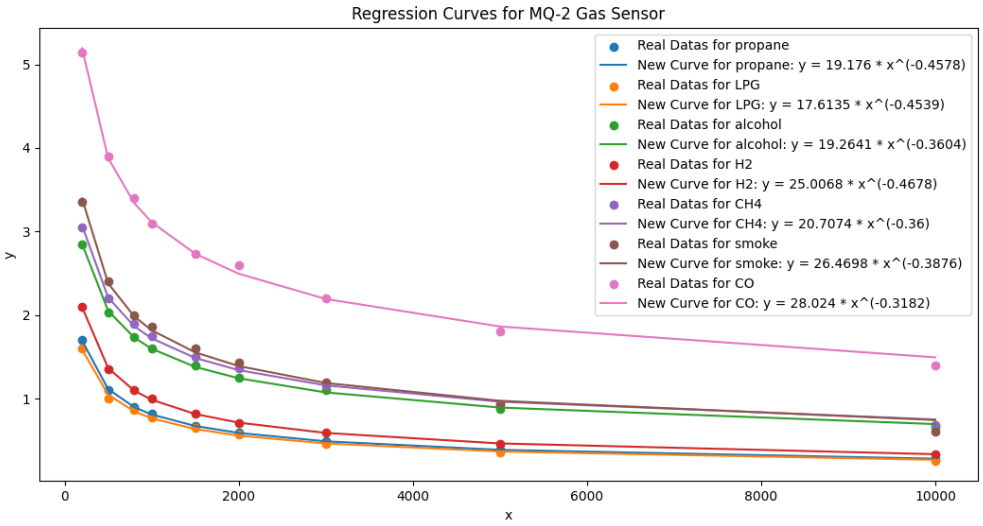

For Regression Formulla:

1) Y = ax^b
2) ppm = a*ratio^b

For Logarithm Formulla:

1) `ppm = 10^((log10(ratio)-b)/m)`
2) `y = mx + n`
3) log(AverageY) = m*log(x/2) + b
4) -b = m*log((x+x0)/2) – log(AverageY)
5) b = log10(AverageY) - m*log((x+x0)/2)
6) m = slope of the line
7) b = intersection point
8) m = log10(y/y0) / log10(x/x0)
9) b = log10(AverageY) - m * log10(x/2)

`ppm = 10^((log10(ratio)-b)/m)` | ppm = a*ratio^b

Gas	m	b
propane	-0.4749	1.3408
LPG	-0.4525	1.2501
alcohol	-0.3561	1.2771
H2	-0.4941	1.4858

Gas	a	b
propane	19.176	-0.4578
LPG	17.6135	-0.4539
alcohol	19.2641	-0.3606
H2	25.0068	-0.4678

CH4	-0.3532	1.2979	CH4	20.7074	-0.36
smoke	-0.3743	1.3929	smoke	26.4698	-0.3876
CO	-0.2538	1.2423	CO	28.024	-0.3182

Regression Calculator Code:

```
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

print("ppm = a*ratio^b")

sensor_name = input("Enter the type of your gas sensor like 'MQ-2': ")

gases = {}
plt.figure(figsize=(8, 6))

while True:
    gas_name = input("Enter the gas name (or press 'Enter' to finish): ")
    if gas_name == '':
        break

    values = []
    values_input = input(f"Enter (x, y) values for {gas_name} as [(x1, y1), (x2, y
2), ...]: ")
    try:
        values = eval(values_input)
        x_values = [value[0] for value in values]
        y_values = [value[1] for value in values]

        x = np.array(x_values)
        y = np.array(y_values)

        def func(x, a, b):
            return a * np.power(x, b)
```

```
        popt, pcov = curve_fit(func, x, y)

        a = round(popt[0], 4)
        b = round(popt[1], 4)

        print(f"create_a for {gas_name}: {a}, create_b for {gas_name}: {b}")

        plt.scatter(x, y, label=f'Real Datas for {gas_name}')
        plt.plot(x, func(x, a, b), label=f'New Curve for {gas_name}: y = {a} * x^(
{b})')

        gases[gas_name] = {'a': a, 'b': b}

    except (SyntaxError, NameError, ValueError) as e:
        print("Invalid input format. Please enter the values as [(x1, y1), (x2, y2
), ...]")

plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.title(f'Regression Curves for {sensor_name} Gas Sensor')
plt.show()
```

Logarithm Calculator Code:

```
from math import log10

print("ppm = pow(10, ((log10(ratio)-b)/m));")

def valueM(y, y0, x, x0):
    return round(log10(y/y0) / log10(x/x0), 4)

def valueB(y, AverageY, x, x0):
    return round(log10(AverageY) - valueM(y, y0, x, x0) * log10((x+x0)/2), 4)

def print_gas_table(gas_data):
    print("Gas    | m       | b")
    for gas, (m, b) in gas_data.items():
```

```
        print(f"{gas.ljust(7)}| {str(m).ljust(8)}| {str(b).ljust(7)}")


MQ_Model = input("Please define your MQ model like MQ-303A: ")


con = input("Does your sensor detect the same concentration range for all gases? (
yes/no): ")


if con.lower() == 'yes':
    x = float(input(f"Define max ppm concentrate point of the graph for {MQ_Model}
(x value): "))
    x0 = float(input(f"Define min ppm concentrate for {MQ_Model} (x0 value): "))


gas_data = {}


while True:
    Gas = input("Name of the gas like LPG (type 'stop' to exit): ")


    if Gas == 'stop':
        break


    if con.lower() == 'no':
        x = float(input(f"Define max ppm concentrate point of the graph for {Gas} (
x value): "))
        x0 = float(input(f"Define min ppm concentrate for {Gas} (x0 value): "))


    y0 = float(input(f"Define first reference point of the graph for {Gas} (y0 val
ue): "))
    y = float(input(f"Define final reference point of the graph for {Gas}  (y valu
e): "))
    AverageY = float(input(f"Define your y value at medium ppm concentration for {
Gas} (AverageY value): "))


    m = valueM(y, y0, x, x0)
    b = valueB(y, AverageY, x, x0)


    gas_data[Gas] = (m, b)


    print("Continue with another gas (yes/no)?")
```

```
    user_input = input()
    if user_input.lower() != 'yes':
        break

print("Your MQ Model is " + str(MQ_Model))
print_gas_table(gas_data)
```