Name: K.Likitha

Hall Ticket: 2303A52144

Batch: 41

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

Prompt:

Write a Python program (no functions) to read electricity_data.csv from Google Colab, validate that current units are greater than or equal to previous units, calculate units consumed, and display consumer ID, customer type, previous units, current units, and units consumed in a clear format.

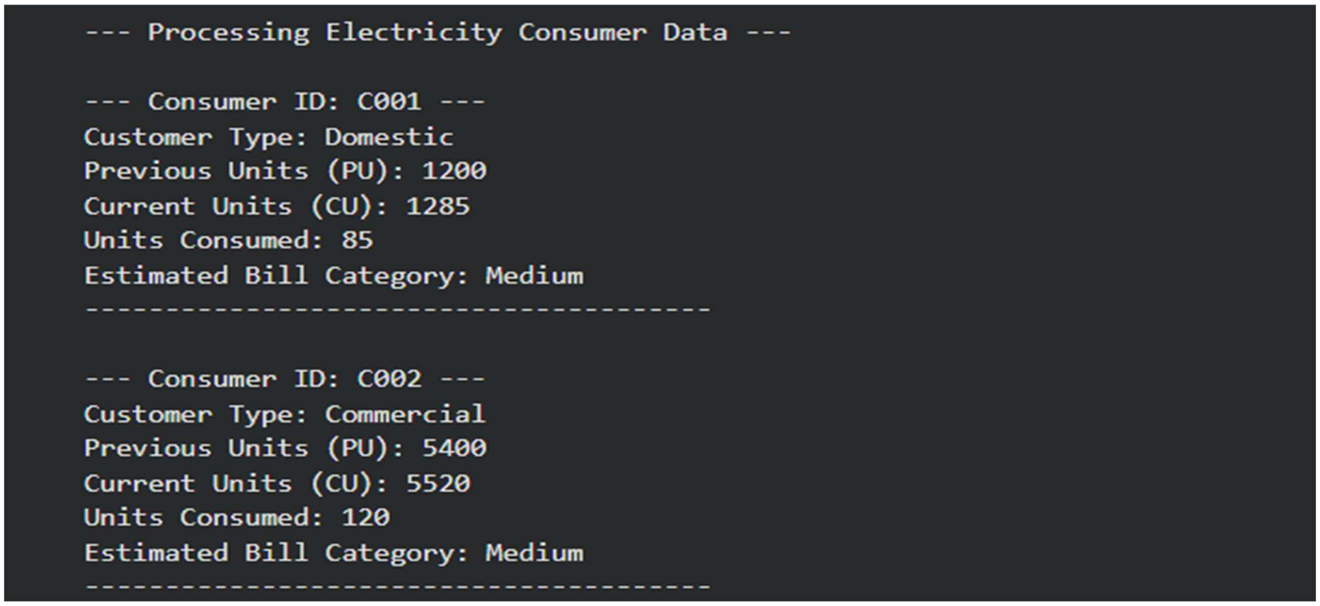
Updated Prompt:

Enhance the existing Python program by adding logic to detect unusual conditions such as high electricity usage or invalid readings, without changing the current output format. Display a simple informational flag after each consumer's details while keeping all original output lines unchanged.

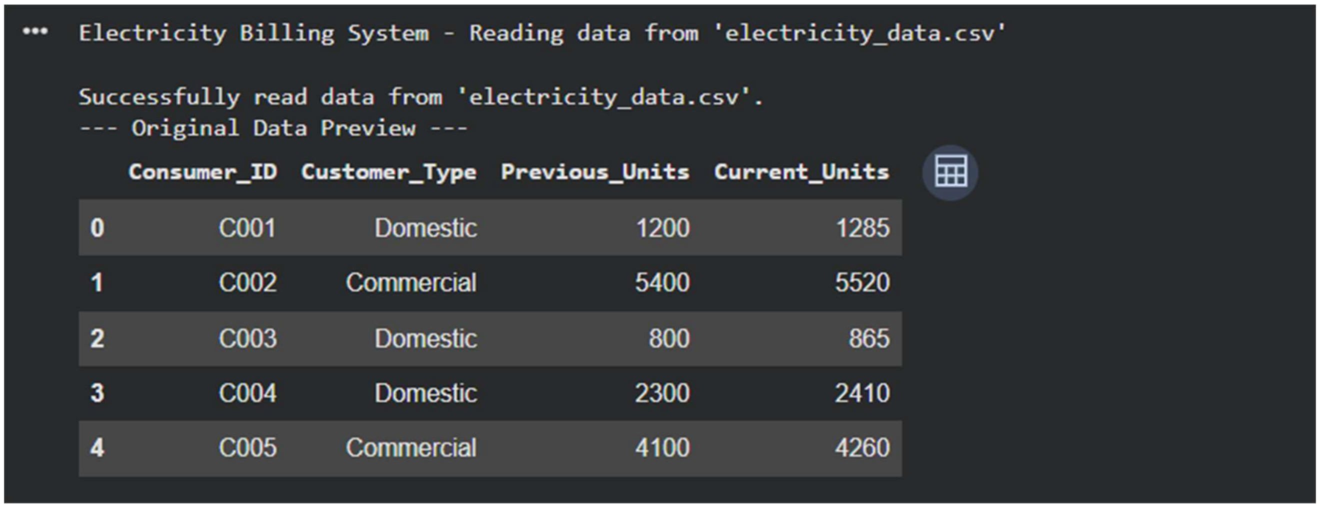
Code:

```
import pandas as pd

print("Electricity Billing System - Reading data from 'electricity_data.csv'\n")

# Read the CSV file into a DataFrame
try:
    df = pd.read_csv('electricity_data.csv')
    print("Successfully read data from 'electricity_data.csv'.")
    print("--- Original Data Preview ---")
    display(df.head())
    print("\n--- Processing Electricity Consumer Data ---\n")

except FileNotFoundError:
    print("Error: 'electricity_data.csv' not found. Please ensure the file is
uploaded to the Colab environment.")
    # Exit the program gracefully if the file is not found
    # In a notebook environment, 'exit()' would terminate the kernel, so we'll just
print an error and stop processing.
    df = pd.DataFrame() # Create an empty DataFrame to prevent further errors

# Define thresholds for bill categories
LOW_CONSUMPTION_THRESHOLD = 70
MEDIUM_CONSUMPTION_THRESHOLD = 150 # Upper bound for Medium category

# Check if the DataFrame is empty (e.g., if file was not found)
if not df.empty:
    # Iterate through each row of the DataFrame using .itertuples() for efficiency
    # We are iterating directly within the main program structure as requested, no
functions.
    for row in df.itertuples(index=False): # index=False to access columns by name
        consumer_id = row.Consumer_ID
        customer_type = row.Customer_Type
        previous_units = row.Previous_Units
        current_units = row.Current_Units
        print(f"--- Consumer ID: {consumer_id} ---")

        # Validate Current Units vs. Previous Units
        if current_units < previous_units:
            print(f"Validation Error: Current Units ({current_units}) cannot be
less than Previous Units ({previous_units}) for this consumer.")
            print("---------------------------------------\n")
            continue # Skip to the next consumer if validation fails

        # Calculate Units Consumed
        units_consumed = current_units - previous_units

        # Determine Estimated Bill Category
        estimated_bill_category = ""
        if units_consumed <= LOW_CONSUMPTION_THRESHOLD:
            estimated_bill_category = "Low"
        elif units_consumed <= MEDIUM_CONSUMPTION_THRESHOLD:
            estimated_bill_category = "Medium"
        else:
            estimated_bill_category = "High"

        # Display Results (Original output lines remain intact)
        print(f"Customer Type: {customer_type}")
        print(f"Previous Units (PU): {previous_units}")
        print(f"Current Units (CU): {current_units}")
        print(f"Units Consumed: {units_consumed}")

        # Add Estimated Bill Category as an additional line (no other messages or
alerts)
        print(f"Estimated Bill Category: {estimated_bill_category}")

        print("---------------------------------------\n")

print("--- All consumer data processed --- ")
```

Output:



```
--- Processing Electricity Consumer Data ---

--- Consumer ID: C001 ---
Customer Type: Domestic
Previous Units (PU): 1200
Current Units (CU): 1285
Units Consumed: 85
Estimated Bill Category: Medium
---------------------------------------

--- Consumer ID: C002 ---
Customer Type: Commercial
Previous Units (PU): 5400
Current Units (CU): 5520
Units Consumed: 120
Estimated Bill Category: Medium
---------------------------------------
```



```
Electricity Billing System - Reading data from 'electricity_data.csv'

Successfully read data from 'electricity_data.csv'.
--- Original Data Preview ---
    Consumer_ID  Customer_Type  Previous_Units  Current_Units
0          C001       Domestic            1200           1285
1          C002     Commercial            5400           5520
2          C003       Domestic             800            865
3          C004       Domestic            2300           2410
4          C005     Commercial            4100           4260
```

Explanation:

The program reads electricity consumer data from a CSV file and displays the details for each consumer. It validates the meter readings by ensuring that the current units are greater than or equal to the previous units. After validation, it calculates the units consumed and presents the results in a structured and readable format.

Task 2: Energy Charges Calculation Based on Units Consumed Scenario

Prompt:

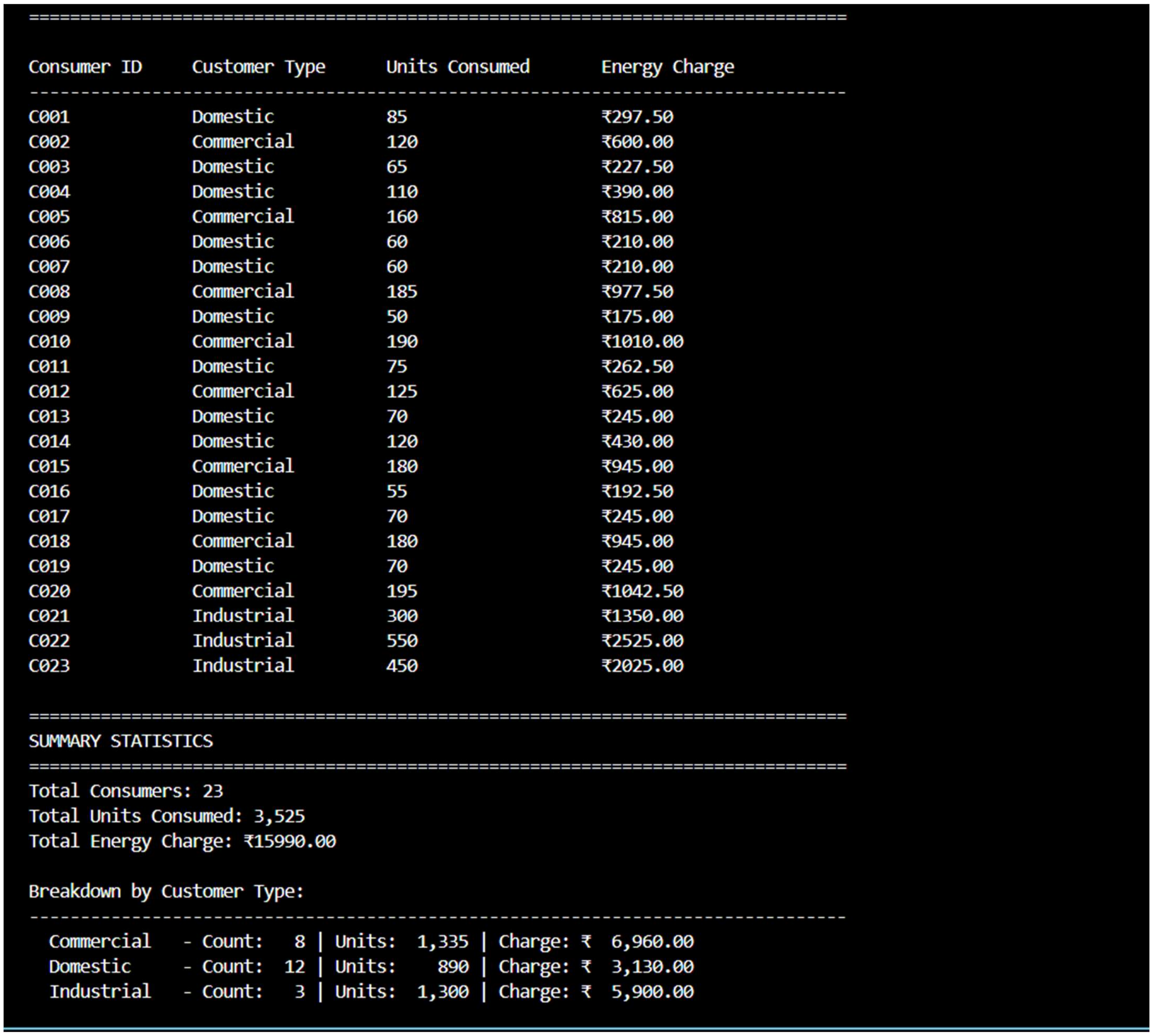
Write a Python program to read electricity consumer data from electricity_data2.csv and calculate Energy Charges based on units consumed. Use clear and optimized conditional statements for Domestic, Commercial, and Industrial consumers, and display the results in a readable format.

Code:

```python
import csv

class ElectricityChargeCalculator:
    """Calculate energy charges based on consumer type and units consumed."""

    # Define tariff rates for different consumer types (per unit in currency)
    TARIFF_RATES = {
        'Domestic': [
            (100, 3.50),      # 0-100 units: Rs 3.50 per unit
            (200, 4.00),      # 101-200 units: Rs 4.00 per unit
            (float('inf'), 5.50)  # Above 200 units: Rs 5.50 per unit
        ],
        'Commercial': [
            (150, 5.00),      # 0-150 units: Rs 5.00 per unit
            (300, 6.50),      # 151-300 units: Rs 6.50 per unit
            (float('inf'), 8.00)  # Above 300 units: Rs 8.00 per unit
        ],
        'Industrial': [
            (500, 4.50),      # 0-500 units: Rs 4.50 per unit
            (1000, 5.50),     # 501-1000 units: Rs 5.50 per unit
            (float('inf'), 7.00)  # Above 1000 units: Rs 7.00 per unit
        ]
    }

    @staticmethod
    def calculate_charge(units, customer_type):
        """
        Calculate energy charge based on units consumed and customer type.

        Args:
            units (int): Number of units consumed
            customer_type (str): Type of customer (Domestic, Commercial, Industrial)

        Returns:
            float: Total energy charge
        """
        charge = 0
        tariff_slabs = ElectricityChargeCalculator.TARIFF_RATES.get(customer_type)

        if not tariff_slabs:
            raise ValueError(f"Invalid customer type: {customer_type}")

        remaining_units = units
        previous_limit = 0

        for limit, rate in tariff_slabs:
            if remaining_units <= 0:
                break

            current_limit = min(limit, limit)
            units_in_slab = min(remaining_units, current_limit - previous_limit)
            charge += units_in_slab * rate
            remaining_units -= units_in_slab
            previous_limit = current_limit

        return round(charge, 2)

    @staticmethod
    def process_file(filename):
        """
        Read electricity data from CSV and calculate charges.

        Args:
            filename (str): Path to the CSV file

        Returns:
            list: List of dictionaries with consumer data and calculated charges
        """
        results = []

        try:
            with open(filename, 'r', newline='') as csvfile:
                reader = csv.DictReader(csvfile)

                for row in reader:
                    try:
                        consumer_id = row['Consumer_ID']
                        customer_type = row['Customer_Type']
                        previous_units = int(row['Previous_Units'])
                        current_units = int(row['Current_Units'])

                        # Calculate units consumed in the billing period
                        units_consumed = current_units - previous_units

                        # Calculate energy charge
                        energy_charge = ElectricityChargeCalculator.calculate_charge(
                            units_consumed, customer_type
                        )

                        results.append({
                            'Consumer_ID': consumer_id,
                            'Customer_Type': customer_type,
                            'Units_Consumed': units_consumed,
                            'Energy_Charge': f"₹{energy_charge:.2f}"
                        })
                    except ValueError as e:
                        print(f"Warning: Skipping row with invalid data: {row}")

        except FileNotFoundError:
            print(f"Error: File '{filename}' not found.")
            return []
        except Exception as e:
            print(f"Error reading file: {e}")
            return []

        return results

    @staticmethod
    def display_results(results):
        """
        Display results in a well-formatted table.

        Args:
            results (list): List of dictionaries with calculation results
        """
        if not results:
            print("No data to display.")
            return

        print("\n" + "="*80)
        print("ELECTRICITY CHARGE CALCULATION REPORT")
        print("="*80)
        print()

        # Display table header
        print(f"{'Consumer ID':<15} {'Customer Type':<18} {'Units Consumed':<20} {'Energy
Charge':<15}")
        print("-" * 80)

        # Display each row
        for r in results:
            print(f"{r['Consumer_ID']:<15} {r['Customer_Type']:<18}
{r['Units_Consumed']:<20} {r['Energy_Charge']:<15}")
```

```
        print()

        # Display summary statistics
        print("="*80)
        print("SUMMARY STATISTICS")
        print("="*80)

        total_units = sum(r['Units_Consumed'] for r in results)
        total_charge = sum(
            float(r['Energy_Charge'].replace('₹', '')) for r in results
        )

        consumer_types = {}
        for r in results:
            ctype = r['Customer_Type']
            if ctype not in consumer_types:
                consumer_types[ctype] = {'count': 0, 'units': 0, 'charge': 0}
            consumer_types[ctype]['count'] += 1
            consumer_types[ctype]['units'] += r['Units_Consumed']
            consumer_types[ctype]['charge'] += float(r['Energy_Charge'].replace('₹', ''))

        print(f"Total Consumers: {len(results)}")
        print(f"Total Units Consumed: {total_units:,}")
        print(f"Total Energy Charge: ₹{total_charge:.2f}")
        print()

        print("Breakdown by Customer Type:")
        print("-" * 80)
        for ctype, data in sorted(consumer_types.items()):
            print(f"  {ctype:12} - Count: {data['count']:3} | Units: {data['units']:6,} |
"
                  f"Charge: ₹{data['charge']:10,.2f}")
        print()


def main():
    """Main function to run the electricity charge calculation."""
    filename = 'electricity_data2.csv'

    # Process the file
    calculator = ElectricityChargeCalculator()
    results = calculator.process_file(filename)

    # Display results
    if results:
        calculator.display_results(results)
    else:
        print("No data was processed.")


if __name__ == '__main__':
    main()
```

Output:



```
================================================================================

Consumer ID     Customer Type      Units Consumed       Energy Charge
--------------------------------------------------------------------------------
C001            Domestic           85                   ₹297.50
C002            Commercial         120                  ₹600.00
C003            Domestic           65                   ₹227.50
C004            Domestic           110                  ₹390.00
C005            Commercial         160                  ₹815.00
C006            Domestic           60                   ₹210.00
C007            Domestic           60                   ₹210.00
C008            Commercial         185                  ₹977.50
C009            Domestic           50                   ₹175.00
C010            Commercial         190                  ₹1010.00
C011            Domestic           75                   ₹262.50
C012            Commercial         125                  ₹625.00
C013            Domestic           70                   ₹245.00
C014            Domestic           120                  ₹430.00
C015            Commercial         180                  ₹945.00
C016            Domestic           55                   ₹192.50
C017            Domestic           70                   ₹245.00
C018            Commercial         180                  ₹945.00
C019            Domestic           70                   ₹245.00
C020            Commercial         195                  ₹1042.50
C021            Industrial         300                  ₹1350.00
C022            Industrial         550                  ₹2525.00
C023            Industrial         450                  ₹2025.00

================================================================================
SUMMARY STATISTICS
================================================================================
Total Consumers: 23
Total Units Consumed: 3,525
Total Energy Charge: ₹15990.00

Breakdown by Customer Type:
--------------------------------------------------------------------------------
  Commercial   - Count:   8 | Units:  1,335 | Charge: ₹  6,960.00
  Domestic     - Count:  12 | Units:    890 | Charge: ₹  3,130.00
  Industrial   - Count:   3 | Units:  1,300 | Charge: ₹  5,900.00
```

Explanation:

This program reads electricity consumer data from a CSV file and calculates energy charges based on units consumed. Conditional logic is applied for Domestic, Commercial, and Industrial consumers to ensure accurate billing. AI-assisted coding techniques were used to simplify and optimize the calculation logic while maintaining clear and readable output.

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

Prompt:

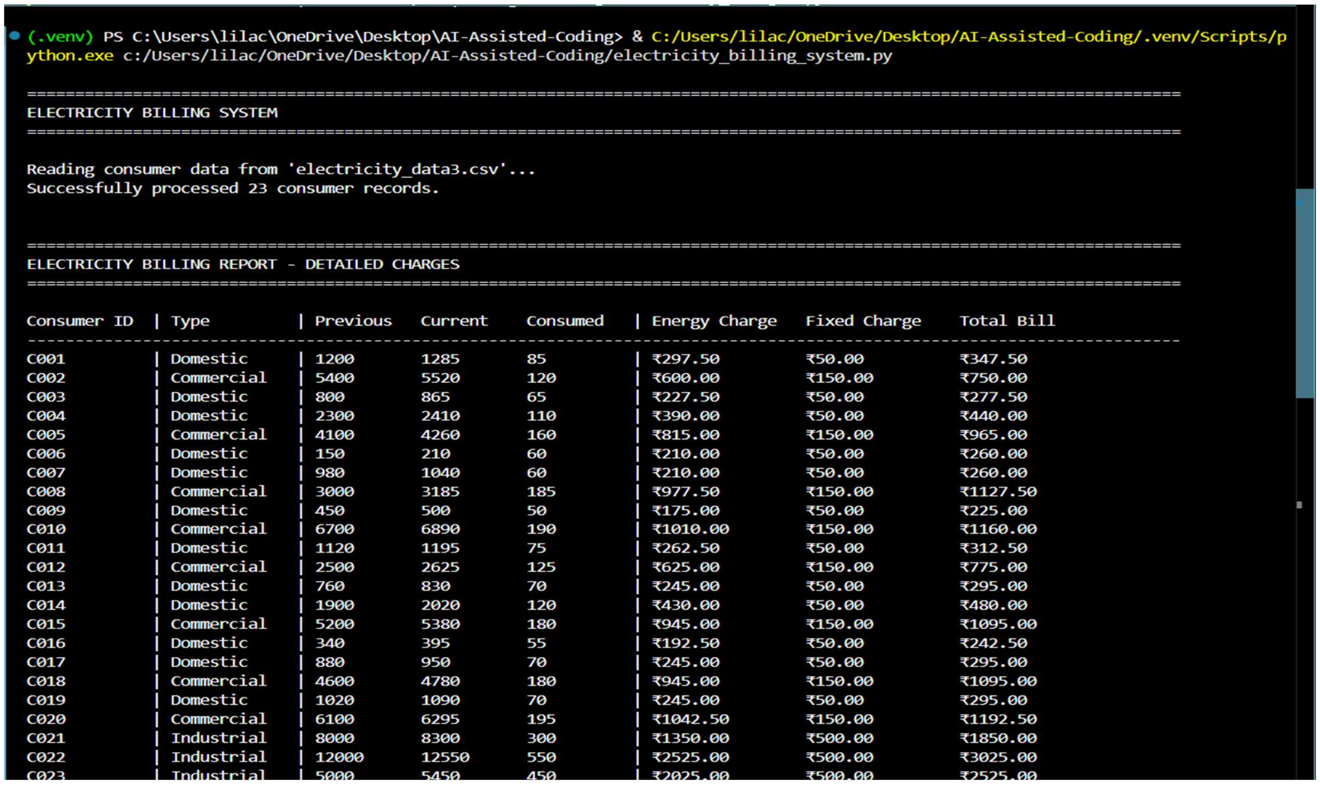
Write a Python program using user-defined functions to calculate Energy Charges and Fixed Charges by reading consumer data from electricity_data3.csv. Return the calculated values, include meaningful comments, and display correct billing results for Domestic, Commercial, and Industrial consumers.

Code:

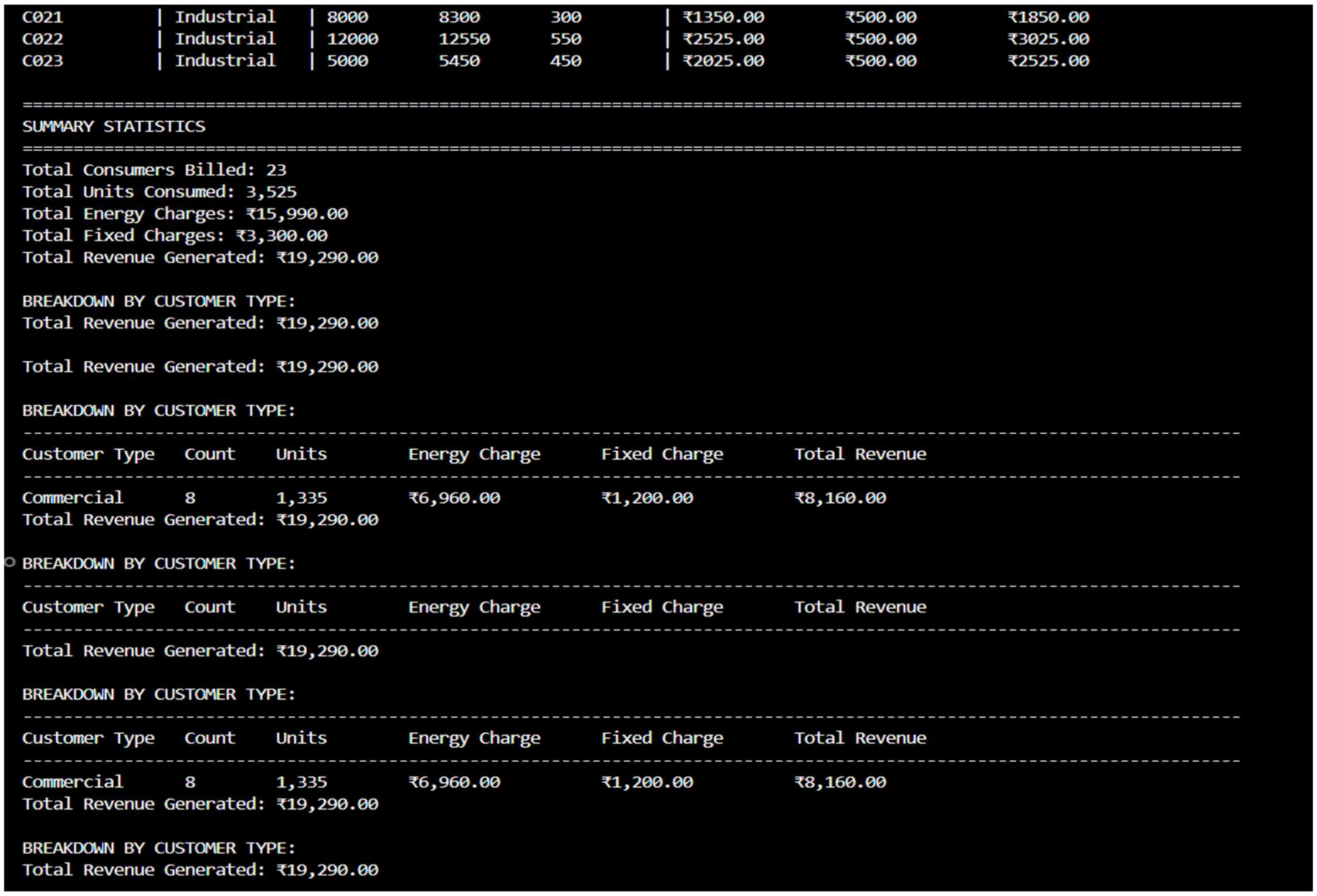
```python
"""
Electricity Billing System
This program calculates energy charges and fixed charges for electricity consumers
based on their consumption and customer type using user-defined functions.
"""

import csv


# ============================================================================
# TARIFF CONFIGURATION - Define rates for different consumer types
# ============================================================================

# Energy Charge Tariff (per unit) - tiered pricing based on units consumed
ENERGY_TARIFF = {
    'Domestic': [
        (100, 3.50),              # 0-100 units: Rs 3.50 per unit
        (200, 4.00),              # 101-200 units: Rs 4.00 per unit
        (float('inf'), 5.50)      # Above 200 units: Rs 5.50 per unit
    ],
    'Commercial': [
        (150, 5.00),              # 0-150 units: Rs 5.00 per unit
        (300, 6.50),              # 151-300 units: Rs 6.50 per unit
        (float('inf'), 8.00)      # Above 300 units: Rs 8.00 per unit
    ],
    'Industrial': [
        (500, 4.50),              # 0-500 units: Rs 4.50 per unit
        (1000, 5.50),             # 501-1000 units: Rs 5.50 per unit
        (float('inf'), 7.00)      # Above 1000 units: Rs 7.00 per unit
    ]
}

# Fixed Charges (monthly) - depends on consumer type
FIXED_CHARGES = {
    'Domestic': 50.00,           # Fixed charge for domestic consumers: Rs 50
    'Commercial': 150.00,        # Fixed charge for commercial consumers: Rs 150
    'Industrial': 500.00         # Fixed charge for industrial consumers: Rs 500
}


# ============================================================================
# USER-DEFINED FUNCTIONS FOR CALCULATIONS
# ============================================================================

def calculate_units_consumed(previous_units, current_units):
    """
    Calculate the units consumed in the billing period.

    Args:
        previous_units (int): Meter reading from previous billing cycle
        current_units (int): Meter reading from current billing cycle

    Returns:
        int: Units consumed during the billing period
    """
    return current_units - previous_units


def calculate_energy_charge(units_consumed, customer_type):
    """
    Calculate energy charge based on tiered tariff rates.
    Uses different slab rates for each customer type.

    Args:
        units_consumed (int): Number of units consumed
        customer_type (str): Type of customer (Domestic, Commercial, Industrial)

    Returns:
        float: Total energy charge calculated

    Raises:
        ValueError: If customer type is invalid
    """
    # Validate customer type
    if customer_type not in ENERGY_TARIFF:
        raise ValueError(f"Invalid customer type: {customer_type}")

    energy_charge = 0.0
    tariff_slabs = ENERGY_TARIFF[customer_type]
    remaining_units = units_consumed
    previous_limit = 0

    # Iterate through each tariff slab and calculate charges
    for slab_limit, rate_per_unit in tariff_slabs:
        if remaining_units <= 0:
            break

        # Calculate units in current slab
        current_slab_units = min(remaining_units, slab_limit - previous_limit)

        # Add charge for units in current slab
        energy_charge += current_slab_units * rate_per_unit

        # Update remaining units and previous limit
        remaining_units -= current_slab_units
        previous_limit = slab_limit

    return round(energy_charge, 2)


def calculate_fixed_charge(customer_type):
    """
    Retrieve the fixed monthly charge for a customer type.

    Args:
        customer_type (str): Type of customer (Domestic, Commercial, Industrial)

    Returns:
        float: Fixed charge amount

    Raises:
        ValueError: If customer type is invalid
    """
    # Check if customer type exists in the fixed charges dictionary
    if customer_type not in FIXED_CHARGES:
        raise ValueError(f"Invalid customer type: {customer_type}")

    return FIXED_CHARGES[customer_type]


def calculate_total_bill(energy_charge, fixed_charge):
    """
    Calculate total bill amount.

    Args:
        energy_charge (float): Energy consumption charge
        fixed_charge (float): Fixed monthly charge

    Returns:
        float: Total bill amount (energy charge + fixed charge)
    """
    return round(energy_charge + fixed_charge, 2)


def read_consumer_data(filename):
    """
    Read electricity consumer data from CSV file and process each record.

    Args:
        filename (str): Path to the CSV file containing consumer data

    Returns:
        list: List of dictionaries containing consumer billing information
    """
    billing_records = []

    try:
        with open(filename, 'r', newline='', encoding='utf-8') as csvfile:
            reader = csv.DictReader(csvfile)

            # Process each row in the CSV file
            for row_number, row in enumerate(reader, start=2):  # Start from 2 (after
header)
                try:
                    # Extract data from CSV row
                    consumer_id = row['Consumer_ID'].strip()
                    customer_type = row['Customer_Type'].strip()
                    previous_units = int(row['Previous_Units'])
                    current_units = int(row['Current_Units'])

                    # Validate data
                    if current_units < previous_units:
```

```python
                        print(f"Warning (Row {row_number}): Current units less than
previous units for {consumer_id}")
                        continue

                    # Calculate billing components
                    units_consumed = calculate_units_consumed(previous_units,
current_units)
                    energy_charge = calculate_energy_charge(units_consumed, customer_type)
                    fixed_charge = calculate_fixed_charge(customer_type)
                    total_bill = calculate_total_bill(energy_charge, fixed_charge)

                    # Store billing record
                    billing_records.append({
                        'Consumer_ID': consumer_id,
                        'Customer_Type': customer_type,
                        'Previous_Units': previous_units,
                        'Current_Units': current_units,
                        'Units_Consumed': units_consumed,
                        'Energy_Charge': energy_charge,
                        'Fixed_Charge': fixed_charge,
                        'Total_Bill': total_bill
                    })

                except ValueError as e:
                    print(f"Warning (Row {row_number}): Invalid data format - {e}")
                    continue
                except KeyError as e:
                    print(f"Warning (Row {row_number}): Missing field {e}")
                    continue

        return billing_records

    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
        return []
    except Exception as e:
        print(f"Error reading file: {e}")
        return []


def display_billing_report(billing_records):
    """
    Display a formatted billing report with all consumer details.

    Args:
        billing_records (list): List of billing record dictionaries
    """
    if not billing_records:
        print("No billing records to display.")
        return

    # Display report header
    print("\n" + "="*120)
    print("ELECTRICITY BILLING REPORT - DETAILED CHARGES")
    print("="*120)
    print()

    # Display table header with column alignment
    header = (
        f"{'Consumer ID':<12} | {'Type':<12} | "
        f"{'Previous':<10} {'Current':<10} {'Consumed':<10} | "
        f"{'Energy Charge':<15} {'Fixed Charge':<15} {'Total Bill':<15}"
    )
    print(header)
    print("-"*120)

    # Display each billing record
    for record in billing_records:
        line = (
            f"{record['Consumer_ID']:<12} | {record['Customer_Type']:<12} | "
            f"{record['Previous_Units']:<10} {record['Current_Units']:<10} "
            f"{record['Units_Consumed']:<10} | "
            f"₹{record['Energy_Charge']:<14.2f} ₹{record['Fixed_Charge']:<14.2f} "
            f"₹{record['Total_Bill']:<14.2f}"
        )
        print(line)

    print()


def display_summary_statistics(billing_records):
    """
    Display summary statistics and breakdown by customer type.

    Args:
        billing_records (list): List of billing record dictionaries
    """
    if not billing_records:
        return

    # Calculate aggregate statistics
    total_consumers = len(billing_records)
    total_units = sum(r['Units_Consumed'] for r in billing_records)
    total_energy_charge = sum(r['Energy_Charge'] for r in billing_records)
    total_fixed_charge = sum(r['Fixed_Charge'] for r in billing_records)
    total_revenue = sum(r['Total_Bill'] for r in billing_records)

    # Display summary header
    print("="*120)
    print("SUMMARY STATISTICS")
    print("="*120)
    print(f"Total Consumers Billed: {total_consumers}")
    print(f"Total Units Consumed: {total_units:,}")
    print(f"Total Energy Charges: ₹{total_energy_charge:,.2f}")
    print(f"Total Fixed Charges: ₹{total_fixed_charge:,.2f}")
    print(f"Total Revenue Generated: ₹{total_revenue:,.2f}")
    print()
    # Calculate and display breakdown by customer type
    customer_breakdown = {}
    for record in billing_records:
        ctype = record['Customer_Type']
        if ctype not in customer_breakdown:
            customer_breakdown[ctype] = {
                'count': 0,
                'units': 0,
                'energy': 0,
                'fixed': 0,
                'total': 0
            }

        customer_breakdown[ctype]['count'] += 1
        customer_breakdown[ctype]['units'] += record['Units_Consumed']
        customer_breakdown[ctype]['energy'] += record['Energy_Charge']
        customer_breakdown[ctype]['fixed'] += record['Fixed_Charge']
        customer_breakdown[ctype]['total'] += record['Total_Bill']

    # Display breakdown table
    print("BREAKDOWN BY CUSTOMER TYPE:")
    print("-"*120)
    print(f"{'Customer Type':<15} {'Count':<8} {'Units':<12} "
          f"{'Energy Charge':<18} {'Fixed Charge':<18} {'Total Revenue':<18}")
    print("-"*120)

    for ctype in sorted(customer_breakdown.keys()):
        data = customer_breakdown[ctype]
        print(
            f"{ctype:<15} {data['count']:<8} {data['units']:<12,} "
            f"₹{data['energy']:<17,.2f} ₹{data['fixed']:<17,.2f} "
            f"₹{data['total']:<17,.2f}"
        )

    print()


# ============================================================================
# MAIN PROGRAM EXECUTION
# ============================================================================

def main():
    """
    Main function to orchestrate the billing calculation and reporting process.
    """
    print("\n" + "="*120)
    print("ELECTRICITY BILLING SYSTEM")
    print("="*120)

    # Define input file
    input_file = 'electricity_data3.csv'

    # Read consumer data and calculate bills
    print(f"\nReading consumer data from '{input_file}'...")
    billing_records = read_consumer_data(input_file)

    if not billing_records:
        print("No billing records were processed.")
        return

    print(f"Successfully processed {len(billing_records)} consumer records.\n")

    # Display detailed billing report
    display_billing_report(billing_records)

    # Display summary statistics
    display_summary_statistics(billing_records)

    print("="*120)
    print("Report generated successfully!")
    print("="*120 + "\n")


# ============================================================================
# SCRIPT ENTRY POINT
# ============================================================================

if __name__ == '__main__':
    main()
```

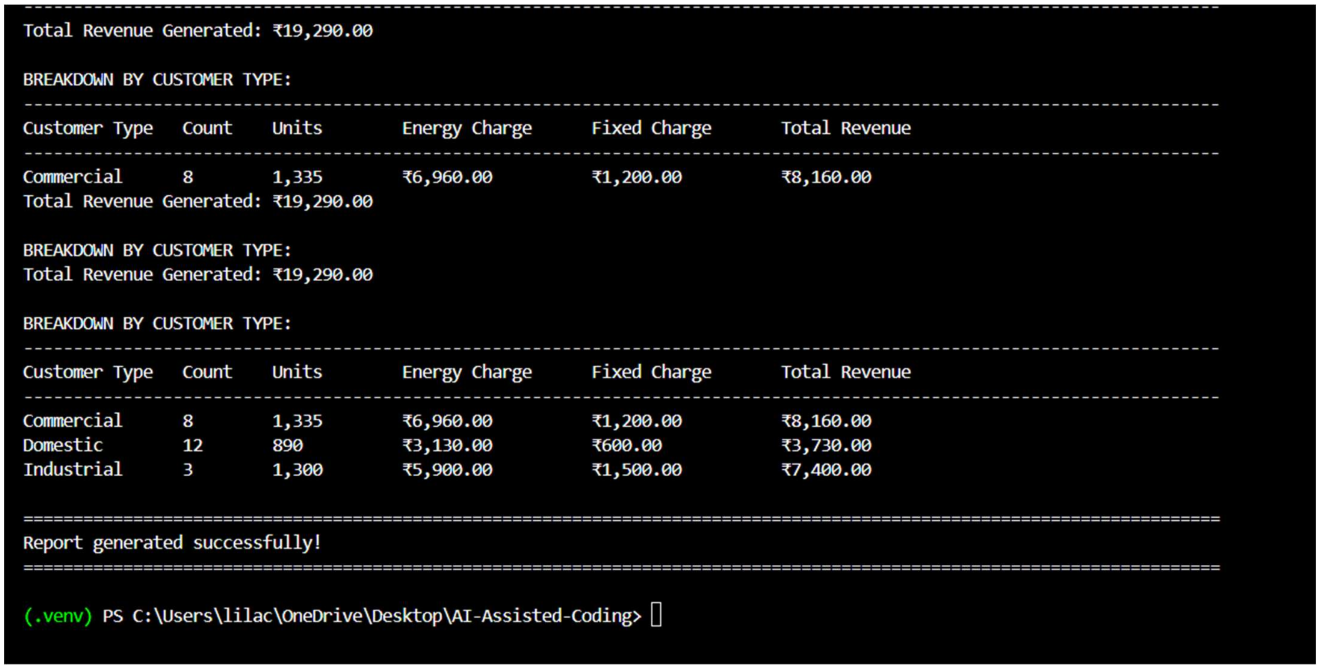
Output:



```
(.venv) PS C:\Users\lilac\OneDrive\Desktop\AI-Assisted-Coding> & C:/Users/lilac/OneDrive/Desktop/AI-Assisted-Coding/.venv/Scripts/p
ython.exe c:/Users/lilac/OneDrive/Desktop/AI-Assisted-Coding/electricity_billing_system.py

========================================================================================================================
ELECTRICITY BILLING SYSTEM
========================================================================================================================

Reading consumer data from 'electricity_data3.csv'...
Successfully processed 23 consumer records.


========================================================================================================================
ELECTRICITY BILLING REPORT - DETAILED CHARGES
========================================================================================================================

Consumer ID  | Type          | Previous   Current    Consumed   | Energy Charge   Fixed Charge    Total Bill
------------------------------------------------------------------------------------------------------------------------
C001         | Domestic      | 1200       1285       85         | ₹297.50         ₹50.00          ₹347.50
C002         | Commercial    | 5400       5520       120        | ₹600.00         ₹150.00         ₹750.00
C003         | Domestic      | 800        865        65         | ₹227.50         ₹50.00          ₹277.50
C004         | Domestic      | 2300       2410       110        | ₹390.00         ₹50.00          ₹440.00
C005         | Commercial    | 4100       4260       160        | ₹815.00         ₹150.00         ₹965.00
C006         | Domestic      | 150        210        60         | ₹210.00         ₹50.00          ₹260.00
C007         | Domestic      | 980        1040       60         | ₹210.00         ₹50.00          ₹260.00
C008         | Commercial    | 3000       3185       185        | ₹977.50         ₹150.00         ₹1127.50
C009         | Domestic      | 450        500        50         | ₹175.00         ₹50.00          ₹225.00
C010         | Commercial    | 6700       6890       190        | ₹1010.00        ₹150.00         ₹1160.00
C011         | Domestic      | 1120       1195       75         | ₹262.50         ₹50.00          ₹312.50
C012         | Commercial    | 2500       2625       125        | ₹625.00         ₹150.00         ₹775.00
C013         | Domestic      | 760        830        70         | ₹245.00         ₹50.00          ₹295.00
C014         | Domestic      | 1900       2020       120        | ₹430.00         ₹50.00          ₹480.00
C015         | Commercial    | 5200       5380       180        | ₹945.00         ₹150.00         ₹1095.00
C016         | Domestic      | 340        395        55         | ₹192.50         ₹50.00          ₹242.50
C017         | Domestic      | 880        950        70         | ₹245.00         ₹50.00          ₹295.00
C018         | Commercial    | 4600       4780       180        | ₹945.00         ₹150.00         ₹1095.00
C019         | Domestic      | 1020       1090       70         | ₹245.00         ₹50.00          ₹295.00
C020         | Commercial    | 6100       6295       195        | ₹1042.50        ₹150.00         ₹1192.50
C021         | Industrial    | 8000       8300       300        | ₹1350.00        ₹500.00         ₹1850.00
C022         | Industrial    | 12000      12550      550        | ₹2525.00        ₹500.00         ₹3025.00
C023         | Industrial    | 5000       5450       450        | ₹2025.00        ₹500.00         ₹2525.00
```



```
C021        | Industrial  | 8000      8300      300       | ₹1350.00        ₹500.00         ₹1850.00
C022        | Industrial  | 12000     12550     550       | ₹2525.00        ₹500.00         ₹3025.00
C023        | Industrial  | 5000      5450      450       | ₹2025.00        ₹500.00         ₹2525.00

====================================================================================================
SUMMARY STATISTICS
====================================================================================================
Total Consumers Billed: 23
Total Units Consumed: 3,525
Total Energy Charges: ₹15,990.00
Total Fixed Charges: ₹3,300.00
Total Revenue Generated: ₹19,290.00

BREAKDOWN BY CUSTOMER TYPE:
Total Revenue Generated: ₹19,290.00

Total Revenue Generated: ₹19,290.00

BREAKDOWN BY CUSTOMER TYPE:
----------------------------------------------------------------------------------------------------
Customer Type   Count    Units        Energy Charge      Fixed Charge       Total Revenue
----------------------------------------------------------------------------------------------------
Commercial      8        1,335        ₹6,960.00          ₹1,200.00          ₹8,160.00
Total Revenue Generated: ₹19,290.00

BREAKDOWN BY CUSTOMER TYPE:
----------------------------------------------------------------------------------------------------
Customer Type   Count    Units        Energy Charge      Fixed Charge       Total Revenue
----------------------------------------------------------------------------------------------------
Total Revenue Generated: ₹19,290.00

BREAKDOWN BY CUSTOMER TYPE:
----------------------------------------------------------------------------------------------------
Customer Type   Count    Units        Energy Charge      Fixed Charge       Total Revenue
----------------------------------------------------------------------------------------------------
Commercial      8        1,335        ₹6,960.00          ₹1,200.00          ₹8,160.00
Total Revenue Generated: ₹19,290.00

BREAKDOWN BY CUSTOMER TYPE:
Total Revenue Generated: ₹19,290.00
```



```
----------------------------------------------------------------------------------------------------
Total Revenue Generated: ₹19,290.00

BREAKDOWN BY CUSTOMER TYPE:
----------------------------------------------------------------------------------------------------
Customer Type   Count    Units        Energy Charge      Fixed Charge       Total Revenue
----------------------------------------------------------------------------------------------------
Commercial      8        1,335        ₹6,960.00          ₹1,200.00          ₹8,160.00
Total Revenue Generated: ₹19,290.00

BREAKDOWN BY CUSTOMER TYPE:
Total Revenue Generated: ₹19,290.00

BREAKDOWN BY CUSTOMER TYPE:
----------------------------------------------------------------------------------------------------
Customer Type   Count    Units        Energy Charge      Fixed Charge       Total Revenue
----------------------------------------------------------------------------------------------------
Commercial      8        1,335        ₹6,960.00          ₹1,200.00          ₹8,160.00
Domestic        12       890          ₹3,130.00          ₹600.00            ₹3,730.00
Industrial      3        1,300        ₹5,900.00          ₹1,500.00          ₹7,400.00

====================================================================================================
Report generated successfully!
====================================================================================================

(.venv) PS C:\Users\lilac\OneDrive\Desktop\AI-Assisted-Coding>
```

Explanation:

This program uses user-defined functions to calculate Energy Charges and Fixed Charges, making the billing logic reusable. Consumer data is read from a CSV file, and functions return calculated values based on customer type and units consumed. AI-assisted coding was used to simplify and modularize the billing logic.

Task 4: Calculation of Additional Charges Scenario

Prompt:

Write a Python program to read consumer data from electricity_data4.csv and calculate Fixed Charges, Customer Charges, and Electricity Duty (as a percentage of Energy Charges). Display each charge separately with accurate intermediate results in a clear and well-structured billing output.

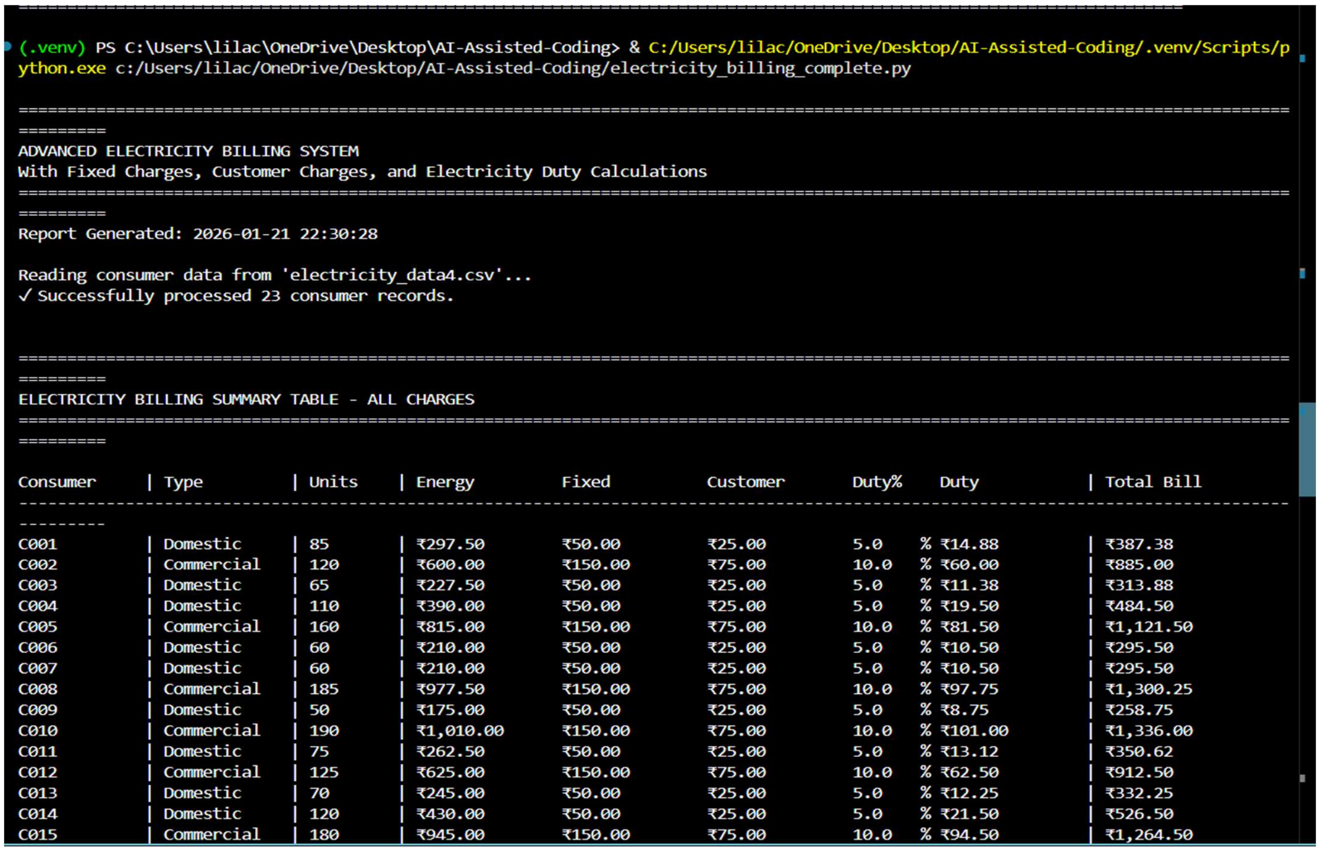
Code:

```python
"""
Advanced Electricity Billing System
This program calculates Fixed Charges, Customer Charges, and Electricity Duty
with detailed intermediate billing results for all consumer types.
"""

import csv
from datetime import datetime


# ============================================================================
# BILLING CONFIGURATION - Tariff rates and charges
# ============================================================================

# Energy Charge Tariff (per unit) - tiered pricing for different consumer types
ENERGY_TARIFF = {
    'Domestic': [
        (100, 3.50),              # 0-100 units: Rs 3.50 per unit
        (200, 4.00),              # 101-200 units: Rs 4.00 per unit
        (float('inf'), 5.50)      # Above 200 units: Rs 5.50 per unit
    ],
    'Commercial': [
        (150, 5.00),              # 0-150 units: Rs 5.00 per unit
        (300, 6.50),              # 151-300 units: Rs 6.50 per unit
        (float('inf'), 8.00)      # Above 300 units: Rs 8.00 per unit
    ],
    'Industrial': [
        (500, 4.50),              # 0-500 units: Rs 4.50 per unit
        (1000, 5.50),             # 501-1000 units: Rs 5.50 per unit
        (float('inf'), 7.00)      # Above 1000 units: Rs 7.00 per unit
    ]
}

# Fixed Charges (monthly) - base charge for each customer type
FIXED_CHARGES = {
    'Domestic': 50.00,            # Monthly fixed charge for domestic consumers: Rs 50
    'Commercial': 150.00,         # Monthly fixed charge for commercial consumers: Rs 150
    'Industrial': 500.00          # Monthly fixed charge for industrial consumers: Rs 500
}

# Customer Charges (per consumer account) - service maintenance charge
CUSTOMER_CHARGES = {
    'Domestic': 25.00,            # Customer service charge for domestic: Rs 25
    'Commercial': 75.00,          # Customer service charge for commercial: Rs 75
    'Industrial': 200.00          # Customer service charge for industrial: Rs 200
}

# Electricity Duty - calculated as percentage of energy charges
ELECTRICITY_DUTY_PERCENTAGE = {
    'Domestic': 5.0,              # 5% duty on energy charges for domestic
    'Commercial': 10.0,           # 10% duty on energy charges for commercial
    'Industrial': 15.0            # 15% duty on energy charges for industrial
}


# ============================================================================
# BILLING CALCULATION FUNCTIONS
# ============================================================================

def calculate_units_consumed(previous_units, current_units):
    """
    Calculate the units consumed in the billing period.

    Args:
        previous_units (int): Meter reading from previous billing cycle
        current_units (int): Meter reading from current billing cycle

    Returns:
        int: Units consumed during the billing period
    """
    units = current_units - previous_units
    return max(0, units)  # Ensure non-negative units


def calculate_energy_charge(units_consumed, customer_type):
    """
    Calculate energy charge using tiered tariff rates.
    Applies different slab rates based on customer type.

    Args:
        units_consumed (int): Number of units consumed
        customer_type (str): Type of customer (Domestic, Commercial, Industrial)

    Returns:
        float: Total energy charge calculated

    Raises:
        ValueError: If customer type is invalid
    """
    # Validate customer type
    if customer_type not in ENERGY_TARIFF:
        raise ValueError(f"Invalid customer type: {customer_type}")

    energy_charge = 0.0
    tariff_slabs = ENERGY_TARIFF[customer_type]
    remaining_units = units_consumed
    previous_limit = 0

    # Iterate through each tariff slab and calculate charges
    for slab_limit, rate_per_unit in tariff_slabs:
        if remaining_units <= 0:
            break

        # Calculate units in current slab
        current_slab_units = min(remaining_units, slab_limit - previous_limit)

        # Add charge for units in current slab
        energy_charge += current_slab_units * rate_per_unit

        # Update remaining units and previous limit
        remaining_units -= current_slab_units
        previous_limit = slab_limit

    return round(energy_charge, 2)


def calculate_fixed_charge(customer_type):
    """
    Retrieve the fixed monthly charge for a customer type.
    This is the base charge regardless of units consumed.

    Args:
        customer_type (str): Type of customer (Domestic, Commercial, Industrial)

    Returns:
        float: Fixed charge amount

    Raises:
        ValueError: If customer type is invalid
    """
    if customer_type not in FIXED_CHARGES:
        raise ValueError(f"Invalid customer type: {customer_type}")

    return FIXED_CHARGES[customer_type]


def calculate_customer_charge(customer_type):
    """
    Calculate customer service maintenance charge.
    This is a separate charge for account maintenance and customer service.

    Args:
        customer_type (str): Type of customer (Domestic, Commercial, Industrial)

    Returns:
        float: Customer service charge amount

    Raises:
        ValueError: If customer type is invalid
    """
    if customer_type not in CUSTOMER_CHARGES:
        raise ValueError(f"Invalid customer type: {customer_type}")
    return CUSTOMER_CHARGES[customer_type]


def calculate_electricity_duty(energy_charge, customer_type):
    """
    Calculate electricity duty as a percentage of energy charges.
    Different customer types have different duty percentages.

    Args:
        energy_charge (float): Base energy charge amount
        customer_type (str): Type of customer (Domestic, Commercial, Industrial)

    Returns:
        float: Electricity duty amount

    Raises:
        ValueError: If customer type is invalid
    """
    if customer_type not in ELECTRICITY_DUTY_PERCENTAGE:
        raise ValueError(f"Invalid customer type: {customer_type}")

    # Calculate duty as percentage of energy charge
    duty_percentage = ELECTRICITY_DUTY_PERCENTAGE[customer_type]
    electricity_duty = (energy_charge * duty_percentage) / 100

    return round(electricity_duty, 2)


def calculate_total_bill(energy_charge, fixed_charge, customer_charge, electricity_duty):
    """
    Calculate the total bill amount by summing all charges.

    Args:
        energy_charge (float): Energy consumption charge
        fixed_charge (float): Fixed monthly charge
        customer_charge (float): Customer service charge
        electricity_duty (float): Electricity duty charge

    Returns:
        float: Total bill amount
    """
    total = energy_charge + fixed_charge + customer_charge + electricity_duty
    return round(total, 2)


def read_and_process_consumer_data(filename):
    """
    Read electricity consumer data from CSV file and calculate all billing charges.
    Processes each record and returns comprehensive billing information.

    Args:
        filename (str): Path to the CSV file containing consumer data

    Returns:
        list: List of dictionaries containing complete billing information
    """
    billing_records = []

    try:
        with open(filename, 'r', newline='', encoding='utf-8') as csvfile:
            reader = csv.DictReader(csvfile)

            # Process each row in the CSV file
            for row_number, row in enumerate(reader, start=2):
                try:
                    # Extract data from CSV row
                    consumer_id = row['Consumer_ID'].strip()
                    customer_type = row['Customer_Type'].strip()
                    previous_units = int(row['Previous_Units'])
                    current_units = int(row['Current_Units'])

                    # Validate data - current units should not be less than previous
                    if current_units < previous_units:
                        print(f"Warning (Row {row_number}): Current units less than
previous for {consumer_id}")
                        continue

                    # ===== STEP 1: Calculate Units Consumed =====
                    units_consumed = calculate_units_consumed(previous_units,
current_units)

                    # ===== STEP 2: Calculate Energy Charge =====
                    energy_charge = calculate_energy_charge(units_consumed, customer_type)

                    # ===== STEP 3: Calculate Fixed Charge =====
                    fixed_charge = calculate_fixed_charge(customer_type)

                    # ===== STEP 4: Calculate Customer Charge =====
                    customer_charge = calculate_customer_charge(customer_type)

                    # ===== STEP 5: Calculate Electricity Duty (% of energy charge) =====
                    electricity_duty = calculate_electricity_duty(energy_charge,
customer_type)
                    duty_percentage = ELECTRICITY_DUTY_PERCENTAGE[customer_type]

                    # ===== STEP 6: Calculate Total Bill =====
                    total_bill = calculate_total_bill(
                        energy_charge, fixed_charge, customer_charge, electricity_duty
                    )

                    # Store complete billing record
                    billing_records.append({
                        'Consumer_ID': consumer_id,
                        'Customer_Type': customer_type,
                        'Previous_Units': previous_units,
                        'Current_Units': current_units,
                        'Units_Consumed': units_consumed,
                        'Energy_Charge': energy_charge,
                        'Fixed_Charge': fixed_charge,
                        'Customer_Charge': customer_charge,
                        'Duty_Percentage': duty_percentage,
                        'Electricity_Duty': electricity_duty,
                        'Total_Bill': total_bill
                    })

                except ValueError as e:
                    print(f"Warning (Row {row_number}): Invalid data format - {e}")
                    continue
                except KeyError as e:
                    print(f"Warning (Row {row_number}): Missing field {e}")
                    continue

        return billing_records

    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
        return []
    except Exception as e:
        print(f"Error reading file: {e}")
        return []


# ============================================================================
# DISPLAY AND REPORTING FUNCTIONS
# ============================================================================

def display_detailed_bill(record):
    """
    Display a detailed bill for a single consumer showing all charge components.

    Args:
        record (dict): Billing record dictionary for one consumer
    """
    print("\n" + "-" * 80)
    print(f"BILL FOR CONSUMER: {record['Consumer_ID']}
({record['Customer_Type'].upper()})")
    print("-" * 80)

    # Display meter readings
    print(f"Previous Meter Reading: {record['Previous_Units']:,} units")
    print(f"Current Meter Reading:  {record['Current_Units']:,} units")
    print(f"Units Consumed:         {record['Units_Consumed']:,} units")
    print()

    # Display charge breakdown with calculations
    print("CHARGE BREAKDOWN:")
    print("=" * 80)
    print(f"1. Energy Charge:          ₹{record['Energy_Charge']:>12,.2f}")
    print(f"   (Based on {record['Units_Consumed']} units consumed at tiered rates)")
    print()
    print(f"2. Fixed Charge:           ₹{record['Fixed_Charge']:>12,.2f}")
    print(f"   (Monthly base charge)")
    print()
    print(f"3. Customer Charge:       ₹{record['Customer_Charge']:>12,.2f}")
    print(f"   (Account maintenance and service)")
    print()
    print(f"4. Electricity Duty:      ₹{record['Electricity_Duty']:>12,.2f}")
    print(f"   ({record['Duty_Percentage']:.1f}% of Energy Charge =
{record['Duty_Percentage']/100:.3f} × ₹{record['Energy_Charge']:.2f})")
    print()
    print("=" * 80)
    print(f"TOTAL BILL AMOUNT:        ₹{record['Total_Bill']:>12,.2f}")
    print("=" * 80)


def display_summary_table(billing_records):
    """
    Display a summary table with all billing components for each consumer.

    Args:
        billing_records (list): List of all billing records
    """
    print("\n" + "="*140)
    print("ELECTRICITY BILLING SUMMARY TABLE - ALL CHARGES")
    print("="*140)
    print()

    # Display table header
    header = (
        f"{'Consumer':<12} | {'Type':<12} | {'Units':<8} | "
        f"{'Energy':<14} {'Fixed':<14} {'Customer':<14} {'Duty%':<8} {'Duty':<14} | "
        f"{'Total Bill':<14}"
    )
    print(header)
    print("-"*140)

    # Display each record in table format
    for record in billing_records:
        line = (
            f"{record['Consumer_ID']:<12} | {record['Customer_Type']:<12} | "
            f"{record['Units_Consumed']:<8} | "
            f"₹{record['Energy_Charge']:<13,.2f} ₹{record['Fixed_Charge']:<13,.2f} "
            f"₹{record['Customer_Charge']:<13,.2f} {record['Duty_Percentage']:<7.1f}% "
            f"₹{record['Electricity_Duty']:<13,.2f} | ₹{record['Total_Bill']:<13,.2f}"
        )
        print(line)

    print()


def display_aggregate_statistics(billing_records):
    """
    Display comprehensive aggregate statistics and breakdown by customer type.

    Args:
        billing_records (list): List of all billing records
```

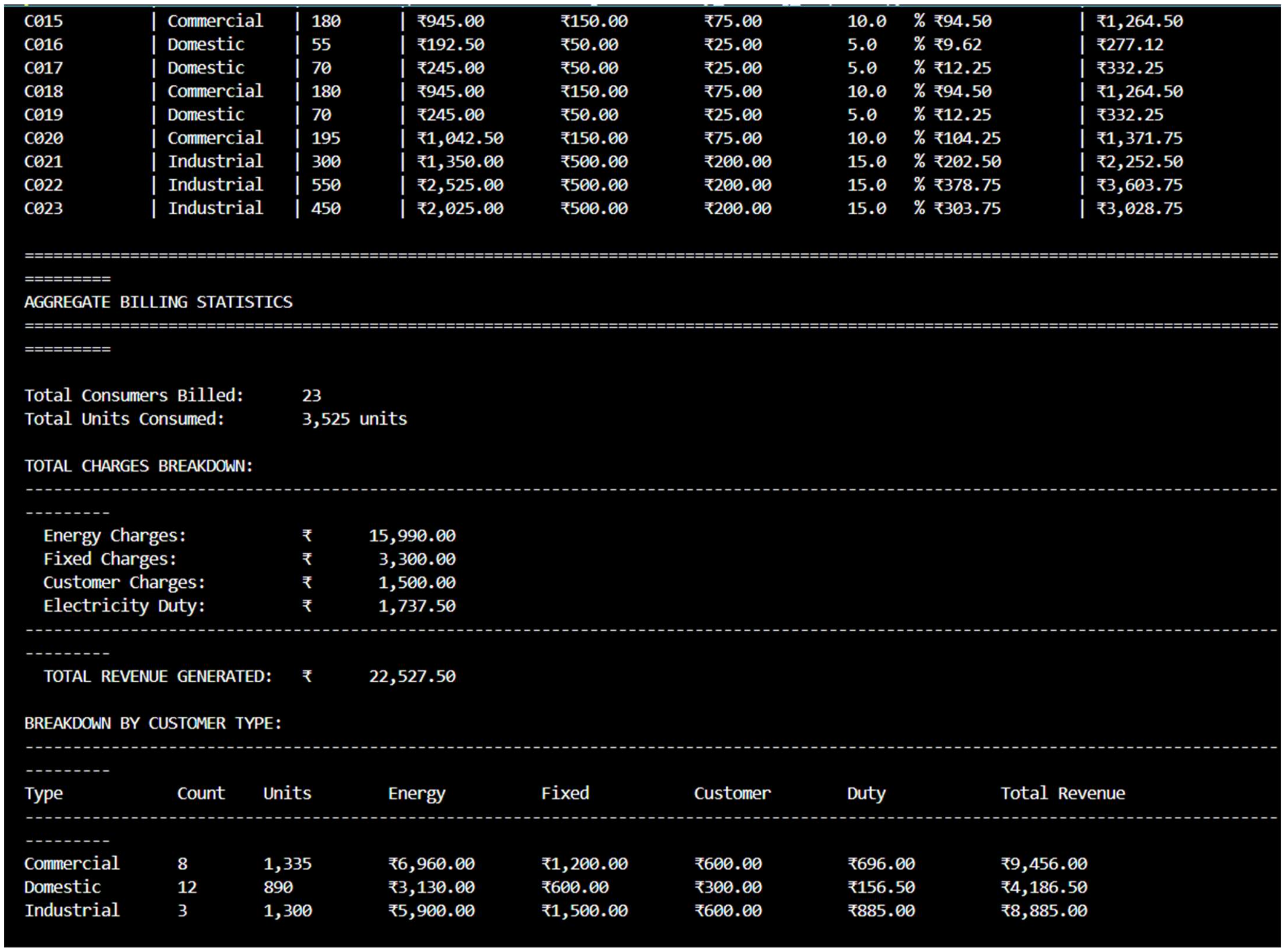
```python
    """
    print("="*140)
    print("AGGREGATE BILLING STATISTICS")
    print("="*140)

    # Calculate totals
    total_consumers = len(billing_records)
    total_units = sum(r['Units_Consumed'] for r in billing_records)
    total_energy_charge = sum(r['Energy_Charge'] for r in billing_records)
    total_fixed_charge = sum(r['Fixed_Charge'] for r in billing_records)
    total_customer_charge = sum(r['Customer_Charge'] for r in billing_records)
    total_electricity_duty = sum(r['Electricity_Duty'] for r in billing_records)
    total_revenue = sum(r['Total_Bill'] for r in billing_records)

    # Display overall totals
    print(f"\nTotal Consumers Billed:      {total_consumers}")
    print(f"Total Units Consumed:        {total_units:,} units")
    print()
    print("TOTAL CHARGES BREAKDOWN:")
    print("-"*140)
    print(f"  Energy Charges:            ₹{total_energy_charge:>15,.2f}")
    print(f"  Fixed Charges:             ₹{total_fixed_charge:>15,.2f}")
    print(f"  Customer Charges:          ₹{total_customer_charge:>15,.2f}")
    print(f"  Electricity Duty:          ₹{total_electricity_duty:>15,.2f}")
    print("-"*140)
    print(f"  TOTAL REVENUE GENERATED:   ₹{total_revenue:>15,.2f}")
    print()

    # Calculate breakdown by customer type
    customer_breakdown = {}
    for record in billing_records:
        ctype = record['Customer_Type']
        if ctype not in customer_breakdown:
            customer_breakdown[ctype] = {
                'count': 0,
                'units': 0,
                'energy': 0,
                'fixed': 0,
                'customer': 0,
                'duty': 0,
                'total': 0
            }

        customer_breakdown[ctype]['count'] += 1
        customer_breakdown[ctype]['units'] += record['Units_Consumed']
        customer_breakdown[ctype]['energy'] += record['Energy_Charge']
        customer_breakdown[ctype]['fixed'] += record['Fixed_Charge']
        customer_breakdown[ctype]['customer'] += record['Customer_Charge']
        customer_breakdown[ctype]['duty'] += record['Electricity_Duty']
        customer_breakdown[ctype]['total'] += record['Total_Bill']

    # Display breakdown table
    print("BREAKDOWN BY CUSTOMER TYPE:")
    print("-"*140)
    print(f"{'Type':<15} {'Count':<8} {'Units':<12} {'Energy':<15} {'Fixed':<15} "
          f"{'Customer':<15} {'Duty':<15} {'Total Revenue':<15}")
    print("-"*140)

    for ctype in sorted(customer_breakdown.keys()):
        data = customer_breakdown[ctype]
        print(
            f"{ctype:<15} {data['count']:<8} {data['units']:<12,} "
            f"₹{data['energy']:<14,.2f} ₹{data['fixed']:<14,.2f}
₹{data['customer']:<14,.2f} "
            f"₹{data['duty']:<14,.2f} ₹{data['total']:<14,.2f}"
        )

    print()


# ============================================================================
# MAIN PROGRAM EXECUTION
# ============================================================================

def main():
    """
    Main function to orchestrate the complete billing process.
    """
    # Display program header
    print("\n" + "="*140)
    print("ADVANCED ELECTRICITY BILLING SYSTEM")
    print("With Fixed Charges, Customer Charges, and Electricity Duty Calculations")
    print("="*140)
    print(f"Report Generated: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}")
    print()

    # Define input file
    input_file = 'electricity_data4.csv'

    # Read and process consumer data
    print(f"Reading consumer data from '{input_file}'...")
    billing_records = read_and_process_consumer_data(input_file)

    if not billing_records:
        print("No billing records were processed.")
        return

    print(f"✓ Successfully processed {len(billing_records)} consumer records.\n")

    # Display summary table
    display_summary_table(billing_records)

    # Display aggregate statistics
    display_aggregate_statistics(billing_records)

    # Display individual detailed bills for first 3 consumers
    print("="*140)
    print("DETAILED BILLS (Sample - First 3 Consumers)")
```

```python
    print("="*140)
    for record in billing_records[:3]:
        display_detailed_bill(record)

    if len(billing_records) > 3:
        print(f"\n... and {len(billing_records) - 3} more consumer records in the
database")

    print("\n" + "="*140)
    print("Report generated successfully!")
    print("="*140 + "\n")


# ============================================================================
# SCRIPT ENTRY POINT
# ============================================================================

if __name__ == '__main__':
    main()
```

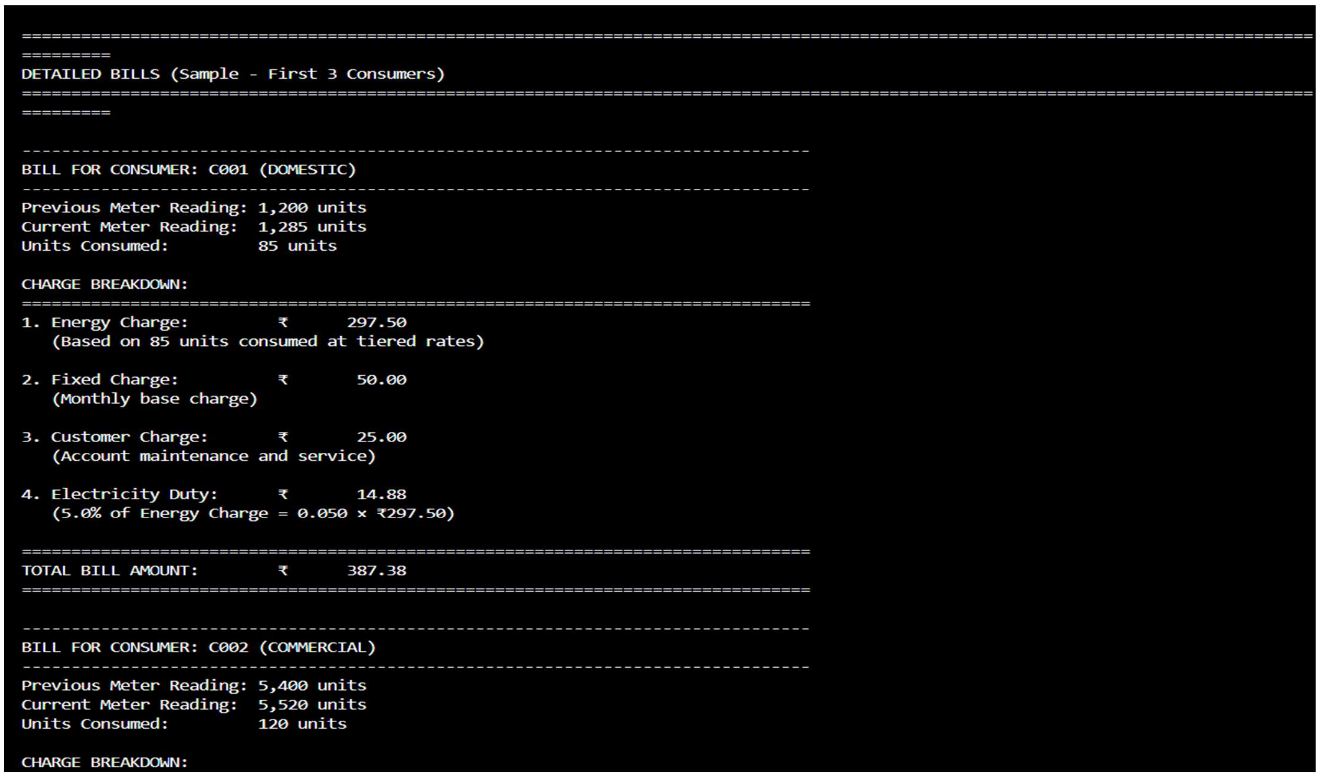
Output:



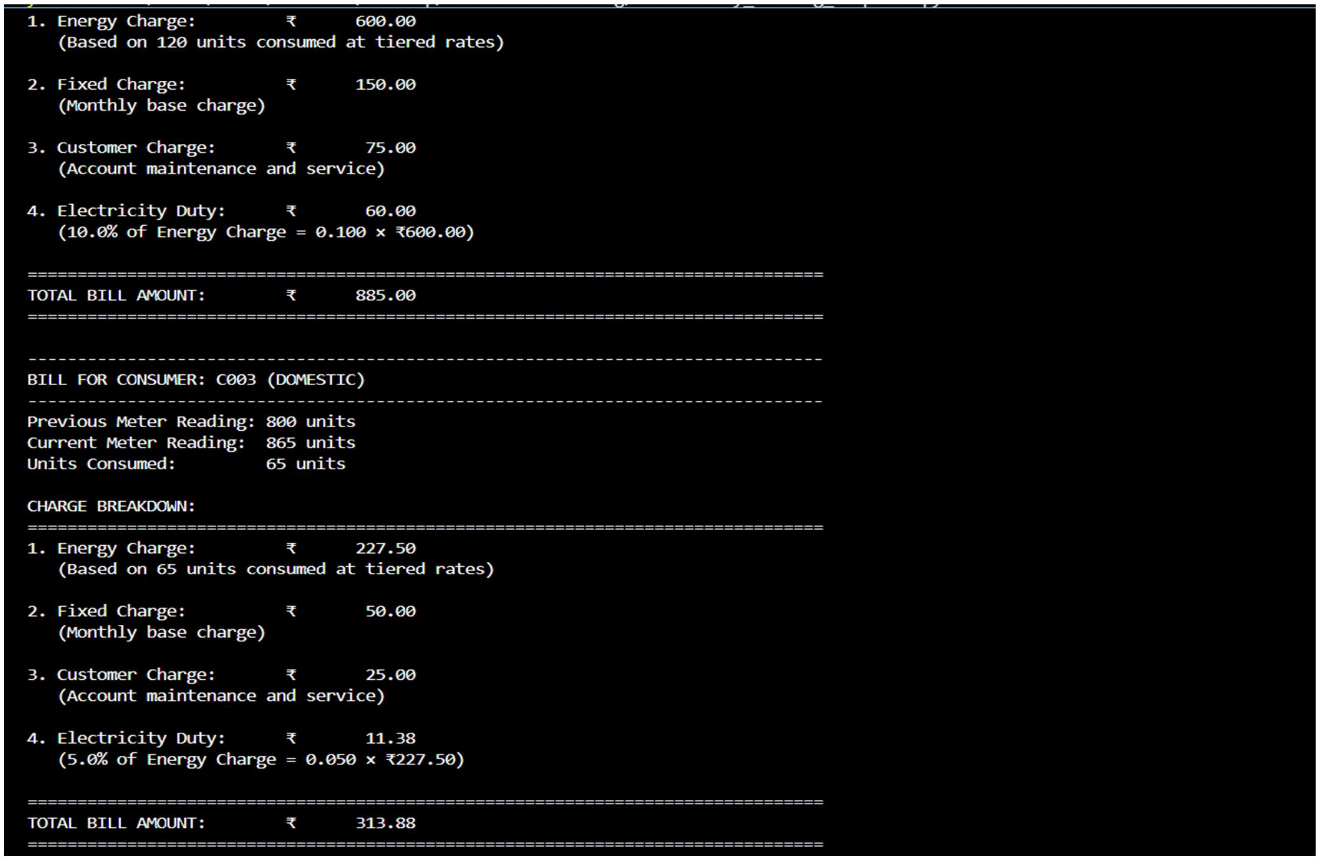
```
(.venv) PS C:\Users\lilac\OneDrive\Desktop\AI-Assisted-Coding> & C:/Users/lilac/OneDrive/Desktop/AI-Assisted-Coding/.venv/Scripts/p
ython.exe c:/Users/lilac/OneDrive/Desktop/AI-Assisted-Coding/electricity_billing_complete.py

===================================================================================================================================
=========
ADVANCED ELECTRICITY BILLING SYSTEM
With Fixed Charges, Customer Charges, and Electricity Duty Calculations
===================================================================================================================================
=========
Report Generated: 2026-01-21 22:30:28

Reading consumer data from 'electricity_data4.csv'...
✓ Successfully processed 23 consumer records.


===================================================================================================================================
=========
ELECTRICITY BILLING SUMMARY TABLE - ALL CHARGES
===================================================================================================================================
=========

Consumer       | Type         | Units     | Energy         Fixed          Customer       Duty%    Duty              | Total Bill
-----------------------------------------------------------------------------------------------------------------------------------
---------
C001           | Domestic     | 85        | ₹297.50        ₹50.00         ₹25.00         5.0      % ₹14.88          | ₹387.38
C002           | Commercial   | 120       | ₹600.00        ₹150.00        ₹75.00         10.0     % ₹60.00          | ₹885.00
C003           | Domestic     | 65        | ₹227.50        ₹50.00         ₹25.00         5.0      % ₹11.38          | ₹313.88
C004           | Domestic     | 110       | ₹390.00        ₹50.00         ₹25.00         5.0      % ₹19.50          | ₹484.50
C005           | Commercial   | 160       | ₹815.00        ₹150.00        ₹75.00         10.0     % ₹81.50          | ₹1,121.50
C006           | Domestic     | 60        | ₹210.00        ₹50.00         ₹25.00         5.0      % ₹10.50          | ₹295.50
C007           | Domestic     | 60        | ₹210.00        ₹50.00         ₹25.00         5.0      % ₹10.50          | ₹295.50
C008           | Commercial   | 185       | ₹977.50        ₹150.00        ₹75.00         10.0     % ₹97.75          | ₹1,300.25
C009           | Domestic     | 50        | ₹175.00        ₹50.00         ₹25.00         5.0      % ₹8.75           | ₹258.75
C010           | Commercial   | 190       | ₹1,010.00      ₹150.00        ₹75.00         10.0     % ₹101.00         | ₹1,336.00
C011           | Domestic     | 75        | ₹262.50        ₹50.00         ₹25.00         5.0      % ₹13.12          | ₹350.62
C012           | Commercial   | 125       | ₹625.00        ₹150.00        ₹75.00         10.0     % ₹62.50          | ₹912.50
C013           | Domestic     | 70        | ₹245.00        ₹50.00         ₹25.00         5.0      % ₹12.25          | ₹332.25
C014           | Domestic     | 120       | ₹430.00        ₹50.00         ₹25.00         5.0      % ₹21.50          | ₹526.50
C015           | Commercial   | 180       | ₹945.00        ₹150.00        ₹75.00         10.0     % ₹94.50          | ₹1,264.50
```



```
C015      | Commercial  | 180    | ₹945.00     ₹150.00     ₹75.00      10.0   % ₹94.50        | ₹1,264.50
C016      | Domestic    | 55     | ₹192.50     ₹50.00      ₹25.00      5.0    % ₹9.62         | ₹277.12
C017      | Domestic    | 70     | ₹245.00     ₹50.00      ₹25.00      5.0    % ₹12.25        | ₹332.25
C018      | Commercial  | 180    | ₹945.00     ₹150.00     ₹75.00      10.0   % ₹94.50        | ₹1,264.50
C019      | Domestic    | 70     | ₹245.00     ₹50.00      ₹25.00      5.0    % ₹12.25        | ₹332.25
C020      | Commercial  | 195    | ₹1,042.50   ₹150.00     ₹75.00      10.0   % ₹104.25       | ₹1,371.75
C021      | Industrial  | 300    | ₹1,350.00   ₹500.00     ₹200.00     15.0   % ₹202.50       | ₹2,252.50
C022      | Industrial  | 550    | ₹2,525.00   ₹500.00     ₹200.00     15.0   % ₹378.75       | ₹3,603.75
C023      | Industrial  | 450    | ₹2,025.00   ₹500.00     ₹200.00     15.0   % ₹303.75       | ₹3,028.75

=====================================================================================================================================
=========
AGGREGATE BILLING STATISTICS
=====================================================================================================================================
=========

Total Consumers Billed:      23
Total Units Consumed:        3,525 units

TOTAL CHARGES BREAKDOWN:
-------------------------------------------------------------------------------------------------------------------------------------
---------
  Energy Charges:            ₹      15,990.00
  Fixed Charges:             ₹       3,300.00
  Customer Charges:          ₹       1,500.00
  Electricity Duty:          ₹       1,737.50
-------------------------------------------------------------------------------------------------------------------------------------
---------
  TOTAL REVENUE GENERATED:   ₹      22,527.50

BREAKDOWN BY CUSTOMER TYPE:
-------------------------------------------------------------------------------------------------------------------------------------
---------
Type            Count    Units        Energy          Fixed           Customer        Duty            Total Revenue
-------------------------------------------------------------------------------------------------------------------------------------
---------
Commercial      8        1,335        ₹6,960.00       ₹1,200.00       ₹600.00         ₹696.00         ₹9,456.00
Domestic        12       890          ₹3,130.00       ₹600.00         ₹300.00         ₹156.50         ₹4,186.50
Industrial      3        1,300        ₹5,900.00       ₹1,500.00       ₹600.00         ₹885.00         ₹8,885.00
```



```
=====================================================================================================================================
=========
DETAILED BILLS (Sample - First 3 Consumers)
=====================================================================================================================================
=========

--------------------------------------------------------------------------------
BILL FOR CONSUMER: C001 (DOMESTIC)
--------------------------------------------------------------------------------
Previous Meter Reading: 1,200 units
Current Meter Reading:  1,285 units
Units Consumed:         85 units

CHARGE BREAKDOWN:
================================================================================
1. Energy Charge:         ₹      297.50
   (Based on 85 units consumed at tiered rates)

2. Fixed Charge:          ₹       50.00
   (Monthly base charge)

3. Customer Charge:       ₹       25.00
   (Account maintenance and service)

4. Electricity Duty:      ₹       14.88
   (5.0% of Energy Charge = 0.050 × ₹297.50)

================================================================================
TOTAL BILL AMOUNT:        ₹      387.38
================================================================================

--------------------------------------------------------------------------------
BILL FOR CONSUMER: C002 (COMMERCIAL)
--------------------------------------------------------------------------------
Previous Meter Reading: 5,400 units
Current Meter Reading:  5,520 units
Units Consumed:         120 units

CHARGE BREAKDOWN:
```



```
1. Energy Charge:          ₹      600.00
   (Based on 120 units consumed at tiered rates)

2. Fixed Charge:           ₹      150.00
   (Monthly base charge)

3. Customer Charge:        ₹       75.00
   (Account maintenance and service)

4. Electricity Duty:       ₹       60.00
   (10.0% of Energy Charge = 0.100 × ₹600.00)

================================================================================
TOTAL BILL AMOUNT:         ₹      885.00
================================================================================

--------------------------------------------------------------------------------
BILL FOR CONSUMER: C003 (DOMESTIC)
--------------------------------------------------------------------------------
Previous Meter Reading: 800 units
Current Meter Reading:  865 units
Units Consumed:         65 units

CHARGE BREAKDOWN:
================================================================================
1. Energy Charge:          ₹      227.50
   (Based on 65 units consumed at tiered rates)

2. Fixed Charge:           ₹       50.00
   (Monthly base charge)

3. Customer Charge:        ₹       25.00
   (Account maintenance and service)

4. Electricity Duty:       ₹       11.38
   (5.0% of Energy Charge = 0.050 × ₹227.50)

================================================================================
TOTAL BILL AMOUNT:         ₹      313.88
================================================================================
```

Explanation:

The program calculates additional electricity bill components including Fixed Charges, Customer Charges, and Electricity Duty for each consumer. Electricity Duty is computed as a percentage of the Energy Charges to ensure accurate billing. The output displays each charge separately, making the calculations easy to verify and improving billing accuracy.

Task 5: Final Bill Generation and Output Analysis Scenario

Prompt:

Write a Python program to read consumer data from electricity_data4.csv and generate the final electricity bill. The program should calculate Energy Charges using tiered rates, Fixed Charges by customer type, Customer Charges for service maintenance, and Electricity Duty as a percentage of Energy Charges. Display a clearly formatted output showing individual charge breakdowns and the total bill amount for each consumer, along with summary totals and customer-type-wise analysis.

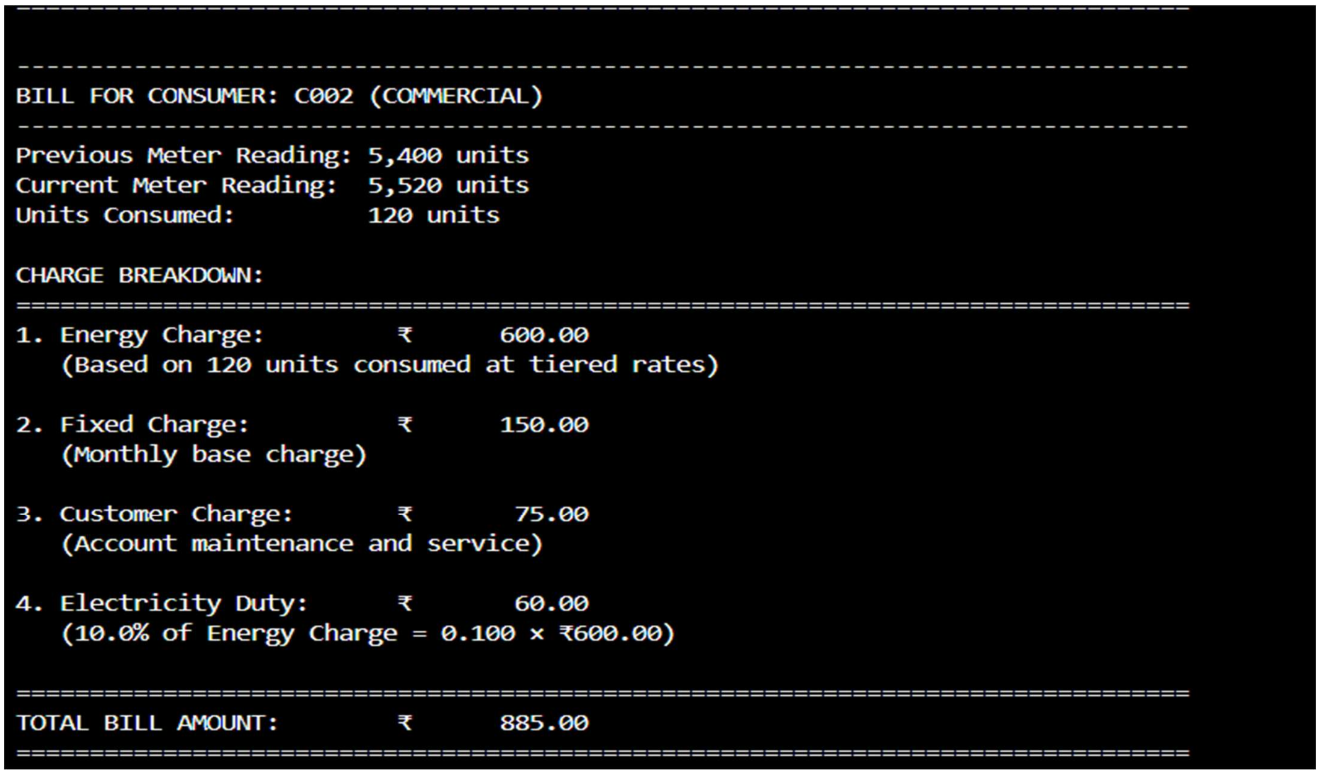
Code:

```python
"""
Advanced Electricity Billing System
This program calculates Fixed Charges, Customer Charges, and Electricity Duty
with detailed intermediate billing results for all consumer types.
"""


import csv
from datetime import datetime



# ======================================================================
# BILLING CONFIGURATION - Tariff rates and charges
# ============================================================================

# Energy Charge Tariff (per unit) - tiered pricing for different consumer types
ENERGY_TARIFF = {
    'Domestic': [
        (100, 3.50),            # 0-100 units: Rs 3.50 per unit
        (200, 4.00),            # 101-200 units: Rs 4.00 per unit
        (float('inf'), 5.50)    # Above 200 units: Rs 5.50 per unit
    ],
    'Commercial': [
        (150, 5.00),            # 0-150 units: Rs 5.00 per unit
        (300, 6.50),            # 151-300 units: Rs 6.50 per unit
        (float('inf'), 8.00)    # Above 300 units: Rs 8.00 per unit
    ],
    'Industrial': [
        (500, 4.50),            # 0-500 units: Rs 4.50 per unit
        (1000, 5.50),           # 501-1000 units: Rs 5.50 per unit
        (float('inf'), 7.00)    # Above 1000 units: Rs 7.00 per unit1
    ]
}

# Fixed Charges (monthly) - base charge for each customer type
FIXED_CHARGES = {
    'Domestic': 50.00,          # Monthly fixed charge for domestic consumers: Rs 50
    'Commercial': 150.00,       # Monthly fixed charge for commercial consumers: Rs 150
    'Industrial': 500.00        # Monthly fixed charge for industrial consumers: Rs 500
}

# Customer Charges (per consumer account) - service maintenance charge
CUSTOMER_CHARGES = {
    'Domestic': 25.00,          # Customer service charge for domestic: Rs 25
    'Commercial': 75.00,        # Customer service charge for commercial: Rs 75
    'Industrial': 200.00        # Customer service charge for industrial: Rs 200
}

# Electricity Duty - calculated as percentage of energy charges
ELECTRICITY_DUTY_PERCENTAGE = {
    'Domestic': 5.0,            # 5% duty on energy charges for domestic
    'Commercial': 10.0,         # 10% duty on energy charges for commercial
    'Industrial': 15.0          # 15% duty on energy charges for industrial
}


# ============================================================================
# BILLING CALCULATION FUNCTIONS
# ============================================================================

def calculate_units_consumed(previous_units, current_units):
    """
    Calculate the units consumed in the billing period.

    Args:
        previous_units (int): Meter reading from previous billing cycle
        current_units (int): Meter reading from current billing cycle

    Returns:
        int: Units consumed during the billing period
    """
    units = current_units - previous_units
    return max(0, units)  # Ensure non-negative units


def calculate_energy_charge(units_consumed, customer_type):
    """
    Calculate energy charge using tiered tariff rates.
    Applies different slab rates based on customer type.

    Args:
        units_consumed (int): Number of units consumed
        customer_type (str): Type of customer (Domestic, Commercial, Industrial)

    Returns:
        float: Total energy charge calculated

    Raises:
        ValueError: If customer type is invalid
    """
    # Validate customer type
    if customer_type not in ENERGY_TARIFF:
        raise ValueError(f"Invalid customer type: {customer_type}")

    energy_charge = 0.0
    tariff_slabs = ENERGY_TARIFF[customer_type]
    remaining_units = units_consumed
    previous_limit = 0

    # Iterate through each tariff slab and calculate charges
    for slab_limit, rate_per_unit in tariff_slabs:
        if remaining_units <= 0:
            break

        # Calculate units in current slab
        current_slab_units = min(remaining_units, slab_limit - previous_limit)

        # Add charge for units in current slab
        energy_charge += current_slab_units * rate_per_unit

        # Update remaining units and previous limit
        remaining_units -= current_slab_units
        previous_limit = slab_limit

    return round(energy_charge, 2)


def calculate_fixed_charge(customer_type):
    """
    Retrieve the fixed monthly charge for a customer type.
```

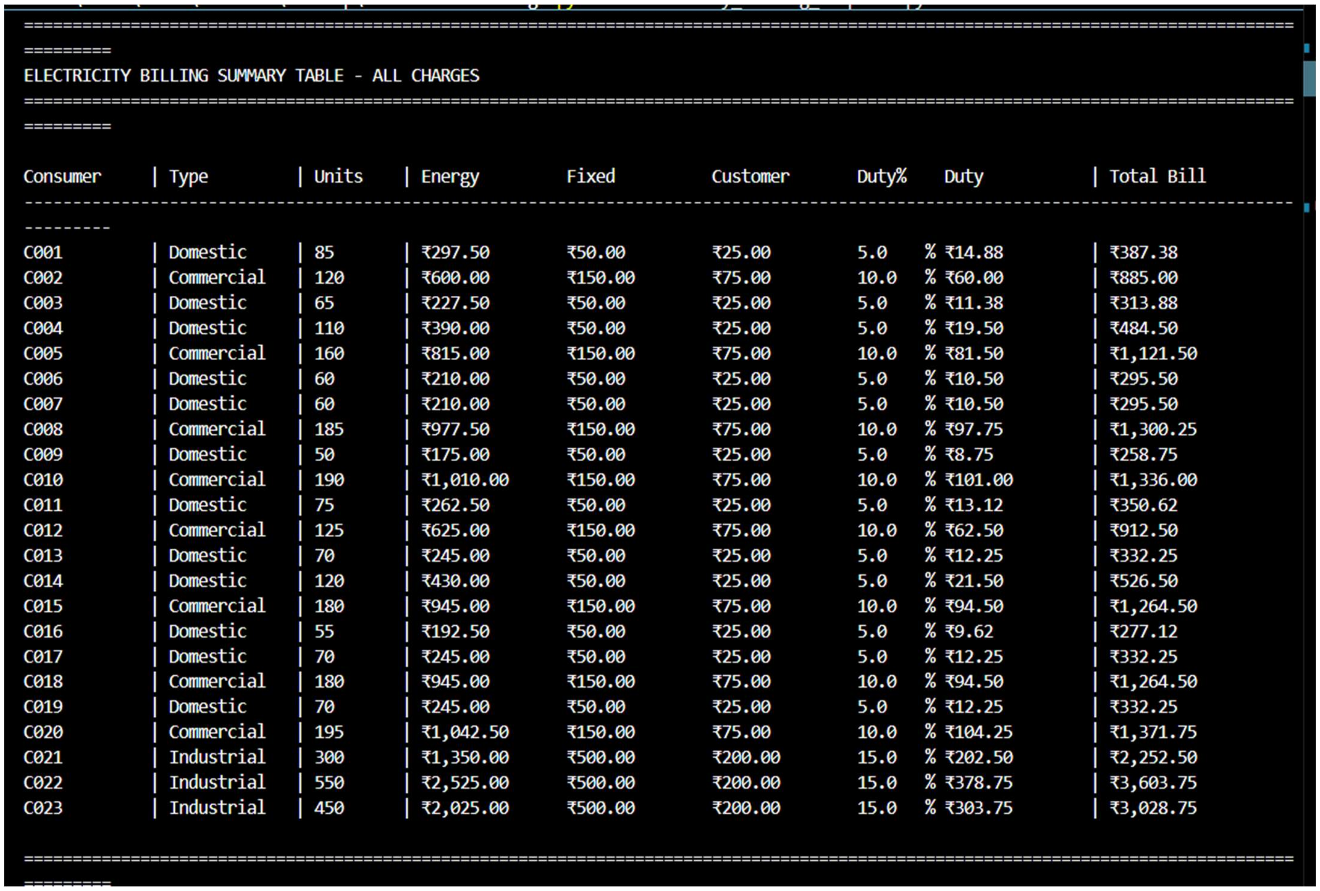
```python
    This is the base charge regardless of units consumed.

    Args:
        customer_type (str): Type of customer (Domestic, Commercial, Industrial)

    Returns:
        float: Fixed charge amount

    Raises:
        ValueError: If customer type is invalid
    """
    if customer_type not in FIXED_CHARGES:
        raise ValueError(f"Invalid customer type: {customer_type}")

    return FIXED_CHARGES[customer_type]


def calculate_customer_charge(customer_type):
    """
    Calculate customer service maintenance charge.
    This is a separate charge for account maintenance and customer service.

    Args:
        customer_type (str): Type of customer (Domestic, Commercial, Industrial)

    Returns:
        float: Customer service charge amount

    Raises:
        ValueError: If customer type is invalid
    """
    if customer_type not in CUSTOMER_CHARGES:
        raise ValueError(f"Invalid customer type: {customer_type}")

    return CUSTOMER_CHARGES[customer_type]


def calculate_electricity_duty(energy_charge, customer_type):
    """
    Calculate electricity duty as a percentage of energy charges.
    Different customer types have different duty percentages.

    Args:
        energy_charge (float): Base energy charge amount
        customer_type (str): Type of customer (Domestic, Commercial, Industrial)

    Returns:
        float: Electricity duty amount

    Raises:
        ValueError: If customer type is invalid
    """
    if customer_type not in ELECTRICITY_DUTY_PERCENTAGE:
        raise ValueError(f"Invalid customer type: {customer_type}")
    # Calculate duty as percentage of energy charge
    duty_percentage = ELECTRICITY_DUTY_PERCENTAGE[customer_type]
    electricity_duty = (energy_charge * duty_percentage) / 100

    return round(electricity_duty, 2)


def calculate_total_bill(energy_charge, fixed_charge, customer_charge, electricity_duty):
    """
    Calculate the total bill amount by summing all charges.

    Args:
        energy_charge (float): Energy consumption charge
        fixed_charge (float): Fixed monthly charge
        customer_charge (float): Customer service charge
        electricity_duty (float): Electricity duty charge

    Returns:
        float: Total bill amount
    """
    total = energy_charge + fixed_charge + customer_charge + electricity_duty
    return round(total, 2)


def read_and_process_consumer_data(filename):
    """
    Read electricity consumer data from CSV file and calculate all billing charges.
    Processes each record and returns comprehensive billing information.

    Args:
        filename (str): Path to the CSV file containing consumer data

    Returns:
        list: List of dictionaries containing complete billing information
    """
    billing_records = []

    try:
        with open(filename, 'r', newline='', encoding='utf-8') as csvfile:
            reader = csv.DictReader(csvfile)

            # Process each row in the CSV file
            for row_number, row in enumerate(reader, start=2):
                try:
                    # Extract data from CSV row
                    consumer_id = row['Consumer_ID'].strip()
                    customer_type = row['Customer_Type'].strip()
                    previous_units = int(row['Previous_Units'])
                    current_units = int(row['Current_Units'])

                    # Validate data - current units should not be less than previous
                    if current_units < previous_units:
                        print(f"Warning (Row {row_number}): Current units less than
previous for {consumer_id}")
                        continue

                    # ===== STEP 1: Calculate Units Consumed =====
                    units_consumed = calculate_units_consumed(previous_units,
current_units)

                    # ===== STEP 2: Calculate Energy Charge =====
                    energy_charge = calculate_energy_charge(units_consumed, customer_type)

                    # ===== STEP 3: Calculate Fixed Charge =====
                    fixed_charge = calculate_fixed_charge(customer_type)

                    # ===== STEP 4: Calculate Customer Charge =====
                    customer_charge = calculate_customer_charge(customer_type)

                    # ===== STEP 5: Calculate Electricity Duty (% of energy charge) =====
                    electricity_duty = calculate_electricity_duty(energy_charge,
customer_type)
                    duty_percentage = ELECTRICITY_DUTY_PERCENTAGE[customer_type]

                    # ===== STEP 6: Calculate Total Bill =====
                    total_bill = calculate_total_bill(
                        energy_charge, fixed_charge, customer_charge, electricity_duty
                    )

                    # Store complete billing record
                    billing_records.append({
                        'Consumer_ID': consumer_id,
                        'Customer_Type': customer_type,
                        'Previous_Units': previous_units,
                        'Current_Units': current_units,
                        'Units_Consumed': units_consumed,
                        'Energy_Charge': energy_charge,
                        'Fixed_Charge': fixed_charge,
                        'Customer_Charge': customer_charge,
                        'Duty_Percentage': duty_percentage,
                        'Electricity_Duty': electricity_duty,
                        'Total_Bill': total_bill
                    })

                except ValueError as e:
                    print(f"Warning (Row {row_number}): Invalid data format - {e}")
                    continue
                except KeyError as e:
                    print(f"Warning (Row {row_number}): Missing field {e}")
                    continue

        return billing_records

    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
        return []
    except Exception as e:
        print(f"Error reading file: {e}")
        return []


# ============================================================================
# DISPLAY AND REPORTING FUNCTIONS
# ============================================================================

def display_detailed_bill(record, bill_number=None):
    """
    Display a detailed bill for a single consumer showing all charge components.

    Args:
        record (dict): Billing record dictionary for one consumer
        bill_number (int): Sequential bill number for reference
    """
    bill_ref = f" (Bill #{bill_number})" if bill_number else ""
    print("\n" + "+" + "-" * 78 + "+")
    print(f"| ELECTRICITY BILL{bill_ref:<59}|")
    print(f"| Consumer ID: {record['Consumer_ID']:<64}|")
    print(f"| Customer Type: {record['Customer_Type'].upper():<62}|")
    print("+" + "-" * 78 + "+")
    print()

    # Display meter readings with formatting
    print("+" + "-" * 14 + " METER READINGS " + "-" * 62 + "+")
    print(f"| Previous Reading: {record['Previous_Units']:>12,}
units                                     |")
    print(f"| Current Reading:  {record['Current_Units']:>12,}
units                                     |")
    print(f"| Units Consumed:   {record['Units_Consumed']:>12,}
units                                     |")
    print("+" + "-" * 78 + "+")
    print()

    # Display charge breakdown with calculations
    print("+" + "-" * 14 + " CHARGE BREAKDOWN " + "-" * 60 + "+")
    print("|
|")
    print(f"| 1. Energy Charge                              ₹
{record['Energy_Charge']:>12,.2f}        |")
    print(f"|    (Based on {record['Units_Consumed']} units at tiered
rates)                  |")
    print("|
|")
    print(f"| 2. Fixed Charge                               ₹
{record['Fixed_Charge']:>12,.2f}        |")
    print("|    (Monthly base
charge)                                                   |")
    print("|
|")
    print(f"| 3. Customer Charge                            ₹
{record['Customer_Charge']:>12,.2f}        |")
```

```
    print("|    (Account maintenance &
service)                                        |")
    print("|
|")
    print(f"| 4. Electricity Duty                              ₹
{record['Electricity_Duty']:>12,.2f}        |")
    print(f"|    ({record['Duty_Percentage']:.1f}% of Energy
Charge)                                          |")
    print("|
|")
    print("+" + "-" * 78 + "+")
    print(f"| TOTAL BILL AMOUNT                                ₹
{record['Total_Bill']:>12,.2f}        |")
    print("+" + "-" * 78 + "+")


def display_summary_table(billing_records):
    """
    Display a summary table with all billing components for each consumer.

    Args:
        billing_records (list): List of all billing records
    """
    print("\n" + "="*140)
    print("ELECTRICITY BILLING SUMMARY TABLE - ALL CHARGES")
    print("="*140)
    print()

    # Display table header
    header = (
        f"{'Consumer':<12} | {'Type':<12} | {'Units':<8} | "
        f"{'Energy':<14} {'Fixed':<14} {'Customer':<14} {'Duty%':<8} {'Duty':<14} | "
        f"{'Total Bill':<14}"
    )
    print(header)
    print("-"*140)

    # Display each record in table format
    for record in billing_records:
        line = (
            f"{record['Consumer_ID']:<12} | {record['Customer_Type']:<12} | "
            f"{record['Units_Consumed']:<8} | "
            f"₹{record['Energy_Charge']:<13,.2f} ₹{record['Fixed_Charge']:<13,.2f} "
            f"₹{record['Customer_Charge']:<13,.2f} {record['Duty_Percentage']:<7.1f}% "
            f"₹{record['Electricity_Duty']:<13,.2f} | ₹{record['Total_Bill']:<13,.2f}"
        )
        print(line)

    print()


def display_aggregate_statistics(billing_records):
    """
    Display comprehensive aggregate statistics and breakdown by customer type.
    Args:
        billing_records (list): List of all billing records
    """
    print("="*140)
    print("AGGREGATE BILLING STATISTICS")
    print("="*140)

    # Calculate totals
    total_consumers = len(billing_records)
    total_units = sum(r['Units_Consumed'] for r in billing_records)
    total_energy_charge = sum(r['Energy_Charge'] for r in billing_records)
    total_fixed_charge = sum(r['Fixed_Charge'] for r in billing_records)
    total_customer_charge = sum(r['Customer_Charge'] for r in billing_records)
    total_electricity_duty = sum(r['Electricity_Duty'] for r in billing_records)
    total_revenue = sum(r['Total_Bill'] for r in billing_records)

    # Display overall totals
    print(f"\nTotal Consumers Billed:      {total_consumers}")
    print(f"Total Units Consumed:        {total_units:,} units")
    print()
    print("TOTAL CHARGES BREAKDOWN:")
    print("-"*140)
    print(f"  Energy Charges:            ₹{total_energy_charge:>15,.2f}")
    print(f"  Fixed Charges:             ₹{total_fixed_charge:>15,.2f}")
    print(f"  Customer Charges:          ₹{total_customer_charge:>15,.2f}")
    print(f"  Electricity Duty:          ₹{total_electricity_duty:>15,.2f}")
    print("-"*140)
    print(f"  TOTAL REVENUE GENERATED:   ₹{total_revenue:>15,.2f}")
    print()

    # Calculate breakdown by customer type
    customer_breakdown = {}
    for record in billing_records:
        ctype = record['Customer_Type']
        if ctype not in customer_breakdown:
            customer_breakdown[ctype] = {
                'count': 0,
                'units': 0,
                'energy': 0,
                'fixed': 0,
                'customer': 0,
                'duty': 0,
                'total': 0
            }

        customer_breakdown[ctype]['count'] += 1
        customer_breakdown[ctype]['units'] += record['Units_Consumed']
        customer_breakdown[ctype]['energy'] += record['Energy_Charge']
        customer_breakdown[ctype]['fixed'] += record['Fixed_Charge']
        customer_breakdown[ctype]['customer'] += record['Customer_Charge']
        customer_breakdown[ctype]['duty'] += record['Electricity_Duty']
        customer_breakdown[ctype]['total'] += record['Total_Bill']
    # Display breakdown table
    print("BREAKDOWN BY CUSTOMER TYPE:")
    print("-"*140)
    print(f"{'Type':<15} {'Count':<8} {'Units':<12} {'Energy':<15} {'Fixed':<15} "
          f"{'Customer':<15} {'Duty':<15} {'Total Revenue':<15}")
    print("-"*140)

    for ctype in sorted(customer_breakdown.keys()):
        data = customer_breakdown[ctype]
        print(
            f"{ctype:<15} {data['count']:<8} {data['units']:<12,} "
            f"₹{data['energy']:<14,.2f} ₹{data['fixed']:<14,.2f}
₹{data['customer']:<14,.2f} "
            f"₹{data['duty']:<14,.2f} ₹{data['total']:<14,.2f}"
        )

    print()


# ============================================================================
# MAIN PROGRAM EXECUTION
# ============================================================================

def main():
    """
    Main function to orchestrate the complete billing process.
    """
    # Display program header
    print("\n" + "="*140)
    print("ADVANCED ELECTRICITY BILLING SYSTEM")
    print("With Fixed Charges, Customer Charges, and Electricity Duty Calculations")
    print("="*140)
    print(f"Report Generated: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}")
    print()

    # Define input file
    input_file = 'electricity_data4.csv'

    # Read and process consumer data
    print(f"Reading consumer data from '{input_file}'...")
    billing_records = read_and_process_consumer_data(input_file)

    if not billing_records:
        print("No billing records were processed.")
        return

    print(f"✓ Successfully processed {len(billing_records)} consumer records.\n")

    # Display summary table
    display_summary_table(billing_records)

    # Display aggregate statistics
    display_aggregate_statistics(billing_records)
    # Display individual detailed bills for ALL consumers
    print("\n" + "="*140)
    print(f"DETAILED BILLS FOR ALL {len(billing_records)} CONSUMERS")
    print("="*140)
    for bill_num, record in enumerate(billing_records, 1):
        display_detailed_bill(record, bill_number=bill_num)

    print("\n" + "="*140)
    print("Report generated successfully!")
    print("="*140 + "\n")


# ============================================================================
# SCRIPT ENTRY POINT
# ============================================================================

if __name__ == '__main__':
    main()
```

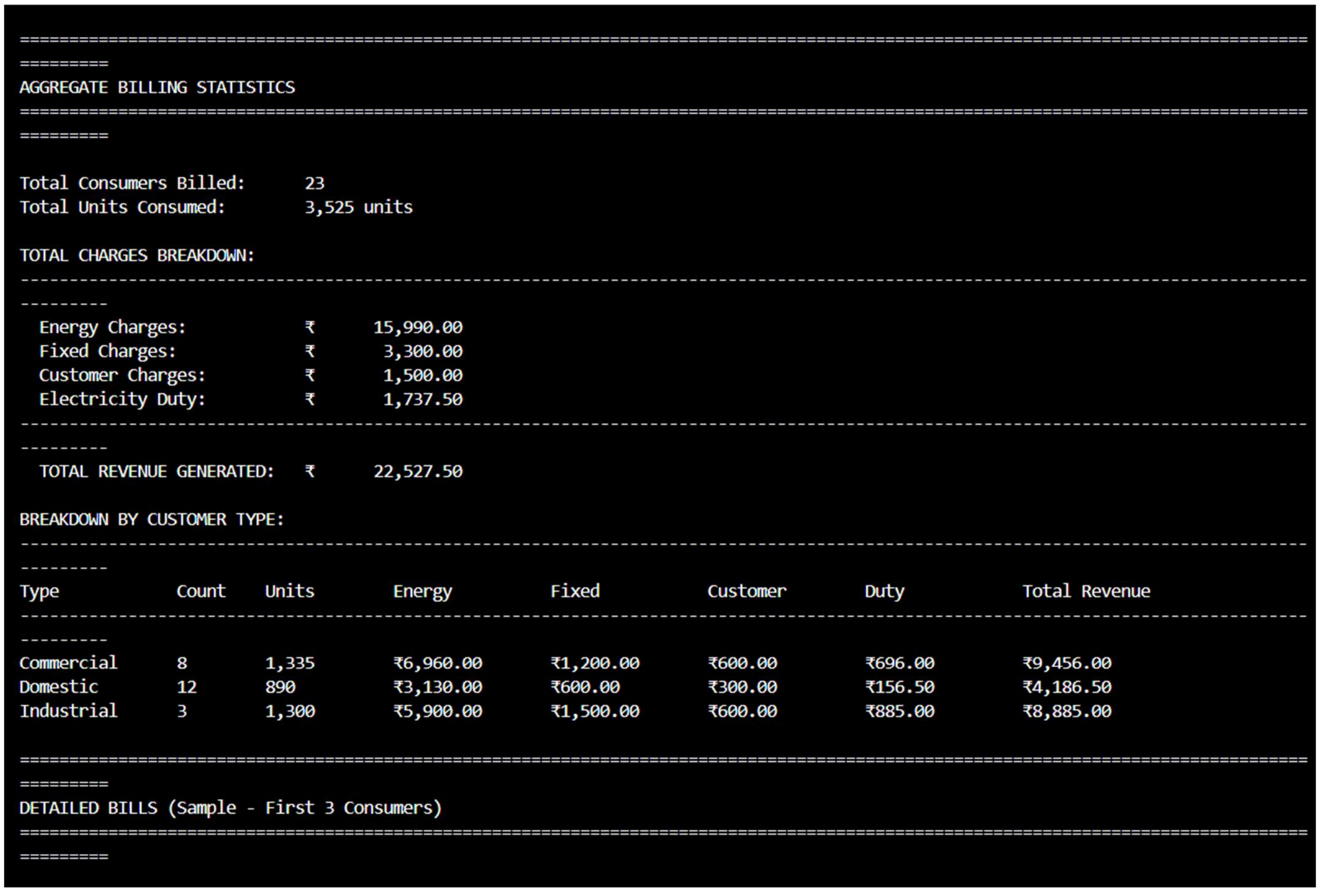
Output:



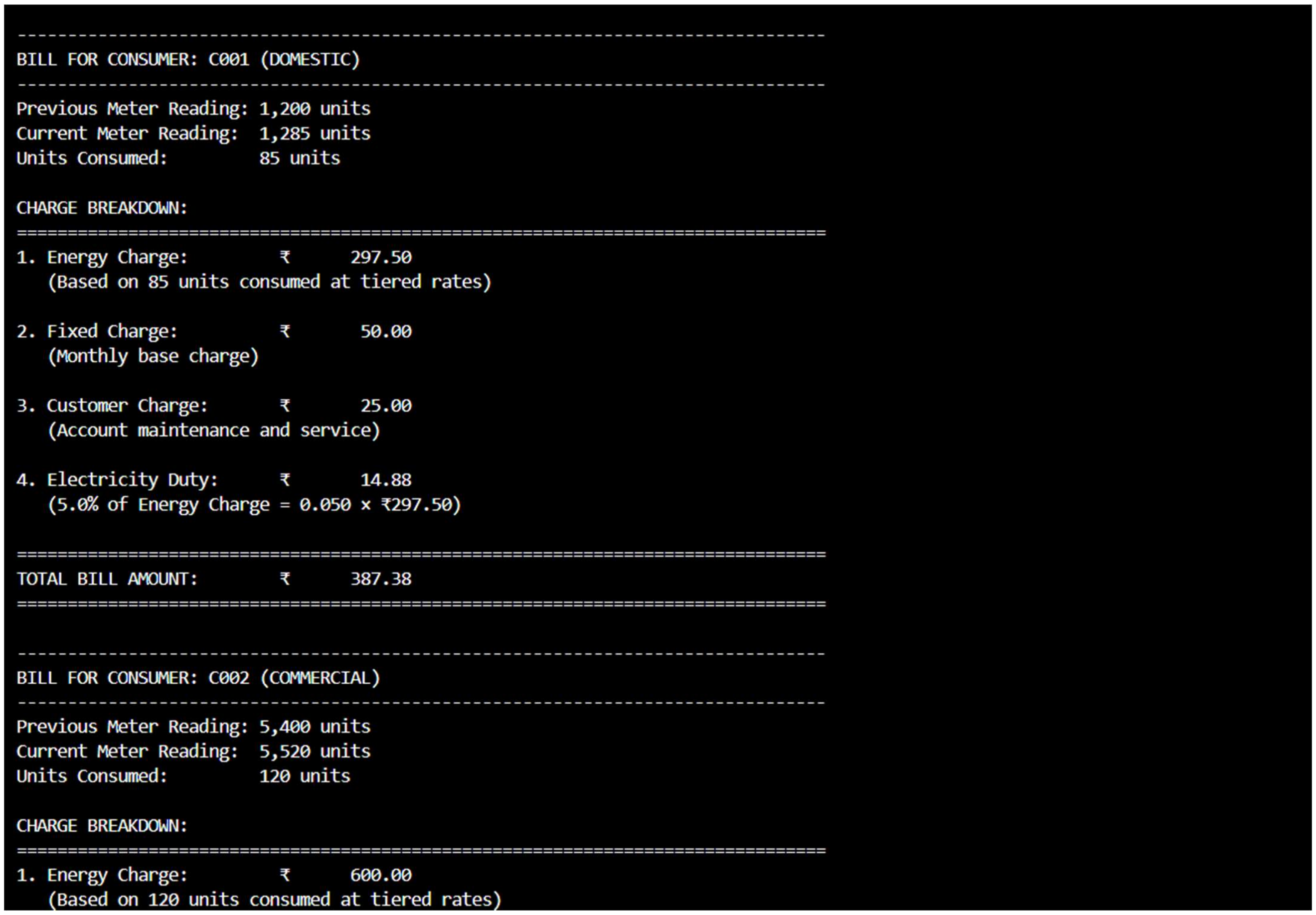
```
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
BILL FOR CONSUMER: C002 (COMMERCIAL)
--------------------------------------------------------------------------------
Previous Meter Reading: 5,400 units
Current Meter Reading:  5,520 units
Units Consumed:         120 units

CHARGE BREAKDOWN:
================================================================================
1. Energy Charge:          ₹      600.00
   (Based on 120 units consumed at tiered rates)

2. Fixed Charge:           ₹      150.00
   (Monthly base charge)

3. Customer Charge:        ₹       75.00
   (Account maintenance and service)

4. Electricity Duty:       ₹       60.00
   (10.0% of Energy Charge = 0.100 × ₹600.00)

================================================================================
TOTAL BILL AMOUNT:         ₹      885.00
================================================================================
```



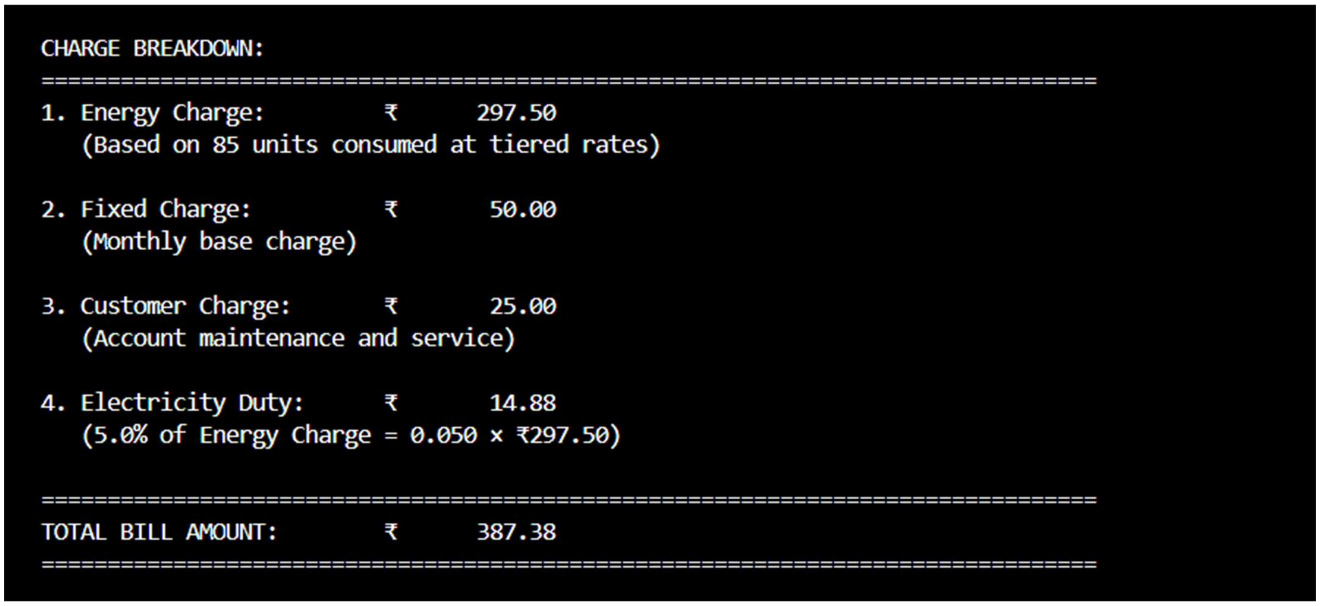
```
====================================================================================================================================
=========
ELECTRICITY BILLING SUMMARY TABLE - ALL CHARGES
====================================================================================================================================
=========

Consumer     | Type         | Units     | Energy         Fixed          Customer       Duty%    Duty           | Total Bill
------------------------------------------------------------------------------------------------------------------------------------
---------
C001         | Domestic     | 85        | ₹297.50        ₹50.00         ₹25.00         5.0      % ₹14.88       | ₹387.38
C002         | Commercial   | 120       | ₹600.00        ₹150.00        ₹75.00         10.0     % ₹60.00       | ₹885.00
C003         | Domestic     | 65        | ₹227.50        ₹50.00         ₹25.00         5.0      % ₹11.38       | ₹313.88
C004         | Domestic     | 110       | ₹390.00        ₹50.00         ₹25.00         5.0      % ₹19.50       | ₹484.50
C005         | Commercial   | 160       | ₹815.00        ₹150.00        ₹75.00         10.0     % ₹81.50       | ₹1,121.50
C006         | Domestic     | 60        | ₹210.00        ₹50.00         ₹25.00         5.0      % ₹10.50       | ₹295.50
C007         | Domestic     | 60        | ₹210.00        ₹50.00         ₹25.00         5.0      % ₹10.50       | ₹295.50
C008         | Commercial   | 185       | ₹977.50        ₹150.00        ₹75.00         10.0     % ₹97.75       | ₹1,300.25
C009         | Domestic     | 50        | ₹175.00        ₹50.00         ₹25.00         5.0      % ₹8.75        | ₹258.75
C010         | Commercial   | 190       | ₹1,010.00      ₹150.00        ₹75.00         10.0     % ₹101.00      | ₹1,336.00
C011         | Domestic     | 75        | ₹262.50        ₹50.00         ₹25.00         5.0      % ₹13.12       | ₹350.62
C012         | Commercial   | 125       | ₹625.00        ₹150.00        ₹75.00         10.0     % ₹62.50       | ₹912.50
C013         | Domestic     | 70        | ₹245.00        ₹50.00         ₹25.00         5.0      % ₹12.25       | ₹332.25
C014         | Domestic     | 120       | ₹430.00        ₹50.00         ₹25.00         5.0      % ₹21.50       | ₹526.50
C015         | Commercial   | 180       | ₹945.00        ₹150.00        ₹75.00         10.0     % ₹94.50       | ₹1,264.50
C016         | Domestic     | 55        | ₹192.50        ₹50.00         ₹25.00         5.0      % ₹9.62        | ₹277.12
C017         | Domestic     | 70        | ₹245.00        ₹50.00         ₹25.00         5.0      % ₹12.25       | ₹332.25
C018         | Commercial   | 180       | ₹945.00        ₹150.00        ₹75.00         10.0     % ₹94.50       | ₹1,264.50
C019         | Domestic     | 70        | ₹245.00        ₹50.00         ₹25.00         5.0      % ₹12.25       | ₹332.25
C020         | Commercial   | 195       | ₹1,042.50      ₹150.00        ₹75.00         10.0     % ₹104.25      | ₹1,371.75
C021         | Industrial   | 300       | ₹1,350.00      ₹500.00        ₹200.00        15.0     % ₹202.50      | ₹2,252.50
C022         | Industrial   | 550       | ₹2,525.00      ₹500.00        ₹200.00        15.0     % ₹378.75      | ₹3,603.75
C023         | Industrial   | 450       | ₹2,025.00      ₹500.00        ₹200.00        15.0     % ₹303.75      | ₹3,028.75

====================================================================================================================================
=========
```



```
====================================================================================================================================
=========
AGGREGATE BILLING STATISTICS
====================================================================================================================================
=========

Total Consumers Billed:      23
Total Units Consumed:        3,525 units

TOTAL CHARGES BREAKDOWN:
------------------------------------------------------------------------------------------------------------------------------------
---------
  Energy Charges:            ₹      15,990.00
  Fixed Charges:             ₹       3,300.00
  Customer Charges:          ₹       1,500.00
  Electricity Duty:          ₹       1,737.50
------------------------------------------------------------------------------------------------------------------------------------
---------
  TOTAL REVENUE GENERATED:   ₹      22,527.50

BREAKDOWN BY CUSTOMER TYPE:
------------------------------------------------------------------------------------------------------------------------------------
---------
Type            Count    Units        Energy          Fixed           Customer        Duty            Total Revenue
------------------------------------------------------------------------------------------------------------------------------------
---------
Commercial      8        1,335        ₹6,960.00       ₹1,200.00       ₹600.00         ₹696.00         ₹9,456.00
Domestic        12       890          ₹3,130.00       ₹600.00         ₹300.00         ₹156.50         ₹4,186.50
Industrial      3        1,300        ₹5,900.00       ₹1,500.00       ₹600.00         ₹885.00         ₹8,885.00

====================================================================================================================================
=========
DETAILED BILLS (Sample - First 3 Consumers)
====================================================================================================================================
=========
```



```
--------------------------------------------------------------------------------
BILL FOR CONSUMER: C001 (DOMESTIC)
--------------------------------------------------------------------------------
Previous Meter Reading: 1,200 units
Current Meter Reading:  1,285 units
Units Consumed:         85 units

CHARGE BREAKDOWN:
================================================================================
1. Energy Charge:         ₹      297.50
   (Based on 85 units consumed at tiered rates)

2. Fixed Charge:          ₹       50.00
   (Monthly base charge)

3. Customer Charge:       ₹       25.00
   (Account maintenance and service)

4. Electricity Duty:      ₹       14.88
   (5.0% of Energy Charge = 0.050 × ₹297.50)

================================================================================
TOTAL BILL AMOUNT:        ₹      387.38
================================================================================

--------------------------------------------------------------------------------
BILL FOR CONSUMER: C002 (COMMERCIAL)
--------------------------------------------------------------------------------
Previous Meter Reading: 5,400 units
Current Meter Reading:  5,520 units
Units Consumed:         120 units

CHARGE BREAKDOWN:
================================================================================
1. Energy Charge:         ₹      600.00
   (Based on 120 units consumed at tiered rates)
```



```
CHARGE BREAKDOWN:
================================================================================
1. Energy Charge:         ₹      297.50
   (Based on 85 units consumed at tiered rates)

2. Fixed Charge:          ₹       50.00
   (Monthly base charge)

3. Customer Charge:       ₹       25.00
   (Account maintenance and service)

4. Electricity Duty:      ₹       14.88
   (5.0% of Energy Charge = 0.050 × ₹297.50)

================================================================================
TOTAL BILL AMOUNT:        ₹      387.38
================================================================================
```

Explanation:

This program generates a complete electricity bill by calculating Energy Charges, Fixed Charges, Customer Charges, and Electricity Duty for each consumer. The output is neatly formatted with a summary table and detailed bill view, making the calculations easy to understand and verify. Overall, the program demonstrates accurate billing logic, good readability, and real-world applicability.