



This is a pet project for an 'analytics engineering' role at an energy company. Generally, analysts on the teams have to manually retrieve and clean data every quarter to understand changes in the sales and capability of different energy types. This process normally takes days and was something that most analysts dread. My job is to automate this process by building a data pipeline. I'll write this data pipeline to pull data each month, helping to provide more rapid insights and free up time for your data consumers.

I will achieve this using the `pandas` library and its powerful parsing features. I'll be working with two raw files; `electricity_sales.csv` and `electricity_capability_nested.json`.

Below is data dictionary for the `electricity_sales.csv` dataset, which I'll be transforming :

Field	Data Type
period	str
stateid	str
stateDescription	str
sectorid	str
sectorName	str
price	float
price-units	str

```
import pandas as pd
import json
```

```
def extract_tabular_data(file_path: str):
    """Extract data from a tabular file format, with pandas."""
    if file_path.endswith('.csv'):
        return pd.read_csv(file_path)
    else:
        raise ValueError("Unsupported file format. Please provide a .csv")

df_csv = extract_tabular_data("electricity_sales.csv")

print(df_csv.head())
```

	period	stateid	...	price	price-units
0	2023-12	HI	...	0.00	cents per kilowatt-hour
1	2023-12	ID	...	9.46	cents per kilowatt-hour
2	2023-12	ID	...	8.88	cents per kilowatt-hour
3	2023-12	ID	...	6.21	cents per kilowatt-hour
4	2023-12	ID	...	NaN	cents per kilowatt-hour

[5 rows x 7 columns]

```
def extract_json_data(file_path):
    """Extract and flatten data from a JSON file."""
    if file_path.endswith('.json'):
        with open(file_path, 'r') as file:
            data = json.load(file)
        return pd.json_normalize(data)
    else:
        raise ValueError("Unsupported file format. Please provide a .csv or .json file.")

df_json = extract_json_data("electricity_capability_nested.json")

print(df_json.head())
```

	period	stateId	...	energySource.capability	energySource.capabilityUnits
0	2023	WI	...	1442.6	megawatts
1	2023	WI	...	1442.6	megawatts
2	2023	WI	...	636.5	megawatts
3	2023	WI	...	124.6	megawatts
4	2023	WI	...	2140.9	megawatts

[5 rows x 7 columns]

```
def transform_electricity_sales_data(raw_data: pd.DataFrame):
    """
    Transform electricity sales to find the total amount of electricity sold
    in the residential and transportation sectors.

    To transform the electricity sales data, you'll need to do the following:
    - Drop any records with NA values in the `price` column. Do this inplace.
    - Only keep records with a `sectorName` of "residential" or "transportation".
    - Create a `month` column using the first 4 characters of the values in `period`.
    - Create a `year` column using the last 2 characters of the values in `period`.
    - Return the transformed `DataFrame`, keeping only the columns `year`, `month`, `stateid`, `price` and `price-units`.
    """
    # Drop records with NA values in the `price` column
    raw_data.dropna(subset=['price'], inplace=True)

    # Only keep records with a `sectorName` of "residential" or "transportation"
    raw_data = raw_data[raw_data['sectorName'].isin(['residential', 'transportation'])]

    # Create a `month` column using the first 4 characters of the values in `period`
    raw_data['month'] = raw_data['period'].str[:4]

    # Create a `year` column using the last 2 characters of the values in `period`
    raw_data['year'] = raw_data['period'].str[-2:]

    # Return the transformed DataFrame, keeping only the specified columns
    return raw_data[['year', 'month', 'stateid', 'price', 'price-units']]

# Example usage
raw_data = pd.read_csv('electricity_sales.csv')
transformed_data = transform_electricity_sales_data(raw_data)
print(transformed_data.head())
```

	year	month	stateid	price	price-units
0	12	2023	HI	0.00	cents per kilowatt-hour
5	12	2023	ID	11.53	cents per kilowatt-hour
6	12	2023	ID	0.00	cents per kilowatt-hour
11	12	2023	IL	14.84	cents per kilowatt-hour
12	12	2023	IL	7.29	cents per kilowatt-hour

```
# Test the functions
raw_electricity_capability_df = extract_json_data("electricity_capability_nested.json")
raw_electricity_sales_df = extract_tabular_data("electricity_sales.csv")

cleaned_electricity_sales_df = transform_electricity_sales_data(raw_electricity_sales_df)

load(raw_electricity_capability_df, "loaded__electricity_capability.parquet")
load(cleaned_electricity_sales_df, "loaded__electricity_sales.csv")
```