**Jupyter Notebook Lab Exercise: Attention & Self-Attention Mechanism**

**Problem Definition:**

The goal of this exercise is to deeply understand the mechanics of a single Scaled Dot-Product Attention "head" by manually performing the calculations and articulating the conceptual meaning of each step. This will provide a solid foundation for grasping how attention enables models to weigh the importance of different parts of a sequence.

**Background & Theory:**

The attention mechanism, as seen in Transformer models, allows a model to dynamically decide which parts of an input sequence are most relevant when processing a specific element. It relies on three fundamental components:

* **Queries (Q):** Represents what you are currently seeking or the element for which you are computing a new representation.
* **Keys (K):** Represents what is available to be looked at or the features of elements that might be relevant.
* **Values (V):** Represents the actual information content associated with each key. If a key is deemed relevant, its value is what gets extracted and used.

The attention process involves several key mathematical steps:

1. **Similarity Calculation (QKT):** Measuring how well each Query aligns with each Key using a dot product.
2. **Scaling (…/dk​​):** Adjusting these similarity scores to ensure stable gradient flow during training, especially for high-dimensional vectors.
3. **Normalization (Softmax):** Converting the scaled scores into a probability distribution (attention weights), indicating the proportional importance of each Key for a given Query.
4. **Weighted Sum (AV):** Combining the Value vectors based on these attention weights to produce a context-aware output for each Query.

**Your "Notebook" Setup: Hypothetical Inputs**

Imagine we have a very simplified scenario.

* **Embedding Dimension:** Let's set this to 2.
* **Query Matrix (Q):** Q=​101​011​​ *(Represents 3 queries/tokens)*
* **Key Matrix (K):** K=​11−1​1−11​​ *(Represents 3 keys/tokens)*
* **Value Matrix (V):** V=​1025​1205​​ *(Represents 3 value vectors, corresponding to the keys)*

**Task 1: Calculate Raw Attention Scores (QKT)**

1. **Step:** Calculate the transpose of the Key matrix (KT).
2. **Step:** Perform the matrix multiplication QKT. This will give you a matrix of raw attention scores.
3. **Conceptual Question:** For the first query (row 0 of Q), what does its dot product with the second key (column 1 of KT) represent?

**Task 2: Scale the Raw Attention Scores**

1. **Step:** Determine the scaling factor, which is the square root of the embedding dimension (dk​).
2. **Step:** Divide every element in your QKT matrix (from Task 1) by this scaling factor.
3. **Conceptual Question:** In a real Transformer where dk​ might be 64 or 128, why is this scaling step particularly important?

**Task 3: Normalize Attention Weights with Softmax**

1. **Step:** Apply the softmax function to the scaled attention scores (from Task 2), **performing this operation row-wise**.
   * Recall: For a vector x=[x1​,x2​,…,xn​], softmax(xi​)=∑j=1n​exj​exi​​. (You might need a calculator for ex values).
2. **Verification:** For each row of your result, verify that the sum of its elements is approximately 1.
3. **Conceptual Question:** What is the primary purpose of applying softmax here? How do these normalized weights dictate "attention"?

**Task 4: Compute Context-Weighted Embeddings**

1. **Step:** Perform the matrix multiplication of your **normalized attention weights** (from Task 3) with the **Value matrix (V)** (given in the "Notebook" Setup).
2. **Conceptual Question:** What does each row of this final matrix (Output) represent? How has the information from the original V matrix been transformed?
3. **Illustrative Example:** Manually show how the first row of your final Output matrix is calculated by using the first row of your normalized attention weights as coefficients for the rows of the V matrix. Write down the explicit sum.

**Final Analysis & Reflection (Write down your answers):**

1. **The "Focus":** Based on your normalized\_attn\_weights matrix from Task 3, which of your original "queries" (row 0, 1, or 2 of Q) seems to pay the most attention to the "key" at index 1 (row 1 of K)? Explain why.
2. **Contextualization:** If the V vector at index 0 represented "apple" and the V vector at index 1 represented "tree", and a query mostly attended to "tree," how would the resulting context-weighted embedding for that query reflect this attention?
3. **Beyond Self-Attention:** Imagine this mechanism was used in an encoder-decoder setup for machine translation. If Q came from the decoder's current state, and K and V came from the encoder's output, what would the "context-weighted embeddings" signify in that scenario?

By completing this exercise, you will have a strong conceptual grasp of the computational flow within a scaled dot-product attention mechanism!