Findings:

In this task, we analyzed a reinforcement learning agent implemented in Python that solves a simplified FrozenLake-style environment using the Q-learning algorithm. The environment consists of a 5x5 grid, with a start state, a goal (win) state, and several hole (lose) states. The agent learns optimal navigation strategies by updating Q-values associated with each state-action pair using an epsilon-greedy policy and iterative updates. To evaluate the efficiency of this implementation, we used a large language model (LLM) to help identify the key computational bottlenecks in the code.

The LLM accurately identified several performance bottlenecks. The most significant one is the repeated linear search over Q-values for each possible action in a given state to find the best action or the maximum Q-value. This process involves manual iteration and comparison over all available actions (in this case, four actions: up, down, left, right), which becomes inefficient as the environment scales. The LLM also noted other inefficiencies, such as the frequent deep copying of the entire Q-table dictionary, redundant reward evaluations, and plotting reward values at every step rather than per episode. All suggestions made logical sense and aligned well with known performance bottlenecks in reinforcement learning implementations, especially when preparing algorithms for hardware acceleration.

Focusing on the most impactful bottleneck—the linear Q-value search—the LLM proposed a hardware implementation of a Q-value Max Selector Module. This module is designed to receive the current state and a list of Q-values and quickly output the best action and its corresponding Q-value. The proposed design leverages parallel comparison logic and constant-time memory access to eliminate the need for a software loop, drastically reducing the decision latency per step.

Finally, the LLM generated a clean and modular SystemVerilog implementation of the Q-value Max Selector. The code defines a combinational logic block that compares four signed 16-bit Q-values in parallel and outputs the maximum value along with the action index (encoded as a 2-bit signal). This hardware component can be easily integrated with a Q-table memory and reinforcement learning controller to accelerate agent decision-making in environments like FrozenLake. Overall, the exercise demonstrates how LLMs can effectively identify algorithmic inefficiencies and aid in designing hardware accelerators for performance-critical AI components.