Polars is extremely fast and memory-efficient, but **Pandas still has advantages in certain areas**, mainly because it has been around longer and has a **very mature ecosystem**.

**1 Pandas advantages / “Pandas-only functions”**

1. **Extensive I/O support**
   * Pandas can read/write almost every format out-of-the-box: CSV, Excel, JSON, HTML tables, SQL, Parquet, Feather, Stata, SAS, Google BigQuery, etc.
   * Polars supports CSV, Parquet, and JSON mainly. Excel support in Polars is limited—you usually need openpyxl or convert via Pandas.
2. **Time series functionality**
   * Pandas has a rich **time series toolkit**: resampling, rolling windows, period/frequency conversions, datetime indexing, timezone handling, etc.
   * Polars can handle basic datetime types but is not yet as comprehensive.
3. **Advanced string and categorical methods**
   * Pandas has a large set of **.str methods** for string manipulation (regex, contains, extract, slicing).
   * Polars also supports strings, but some operations are not as extensive or user-friendly.
4. **Integration with the Python ecosystem**
   * Many libraries (like **statsmodels**, **scikit-learn**, **seaborn**, **matplotlib**) expect Pandas DataFrames.
   * Polars often needs conversion back to Pandas for compatibility.
5. **Indexing and multi-index**
   * Pandas allows **hierarchical/multi-level indexing** which is useful for complex pivot tables and grouped operations.
   * Polars does not have multi-indexing.
6. **Pivot tables and cross-tabulations**
   * Pandas has built-in pivot\_table and crosstab with flexible aggregation functions.
   * Polars can mimic this, but the syntax and flexibility are more limited.
7. **Mature community and documentation**
   * Pandas has a **large user base**, lots of tutorials, and many edge-case solutions.
   * Polars is newer, so fewer community resources exist.

**2 When Polars is better**

* Very large datasets that **don’t fit in memory** or need **multi-threaded processing**.
* Columnar operations, filtering, grouping, and aggregations that are **compute-heavy**.
* Lazy evaluation for building **query pipelines** efficiently.

**In short:**

* Use **Polars** for speed, memory efficiency, and large-scale numeric data.
* Use **Pandas** when you need **time series, Excel/SQL integration, pivot tables, multi-indexing, or compatibility with other Python libraries**.