**Explanation**

**1. Compute Norms Using cublasDnrm2**

* **Purpose:**
  + Compute the Euclidean norm (2-norm) of each column vector in mtxY\text{mtxY}mtxY.
* **Implementation:**
  + Loop over each column (i from 0 to numCols - 1).
  + Call cublasDnrm2 to compute the norm of column i.
    - **Parameters:**
      * n: Number of elements in the vector (numRows).
      * x: Pointer to the start of column i in mtxY\_d (mtxY\_d + i \* numRows).
      * incx: Increment between elements (1 since columns are contiguous).
      * result: Pointer to store the computed norm (norms\_d + i).
* **Note:**
  + norms\_d is a device array allocated to store the norms of all columns.

**2. Compute Inverses of Norms Using a Custom Kernel**

* **Purpose:**
  + Compute the reciprocal of each norm to use for scaling (i.e., normalize the columns).
* **Implementation:**
  + Define a CUDA kernel compute\_inverses that computes 1.0 / norms[idx] for each element.
  + Launch the kernel with appropriate grid and block dimensions.
    - **Grid Size:**
      * Calculated as (numCols + blockSize - 1) / blockSize to cover all columns.
    - **Block Size:**
      * Set to 256 threads per block (can be adjusted based on hardware).
* **Handling Edge Cases:**
  + If a norm is zero, set the inverse to zero to avoid division by zero.

**3. Scale Each Column Using cublasDscal**

* **Purpose:**
  + Multiply each element of column i by the inverse of its norm to normalize the column.
* **Implementation:**
  + Loop over each column (i from 0 to numCols - 1).
  + Call cublasDscal to scale column i by norms\_d[i].
    - **Parameters:**
      * n: Number of elements in the vector (numRows).
      * alpha: Pointer to the scaling factor (norms\_d + i).
      * x: Pointer to the start of column i in mtxY\_d (mtxY\_d + i \* numRows).
      * incx: Increment between elements (1).

**4. Memory Management**

* **Device Memory Allocation:**
  + norms\_d is allocated on the device to store norms and their inverses.
  + No host-device transfers are needed for norms, as all computations are on the device.
* **Freeing Memory:**
  + After normalization, norms\_d is freed to prevent memory leaks.