LOAD BIG DATA EFFICIENTLY

PART 4: PREDICATE PUSHDOWN BOOSTING YOUR PERFORMANCE

- ***What is Predicate Pushdown***
- ***How does it work in the Spark UI***

{json}

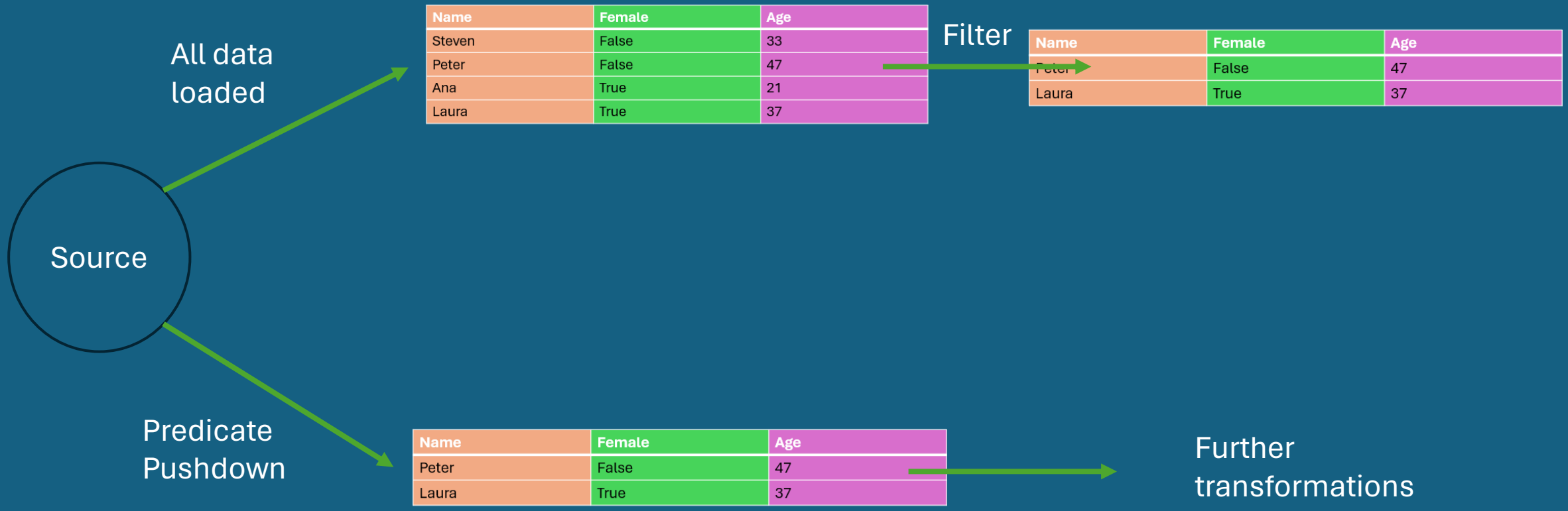
Predicate Pushdown

Name	Female	Age
Steven	False	33
Peter	False	47
Ana	True	21
Laura	True	37

Goal: Get all data WHERE Age > 35

```
sdf.filter(f.col("Age") > 35)
```

Predicate Pushdown



Name	Female	Age
Steven	False	33
Peter	False	47
Ana	True	21
Laura	True	37

Name	Female	Age
[illegible]	False	47
Laura	True	37

Name	Female	Age
Peter	False	47
Laura	True	37

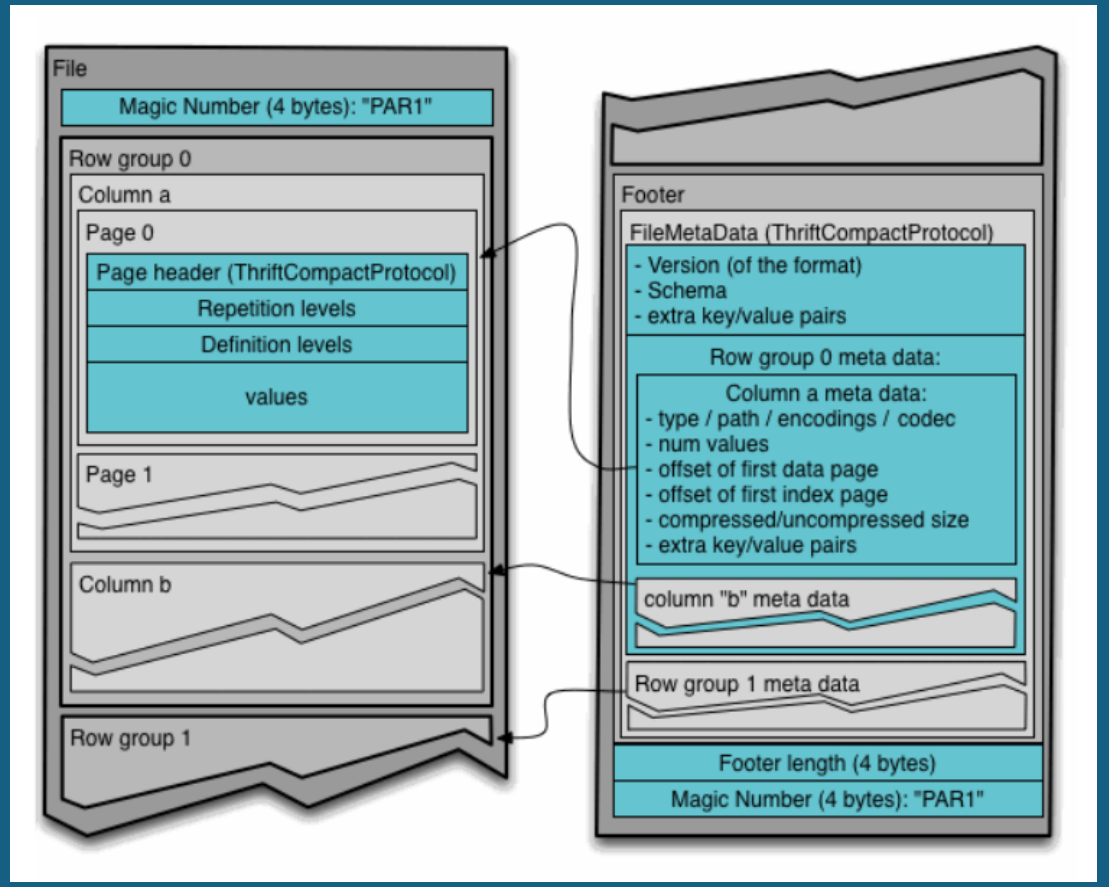
Predicate Pushdown

- **Predicate Pushdown is an optimization technique filtering data at the source and often relies on statistics**

- **Benefits:**
  - Less I/O meaning less data to load
  - Less memory usage
  - Faster queries

- **Parquet supports Predicate Pushdown using statistics saved in meta data footer**
- **Since Spark 3.1.0 also possible on Avro, CSV, JSON**

- Row Groups are a logical division on row level of a parquet defaulting to 128 MB
- Column part relates to column chunk of row groups
- Pages are invisible units where the encoding and compression happens
- Footer containing file metadata which can be used for predicate pushdown:
  - File level: num rows/ columns, schema
  - Row group: num rows/ columns
  - Column level: min, max, null count, distinct value counts, page indexes etc.



Parquet format: Documentation | Parquet (apache.org)
Parquet Page Indexes: Speeding Up SELECT Queries with Parquet Page Indexes - Cloudera Blog

Load time and output rows for column/ row filters

Format	Load all data	Column filter	Row filter
JSON	16 s (10,000,000 rows)	5 s (10,000,000 rows)	4 s (300 rows)
CSV	13 s (10,000,000 rows))	4 s (10,000,000 rows)	4 s (308 rows)
PARQUET	2 s (10,000,000 rows)	0.6 s (10,000,000 rows)	0.1 s (20,000 rows)
AVRO	3 s (10,000,000 rows)	2 s (10,000,000 rows)	1 s (300 rows)

Summary

- Filter and select statements close to the source reduce data load
- Predicate Pushdown is more efficient with Parquet
- The structure/grouping of your data e.g. repartitioning during writes has (in Parquet also the order of data)