**Exercise 2: Prompt Flow with Python, Cosmos DB, and ChatGPT Integration**

**Objective**

This exercise focuses on building a multi-step Prompt Flow application using Python, Cosmos DB, and ChatGPT. The aim is to automate the recommendation process for infrastructure provisioning based on user requirements and validate it against available hosting products.

**Steps and Tasks**

**Part 0: Input, Customer Name, email and question. Where as name, email is given once, question can be repeated if needed.**

**Part 1: Prompt Flow to Recommend Minimal Cluster Size**

1. **Template for ChatGPT Prompt:**
   * Build a prompt template to capture user requirements and recommend PostgreSQL cluster components.
   * The prompt should dynamically accept user inputs such as:
     + Workload type (read-heavy, write-heavy, balanced)
     + Data size (inserts/day)
     + Number of users
   * Expected output is a JSON list containing cluster recommendations:

[

{"component": "PostgreSQL Master", "RAM": "32GB", "Cores": "4", "Bandwidth": "1TB", "Storage": "300GB"},

{"component": "PostgreSQL Replica", "RAM": "16GB", "Cores": "2", "Bandwidth": "500GB", "Storage": "200GB"}

]

If question asked about kafka cluster, the list may include as below, ask the chatgpt to provide results in GB, by unit. The below is output from chatgpt, is just an example format. Design properly yourself

[

{"component": "ZooKeeper", "RAM": "16", "Cores": "4", "Bandwidth": "100", "Storage": "300 "},

{"component": "Broker 1", "RAM": "16", "Cores": "8", "Bandwidth": "500", "Storage": "1000"}

{"component": "Broker 2", "RAM": "16", "Cores": "8", "Bandwidth": "500", "Storage": "1000"}

{"component": "Broker 3", "RAM": "16", "Cores": "8", "Bandwidth": "500", "Storage": "1000"}

{"component": "Broker 4", "RAM": "16", "Cores": "8", "Bandwidth": "500", "Storage": "1000"}

]

**Python Implementation for Flow Step 1:**

* + Use Azure OpenAI Python SDK to pass user input to ChatGPT and parse the response.

**Part 2: Validate Cluster Size Against Available Hosting Plans**

1. **Product Dataset (cosmos db):**
   * Add or generate data for Products in Cosmos db [already done in exercise 2), including specifications (Cores, RAM, Storage, Bandwidth, Monthly Cost) if needed in case
2. **Python Program Tasks:**
   * Read the Products container.
   * Compare the ChatGPT-recommended cluster specs with available hosting plans.
   * Select plans that meet or exceed the requirements. If no suitable plan exists:
     + Suggest adding a custom plan to the database as per your interest.

**Part 3: Query Cosmos DB for Matching Plans**

1. **Cosmos DB Integration:**
   * Query Cosmos DB to fetch plans matching the ChatGPT-recommended specifications.
2. **Python Program Tasks:**
   * Use Cosmos DB Python SDK to query the database for products meeting or exceeding the recommended specs.
   * Collect the filtered results.

**Part 4: Pass Filtered Data Back to ChatGPT**

1. **ChatGPT Task:**
   * Pass the filtered results (list of plans) to ChatGPT with the following prompt:

Based on the user requirements, recommend a product or set of products from the list provided that best suit the needs.

1. **Expected Output:**

ChatGPT will return the best-suited plans.

**Part 5: Use Python, Pandas in the flow to calculate total units of each server, return json output from there.**

Calculate the approximate monthly price for the chosen products, including an 18% tax. along with the total monthly price including tax. Output should in json format, tax and total in separate document in the list

[

{id: 2, name: Intel Nano, cores: 2, RAM: 16 gb,Bandwidth 2 Tb, unit\_price: 50 USD, qty: 2, amount\_before\_tax: 100, tax: 18, total\_amount: 118},

{id: 20, name: AMD suupreme, cores: 2, RAM: 16 …}

]

Expected Result: Above json must be returned.

Part 6:

Ask GPT to generate nicely formatted table/ASCII (ASCII preferable) from above json, along with total amount including tax

**Output:**

Nicely formatted output in plain ASCII text

Part 7:

**Sending Emails to Customers**:

* Use Azure's **SendGrid Email API** to send emails to customers with the output of the recommendation process.
* Email will include:
  + Customer's Name and Email ID.
  + Recommended products with calculated prices (subtotal, tax, total).
* Include a fallback to log the email content locally if email fails.

**Storing Data in Azure Blob Storage**:

* Save the recommendation output and email content as a Markdown (.md) file in a blob container:
  + Path: /enquiries/<date>-<time>-enquiry.md.