ETL vs. ELT in Python

ETL (Extract, Transform, Load) and ELT (Extract, Load, Transform) are two data processing strategies used in data pipelines.

The key difference is **when and where the data transformation happens**.

---

1. ETL (Extract, Transform, Load)

- **Definition**: Data is extracted from source(s), transformed in an intermediary system (e.g., a Python script, Spark, or Airflow), and then loaded into the final destination (e.g., a database, data warehouse).
- **Use Case**: Suitable when the data needs preprocessing before loading, especially if the target system has limited processing power.

Python ETL Example

```python
import pandas as pd

# Extract
data = pd.read_csv("raw_data.csv")

# Transform
data['new_column'] = data['existing_column'].apply(lambda x: x * 2)  # Example transformation

# Load
data.to_sql('processed_table',
            con=database_connection,
            if_exists='replace', index=False)

```

Pros of ETL:

✔ Ensures only clean, structured data is stored
✔ Reduces the load on the data warehouse
✔ Better for structured and consistent data

Cons of ETL:

❌ Can be slow for large datasets
❌ Requires a separate transformation step before loading

---

2. ELT (Extract, Load, Transform)

- **Definition**: Data is extracted from source(s), loaded **as is** into a data warehouse (e.g., BigQuery, Snowflake, Redshift), and then transformed **inside the warehouse** using SQL or Python-based processing.
- **Use Case**: Best when working with **large, unstructured data** that needs scalable processing.

Python ELT Example

```python
import sqlalchemy

# Extract
data = pd.read_csv("raw_data.csv")

# Load (raw data stored in the data warehouse)
engine = sqlalchemy.create_engine("postgresql://user:password@host/dbname")
data.to_sql('raw_table', con=engine, if_exists='replace', index=False)

# Transform (SQL transformation inside the data warehouse)
with engine.connect() as conn:
    conn.execute("""
        INSERT INTO processed_table
        SELECT *, existing_column * 2 AS new_column FROM raw_table
    """)
```

Pros of ELT:

✔ More scalable for big data
✔ Uses powerful cloud-based transformations (SQL, dbt, Spark)
✔ Ideal for modern data lakes and real-time analytics

Cons of ELT:

❌ Requires a powerful data warehouse
❌ More complex permission and governance management

Which One to Use in Python?

- **Use ETL if**:
    - You need structured, cleaned data before loading.
    - Your target system has limited processing power.
    - You are working with smaller datasets in Pandas, Airflow, or Prefect.
- **Use ELT if**:
    - You're working with large datasets and cloud data warehouses.
    - You want to leverage Snowflake, BigQuery, or Redshift for transformation.
    - You're using dbt (data build tool) for SQL-based transformations.