Query Plans

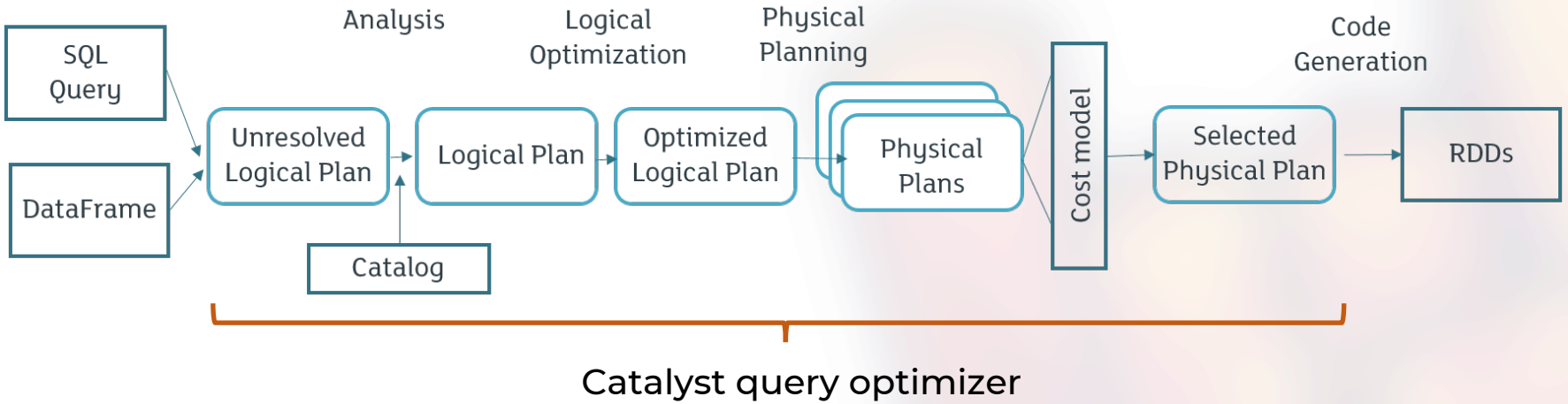
Objective

Understand how Spark "compiles" a SQL/DataFrame job

Read query plans

(later) Predict job performance based on query plans

How a SQL Job Runs



When you run a SQL job

- Spark knows the DF dependencies in advance – unresolved logical transformation plan
- Catalyst resolves references and expression types – resolved logical plan
- Catalyst compresses and pattern matches on the plan tree – optimized logical plan
- Catalyst generates physical execution plans

How to Read Query Plans

```
val ds1 = spark.range(1, 10000000)
val ds2 = spark.range(1, 100000000, 2)
val ds3 = ds1.repartition(7)
val ds4 = ds2.repartition(9)
val ds5 = ds3.selectExpr("id * 3 as id")
val joined = ds5.join(ds4, "id")
val sum = joined.selectExpr("sum(id)")
sum.explain()
```

How to Read Query Plans

```
val ds1 = spark.range(1, 100000000)
val ds2 = spark.range(1, 100000000, 2)
val ds3 = ds1.repartition(7)
val ds4 = ds2.repartition(9)
val ds5 = ds3.selectExpr("id * 3 as id")
val joined = ds5.join(ds4, "id")
val sum = joined.selectExpr("sum(id)")
sum.explain()
```

```
== Physical Plan ==
*(7) HashAggregate(keys=[], functions=[sum(id#73L)])
+- Exchange SinglePartition
   +- *(6) HashAggregate(keys=[], functions=[partial_sum(id#73L)])
      +- *(6) Project [id#73L]
         +- *(6) SortMergeJoin [id#73L], [id#69L], Inner
            :- *(3) Sort [id#73L ASC NULLS FIRST], false, 0
            :  +- Exchange hashpartitioning(id#73L, 200)
            :     +- *(2) Project [(id#67L * 3) AS id#73L]
            :        +- Exchange RoundRobinPartitioning(7)
            :           +- *(1) Range (1, 10000000, step=1, splits=12)
            +- *(5) Sort [id#69L ASC NULLS FIRST], false, 0
               +- Exchange hashpartitioning(id#69L, 200)
                  +- Exchange RoundRobinPartitioning(9)
                     +- *(4) Range (1, 100000000, step=2, splits=12)
```

How to Read Query Plans

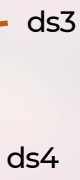
```
val ds1 = spark.range(1, 100000000)
val ds2 = spark.range(1, 100000000, 2)
val ds3 = ds1.repartition(7)
val ds4 = ds2.repartition(9)
val ds5 = ds3.selectExpr("id * 3 as id")
val joined = ds5.join(ds4, "id")
val sum = joined.selectExpr("sum(id)")
sum.explain()
```

```
== Physical Plan ==
*(7) HashAggregate(keys=[], functions=[sum(id#73L)])
+- Exchange SinglePartition
   +- *(6) HashAggregate(keys=[], functions=[partial_sum(id#73L)])
      +- *(6) Project [id#73L]
         +- *(6) SortMergeJoin [id#73L], [id#69L], Inner
            :- *(3) Sort [id#73L ASC NULLS FIRST], false, 0
            :  +- Exchange hashpartitioning(id#73L, 200)
            :     +- *(2) Project [(id#67L * 3) AS id#73L]
            :        +- Exchange RoundRobinPartitioning(7)
            :           +- *(1) Range (1, 10000000, step=1, splits=12)
            +- *(5) Sort [id#69L ASC NULLS FIRST], false, 0
               +- Exchange hashpartitioning(id#69L, 200)
                  +- Exchange RoundRobinPartitioning(9)
                     +- *(4) Range (1, 100000000, step=2, splits=12)
```

How to Read Query Plans

```scala
val ds1 = spark.range(1, 100000000)
val ds2 = spark.range(1, 100000000, 2)
val ds3 = ds1.repartition(7)
val ds4 = ds2.repartition(9)
val ds5 = ds3.selectExpr("id * 3 as id")
val joined = ds5.join(ds4, "id")
val sum = joined.selectExpr("sum(id)")
sum.explain()
```

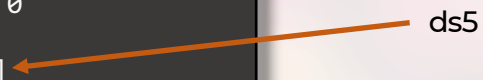
```
== Physical Plan ==
*(7) HashAggregate(keys=[], functions=[sum(id#73L)])
+- Exchange SinglePartition
   +- *(6) HashAggregate(keys=[], functions=[partial_sum(id#73L)])
      +- *(6) Project [id#73L]
         +- *(6) SortMergeJoin [id#73L], [id#69L], Inner
            :- *(3) Sort [id#73L ASC NULLS FIRST], false, 0
            :  +- Exchange hashpartitioning(id#73L, 200)
            :     +- *(2) Project [(id#67L * 3) AS id#73L]
            :        +- Exchange RoundRobinPartitioning(7)
            :           +- *(1) Range (1, 10000000, step=1, splits=12)
            +- *(5) Sort [id#69L ASC NULLS FIRST], false, 0
               +- Exchange hashpartitioning(id#69L, 200)
                  +- Exchange RoundRobinPartitioning(9)
                     +- *(4) Range (1, 100000000, step=2, splits=12)
```



How to Read Query Plans

```
val ds1 = spark.range(1, 100000000)
val ds2 = spark.range(1, 100000000, 2)
val ds3 = ds1.repartition(7)
val ds4 = ds2.repartition(9)
val ds5 = ds3.selectExpr("id * 3 as id")
val joined = ds5.join(ds4, "id")
val sum = joined.selectExpr("sum(id)")
sum.explain()
```

```
== Physical Plan ==
*(7) HashAggregate(keys=[], functions=[sum(id#73L)])
+- Exchange SinglePartition
   +- *(6) HashAggregate(keys=[], functions=[partial_sum(id#73L)])
      +- *(6) Project [id#73L]
         +- *(6) SortMergeJoin [id#73L], [id#69L], Inner
            :- *(3) Sort [id#73L ASC NULLS FIRST], false, 0
            :  +- Exchange hashpartitioning(id#73L, 200)
            :     +- *(2) Project [(id#67L * 3) AS id#73L]
            :        +- Exchange RoundRobinPartitioning(7)
            :           +- *(1) Range (1, 10000000, step=1, splits=12)
            +- *(5) Sort [id#69L ASC NULLS FIRST], false, 0
               +- Exchange hashpartitioning(id#69L, 200)
                  +- Exchange RoundRobinPartitioning(9)
                     +- *(4) Range (1, 100000000, step=2, splits=12)
```



How to Read Query Plans

```
val ds1 = spark.range(1, 100000000)
val ds2 = spark.range(1, 100000000, 2)
val ds3 = ds1.repartition(7)
val ds4 = ds2.repartition(9)
val ds5 = ds3.selectExpr("id * 3 as id")
val joined = ds5.join(ds4, "id")
val sum = joined.selectExpr("sum(id)")
sum.explain()
```

```
== Physical Plan ==
*(7) HashAggregate(keys=[], functions=[sum(id#73L)])
+- Exchange SinglePartition
   +- *(6) HashAggregate(keys=[], functions=[partial_sum(id#73L)])
      +- *(6) Project [id#73L]
         +- *(6) SortMergeJoin [id#73L], [id#69L], Inner
            :- *(3) Sort [id#73L ASC NULLS FIRST], false, 0
            :  +- Exchange hashpartitioning(id#73L, 200)
            :     +- *(2) Project [(id#67L * 3) AS id#73L]
            :        +- Exchange RoundRobinPartitioning(7)
            :           +- *(1) Range (1, 10000000, step=1, splits=12)
            +- *(5) Sort [id#69L ASC NULLS FIRST], false, 0
               +- Exchange hashpartitioning(id#69L, 200)
                  +- Exchange RoundRobinPartitioning(9)
                     +- *(4) Range (1, 100000000, step=2, splits=12)
```

Preparation for join:
a shuffle and a sort

(why?)

How to Read Query Plans

```scala
val ds1 = spark.range(1, 100000000)
val ds2 = spark.range(1, 100000000, 2)
val ds3 = ds1.repartition(7)
val ds4 = ds2.repartition(9)
val ds5 = ds3.selectExpr("id * 3 as id")
val joined = ds5.join(ds4, "id")
val sum = joined.selectExpr("sum(id)")
sum.explain()
```

```
== Physical Plan ==
*(7) HashAggregate(keys=[], functions=[sum(id#73L)])
+- Exchange SinglePartition
   +- *(6) HashAggregate(keys=[], functions=[partial_sum(id#73L)])
      +- *(6) Project [id#73L]
         +- *(6) SortMergeJoin [id#73L], [id#69L], Inner
            :- *(3) Sort [id#73L ASC NULLS FIRST], false, 0
            :  +- Exchange hashpartitioning(id#73L, 200)
            :     +- *(2) Project [(id#67L * 3) AS id#73L]
            :        +- Exchange RoundRobinPartitioning(7)
            :           +- *(1) Range (1, 10000000, step=1, splits=12)
            +- *(5) Sort [id#69L ASC NULLS FIRST], false, 0
               +- Exchange hashpartitioning(id#69L, 200)
                  +- Exchange RoundRobinPartitioning(9)
                     +- *(4) Range (1, 100000000, step=2, splits=12)
```

joined

How to Read Query Plans

```
val ds1 = spark.range(1, 100000000)
val ds2 = spark.range(1, 100000000, 2)
val ds3 = ds1.repartition(7)
val ds4 = ds2.repartition(9)
val ds5 = ds3.selectExpr("id * 3 as id")
val joined = ds5.join(ds4, "id")
val sum = joined.selectExpr("sum(id)")
sum.explain()
```

```
== Physical Plan ==
*(7) HashAggregate(keys=[], functions=[sum(id#73L)])
+- Exchange SinglePartition
   +- *(6) HashAggregate(keys=[], functions=[partial_sum(id#73L)])
      +- *(6) Project [id#73L]
         +- *(6) SortMergeJoin [id#73L], [id#69L], Inner
            :- *(3) Sort [id#73L ASC NULLS FIRST], false, 0
            :  +- Exchange hashpartitioning(id#73L, 200)
            :     +- *(2) Project [(id#67L * 3) AS id#73L]
            :        +- Exchange RoundRobinPartitioning(7)
            :           +- *(1) Range (1, 10000000, step=1, splits=12)
            +- *(5) Sort [id#69L ASC NULLS FIRST], false, 0
               +- Exchange hashpartitioning(id#69L, 200)
                  +- Exchange RoundRobinPartitioning(9)
                     +- *(4) Range (1, 100000000, step=2, splits=12)
```

keeps only one of the id columns

How to Read Query Plans

```scala
val ds1 = spark.range(1, 100000000)
val ds2 = spark.range(1, 100000000, 2)
val ds3 = ds1.repartition(7)
val ds4 = ds2.repartition(9)
val ds5 = ds3.selectExpr("id * 3 as id")
val joined = ds5.join(ds4, "id")
val sum = joined.selectExpr("sum(id)")
sum.explain()
```

```
== Physical Plan ==
*(7) HashAggregate(keys=[], functions=[sum(id#73L)])
+- Exchange SinglePartition
   +- *(6) HashAggregate(keys=[], functions=[partial_sum(id#73L)])
      +- *(6) Project [id#73L]
         +- *(6) SortMergeJoin [id#73L], [id#69L], Inner
            :- *(3) Sort [id#73L ASC NULLS FIRST], false, 0
            :  +- Exchange hashpartitioning(id#73L, 200)
            :     +- *(2) Project [(id#67L * 3) AS id#73L]
            :        +- Exchange RoundRobinPartitioning(7)
            :           +- *(1) Range (1, 10000000, step=1, splits=12)
            +- *(5) Sort [id#69L ASC NULLS FIRST], false, 0
               +- Exchange hashpartitioning(id#69L, 200)
                  +- Exchange RoundRobinPartitioning(9)
                     +- *(4) Range (1, 100000000, step=2, splits=12)
```

sum:

- sum per partition
- shuffle (bring partial results)
- sum all

How to Read Query Plans

```
val ds1 = spark.range(1, 100000000)
val ds2 = spark.range(1, 100000000, 2)
val ds3 = ds1.repartition(7)
val ds4 = ds2.repartition(9)
val ds5 = ds3.selectExpr("id * 3 as id")
val joined = ds5.join(ds4, "id")
val sum = joined.selectExpr("sum(id)")
sum.explain()
```

```
== Physical Plan ==
*(7) HashAggregate(keys=[], functions=[sum(id#73L)])
+- Exchange SinglePartition
   +- *(6) HashAggregate(keys=[], functions=[partial_sum(id#73L)])
      +- *(6) Project [id#73L]
         +- *(6) SortMergeJoin [id#73L], [id#69L], Inner
            :- *(3) Sort [id#73L ASC NULLS FIRST], false, 0
            :  +- Exchange hashpartitioning(id#73L, 200)
            :     +- *(2) Project [(id#67L * 3) AS id#73L]
            :        +- Exchange RoundRobinPartitioning(7)
            :           +- *(1) Range (1, 10000000, step=1, splits=12)
            +- *(5) Sort [id#69L ASC NULLS FIRST], false, 0
               +- Exchange hashpartitioning(id#69L, 200)
                  +- Exchange RoundRobinPartitioning(9)
                     +- *(4) Range (1, 100000000, step=2, splits=12)
```

How many stages?

How many shuffles?

In which order?

How to Read Query Plans

```
val ds1 = spark.range(1, 100000000)
val ds2 = spark.range(1, 100000000, 2)
val ds3 = ds1.repartition(7)
val ds4 = ds2.repartition(9)
val ds5 = ds3.selectExpr("id * 3 as id")
val joined = ds5.join(ds4, "id")
val sum = joined.selectExpr("sum(id)")
sum.explain()
```

```
== Physical Plan ==
*(7) HashAggregate(keys=[], functions=[sum(id#73L)])
+- Exchange SinglePartition
   +- *(6) HashAggregate(keys=[], functions=[partial_sum(id#73L)])
      +- *(6) Project [id#73L]
         +- *(6) SortMergeJoin [id#73L], [id#69L], Inner
            :- *(3) Sort [id#73L ASC NULLS FIRST], false, 0
            :  +- Exchange hashpartitioning(id#73L, 200)
            :     +- *(2) Project [(id#67L * 3) AS id#73L]
            :        +- Exchange RoundRobinPartitioning(7)
            :           +- *(1) Range (1, 10000000, step=1, splits=12)
            +- *(5) Sort [id#69L ASC NULLS FIRST], false, 0
               +- Exchange hashpartitioning(id#69L, 200)
                  +- Exchange RoundRobinPartitioning(9)
                     +- *(4) Range (1, 100000000, step=2, splits=12)
```

Spark can further optimize how the jobs are actually executed!

Exercises

Look at the query plans and try to understand what I've done in code!

Plan 1:

```
== Physical Plan ==
*(1) Project [firstName#153, lastName#155, (cast(salary#159 as double) / 1.1) AS salary_EUR#168]
+- *(1) FileScan csv [firstName#153,lastName#155,salary#159] Batched: false, Format: CSV, Location:
InMemoryFileIndex[file:/tmp/employees_headers.csv], PartitionFilters: [], PushedFilters: [], ReadSchema:
struct<firstName:string,lastName:string,salary:string>
```

Pause the video!

```
val employeesDF = spark.read.option("header", true).csv("/tmp/employees_headers.csv")
val empEur = employeesDF.selectExpr("firstName", "lastName", "salary / 1.1 as salary_EUR")
```

Exercises

Look at the query plans and try to understand what I've done in code!

Plan 2:

```
== Physical Plan ==
*(2) HashAggregate(keys=[dept#156], functions=[avg(cast(salary#181 as bigint))])
+- Exchange hashpartitioning(dept#156, 200)
   +- *(1) HashAggregate(keys=[dept#156], functions=[partial_avg(cast(salary#181 as bigint))])
      +- *(1) Project [dept#156, cast(salary#159 as int) AS salary#181]
         +- *(1) FileScan csv [dept#156,salary#159] Batched: false, Format: CSV, Location:
InMemoryFileIndex[file:/tmp/employees_headers.csv], PartitionFilters: [], PushedFilters: [], ReadSchema:
struct<dept:string,salary:string>
```

Pause the video!

```
val avgSals = employeesDF
  .selectExpr("dept", "cast(salary as int) as salary")
  .groupBy("dept")
  .avg("salary")
```

Exercises

Look at the query plans and try to understand what I've done in code!

Plan 3:

```
== Physical Plan ==
*(5) Project [id#195L]
+- *(5) SortMergeJoin [id#195L], [id#197L], Inner
   :- *(2) Sort [id#195L ASC NULLS FIRST], false, 0
   :  +- Exchange hashpartitioning(id#195L, 200)
   :     +- *(1) Range (1, 10000000, step=3, splits=6)
   +- *(4) Sort [id#197L ASC NULLS FIRST], false, 0
      +- Exchange hashpartitioning(id#197L, 200)
         +- *(3) Range (1, 10000000, step=5, splits=6)
```

Pause the video!

```
val ds1 = spark.range(1, 10000000, 3)
val ds2 = spark.range(1, 10000000, 5)
val j1 = ds1.join(ds2, "id")
```

Query Plans

A query plan

- describes all the operations Spark will execute when the action is triggered
- has information about partitioning scheme
- has information about the number of partitions in advance
- shows job stages
- is shown by *dataFrame.explain()*

Explain (true) will give

- the parsed logical plan
- the analyzed logical plan
- the optimized logical plan (via Catalyst)
- the physical execution plan (generated by Catalyst)

Spark works like a compiler

To Remember

Query plans = layout of Spark computations (before they run)

Whenever you see "exchange", that's a shuffle

Number of shuffles = number of stages

Number of tasks = number of partitions of each intermediate DF

Sometimes Spark already optimizes some plans!

Spark rocks