### **Title: Hierarchical Data Processing with Advanced Calculations and Robust Error Handling**

### **As a**

Business Analyst responsible for data integrity and reporting in Salesforce,

### **I want**

to have an invocable action that enables hierarchical data processing within Salesforce Flow, allowing AVG, MIN, MAX, and SUM calculations at each node,

### **So that**

I can generate accurate, efficient, and real-time data analysis while ensuring seamless error handling and optimal system performance.

### **Background & Business Need:**

SLEEQ is undertaking a high-complexity Salesforce project requiring advanced hierarchical data calculations and rigorous error management. Businesses dealing with nested data structures—such as organizational hierarchies, product categorization, or financial rollups—need reliable and scalable solutions. Without an efficient way to process hierarchical data, organizations may experience inefficiencies, inconsistencies, and performance issues in data reporting.

The proposed invocable action will allow Salesforce Flow to process hierarchical data efficiently while dynamically adapting to various data structures. This ensures businesses can leverage real-time calculations without exceeding governor limits or encountering system failures due to unhandled exceptions.

### **Acceptance Criteria:**

1. **Hierarchical Data Processing:**
   1. The invocable action must traverse hierarchical data structures efficiently.
   2. It should process multiple levels of hierarchy, performing AVG, MIN, MAX, and SUM calculations on designated numeric fields.
2. **Dynamic Apex Utilization:**
   1. The solution must incorporate dynamic SOQL queries and adaptable logic to accommodate different hierarchical data models.
   2. Governor limit optimization strategies must be implemented to ensure scalable performance.
3. **Error Handling & Logging:**
   1. Any system failure must be captured and logged with relevant details.
   2. The solution must notify users in case of failures, ensuring business continuity.
4. **Salesforce Flow Integration:**
   1. The invocable action should be seamlessly usable within Salesforce Flow.
   2. The calculated results should be easily retrievable and displayable within Flow.
5. **Test Coverage:**
   1. A robust test class must cover all possible scenarios, including edge cases.
   2. The test class should ensure high code coverage and validate error-handling mechanisms.