```
─────────────────── Traceback (most recent call last) ───────────────────
 in <module>:1

 ❱ 1 cluster = data.ClusterData(graph, 5)
   2

 /usr/local/lib/python3.10/dist-packages/torch_geometric/deprecation.py:27 in wrapper

   24 │   │   │   if details is not None:
   25 │   │   │   │   out += f", {details}"
   26 │   │   │   warnings.warn(out)
 ❱ 27 │   │   │   return func(*args, **kwargs)
   28 │   │
   29 │   │   return wrapper
   30

 /usr/local/lib/python3.10/dist-packages/torch_geometric/loader/cluster.py:86 in __init__

    83 │   │   │   if log:  # pragma: no cover
    84 │   │   │   │   print('Computing METIS partitioning...', file=sys.stderr)
    85 │   │   │
 ❱  86 │   │   │   cluster = self._metis(data.edge_index, data.num_nodes)
    87 │   │   │   self.partition = self._partition(data.edge_index, cluster)
    88 │   │   │
    89 │   │   │   if save_dir is not None:

 /usr/local/lib/python3.10/dist-packages/torch_geometric/loader/cluster.py:132 in _metis

   129 │   │   │   ).to(edge_index.device)
   130 │   │
   131 │   │   if cluster is None:
 ❱ 132 │   │   │   raise ImportError(f"'{self.__class__.__name__}' requires either "
   133 │   │   │   │   │   │   │     f"'pyg-lib' or 'torch-sparse'")
   134 │   │
   135 │   │   return cluster
─────────────────────────────────────────────────────────────────────────
ImportError: 'ClusterData' requires either 'pyg-lib' or 'torch-sparse'

─────────────────── Traceback (most recent call last) ───────────────────
 in <module>:1

 ❱ 1 sampler = data.NeighborSampler(graph.edge_index, sizes=[3,10], batch_size=4,
   2 │   │   │   │   │   │   │   │     shuffle=False)
   3

 /usr/local/lib/python3.10/dist-packages/torch_geometric/deprecation.py:27 in wrapper

   24 │   │   │   if details is not None:
   25 │   │   │   │   out += f", {details}"
   26 │   │   │   warnings.warn(out)
 ❱ 27 │   │   │   return func(*args, **kwargs)
   28 │   │
   29 │   │   return wrapper
   30

 /usr/local/lib/python3.10/dist-packages/torch_geometric/loader/neighbor_sampler.py:147 in
 __init__

   144 │   │   │   │   num_nodes = int(edge_index.max()) + 1
   145 │   │   │
   146 │   │   │   value = torch.arange(edge_index.size(1)) if return_e_id else None
 ❱ 147 │   │   │   self.adj_t = SparseTensor(row=edge_index[0], col=edge_index[1],
   148 │   │   │   │   │   │   │   │     value=value,
   149 │   │   │   │   │   │   │   │     sparse_sizes=(num_nodes, num_nodes)).t()
   150 │   │   else:

 /usr/local/lib/python3.10/dist-packages/torch_geometric/typing.py:163 in __init__

   160 │   │   │   is_sorted: bool = False,
   161 │   │   │   trust_data: bool = False,
   162 │   │   ):
 ❱ 163 │   │   │   raise ImportError("'SparseTensor' requires 'torch-sparse'")
   164 │   │
   165 │   │   @classmethod
   166 │   │   def from_edge_index(
─────────────────────────────────────────────────────────────────────────
ImportError: 'SparseTensor' requires 'torch-sparse'
```