Programming Interview Problem: Data Analysis and Optimization

Overview: In this interview problem, you'll work with a dataset and apply a custom function to analyze the data and optimize parameters. This task tests your ability to manipulate data, implement optimization algorithms, and write clean, efficient code.

Problem Description: You are provided with a function `func02` that takes in two numeric variables (`a`, `b`) and one categorical variable (`c`), and returns two numeric values (`g`, `h`). Your task is to analyze a large dataset and answer five questions of increasing complexity by applying this function.

You may use standard libraries for data manipulation and numerical methods. Immediately after the line that reads the file, please add a comment that gives your full name and the name of the company that you are applying to.

We ask you that you use C++ programming language to write a code that finds the answers to the questions below. Feel free to use any 3rd party libraries that you find appropriate and would be typically used in a production code (e.g. boost, Eigen, etc.). We provide also python definition of the function in case you would like to cross-check the answers.

C++ Implementation:

```
std::pair<double, double> func02(double a, double b, const
std::string& c) {
    // Complex but deterministic transformations
    double phaseA = std::sin(M_PI * a / 4) * std::cos(M_PI * a /
6);
    double phaseB = std::sin(M_PI * b / 3) * std::cos(M_PI * b /
5);

    // Category-based multipliers with fixed rules
    double categoryMultiplier = 1.0;
    if (c == "p") categoryMultiplier = 1.2;
    else if (c == "q") categoryMultiplier = 0.8;
    else if (c == "r") categoryMultiplier = 1.5;
    else if (c == "s") categoryMultiplier = 0.9;
```

```
    // Complex quadratic form centered at (3,0)
    double baseG = 10.0 + 0.1 - 0.5 * std::pow(a - 3.0, 2);

    // Apply transformations with controlled precision
    double g = baseG * (1 + phaseA * 0.1) * categoryMultiplier;

    // Category-dependent output calculation
    double h;
    if (c == "p") {
        h = g * (a + std::abs(std::sin(a * 7)));
    } else if (c == "q") {
        h = g / (std::abs(b) + 0.001) * (1 + std::abs(std::cos(b
* 5)));
    } else {
        h = g - std::log(1 + std::abs(a * b)) * (1 +
std::abs(phaseA * phaseB));
    }

    // Round to prevent floating point discrepancies
    g = std::round(g * 1000000) / 1000000;
    h = std::round(h * 1000000) / 1000000;

    return std::make_pair(g, h);
}
```

Python Implementation:

```
 def func02(a, b, c):
    # Convert inputs to ensure consistency
    a = float(a)
    b = float(b)

    # Complex but deterministic transformations
    phaseA = np.sin(np.pi * a / 4) * np.cos(np.pi * a / 6)
    phaseB = np.sin(np.pi * b / 3) * np.cos(np.pi * b / 5)

    # Category-based multipliers with fixed rules
    categoryMultiplier = 1.0
```

```
    if c == 'p':
        categoryMultiplier = 1.2
    elif c == 'q':
        categoryMultiplier = 0.8
    elif c == 'r':
        categoryMultiplier = 1.5
    elif c == 's':
        categoryMultiplier = 0.9

    # Complex quadratic form centered at (3,0)
    baseG = 10.0 + 0.1 - 0.5 * np.power(a - 3.0, 2)

    # Apply transformations with controlled precision
    g = baseG * (1 + phaseA * 0.1) * categoryMultiplier

    # Category-dependent output calculation
    if c == 'p':
        h = g * (a + abs(np.sin(a * 7)))
    elif c == 'q':
        h = g / (abs(b) + 0.001) * (1 + abs(np.cos(b * 5)))
    else:
        h = g - np.log(1 + abs(a * b)) * (1 + abs(phaseA *
phaseB))

    # Round to prevent floating point discrepancies
    return (
        round(g * 1000000) / 1000000,
        round(h * 1000000) / 1000000
    )
```

Dataset:
You are provided with a **CSV file** containing approximately 500,000 rows and 4 columns:

- `f`: A categorical column containing city names
- `a`: A numeric column
- `b`: A numeric column
- `c`: A categorical column with values 'p', 'q', 'r', or 's'

Data Cleaning Instructions: Before answering any question, you should first clean the dataset by excluding all rows where either 'a' or 'b' contains non-numeric values. All questions below should be answered using only this cleaned dataset.

Questions:
Answer the following questions, providing numerical answers with the specified precision:

1. *Basic Function Application (10 points)* Apply the function to all rows in the cleaned dataset. Return the average value of 'g' rounded to 4 decimal places.
2. *Filtered Calculation (15 points)* Apply the function only to rows where 'f' is 'Tokyo'. Return the sum of all 'h' values where 'g' is greater than 1. Round to 2 decimal places.
3. *Grouped Analysis (20 points)* For each unique value of 'c' (only consider 'p', 'q', 'r', 's'), calculate the median 'g' value. Return the sum of these four medians, rounded to 3 decimal places.
4. *Optimization Problem (30 points)* For rows where 'f' is 'Paris', find the value of 'a' that maximizes the average output value of 'h' when used in place of the actual 'a' values. In other words, use a single fixed value for 'a' (ignoring the values in the dataset), while using the provided values for 'b' and 'c'. Your solution must use a proper optimization algorithm (not an exhaustive search) and achieve a precision of at least 0.0001. Return both this optimal value of 'a' and the resulting maximum average 'h' value as a tuple, with both values rounded to 2 decimal places. Run the optimization algorithm over the range of values ‘a’ presented in the entire dataset.
5. *Parameter Sensitivity (25 points)* For rows where 'f' is 'New York', determine how sensitive the average value of 'g' is to small changes in 'a'. Specifically, calculate the approximate derivative of the average 'g' with respect to 'a' at the point where 'a' equals 1.0 for all rows. Use a small delta approach with $\Delta a = 0.0001$. Return this sensitivity value rounded to 6 decimal places.

Submission Format:

1. Submit a plain text file containing your 5 answers, one per line, in the order of the questions. For question 4, provide the two values separated by a comma.

Example submission:

```
9.9321
995395.03
98.148
9.29, 95.31
9.999950
```

2. Attach your source code for review. Your code will be evaluated not only for correctness but also for clarity, efficiency, and elegance.