Solutions - Problem Set 1

1. Pivoting

Easy

Solution: The pivoted DataFrame will look like this:

```
result = df.pivot(index='Date', columns='Product',
    ↪ values='Sales')
```

Output:

```
Product          A      B
Date
2024-11-01   100.0  200.0
2024-11-02   150.0    NaN
```

—

Medium

Solution: The pivot for both `Temperature` and `Humidity`:

```
result = df.pivot(index='City', columns='Date',
    ↪ values=['Temperature', 'Humidity'])
```

Output:

```
                Temperature                 Humidity
Date         2024-11-01 2024-11-02 2024-11-01 2024-11-02
City
A                    20         22         60         65
B                    25         26         70         72
```

—

Hard

Solution: Using `pivot_table` to calculate the averages:

```
result = df.pivot_table(index='City', columns='Date'
    ↪ , aggfunc='mean')
```

Output:

	Humidity		Temperature	
Date	2024-11-01	2024-11-02	2024-11-01	2024-11-02
City				
A	60	65	20	22
B	70	72	25	26

—

2. Melting

Easy

Solution: Unpivot the DataFrame to long format:

```
result = df.reset_index().melt(id_vars='City',
    ↪ var_name='Date', value_name='Temperature')
```

Output:

	City	Date	Temperature
0	A	2024-11-01	20
1	B	2024-11-01	25
2	A	2024-11-02	22
3	B	2024-11-02	26

—

Medium

Solution: Melt the DataFrame to extract the categories:

```
result = df.melt(id_vars='City', var_name='Category'
    ↪ , value_name='Value')
```

Output:

```
  City             Category  Value
0    A    Temp_2024-11-01     20
1    B    Temp_2024-11-01     25
2    A    Temp_2024-11-02     22
3    B    Temp_2024-11-02     26
4    A  Humidity_2024-11-01     60
...
```

Hard

Solution: Split the `Category` column into `Measurement` and `Date`:

```
df[['Measurement', 'Date']] = df['Category'].str.
    ↪ split('_', expand=True)
```

Output:

```
  City Measurement        Date  Value
0    A        Temp  2024-11-01     20
1    B        Temp  2024-11-01     25
2    A        Temp  2024-11-02     22
3    B        Temp  2024-11-02     26
4    A    Humidity  2024-11-01     60
...
```

---

3. Stacking/Unstacking

Easy

Solution: Stack the DataFrame:

```
result = df.stack()
```

Output:

```
City    Date
A       2024-11-01    20
        2024-11-02    22
B       2024-11-01    25
        2024-11-02    26
dtype: int64
```

—

Medium

Solution: Unstack the DataFrame to move `Date` as columns:

```
result = df.unstack(level='Date')
```

Output:

```
          Humidity                 Temperature
Date    2024-11-01 2024-11-02    2024-11-01 2024-11-02
City
A               60         65            20         22
B               70         72            25         26
```

—

Hard

Solution: Stack and reset the DataFrame to long format:

```
result = df.stack().reset_index(name='Value')
```

Output:

```
  City        Date  Measurement  Value
0    A  2024-11-01   Humidity      60
1    A  2024-11-01   Temperature   20
...
```

—

4. Method Chaining

Easy

Solution: Filter rows where `Sales > 150`:

```
result = df[df['Sales'] > 150].reset_index(drop=True
    ↪ )
```

Output:

```
  City        Date  Sales
0    B  2024-11-01    200
1    B  2024-11-02    250
```

—

Medium

Solution: Calculate total `Sales` per `City`:

```
result = df.groupby('City')['Sales'].sum().
    ↪ reset_index()
```

Output:

```
  City  Sales
0    A    250
1    B    450
```

—

Hard

Solution: Calculate and sort total `Sales` per `City`:

```
result = df.groupby('City')['Sales'].sum().
    ↪ reset_index().sort_values(by='Sales', ascending
    ↪ =False)
```

Output:

```
  City  Sales
0    B    450
1    A    250
```

—