# Polars cheat sheet

Open in Colab

## General

### Install

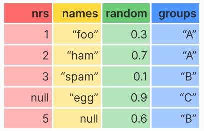
```
pip install polars
```

### Import

```
import polars as pl
```

## Creating/reading DataFrames

Create DataFrame



| nrs | names | random | groups |
|---|---|---|---|
| 1 | "foo" | 0.3 | "A" |
| 2 | "ham" | 0.7 | "A" |
| 3 | "spam" | 0.1 | "B" |
| null | "egg" | 0.9 | "C" |
| 5 | null | 0.6 | "B" |

```
df = pl.DataFrame(
  {
    "nrs": [1, 2, 3, None, 5],
    "names": ["foo", "ham", "spam", "egg", None],
    "random": [0.3, 0.7, 0.1, 0.9, 0.6],
    "groups": ["A", "A", "B", "C", "B"],
  }
)
```

Read CSV

```
df = pl.read_csv("https://j.mp/iriscsv",
                 has_header=True)
```

Read parquet

```
df = pl.read_parquet("path.parquet")
```

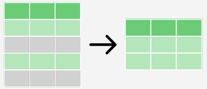
## Expressions

Polars expressions can be performed in sequence. This improves readability of code.

```
df \
  .filter(pl.col("nrs") < 4) \
  .groupby("groups") \
  .agg(
    pl \
      .all() \
      .sum()
)
```

by Franz Diebold (diebold.io)

## Subset Observations - rows



Filter: Extract rows that meet logical criteria.

```
df.filter(pl.col("random") > 0.5)
df.filter(
  (pl.col("groups") == "B")
  & (pl.col("random") > 0.5)
)
```

Sample

```
# Randomly select fraction of rows.
df.sample(frac=0.5)

# Randomly select n rows.
df.sample(n=2)
```

Select first and last rows

```
# Select first n rows
df.head(n=2)

# Select last n rows.
df.tail(n=2)
```

## Subset Variables - columns

Select multiple columns with specific names

```
df.select(["nrs", "names"])
```

Select columns whose name matches regex

```
df.select(pl.col("^n.*$"))
```

## Subsets - rows and columns

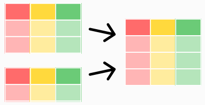
Select rows 2-4

```
df[2:4, :]
```

Select columns in positions 1 and 3 (first column is 0)

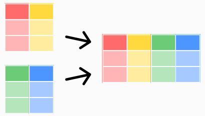
```
df[:, [1, 3]]
```

## Reshaping Data – Change layout, sorting, renaming



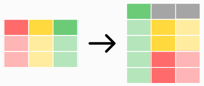
Append rows of DataFrames

```
pl.concat([df, df2])
```



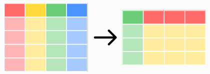
Append columns of DataFrames

```
pl.concat([df, df3], how="horizontal")
```



Gather columns into rows

```
df.melt(
  id_vars="nrs",
  value_vars=["names", "groups"]
)
```



Spread rows into columns

```
df.pivot(values="nrs", index="groups",
         columns="names")
```

Order rows by values of a column

```
# low to high
df.sort("random")

# high to low
df.sort("random", reverse=True)
```

Rename the columns of a DataFrame

```
df.rename({"nrs": "idx"})
```

Drop columns from DataFrame

```
df.drop(["names", "random"])
```

## Summarize Data

Count number of rows with each unique value of variable

```
df["groups"].value_counts()
```

# of rows in DataFrame

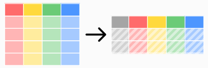
```
len(df)
# or
df.height
```

Tuple of # of rows, # of columns in DataFrame

```
df.shape
```

# of distinct values in a column

```
df["groups"].n_unique()
```



Basic descriptive and statistics for each column

```
df.describe()
```

Aggregation functions

```
df.select(
    [
        # Sum values
        pl.sum("random").alias("sum"),

        # Minimum value
        pl.min("random").alias("min"),

        # Maximum value
        pl.max("random").alias("max"),
        # or
        pl.col("random").max().alias("other_max")

        # Standard deviation
        pl.std("random").alias("std dev"),

        # Variance
        pl.var("random").alias("variance"),

        # Median
        pl.median("random").alias("median"),

        # Mean
        pl.mean("random").alias("mean"),

        # Quantile
        pl.quantile("random", 0.75) \
          .alias("quantile_0.75"),
        # or
        pl.col("random").quantile(0.75) \
          .alias("other_quantile_0.75"),
    ]
)
```