

Patient details such as,
patient name, place, mobile number, medical symptoms, disease

email, documents, scan reports

those data are stored in database (e.g., postgres)

we need to do medical suggestions or doctor appointment or notification whatever the action we need a knowledge base

so we are transporting the data from database to elasticsearch via connector, extractor, loader

after that the indexed data from elasticsearch will be embed.

Chatgpt prompt 11/11/2025

let's move forward to loader dag:

```
def load_to_elasticsearch(transformation_result: Dict[str, Any], **context) -> bool:
```

```
    """
```

```
    Load transformed data to Elasticsearch.
```

```
    Args:
```

```
        transformation_result: Result from transformation task
```

```
    Returns:
```

```
        True if successful
```

```
    """
```

```
    try:
```

```
        table_name = transformation_result['table_name']
```

```
        logger.info(f"Loading data to Elasticsearch for table: {table_name}")
```

```
        # Get data
```

```
        data = transformation_result['data']
```

```
    if not data:
```

```
        logger.warning(f"No data to load for {table_name}")
```

```
        return True
```

```
    # Get Elasticsearch connector
```

```
    conn_manager = ConnectionManager()
```

```
    es_conn_id = context['task_instance'].xcom_pull(
```

```

        task_ids='create_elasticsearch_connector',
        key='es_conn_id'
    )
    es_connector = conn_manager.get_connection(es_conn_id)

    if not es_connector:
        raise ValueError(f"Elasticsearch connector not found: {es_conn_id}")

    # Get index configuration
    index_config = CONFIG['loading']['index_mappings'].get(table_name, {})
    index_name = index_config.get('index_name', f"etl_{table_name}")
    id_field = index_config.get('id_field')

    # Create index if it doesn't exist
    if not es_connector.index_exists(index_name):
        es_connector.create_index(index_name)
        logger.info(f"Created index: {index_name}")

    # Create loader
    loader = LoaderFactory.create_loader(
        loader_type="elasticsearch",
        connector=es_connector,
        index_name=index_name,
        id_field=id_field,
        bulk_size=CONFIG['loading']['elasticsearch'].get('bulk_size', 1000),
        max_retries=CONFIG['loading']['elasticsearch'].get('max_retries', 3),
        raise_on_error=CONFIG['loading']['elasticsearch'].get('raise_on_error',
False)
    )

    # Load data in bulk
    success = loader.load_batch(data)

    if success:
        logger.info(f"Successfully loaded {len(data)} documents to
{index_name}")
    else:
        logger.warning(f"Load completed with some errors for {index_name}")

    return success

except Exception as e:

```

```
logger.error(f"Error loading to Elasticsearch: {e}")  
raise
```

here i have a doubt. the input is transformation result. okay it's in params we pass the output of transformation fine. but inside, we write transformation result. hey our output of the transformation is just named as result. the word transformation_result is not defined yet. then how it know it means the output of the transformation. and also we didn't define the xcom push and pull then how it knows? can you clarify my doubt