# 2. Morpheus Data Split: 데이터 분할 및 복제 로직

## 프로젝트 개요

Exercise-split-data.ipynb를 통해, 들어오는 단일 메시지(MessageMeta)의 데이터를 **필터 조건에 따라 분할**하고, 그 중 필터링된 메시지만을 **별도의 출력 포트로** 보내는 커스텀 스테이지를 구현했습니다.

## ⚙️ 주요 기술 및 역량

* **기술:** Morpheus Stage (PassThruTypeMixin 미사용), MRC Builder, compute\_schema.
* **역량:**
  1. **조건부 데이터 분할:** 메시지 페이로드에서 특정 조건(예: data\_len > 100)을 만족하는 행만 분리하여 새로운 MessageMeta를 생성하는 로직 구현.
  2. **다중 출력 포트 관리:** 커스텀 스테이지에서 \_create\_ports(1, 2)를 선언하고, \_build 메서드에서 **분할된 데이터**와 **원본 데이터**를 각각 다른 출력 노드로 전송하도록 구현.

## 🛠️ 구현 내용 (Exercise-split-data.ipynb)

1. **스테이지 정의:** FilterSplitStage를 정의하고 filter\_fn을 인수로 받아 내부 필터 로직을 유연하게 설정할 수 있도록 했습니다.
2. **핵심 분할 로직:** on\_data 메서드 내에서 message.get\_data().query()를 사용하여 데이터프레임을 필터링하고, 필터링된 데이터프레임으로 **새로운 MessageMeta 객체를 생성**하여 출력 포트 중 하나로 보냈습니다.
3. **Pass-Through:** 원본 메시지는 변경 없이 다른 출력 포트로 보내어, \*\*"필터링된 데이터"\*\*와 **"필터링되지 않은 원본"** 데이터를 동시에 다운스트림에서 활용할 수 있는 구조를 완성했습니다.

## 📌 주요 코드 로직 (핵심 분할 로직 추상화)

class FilterSplitStage(Stage):  
 def \_\_init\_\_(self, config, filter\_fn):  
 super().\_\_init\_\_(config)  
 self.\_filter\_fn = filter\_fn  
 self.\_create\_ports(1, 2) # 1 input, 2 outputs  
  
 # ... compute\_schema, name 등 정의 ...  
  
 def \_build(self, builder, input\_nodes):  
 # MRC 노드 생성 및 연결 (복제/분할 로직을 담당하는 내부 노드)  
 split\_node = builder.make\_node("split-logic", ops.map(self.on\_data))  
 builder.make\_edge(input\_nodes[0], split\_node)  
  
 # 출력 노드 1: 필터링된 데이터  
 filtered\_output = builder.make\_node("filtered-out", ops.map(lambda x: self.\_filter\_fn(x)))  
 # 출력 노드 2: 원본 전체 데이터 (Pass-through)  
 passthrough\_output = builder.make\_node("passthrough-out", ops.pass\_thru())  
  
 # (주의: 실제 구현에서는 split\_node 내부에서 조건에 따라 데이터를 두 출력으로 보냄)  
 # 이 예시에서는 로직이 복잡하여 단순화하였으며, 핵심은 두 개의 독립된 출력을 반환하는 것임.  
 return [filtered\_output, passthrough\_output]