Introduction to ETL and ELT Pipelines

ETL AND ELT IN PYTHON

Jake Roach
Data Engineer

datacamp

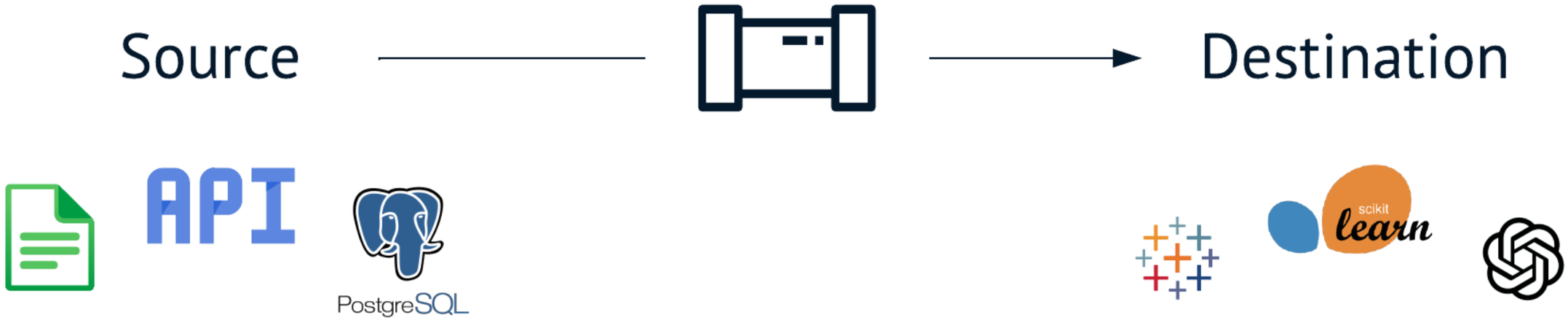
Business Intelligence

Machine Learning

Artificial Intelligence

Data pipelines

... are responsible for moving data from a source to a destination, and transforming it somewhere along the way.



ETL

- Extract, **transform**, load
- Traditional data pipeline design pattern
- Sources may be tabular or non-tabular
- Leverage Python with `pandas`

ELT

- Extract, **load**, transform
- More recent pattern
- Data warehouses
- Typically tabular data

Extract, transform, load (ETL)

```
def load(data_frame, target_table):
    # Some custom-built Python logic to load data to SQL
    data_frame.to_sql(name=target_table, con=POSTGRES_CONNECTION)
    print(f"Loading data to the {target_table} table")

# Now, run the data pipeline
extracted_data = extract(file_name="raw_data.csv")
transformed_data = transform(data_frame=extracted_data)
load(data_frame=transformed_data, target_table="cleaned_data")
```

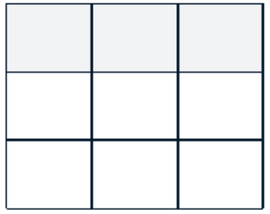
```
Extracting data from raw_data.csv
Transforming data to remove 'null' records
Loading data to the cleaned_data table
```

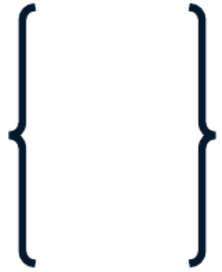
Extract, load, transform (ELT)

```
...
def transform(source_table, target_table):
    data_warehouse.run_sql("""
        CREATE TABLE {target_table} AS
          SELECT
              <field-name>, <field-name>, ...
          FROM {source_table};
    """)

# Similar to ETL pipelines, call the extract, load, and transform functions
extracted_data = extract(file_name="raw_data.csv")
load(data_frame=extracted_data, table_name="raw_data")
transform(source_table="raw_data", target_table="cleaned_data")
```

We'll also take a look at...





Tabular and
non-tabular data

pandas

- Data transformation
- Writing data to disk, SQL databases

- Unit-testing
- Monitoring
- Deploying to production

Let's practice!

ETL AND ELT IN PYTHON

Building ETL and ELT Pipelines

ETL AND ELT IN PYTHON

Jake Roach
Data Engineer

datacamp

Extract Data from a CSV File

```
import pandas as pd

# Read in the CSV file to a DataFrame
data_frame = pd.read_csv("raw_data.csv")
```

`read_csv()`

- Takes a file path, returns a DataFrame
- `delimiter`, `header`, `engine`

```
# Output the first few rows
data_frame.head()
```

`.head()`

- Outputs the first `n` number of a DataFrame

```
   name          num_firms   total_income
0  Advertising      58          3892.41
1  Apparel          39          5422.69
...
49 Trucking         35          17324.36
```

Filtering a DataFrame

```
   name          num_firms   total_income
0  Advertising      58          3892.41
1  Apparel          39          5422.69
...
49 Trucking         35          17324.36
```

```
      name          num_firms
1     Apparel          39
37    Apparel          61
```

```
# First, by rows
data_frame.loc[data_frame["name"] == "Apparel", :]

# Then, by columns
data_frame.loc[:, ["name", "num_firms"]]
```

`.loc`

- Filters a DataFrame
- `:` means "all"

Write a DataFrame to a CSV File

```
# Write a DataFrame to a .csv file
data_frame.to_csv("cleaned_data.csv")
```

`.to_csv()`

- Takes a `path`, creates DataFrame from file stored at that `path`
- Can take other parameters to customize the output

Other options, like:

`.to_json()`, `.to_excel()`, `.to_sql()`

Running SQL Queries

```
data_warehouse.execute(  # Use Python clients or other tools to run SQL queries
    """
    CREATE TABLE total_sales AS
        SELECT
            ds,
            SUM(sales)
        FROM raw_sales_data
        GROUP BY ds;
    """
)
```

- Tools like `.execute()` to run SQL queries

Putting it all together!

```
# Define extract(), transform(), and load() functions
...

def transform(data_frame, value):
    return data_frame.loc[data_frame["name"] == value, ["name", "num_firms"]]

# First, extract data from a .csv
extracted_data = extract(file_name="raw_data.csv")

# Then, transform the `extracted_data`
transformed_data = transform(data_frame=extracted_data, value="Apparel")

# Finally, load the `transformed_data`
load(data_frame=transformed_data, file_name="cleaned_data.csv")
```

Let's practice!

ETL AND ELT IN PYTHON