Polars

This guide covers installing Polars and core functionality like reading/writing data, using expressions to select, filter, add columns, group data, and joining/concatenating DataFrames. Do to the high performance nature of Polars many projects are interested in using Polars which is written in Rust.

Results including reading parquet (lower is better)

Key Points:

- Polars supports reading/writing many data formats like CSV and Parquet
- Expressions allow modular DataFrame transformations
- Filter, select, group_by, join, and concat combine to form complex workflows

Reflection Questions:

- What file formats does Polars support reading and writing?
- When might you use join versus concat to combine DataFrames?
- How could expressions modularly build up a complex DataFrame pipeline?
- In what ways can filtered or aggregated data provide insights?
- How might Polars scale compared to Pandas for large data?

Challenges:

- Read in JSON data and write to Parquet format
- Join two datasets then perform group_by aggregation
- Select subset of columns, filter rows, and add new column
- Build expression to find correlations between columns
- Use Polars for analysis that exceeds Pandas capabilities

```python
import polars as pl

# Create DataFrame
data = [{"fruit": "apple", "count": 10, "price": 0.50},
        {"fruit": "banana", "count": 20, "price": 0.25}]
df = pl.from_dicts(data)

# Expressions to select, filter, aggregate
sel = df.select(["fruit", "count"]) # Select columns
filt = sel.filter(pl.col("fruit") == "apple") # Filter rows
agg = filt.groupby("fruit").agg(pl.col("count").sum()) # Aggregate

print(agg)
```