Table of Contents

[How To Analyze This Document? 3](#_Toc183066107)

[Question 1– Lambda Function 4](#_Toc183066108)

[Codestral-mamba:7b 4](#_Toc183066109)

[Granite3-dense:2b 4](#_Toc183066110)

[Granite3-dense:8b 4](#_Toc183066111)

[Llama3.1:8b 5](#_Toc183066112)

[Starcoder2 5](#_Toc183066113)

[Question 2 – Quicksort Code 6](#_Toc183066114)

[Codestral-mamba:7b 6](#_Toc183066115)

[Granite3-dense:2b 6](#_Toc183066116)

[Granite3-dense:8b 6](#_Toc183066117)

[Llama3.1:8b 7](#_Toc183066118)

[Starcoder2: 7](#_Toc183066119)

[Question 3 – Constructor/Destructor in Java 8](#_Toc183066120)

[Codestral-mamba:7b 8](#_Toc183066121)

[Granite3-dense:2b 8](#_Toc183066122)

[Granite3-dense:8b 8](#_Toc183066123)

[Llama3.1:8b 9](#_Toc183066124)

[Starcoder2 9](#_Toc183066125)

[Question 4 – Binary Search Code 10](#_Toc183066126)

[Codestral-mamba:7b 10](#_Toc183066127)

[Granite3-dense:2b 10](#_Toc183066128)

[Granite3-dense:8b 11](#_Toc183066129)

[Llama3.1:8b 11](#_Toc183066130)

[Starcoder2 11](#_Toc183066131)

[Question 5 – Debugging the Code 12](#_Toc183066132)

[Codestral-mamba:7b 12](#_Toc183066133)

[Granite3-dense:2b 12](#_Toc183066134)

[Granite3-dense:8b 12](#_Toc183066135)

[Llama3.1:8b 13](#_Toc183066136)

[Starcoder2 13](#_Toc183066137)

[Question 6 – Port the code to Java 15](#_Toc183066138)

[Codestral-mamba:7b 15](#_Toc183066139)

[Granite3-dense:2b 15](#_Toc183066140)

[Granite3dense:8b 16](#_Toc183066141)

[Llama3.1:8b 16](#_Toc183066142)

[Starcoder2 17](#_Toc183066143)

[Question 7 – Refactor Into Quarkus 18](#_Toc183066144)

[Codestral-mamba:7b 18](#_Toc183066145)

[Granite3-dense:2b 18](#_Toc183066146)

[Granite3dense:8b 18](#_Toc183066147)

[Llama3.1:8b 18](#_Toc183066148)

[Starcoder2: 19](#_Toc183066149)

[Question 8 – Optimization Of Code 20](#_Toc183066150)

[Codestral-mamba:7b 20](#_Toc183066151)

[Granite3-dense:2b 20](#_Toc183066152)

[Granite3-dense:8b 20](#_Toc183066153)

[Llama3.1:8b 21](#_Toc183066154)

[Starcoder2 21](#_Toc183066155)

[Question 9 – Documentation Of Code 22](#_Toc183066156)

[Question 10 – Unit Test of Code 23](#_Toc183066157)

[Codestral-mamba:7b 23](#_Toc183066158)

[Granite3-dense:2b 23](#_Toc183066159)

[Granite3-dense:8b 24](#_Toc183066160)

[Llama3.1:8b 24](#_Toc183066161)

[Starcoder2 24](#_Toc183066162)

# How To Analyze This Document?

This document contains several test cases ran on five different models - Llama 3.2:8b, granite3-dense:2b, granite3-dense:8b, Codestral\_mamba:7b and Starcoder2.

The test cases cover different scenarios and topics ranging from a simple theoretical question to documenting a code, from generating a code to debugging the code.

The text in red means that it was a problematic behavior by the model, while the text in green means that the behavior was good.

# Question 1– Lambda Function

Query - What is Lambda function in python?

### Codestral\_mamba:7b

###### Execution 1-3: PASS

Format used in all three executions: Definition -> Usage -> Two examples. Results of all three execution is similar. Simple definition with well explained examples.

### Granite3-dense:2b

###### Execution 1-3: PASS

Format used in all three executions: Definition -> Usage -> explained with example. Results of all three execution is similar. Simple definition with well explained examples.

### Granite3-dense:8b

###### Execution 1: PASS

Format: Definition -> Usage syntax -> Example. But no explanation of the example was provided.

###### Execution 2: PASS

Better result. Format: Definition -> Usage syntax -> explained with example.

###### Execution 3: PASS

Same result as execution 2.

### Llama3.1:8b

###### Execution 1: PASS

Format used: Definition -> Usage Syntax -> Example -> Use Cases -> Best Practices -> Real-world Examples -> Explanation of examples.

###### Execution 2: PASS

Format used: Definition -> Usage Syntax -> Example 1-3 -> Advantages -> Best practices.

###### Execution 3: PASS

Format used: Definition -> Usage Syntax -> Examples use cases -> Advantages.

Everything is well written, but it might be an overkill for some users as it takes a lot of time to print this information for such a simple question.

Starcoder2

FAIL => Only prints one line: A lambda function is a small anonymous function.

# Question 2 – Quicksort Code

Query - Write code for quick-sort algorithm in Python.

### Codestral\_mamba:7b

###### Execution 1: PASS

Prints a full undocumented python code with brief explanation of the code.

###### Execution 2: PASS

Prints an undocumented python code which is different from the first execution. It uses list comprehension instead of fully writing the loops. Also writes a brief explanation of the code.

###### Execution 3: PASS

Similar result as execution 2.

### Granite3-dense:2b

###### Execution 1-3: PASS

Quicksort function is correct. It gives simple explanation and explains with examples.

### Granite3-dense:8b

###### Execution 1: PASS

The code accurately implements the in-place quick-sort algorithm using the partition function and recursion. It includes an example array, sorts it, and prints the result, making it complete and well-explained.

###### Execution 2: PASS

Same result as execution 1. It is accurate, but it sorts arrays in a new list rather than in-place, which could be clarified. It does not include an example or demonstrate output.

###### Execution 3: FAIL

Failure Reason: incorrect reference to the Lomuto partition scheme makes the response misleading.

### Llama3.1:8b

###### Execution 1: PASS

The output contains -> A well-documented quicksort function with usage code, the explanation of how quicksort works and advantages of quicksort.

###### Execution 2: PASS

Adds more detail to the previous result like time complexity and best practices to use the code.

Execution 3: PASS

Similar result as previous result but with slightly more explanation.

Starcoder2:   
  
FAIL => Explains quick-sort algorithm in a couple of lines. Does not write a single line of code.

# Question 3 – Constructor/Destructor in Java

Query - How do you declare constructors and destructors in Java?

### Codestral\_mamba:7b

###### Execution 1: PASS

Gives simple definition of constructors and destructors, along with an example.

###### Execution 2: PASS

Similar result as execution 1. Also adds the syntax for finalize () method.

###### Execution 3: PASS

Similar result as execution 2 with slight modification in the code.

### Granite3-dense:2b

###### Execution 1: PASS

Explains constructors and the lack of destructors in Java. It provides examples for both constructors and the finalize() method

###### Execution 2: PASS

Identical to Execution 1. It correctly explains constructors and destructors in Java, including examples for both.

###### Execution 3: PASS

This response is also identical to the first two. It accurately describes constructors and destructors in Java, supported by examples.

### Granite3-dense:8b

###### Execution 1: PASS

Gives simple definition of constructors and destructors, along with usage syntax of constructor and finalize () function.

###### Execution 2: PASS

Similar result as execution 1, just adds a little more explanation.

###### Execution 3: PASS

Slightly improved version of first two executions but nothing different in the format of the answer.

### Llama3.1:8b

###### Execution 1: PASS

Gives a detailed explanation about constructors and destructors, their basic syntax, the rules to use them, and a lot of examples.

###### Execution 2: FAIL

Gives incomplete output -> it starts explaining the types of constructors and explains a few of them but stops printing abruptly.

Failure Reason: Incomplete code.

###### Execution 3: PASS

Explains constructors, destructors and types of constructors. It also explains a few unnecessary things like difference between constructors and methods, which could have been ignored.

Starcoder2

FAIL => Writes the definition of constructors in a couple of lines. Does not write a single line of code

# Question 4 – Binary Search Code

Query – Write code for Binary Search along with the documentation of the code.

### Codestral\_mamba:7b

###### Execution 1: PASS

Prints a documented binary search code with example and brief explanation.

###### Execution 2: PASS

Prints a documented binary search code like execution 1 with additional details in the explanation.

###### Execution 3: PASS

Prints a binary search code with more detailed documentation and more detailed explanation of the code.

### Granite3-dense:2b

###### Execution 1: PASS

The code accurately implements a recursive binary search with detailed documentation explaining parameters, return values, and algorithm logic. However, it does not include an example of how to use the function.

###### Execution 2: PASS

The code and documentation are identical to Execution 1. The implementation is accurate, but it lacks an example of the function in action.

###### Execution 3: PASS

The code is identical to the previous two executions. It correctly implements binary search and provides comprehensive documentation, but it does not include an example demonstrating its usage.

### Granite3-dense:8b

###### Execution 1: PASS

The implementation is accurate and includes a definition of binary search, explaining its purpose and input parameters. The usage of the algorithm is described and the example is well-explained

###### Execution 2: PASS

Identical to Execution 1in terms of functionality. Detail usage explanation. The example is again well-written but lacks an actual demonstration of inputs and outputs for practical illustration.

###### Execution 3: PASS

Same result as execution 1.

### Llama3.1:8b

###### Execution 1: PASS

Prints full code of binary search with a lot of documentation and explanation. It also prints the time complexity of binary search.

###### Execution 2: PASS

Prints a documented Java code for binary search along with explanation of the code. It also explains the time and space complexity of the code.

###### Execution 3: PASS

Similar result as execution 2, but this time the code is in Python.

Starcoder2

FAIL => Explains binary search in a small paragraph. Does not write code.

# Question 5 – Debugging the Code

Query - Fix the bugs in the code: @documentation\_with\_bugs.py

The result for question 5 can be perceived as least bad to worst. In that case, we can look at the detailed result as:

### Codestral\_mamba:7b

###### Execution 1: PASS

Best result out of all. Corrects all the bugs in the code. Does not explain all the bugs but correct all of them.

###### Execution 2: FAIL

It does not fully fix the code as it left a bug at the end of the code.

Failure Reason: Does not fix the code.

###### Execution 3: FAIL

Fixed all the bugs in the code, but the explanation which is provided is unrelated to the bugs fixed, i.e., the bug which was fixed was different, and the explanation provided was different.

Failure Reason: Incorrect explanation provided.

### Granite3-dense:2b

###### Execution 1-3: FAIL

Could only debug the first class and does not provide output of the second class and main function.

Failure Reason: Incomplete code.

### Granite3-dense:8b

###### Execution 1: FAIL

Does not read the full code either but reads a few more line .In one bug, it removes the buggy line instead of fixing it in the code.

Failure Reason: Incomplete code.

###### Execution 2-3: FAIL

Reads a few more lines and fixes a couple more bugs but is still not able to read and finish the full code.

Failure Reason: Incomplete code.

### Llama3.1:8b

###### Execution 1: PASS

Fixes all the bugs. Also adds some exception handling to the code.

###### Execution 2: PASS

Fixes all the bugs, also adds some documentation to the code.

###### Execution 3: PASS

Fixes all the bugs, also does some unnecessary code changes like changing the name of the function. Does not do documentation and error handling in this case.

NOTE: In all the three scenarios of Llama, it did not provide a satisfactory explanation. It gave details of what it added to the code (like error handling) but does not provide good explanation of the bug fixes.

Starcoder2   
  
FAIL => Simply returns the error from the execution, instead of fixing the bugs.

# Question 6 – Port the code to Java

Query - Port the code @documentation.py to java

### Codestral\_mamba:7b

###### Execution 1: FAIL

Converts the code and gives a brief explanation. But the converted Java code is broken and does not work as intended.

Failure Reason: Output code has errors.

###### Execution 2: PASS

Converts the code correctly but does not give any explanation.

###### Execution 3: FAIL

Converts the code into multiple classes which needs to be created as separate files. This can be an undesirable behavior.

Failure Reason: Does not convert the code into one file.

### Granite3-dense:2b

###### Execution 1: FAIL

Creates two public classes from the original code. Does not import any library/class. Also does not convert the main function. Gives a brief description of the code.

Failure Reason: Incomplete code.

###### Execution 2: FAIL

This time the classes are not public, but it still does not import any library, nor does it convert the main function. Gives a detailed description of the conversion.

Failure Reason: Incomplete code.

Execution 3: FAIL

Again, makes the classes as public and does not import any library. But this time it converts the main function.

Failure Reason: Incomplete code.

ALL three executions give an unusable code.

### Granite3-dense:8b

###### Execution 1: FAIL

Prints incomplete code (prints both classes but incomplete main function -> prints only a few cases of switch statement). Also makes both classes as public.

Failure Reason: Incomplete code.

###### Execution 2: FAIL

Prints incomplete code again but this time prints a few more cases of the switch statement. Does not make the classes public in this case.

Failure Reason: Incomplete code.

###### Execution 3: FAIL

Similar code length read (incomplete) in this case as well. Again, makes both classes as public.

Failure Reason: Incomplete code.

### Llama3.1:8b

###### Execution 1: FAIL

Converts full code into Java but leaves out some print statement. If those print statements were there, then it would have been a perfect conversion. It also gives the explanation of how functions and objects in Python were mapped to Java.

Failure Reason: Incomplete code.

###### Execution 2: FAIL

Incorrectly converts the Python code into multiple public classes.

Pros: Tells in the beginning how to use those classes in separate files.

Failure Reason: Output code has errors.

###### Execution 3: PASS

Finally converts the full code accurately. Gives a brief description about the functions and objects used.

NOTE: In all three executions, the model changed the names of a few methods and variables. This is an undesirable behavior as it might break the dependency of the code on some other files.

Starcoder2

FAIL Prints a few lines of gibberish code.

# Question 7 – Refactor Into Quarkus

Query - Refactor the above generated code to quarkus.

### Codestral\_mamba:7b (Rank: )

###### Execution 1:

###### Execution 2:

###### Execution 3:

### Granite3-dense:2b(Rank: )

Execution 1-3: FAIL

Execution 2: FAIL

Execution 3: FAIL

Failure Reason: Incomplete Quarkus Refactoring,syntax errors.

### Granite3-dense:8b

Execution 1: FAIL

The refactored code provided does not fully incorporate Quarkus features.

Execution 2: FAIL

Execution 3: FAIL

### Llama3.1:8b

Execution 1:

Failure Reason: It successfully translates the structure and logic but falls short of fully integrating Quarkus features like REST endpoints, dependency injection, and database handling

Execution 2:

Failure Reason: like execution1 and Incomplete code.

Execution 3:

Failure Reason: like execution1 and Incomplete code.

Starcoder2:

FAIL Prints a few lines of gibberish code.

# Question 8 – Optimization Of Code

Query – Optimize the code @code\_to\_optimize.py.

[The input code was an unoptimized version of bubble sort]

### Codestral\_mamba:7b

###### Execution 1: PASS

Converts the code into quick-sort and explains that quick-sort has better time complexity than bubble sort.

###### Execution 2: PASS

Same behavior has the first execution, but with more explanation.

###### Execution 3: PASS

Again, converts into quick sort. This time it uses list comprehension instead of the full code.

### Granite3-dense:2b

###### Execution 1: PASS

Format Used: Definition -> Optimization -> Explained with Example. The response correctly optimizes the Bubble Sort by reducing unnecessary iterations and provides a clear explanation of the optimization logic and its impact.

###### Execution 2: PASS

Like execution1

Execution 3: PASS

Like execution1&2

### Granite3-dense:8b

###### Execution 1: PASS

Format Used: Definition -> Optimization -> Explained with Example. It is concise, accurate, and well-explained.

###### Execution 2: PASS

Similar to execution1.

###### Execution 3: FAIL

It attempts to introduce alternative algorithms (e.g., QuickSort) but fails to provide correct or executable implementations.

### Llama3.1:8b

###### Execution 1: PASS

This execution converts the unoptimized bubble sort into an optimized bubble sort. It also explains the optimization done by the model, like reducing number of comparisons.

UNDESIRABLE BEHAVIOUR: It changed the name of the function given as input.

###### Execution 2: FAIL

It returns a similar output as execution 1, but this time the code does not work. The output algorithm does not sort the input array.

Failure Reason: Incorrect code.

###### Execution 3: PASS

It optimizes the input code by just a slight margin. It also explains how the in-built “sort” function of Python is a better choice to use.

Starcoder2

Prints a few lines of gibberish code.

# Question 9 – Documentation Of Code

Query – Write documentation comment of the code @documentation.py

### Codestral\_mamba:7b

### Granite3-dense:2b

### Granite3-dense:8b

### Llama3.1:8b

Starcoder2

# Question 10 – Unit Test of Code

Query -Create Unit test for the code @documentation.py

### Codestral\_mamba:7b

###### Execution 1: PASS

Give unit test of the code using unittest library of python. Runs 5 test case and explains the output code.

###### Execution 2: PASS

Similar code and output as 1st execution.

###### Execution 3: PASS

Creates a separate file for each subclass to unit-test the code and explains how to run these files.

### Granite3-dense:2b

###### Execution 1: FAIL

Incorrect imports, flawed balance assertions, and incomplete account setup.

Execution 2: FAIL

Same as execution1 with identical issues

Execution 3: FAIL

Same as execution1 with identical issues

### Granite3-dense:8b

###### Execution 1: FAIL

Does not even read the first class.

###### Execution 2: PASS

It tests all the critical methods of the BankAccount class. The use of setUp simplifies test initialization, and the assertions validate functionality thoroughly. However, it does not test the BankSystem class, which could make it more comprehensive.

###### Execution 3: PASS

### Covers both BankAccount and BankSystem but contains a minor logic error. Better than execution2.

### Llama3.1:8b

###### Execution 1: FAIL

Gives incorrect code for testing. The output code has attribute error.

###### Execution 2: PASS

Creates testing classes for both the input classes and explains where the code will fail.

###### Execution 3: PASS

Gives modular code to test the two classes separately in separate files.

Starcoder2

FAIL: Prints a few lines of gibberish code.