed Charge
df['Total Bill (Energy + Fixed)'] = df['Energy Charge'] + df['Fixed Charge']

# Display the DataFrame with the new charges
print("\n--- Customer Details with Energy, Fixed, and Total Charges ---")
for index, row in df.iterrows():
    print(f"--- Customer {row['Serial Number']} ---")
    print(f"Customer Name: {row['Name']} (ID: {row['Customer ID']})")
    print(f"Customer Type: {row['Type of Customer']}")
    print(f"Units Consumed: {row['Units Consumed']}")
    print(f"Energy Charge: {row['Energy Charge']:.2f}")
    print(f"Fixed Charge: {row['Fixed Charge']:.2f}")
    print(f"Total Bill: {row['Total Bill (Energy + Fixed)']:.2f}")
    print("\n")
```

Output:

```
--- Customer Details with All Charges ---
--- Customer 1 ---
Customer Name: Alice Smith (ID: CUST001)
Customer Type: Domestic
Units Consumed: 50
Energy Charge: 250.00
Fixed Charge: 50.00
Customer Charge: 20.00
Electricity Duty: 12.50
Total Bill: 332.50


--- Customer 2 ---
Customer Name: Bob Johnson (ID: CUST002)
Customer Type: Commercial
Units Consumed: 250
Energy Charge: 2000.00
Fixed Charge: 150.00
Customer Charge: 75.00
Electricity Duty: 140.00
Total Bill: 2365.00


--- Customer 3 ---
Customer Name: Charlie Brown (ID: CUST003)
Customer Type: Industrial
Units Consumed: 500
Energy Charge: 5000.00
Fixed Charge: 500.00
Customer Charge: 200.00
Electricity Duty: 500.00
Total Bill: 6200.00


--- Customer 4 ---
Customer Name: Diana Prince (ID: CUST004)
Customer Type: Domestic
Units Consumed: 70
Energy Charge: 350.00
Fixed Charge: 50.00
Customer Charge: 20.00
Electricity Duty: 17.50
Total Bill: 437.50


--- Customer 5 ---
Customer Name: Eve Adams (ID: CUST005)
Customer Type: Commercial
Units Consumed: 200
Energy Charge: 1600.00
Fixed Charge: 150.00
Customer Charge: 75.00
Electricity Duty: 112.00
Total Bill: 1937.00


--- Customer 6 ---
Customer Name: Frank White (ID: CUST006)
Customer Type: Industrial
Units Consumed: 800
Energy Charge: 8000.00
Fixed Charge: 500.00
Customer Charge: 200.00
Electricity Duty: 800.00
Total Bill: 9500.00
```

Justification:

In this task not only calculating the energy charges, we are also adding the fixed charges to each customer using functions, we are also adding the required comments for better understanding and easy readability.

Task 4: Calculation of Additional Charges

Prompt:

Add additional charges like fixed charges, customer charges, electricity duty (percentage of EC) and improve billing accuracy

Code:

```python
# Define tariff rates (reusing previous definition for energy charge calculation)
tariff_rates = {
    'Domestic': {
        'slab1_units': 100, 'slab1_rate': 5,
        'slab2_units': 200, 'slab2_rate': 7,
        'slab3_rate': 10
    },
    'Commercial': {
        'slab1_units': 500, 'slab1_rate': 8,
        'slab2_rate': 12
    },
    'Industrial': {
        'slab1_units': 1000, 'slab1_rate': 10,
        'slab2_rate': 15
    }
}

# Define fixed charge rates per customer type (example values)
fixed_charge_rates = {
    'Domestic': 50,
    'Commercial': 150,
    'Industrial': 500
}

# Define customer charge rates per customer type (example values)
customer_charge_rates = {
    'Domestic': 20,
    'Commercial': 75,
    'Industrial': 200
}

# Define electricity duty percentage per customer type (example values)
electricity_duty_percentage = {
    'Domestic': 0.05, # 5% of energy charge
    'Commercial': 0.07, # 7% of energy charge
    'Industrial': 0.10 # 10% of energy charge
}
```

```python
def calculate_all_charges(customer_type, units_consumed):
    """
    Calculates energy charges, fixed charges, customer charges, and electricity duty
    for a given customer.

    Args:
        customer_type (str): The type of the customer (e.g., 'Domestic', 'Commercial', 'Industrial').
        units_consumed (int): The number of units consumed by the customer.

    Returns:
        tuple: A tuple containing (energy_charge, fixed_charge, customer_charge, electricity_duty).
    """
    energy_charge = 0
    fixed_charge = 0
    customer_charge = 0
    electricity_duty = 0

    # Calculate Fixed Charge
    fixed_charge = fixed_charge_rates.get(customer_type, 0)

    # Calculate Customer Charge
    customer_charge = customer_charge_rates.get(customer_type, 0)

    # Handle zero units consumed for energy charge
    if units_consumed == 0:
        return energy_charge, fixed_charge, customer_charge, electricity_duty

    # Calculate Energy Charge based on customer type and slab rates
    if customer_type == 'Domestic':
        if units_consumed <= tariff_rates['Domestic']['slab1_units']:
            energy_charge = units_consumed * tariff_rates['Domestic']['slab1_rate']
        elif units_consumed <= (tariff_rates['Domestic']['slab1_units'] + tariff_rates['Domestic']['slab2_units']):
            energy_charge = (tariff_rates['Domestic']['slab1_units'] * tariff_rates['Domestic']['slab1_rate']) + \
                            ((units_consumed - tariff_rates['Domestic']['slab1_units']) * tariff_rates['Domestic']['slab2_rate'])
        else:
            energy_charge = (tariff_rates['Domestic']['slab1_units'] * tariff_rates['Domestic']['slab1_rate']) + \
                            (tariff_rates['Domestic']['slab2_units'] * tariff_rates['Domestic']['slab2_rate']) + \
                            ((units_consumed - tariff_rates['Domestic']['slab1_units'] - tariff_rates['Domestic']['slab2_units']) * tariff_rates['Domestic']['slab3_rate'])
```

```
    elif customer_type == 'Commercial':
        if units_consumed <= tariff_rates['Commercial']['slab1_units']:
            energy_charge = units_consumed * tariff_rates['Commercial']['slab1_rate']
        else:
            energy_charge = (tariff_rates['Commercial']['slab1_units'] * tariff_rates['Commercial']['slab1_rate']) + \
                            ((units_consumed - tariff_rates['Commercial']['slab1_units']) * tariff_rates['Commercial']['slab2_rate'])

    elif customer_type == 'Industrial':
        if units_consumed <= tariff_rates['Industrial']['slab1_units']:
            energy_charge = units_consumed * tariff_rates['Industrial']['slab1_rate']
        else:
            energy_charge = (tariff_rates['Industrial']['slab1_units'] * tariff_rates['Industrial']['slab1_rate']) + \
                            ((units_consumed - tariff_rates['Industrial']['slab1_units']) * tariff_rates['Industrial']['slab2_rate'])

    # Calculate Electricity Duty (percentage of energy charge)
    duty_percentage = electricity_duty_percentage.get(customer_type, 0)
    electricity_duty = energy_charge * duty_percentage

    return energy_charge, fixed_charge, customer_charge, electricity_duty

# Apply the function to the DataFrame to calculate Energy, Fixed, Customer Charges and Electricity Duty
df[['Energy Charge', 'Fixed Charge', 'Customer Charge', 'Electricity Duty']] = df.apply(
    lambda row: calculate_all_charges(row['Type of Customer'], row['Units Consumed']),
    axis=1,
    result_type='expand'
)

# Calculate Total Bill by summing all charges
df['Total Bill (All Charges)'] = df['Energy Charge'] + df['Fixed Charge'] + df['Customer Charge'] + df['Electricity Duty']

# Display the DataFrame with the new charges
print("\n--- Customer Details with All Charges ---")
for index, row in df.iterrows():
    print(f"--- Customer {row['Serial Number']} ---")
    print(f"Customer Name: {row['Name']} (ID: {row['Customer ID']})")
    print(f"Customer Type: {row['Type of Customer']}")
    print(f"Units Consumed: {row['Units Consumed']}")
    print(f"Energy Charge: {row['Energy Charge']:.2f}")
    print(f"Fixed Charge: {row['Fixed Charge']:.2f}")
    print(f"Customer Charge: {row['Customer Charge']:.2f}")
    print(f"Electricity Duty: {row['Electricity Duty']:.2f}")
    print(f"Total Bill: {row['Total Bill (All Charges)']:.2f}")
    print("\n")
```

Output:

```
--- Customer Details with All Charges ---
--- Customer 1 ---
Customer Name: Alice Smith (ID: CUST001)
Customer Type: Domestic
Units Consumed: 50
Energy Charge: 250.00
Fixed Charge: 50.00
Customer Charge: 20.00
Electricity Duty: 12.50
Total Bill: 332.50


--- Customer 2 ---
Customer Name: Bob Johnson (ID: CUST002)
Customer Type: Commercial
Units Consumed: 250
Energy Charge: 2000.00
Fixed Charge: 150.00
Customer Charge: 75.00
Electricity Duty: 140.00
Total Bill: 2365.00


--- Customer 3 ---
Customer Name: Charlie Brown (ID: CUST003)
Customer Type: Industrial
Units Consumed: 500
Energy Charge: 5000.00
Fixed Charge: 500.00
Customer Charge: 200.00
Electricity Duty: 500.00
Total Bill: 6200.00


--- Customer 4 ---
Customer Name: Diana Prince (ID: CUST004)
Customer Type: Domestic
Units Consumed: 70
Energy Charge: 350.00
Fixed Charge: 50.00
Customer Charge: 20.00
Electricity Duty: 17.50
Total Bill: 437.50



--- Customer 5 ---
Customer Name: Eve Adams (ID: CUST005)
Customer Type: Commercial
Units Consumed: 200
Energy Charge: 1600.00
Fixed Charge: 150.00
Customer Charge: 75.00
Electricity Duty: 112.00
Total Bill: 1937.00


--- Customer 6 ---
Customer Name: Frank White (ID: CUST006)
Customer Type: Industrial
Units Consumed: 800
Energy Charge: 8000.00
Fixed Charge: 500.00
Customer Charge: 200.00
Electricity Duty: 800.00
Total Bill: 9500.00
```

Justification:

In this task we are also adding few other charges to bill like Customer charges, Electricity Duty(5% of Energy Charges). We have done this using functions and required comments were kept for easy understanding purpose.

Task 5: Final Bill Generation and Output Analysis Scenario

Prompt:

Display the final bill clearly with each charges, total bill is sum of EC, FC, CC, ED

Code:

```
# Display the DataFrame with all charges
print("\n--- Final Customer Bill Details ---")
for index, row in df.iterrows():
    print(f"--- Customer {row['Serial Number']} ---")
    print(f"Customer Name: {row['Name']} (ID: {row['Customer ID']})")
    print(f"Customer Type: {row['Type of Customer']}")
    print(f"Units Consumed: {row['Units Consumed']}")
    print(f"Energy Charge (EC): {row['Energy Charge']:.2f}")
    print(f"Fixed Charge (FC): {row['Fixed Charge']:.2f}")
    print(f"Customer Charge (CC): {row['Customer Charge']:.2f}")
    print(f"Electricity Duty (ED): {row['Electricity Duty']:.2f}")
    print(f"Total Bill (EC + FC + CC + ED): {row['Total Bill (All Charges)']:.2f}")
    print("\n")
```

Output:

```
...
    --- Final Customer Bill Details ---
    --- Customer 1 ---
    Customer Name: Alice Smith (ID: CUST001)
    Customer Type: Domestic
    Units Consumed: 50
    Energy Charge (EC): 250.00
    Fixed Charge (FC): 50.00
    Customer Charge (CC): 20.00
    Electricity Duty (ED): 12.50
    Total Bill (EC + FC + CC + ED): 332.50


    --- Customer 2 ---
    Customer Name: Bob Johnson (ID: CUST002)
    Customer Type: Commercial
    Units Consumed: 250
    Energy Charge (EC): 2000.00
    Fixed Charge (FC): 150.00
    Customer Charge (CC): 75.00
    Electricity Duty (ED): 140.00
    Total Bill (EC + FC + CC + ED): 2365.00


    --- Customer 3 ---
    Customer Name: Charlie Brown (ID: CUST003)
    Customer Type: Industrial
    Units Consumed: 500
    Energy Charge (EC): 5000.00
    Fixed Charge (FC): 500.00
    Customer Charge (CC): 200.00
    Electricity Duty (ED): 500.00
    Total Bill (EC + FC + CC + ED): 6200.00
```

```
--- Customer 4 ---
Customer Name: Diana Prince (ID: CUST004)
Customer Type: Domestic
Units Consumed: 70
Energy Charge (EC): 350.00
Fixed Charge (FC): 50.00
Customer Charge (CC): 20.00
Electricity Duty (ED): 17.50
Total Bill (EC + FC + CC + ED): 437.50


--- Customer 5 ---
Customer Name: Eve Adams (ID: CUST005)
Customer Type: Commercial
Units Consumed: 200
Energy Charge (EC): 1600.00
Fixed Charge (FC): 150.00
Customer Charge (CC): 75.00
Electricity Duty (ED): 112.00
Total Bill (EC + FC + CC + ED): 1937.00


--- Customer 6 ---
Customer Name: Frank White (ID: CUST006)
Customer Type: Industrial
Units Consumed: 800
Energy Charge (EC): 8000.00
Fixed Charge (FC): 500.00
Customer Charge (CC): 200.00
Electricity Duty (ED): 800.00
Total Bill (EC + FC + CC + ED): 9500.00
```

Justification:

Here, in this task we are finally printing the total bill in perfect arrangements and with full details.