Polars 치트시트

Polars란?

Polars는 Rust로 빌드된 초고속 데이터프레임 라이브러리입니다. 멀티코어 프로세싱을 활용하여 Pandas보다 훨씬 빠른 성능을 제공하며, 직관적인 API를 가지고 있습니다.

```
import polars as pl
```

데이터 생성 및 읽기

DataFrame 생성

- 딕셔너리로부터 생성:
  ```
  data = {'A': [1, 2, 3], 'B': [4, 5, 6]}
  df = pl.DataFrame(data)
  ```
- NumPy 배열로부터 생성: `df = pl.DataFrame(np.random.rand(4, 3), columns=['a', 'b', 'c'])`

파일 읽기

- CSV 읽기: `pl.read_csv('data.csv')`
- Parquet 읽기: `pl.read_parquet('data.parquet')`
- JSON 읽기: `pl.read_json('data.json')`

표현식 (Expressions)

Polars의 핵심 개념. 컬럼에 대한 변환 작업을 정의하며, 지연 평가(Lazy Evaluation)와 최적화를 가능하게 합니다.

- 컬럼 선택: `pl.col('column_name')`
- 모든 컬럼 선택: `pl.all()`
- 여러 컬럼 선택: `pl.col(['col1', 'col2'])`
- 정규식으로 컬럼 선택: `pl.col("^temp_.*$")`

데이터 탐색 및 선택

탐색

- `df.head(n)`: 처음 n개 행
- `df.tail(n)`: 마지막 n개 행
- `df.describe()`: 기술 통계
- `df.shape`: (행, 열) 개수
- `df.columns`: 컬럼 이름
- `df.dtypes`: 컬럼별 데이터 타입

선택 (`.select()`)

`select` 메서드 안에 표현식을 사용하여 데이터를 선택하고 조작합니다.

```
df.select([
    pl.col("A"), # A 컬럼 선택
    pl.col("B").sort(descending=True), # B 컬럼 내림차순 정렬
    (pl.col("A") + pl.col("B")).alias("A+B") # 새 컬럼 생성
])
```

필터링 (`.filter()`)

```
df.filter(pl.col("A") > 2)
df.filter((pl.col("A") > 2) & (pl.col("B") < 6))
```

데이터 조작

새 컬럼 추가 (`.with_columns()`)

```
df.with_columns([
    (pl.col("A") * 2).alias("A*2"),
    pl.lit(5).alias("five") # 리터럴 값으로 새 컬럼
])
```

GroupBy 및 집계 (`.group_by().agg()`)

```
df.group_by("category").agg([

pl.sum("values").alias("sum_values"),

pl.mean("values").alias("mean_values"),
    pl.count().alias("count")
])
```

데이터 결합 (Joins)

```
df1.join(df2, on="key_column", how="inner")
```

- `how`: `inner`, `left`, `outer`, `semi`, `anti`

지연 평가 (Lazy API)

작업을 즉시 실행하지 않고, 최적화된 실행 계획을 세운 후 `.collect()`가 호출될 때 한 번에 실행합니다. 대용량 데이터 처리 시 매우 효율적입니다.

```
lazy_df = pl.read_csv("data.csv").lazy()

result = (
    lazy_df
    .filter(pl.col("A") > 5)
    .group_by("category")
    .agg(pl.sum("B"))
).collect()
```

유용한 표현식

- `when-then-otherwise` (if-else):
  ```
  pl.when(pl.col("A") > 5).then(pl.lit("high")).otherwise(pl.lit("low"))
  ```
- 윈도우 함수 (Window Functions): `pl.col("value").sum().over("category")`
- 문자열 함수:
  - `pl.col("str_col").str.contains("pattern")`
  - `pl.col("str_col").str.replace("a", "b")`
- 날짜/시간 함수:
  - `pl.col("date_col").dt.year()`
  - `pl.col("date_col").dt.strftime("%Y-%m-%d")`

데이터 쓰기

- CSV로 저장: `df.write_csv('output.csv')`
- Parquet으로 저장: `df.write_parquet('output.parquet')`