Download DataFrame DataCamp

```
# Melt the data to long format
df_melted = df.melt(id_vars=["country_name", "continent"],
                    value_vars=[str(year) for year in range(2010, 2022)],
                    var_name="year", value_name="unemployment_rate")

# Set the figure size and style
plt.figure(figsize=(14, 7))
sns.boxplot(data=df_melted, x="year", y="unemployment_rate", hue="continent")

# Customize the plot
plt.title("Unemployment Rate Variation by Year for Each Continent")
plt.xlabel("Year")
plt.ylabel("Unemployment Rate (%)")
plt.legend(title="Continent", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.grid(True, axis='y')

plt.show()
```

in this code not understand the following part -

```
df_melted = df.melt(id_vars=["country_name", "continent"],
                    value_vars=[str(year) for year in range(2010, 2022)],
                    var_name="year", value_name="unemployment_rate")
```

Great question! Let's break down that `df.melt(...)` line in simple terms.

🧠 What does `.melt()` do?

The `.melt()` function **reshapes a DataFrame** from wide format to long format.

🔍 Your original DataFrame looks like this (wide format):

country_name	continent	2010	2011	2012	...	2021
India	Asia	5.3	5.1	5.4	...	7.2
Brazil	America	6.1	6.4	6.8	...	9.1

Each year is a separate column — this is *wide* format.

✅ After using `melt`, it becomes (long format):

country_name	continent	year	unemployment_rate
India	Asia	2010	5.3
India	Asia	2011	5.1
...	...	...	...
Brazil	America	2021	9.1

Now each **row** represents the unemployment rate for **one country in one year**, which is much better for plotting and analysis.

🔍 Breaking down the melt line:

```python
df_melted = df.melt(
    id_vars=["country_name", "continent"],  # Columns to keep as they are
    value_vars=[str(year) for year in range(2010, 2022)],  # Columns to unpivot (years)
    var_name="year",  # New column name for the years
    value_name="unemployment_rate"  # New column name for the rates
)
```

- `id_vars` : These columns stay fixed (won’t be melted). We still want to know which country and continent each row belongs to.
- `value_vars` : The year columns that will be “unpivoted” — each value becomes a row.
- `var_name` : Name of the new column that will store the years.
- `value_name` : Name of the new column that will store the unemployment rates.

Why this helps for plotting:

Most plotting libraries (like Seaborn) work best with **long-format data**, where each row represents one observation.

Would you like me to show how this transformation looks visually with example rows?

to wide to long , i think this process add more rows

Yes — you're **absolutely right**! Converting from **wide to long** format using `.melt()` **increases the number of rows** in your DataFrame. Here's how and why:

🎯 Example:

Suppose you start with this wide-format DataFrame:

country_name	continent	2010	2011	2012
India	Asia	5.3	5.1	5.4
Brazil	America	6.1	6.4	6.8

That’s:

- **2 rows** (countries)
- **3 year columns**

Total data points = **2 countries × 3 years = 6 values**

After `.melt()` (long format):

country_name	continent	year	unemployment_rate
India	Asia	2010	5.3
India	Asia	2011	5.1
India	Asia	2012	5.4
Brazil	America	2010	6.1
Brazil	America	2011	6.4
Brazil	America	2012	6.8

Now you have:

- **6 rows** (1 per country-year pair)
- Only 1 value column: `unemployment_rate`

✅ Summary:

- Yes, **rows increase**.
- But **columns decrease**.
- This makes the data **tidy and easier to plot/analyze** — especially for time series or categorical plots.

Would you like to practice this on your own small example to try it yourself?