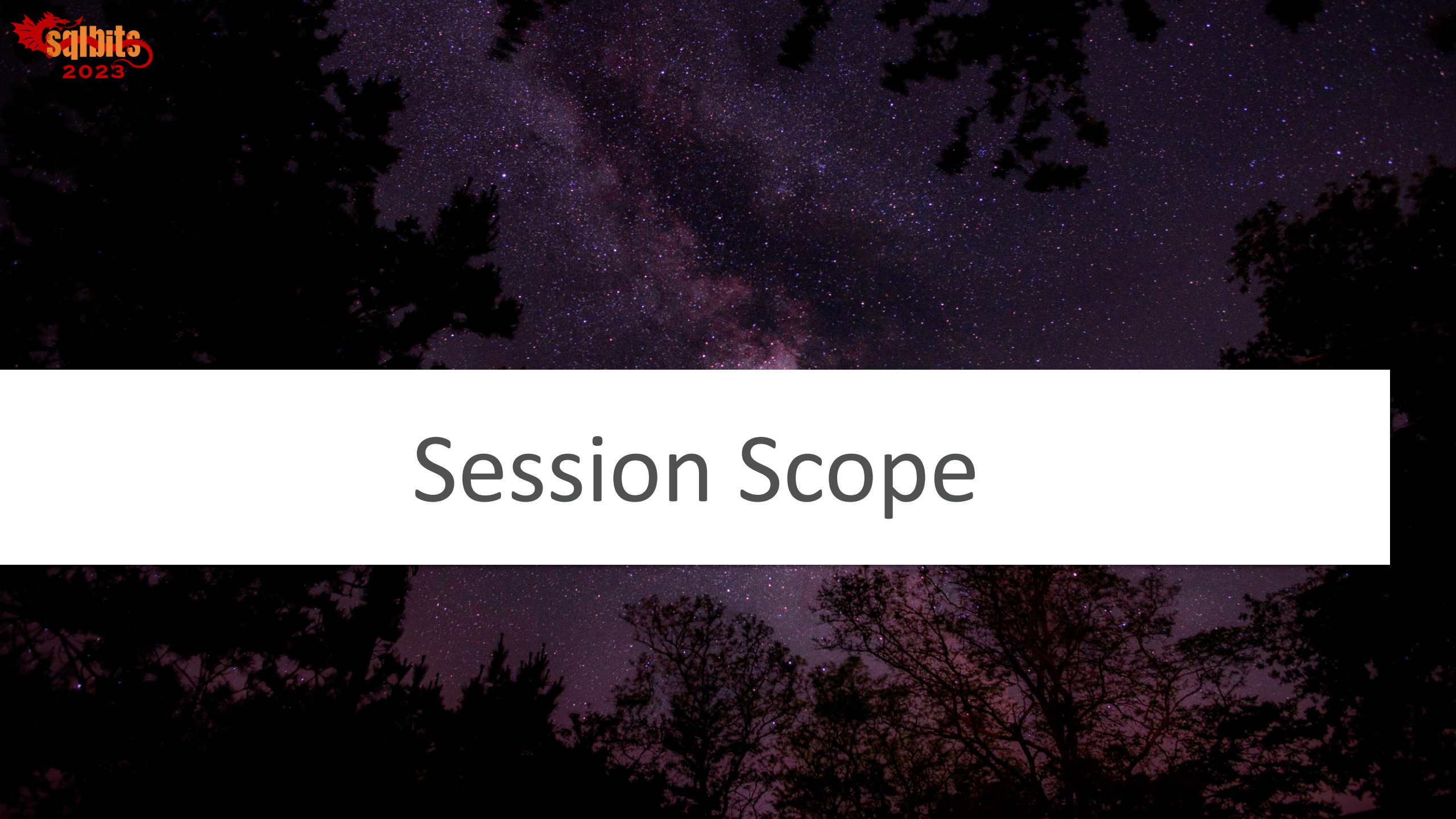
Spark Execution Plans

@falekmiah

falekmiah.com

FalekMiah01

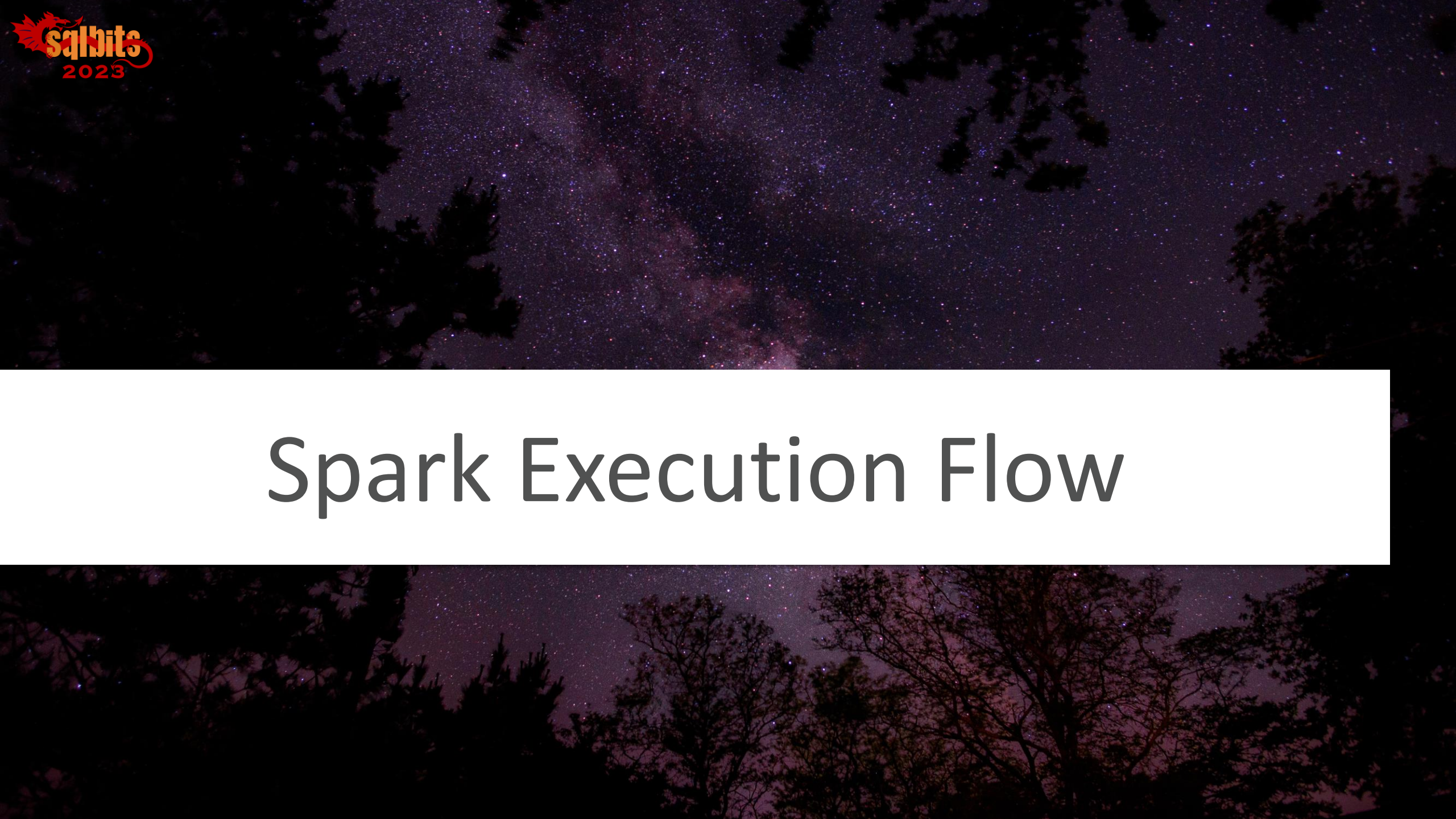


Session Scope

Session Scope

- Optimise Performance
  - Daunting
  - Where to start!

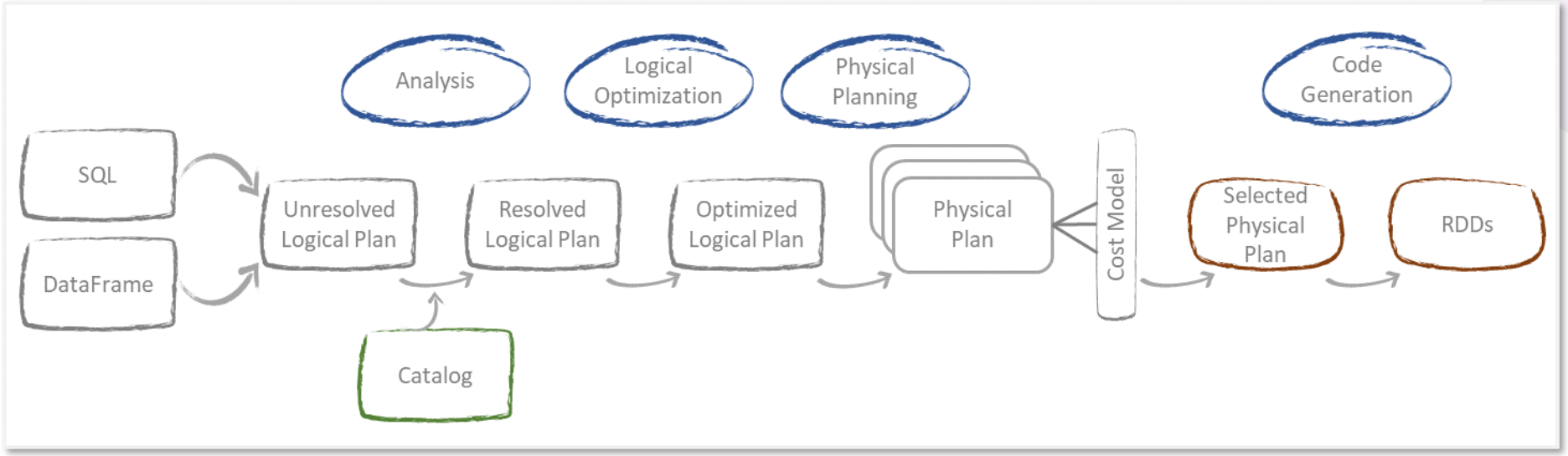
- Spark Execution Plans
  - Execution Plans
  - Execution Flow
  - Adaptive Query Execution (AQE)
  - Spark UI



Spark Execution Flow

Spark Execution Flow

- All Spark Applications use Catalyst Optimizer



Spark Execution Plans

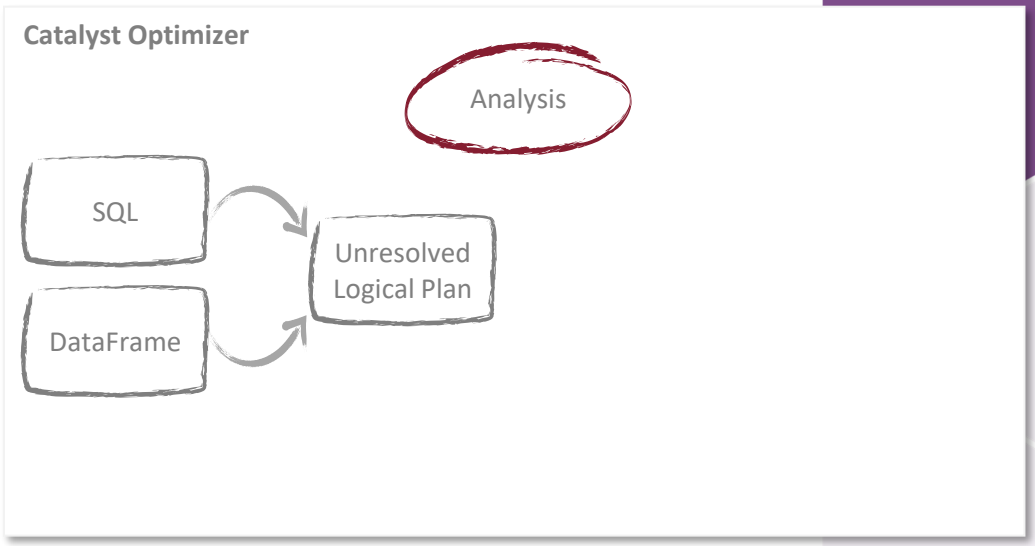
Spark Execution Plans

- Logical Plan
  - Unresolved Logical Plan
  - Resolved Logical Plan
  - Optimized Logical Plan

- Physical Plan

Logical Plan

- **Unresolved Logical Plan** (Parsed Logical Plan)
  - Identifies the `Unresolved` objects
  - Flags unvalidated objects as `Unresolved`



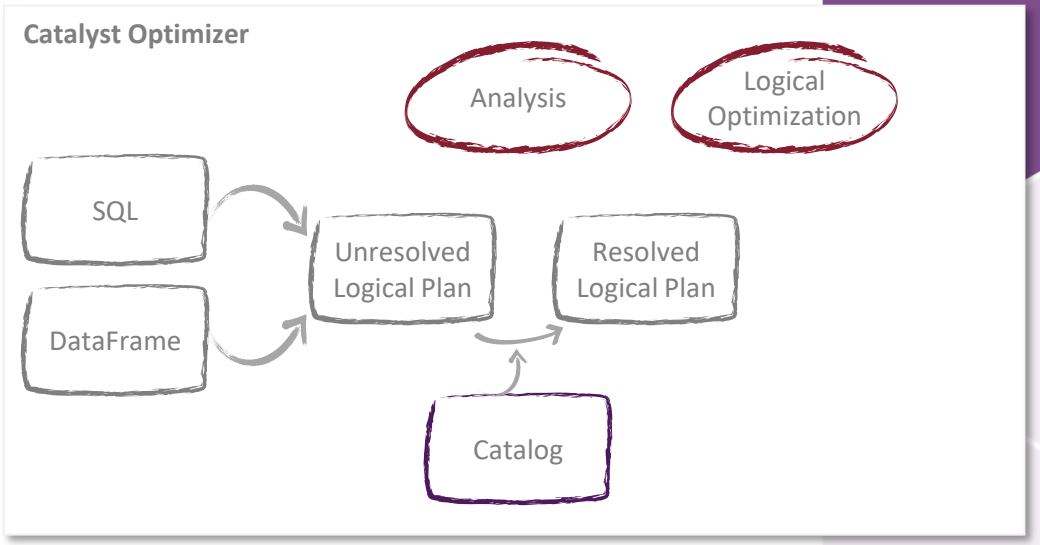

```
1  deltaDF.explain(True)
```

```
== Parsed Logical Plan ==
'Sort ['totalSales DESC NULLS LAST], true
+- 'Aggregate ['saleDate], ['saleDate, 'sum(('quantity * 'price)) AS totalSales#1238]
   +- 'Filter ('i.itemID = 4)
      +- 'Join Inner, ('i.itemID = 's.itemID)
         :- 'SubqueryAlias s
         :  +- 'UnresolvedRelation [sales], [], false
         +- 'SubqueryAlias i
            +- 'UnresolvedRelation [items], [], false
```

Logical Plan

- **Unresolved Logical Plan** (Parsed Logical Plan)
  - Flags unvalidated objects as `Unresolved`

- **Resolved Logical Plan** (Analyzed Logical Plan)
  - Validates the `Unresolved` objects
  - Uses `Catalog` metadata repository

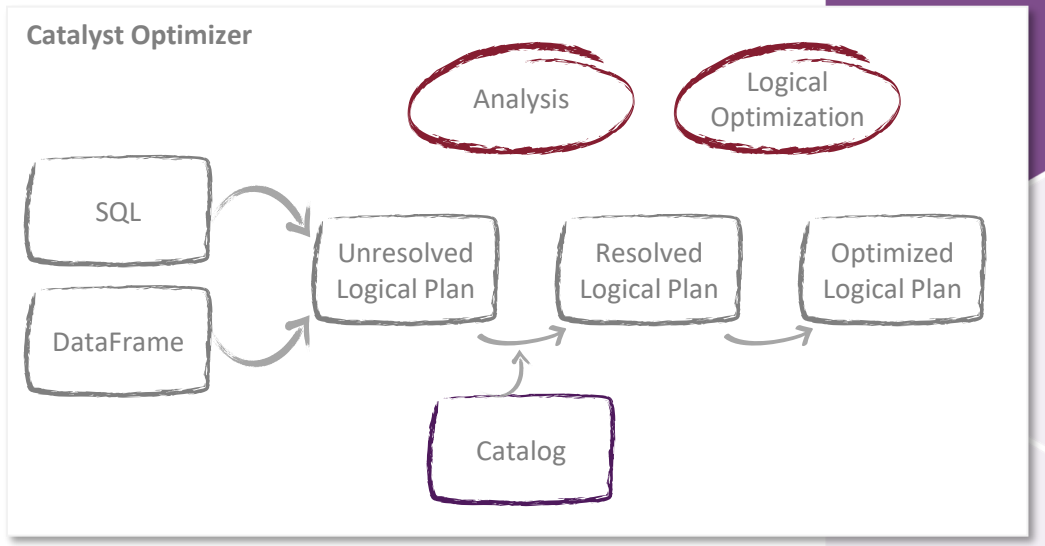



```
1  deltaDF.explain(True)
```

```
== Analyzed Logical Plan ==
saleDate: date, totalSales: double
Sort [totalSales#1238 DESC NULLS LAST], true
+- Aggregate [saleDate#1244], [saleDate#1244, sum((cast(quantity#1243 as float) * price#1251)) AS totalSales#1238]
   +- Filter (itemID#1249 = 4)
      +- Join Inner, (itemID#1249 = itemID#1242)
         :- SubqueryAlias s
         :  +- SubqueryAlias spark_catalog.fmsandbox.sales
         :     +- Relation spark_catalog.fmsandbox.sales[itemID#1242,quantity#1243,saleDate#1244] parquet
         +- SubqueryAlias i
            +- SubqueryAlias spark_catalog.fmsandbox.items
               +- Relation spark_catalog.fmsandbox.items[itemID#1249,itemName#1250,price#1251,effectiveDate#1252] parquet
```

Logical Plan

- **Unresolved Logical Plan** (Parsed Logical Plan)
  - Flags unvalidated objects as `Unresolved`
- **Resolved Logical Plan** (Analyzed Logical Plan)
  - Validates the `Unresolved` objects
  - Uses `Catalog` metadata repository
- **Optimized Logical Plan**
  - Applies predicates or rules to further optimize the plan




```
1  deltaDF.explain(True)
```

```
== Optimized Logical Plan ==
Sort [totalSales#1238 DESC NULLS LAST], true
+- Aggregate [saleDate#1244], [saleDate#1244, sum((cast(quantity#1243 as float) * price#1251)) AS totalSales#
   +- Project [quantity#1243, saleDate#1244, price#1251]
      +- Join Inner, (itemID#1249 = itemID#1242)
         :- Filter (isnotnull(itemID#1242) AND (itemID#1242 = 4))
         :  +- Relation spark_catalog.fmsandbox.sales[itemID#1242,quantity#1243,saleDate#1244] parquet
         +- Project [itemID#1249, price#1251]
            +- Filter (isnotnull(itemID#1249) AND (itemID#1249 = 4))
               +- Relation spark_catalog.fmsandbox.items[itemID#1249,itemName#1250,price#1251,effectiveDate#1
```

Physical Plan

- Is how the Logical Plan will be **executed** on the cluster
- Generates different execution **strategies**
- Compares them through a **Cost Model**
- Selects the **best optimal plan/strategy** as the “Best Physical Plan”

Physical Plan

```
1  deltaDF.explain()
```

Python

```
== Physical Plan ==
Sort [totalSales#1238 DESC NULLS LAST], true, 0
+- Exchange rangepartitioning(totalSales#1238 DESC NULLS LAST, 200), ENSURE_REQUIREMENTS, [plan_id=981]
   +- *(3) HashAggregate(keys=[saleDate#1244], functions=[finalmerge_sum(merge sum#1259) AS sum((cast(quantity#1243 as float) * price#1251))#1255])
      +- Exchange hashpartitioning(saleDate#1244, 200), ENSURE_REQUIREMENTS, [plan_id=977]
         +- *(2) HashAggregate(keys=[saleDate#1244], functions=[partial_sum((cast(quantity#1243 as float) * price#1251)) AS sum#1259])
            +- *(2) Project [quantity#1243, saleDate#1244, price#1251]
               +- *(2) BroadcastHashJoin [itemID#1242], [itemID#1249], Inner, BuildRight, false
                  :- *(2) Filter (isnotnull(itemID#1242) AND (itemID#1242 = 4))
                  :  +- *(2) ColumnarToRow
                  :     +- FileScan parquet spark_catalog.fmsandbox.sales[itemID#1242,quantity#1243,saleDate#1244] Batched: true, DataFilters: [isnotnull(itemID#1242), (itemID#1
242 = 4)], Format: Parquet, Location: PreparedDeltaFileIndex(1 paths)[dbfs:/user/hive/warehouse/fmsandbox.db/sales], PartitionFilters: [], PushedFilters: [IsNotNull(itemID), Equ
alTo(itemID,4)], ReadSchema: struct<itemID:int,quantity:int,saleDate:date>
                  +- BroadcastExchange HashedRelationBroadcastMode(List(cast(input[0, int, false] as bigint)),false), [plan_id=971]
                     +- *(1) Filter (isnotnull(itemID#1249) AND (itemID#1249 = 4))
                        +- *(1) ColumnarToRow
                           +- FileScan parquet spark_catalog.fmsandbox.items[itemID#1249,price#1251] Batched: true, DataFilters: [isnotnull(itemID#1249), (itemID#1249 = 4)], For
mat: Parquet, Location: PreparedDeltaFileIndex(1 paths)[dbfs:/user/hive/warehouse/fmsandbox.db/items], PartitionFilters: [], PushedFilters: [IsNotNull(itemID), EqualTo(itemID,
4)], ReadSchema: struct<itemID:int,price:float>
```

Generate Execution Plans

Generate Execution Plans


```
.explain()
```

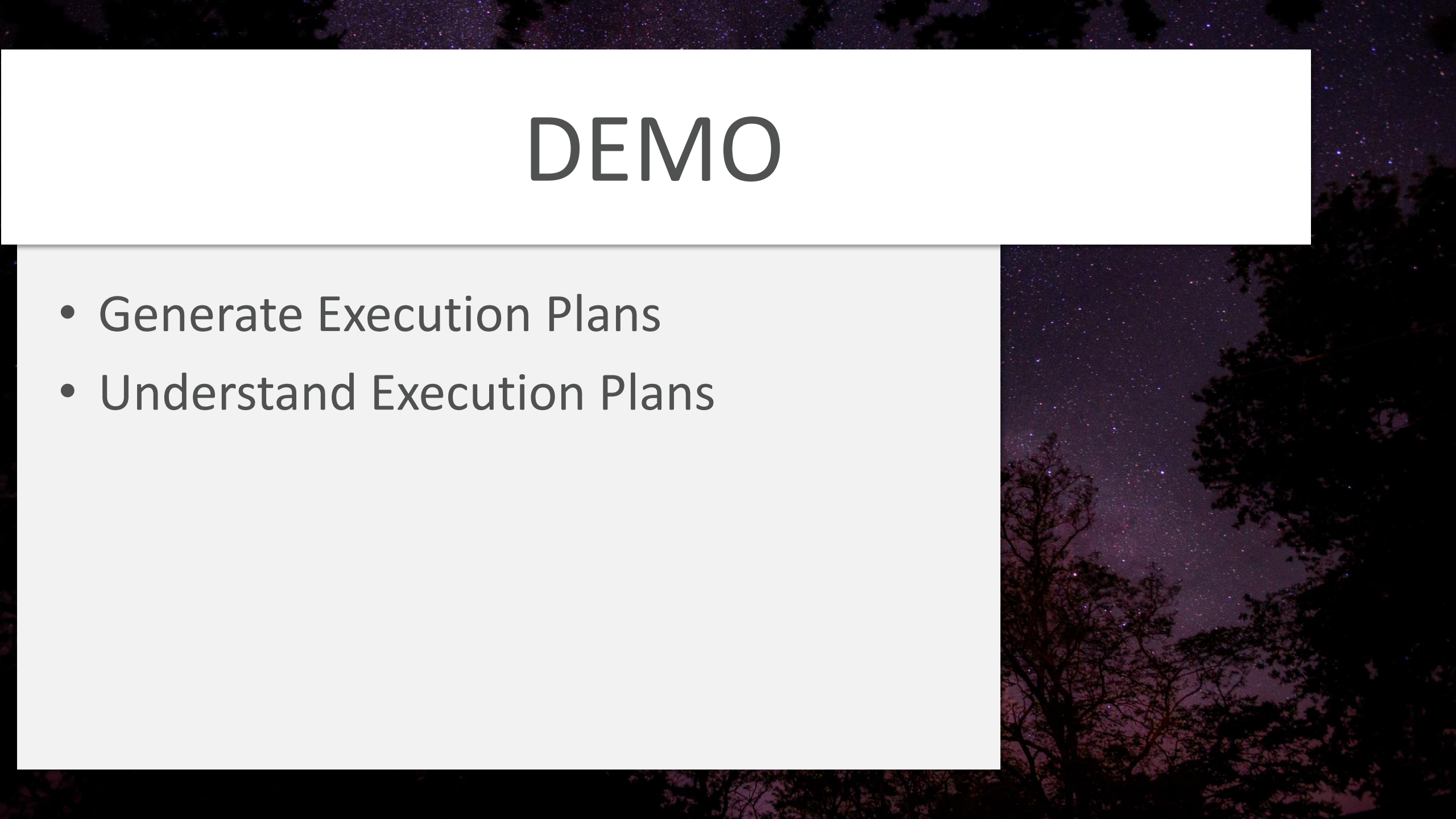
```
.explain(True) or .explain(mode="extended")

.explain(mode="codegen")

.explain(mode="cost")

.explain(mode="formatted")
```

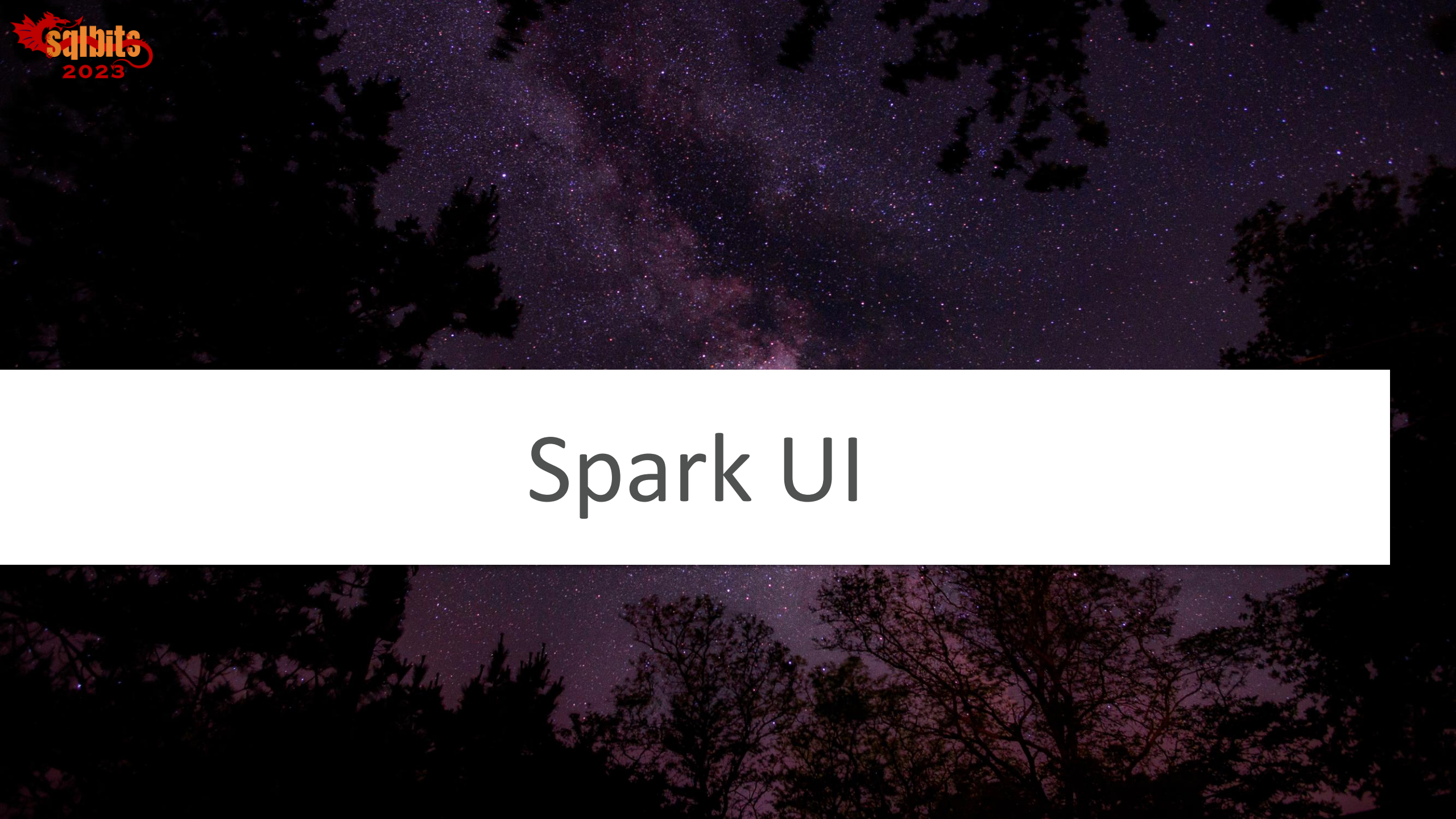
```
EXPLAIN
```

```
EXPLAIN [ EXTENDED | CODEGEN | COST | FORMATTED ]
```

DEMO

- Generate Execution Plans
- Understand Execution Plans





Spark UI

Spark UI

- Monitor Spark Application
- Insight Into Executions and Workload
- Debugging
- Displays queries, jobs, DAG, and query plans

Spark UI - Jobs

Jobs | Stages | Storage | Environment | Executors | SQL / DataFrame | JDBC/ODBC Server | Structured Streaming

Spark Jobs (?)

User: root
Total Uptime: 4.6 min
Scheduling Mode: FAIR
Completed Jobs: 11

▸ Event Timeline

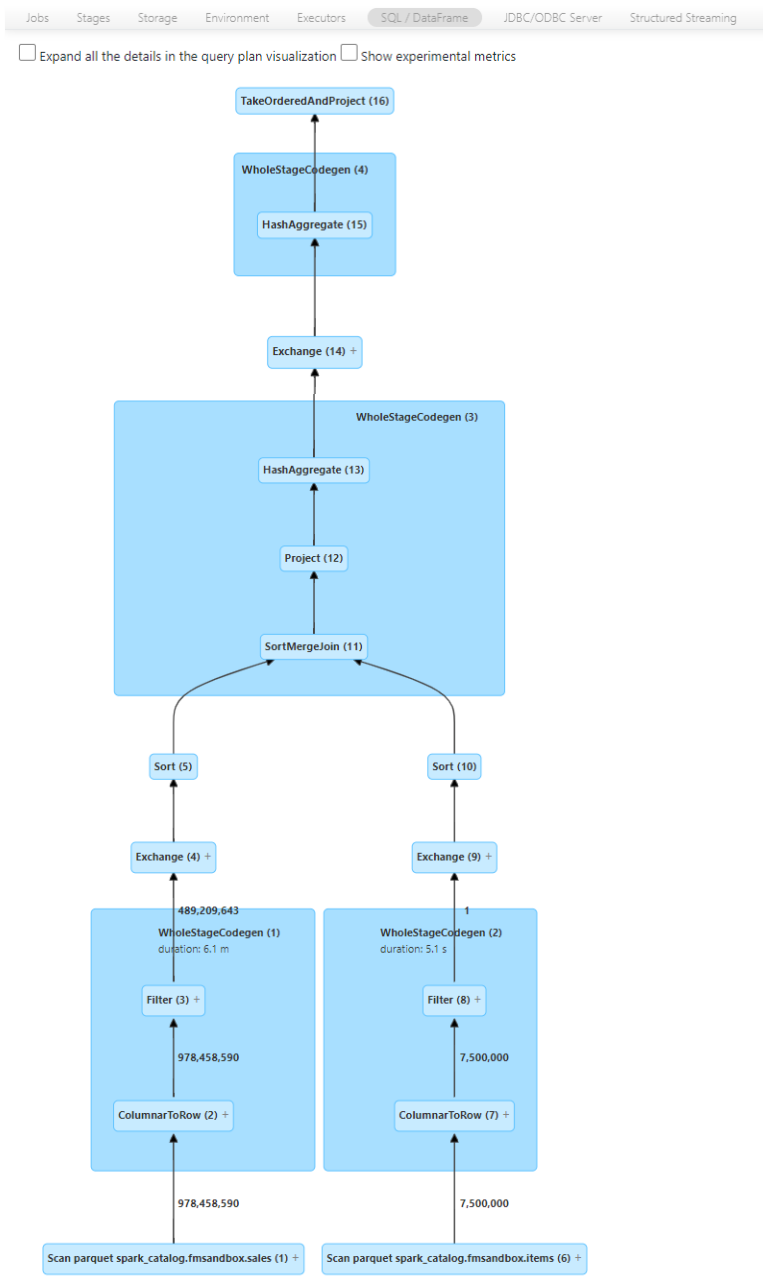
▾ Completed Jobs (11)

Page: 1

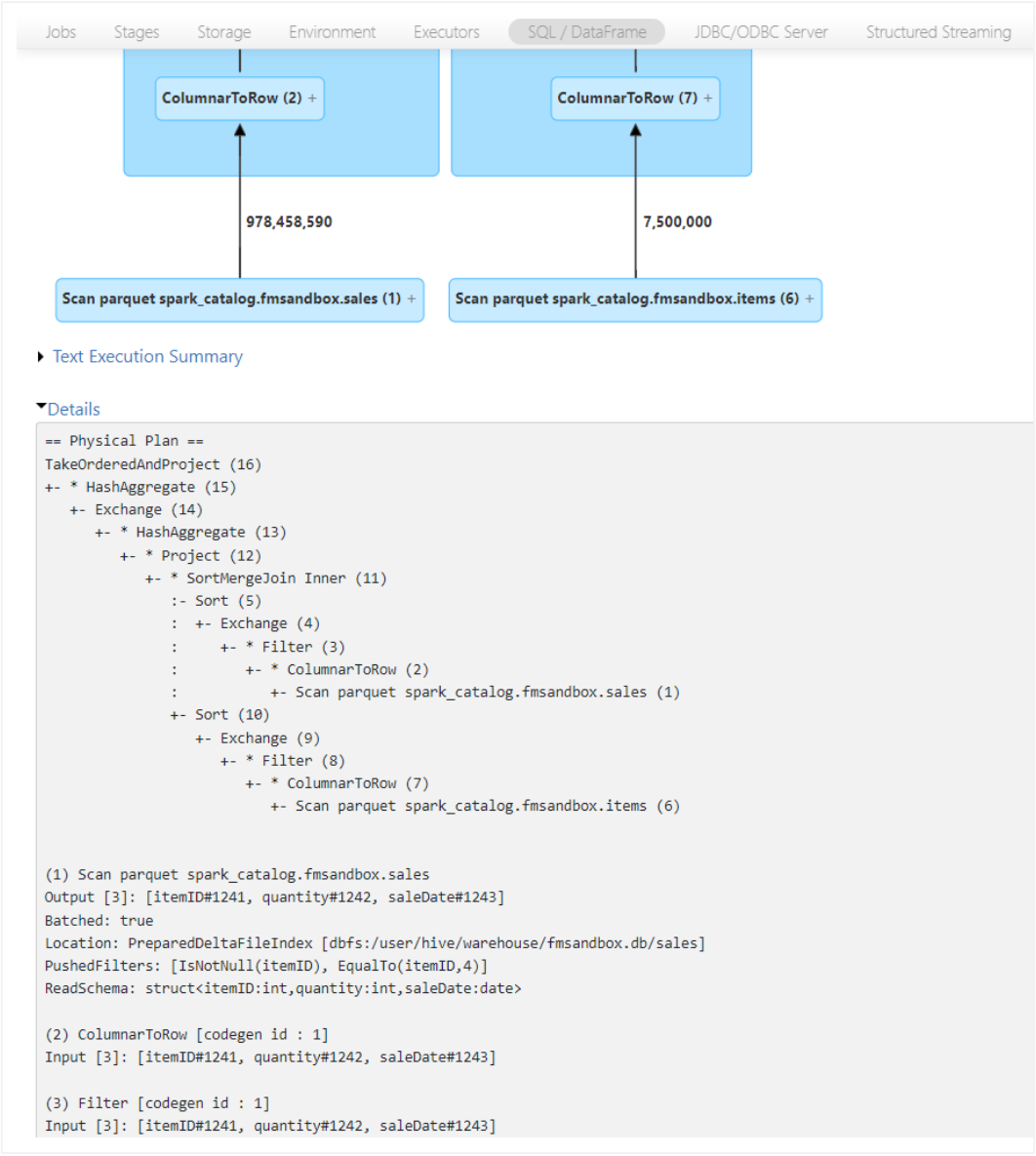
1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id (Job Group)	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
10 (3147736733626348505_6685574054404766343_326aa3d54f44435ea5e7107c817fbe4b)	try: def ___databricks_percent_sql(): i... executeCollect at DatasetRefCache.scala:71	2023/02/02 15:54:58	0.3 s	1/1	4/4
9 (3147736733626348505_6685574054404766343_326aa3d54f44435ea5e7107c817fbe4b)	try: def ___databricks_percent_sql(): i... executeCollect at DatasetRefCache.scala:71	2023/02/02 15:54:58	0.3 s	1/1	4/4
8 (3147736733626348505_8306147359562026961_53e17d4711ef4d1c9cea51c4107e89a9)	# Drop "items" and "sales" delta table spark.sq... first at Snapshot.scala:238	2023/02/02 15:54:56	0.2 s	2/2 (1 skipped)	2/2 (4 skipped)
7 (3147736733626348505_8306147359562026961_53e17d4711ef4d1c9cea51c4107e89a9)	# Drop "items" and "sales" delta table spark.sq... collect at Checksum.scala:387	2023/02/02 15:54:56	0.3 s	2/2	5/5
6 (3147736733626348505_8306147359562026961_53e17d4711ef4d1c9cea51c4107e89a9)	# Drop "items" and "sales" delta table spark.sq...	2023/02/02 15:54:55	0 ms	0/0	0/0
5 (3147736733626348505_8306147359562026961_53e17d4711ef4d1c9cea51c4107e89a9)	# Drop "items" and "sales" delta table spark.sq... write at WriteIntoDeltaCommand.scala:70	2023/02/02 15:52:40	2.2 min	1/1	4/4

Spark UI - DAG


Jobs
Stages
Storage
Environment
Executors
SQL / DataFrame
JDBC/ODBC Server
Structured Streaming
Expand all the details in the query plan visualization
Show experimental metrics
TakeOrderedAndProject (16)
WholeStageCodegen (4)
HashAggregate (15)
Exchange (14)
WholeStageCodegen (3)
HashAggregate (13)
Project (12)
SortMergeJoin (11)
Sort (5)
Sort (10)
Exchange (4)
Exchange (9)
489,209,643
1
WholeStageCodegen (1)
duration: 6.1 m
WholeStageCodegen (2)
duration: 5.1 s
Filter (3)
Filter (8)
978,458,590
7,500,000
ColumnarToRow (2)
ColumnarToRow (7)
978,458,590
7,500,000
Scan parquet spark_catalog.fmsandbox.sales (1)
Scan parquet spark_catalog.fmsandbox.items (6)

Spark UI - Query Plan

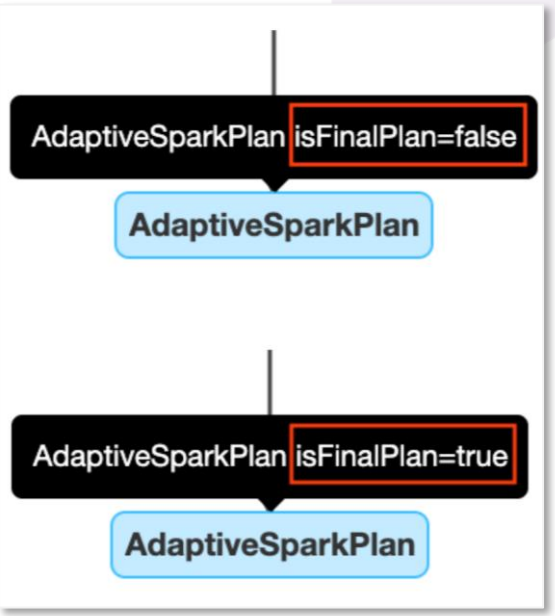


Adaptive Query Execution (AQE)

Adaptive Query Execution (AQE)

- Optimizes further
- Changes Query Plan
  - Uses Runtime Statistic
  - Increases Query Performance
- Visible in Spark UI
- Enable using Spark Configuration settings

```
AdaptiveSparkPlan isFinalPlan=true
+- == Final Plan ==
   *(3) BroadcastHashJoin [key#13], [a#23], Inner, BuildLeft, false
   :- BroadcastQueryStage 2, Statistics(sizeInBytes=1024.0 KiB, rowCount=1,
   :  +- BroadcastExchange
   :        ...
+- == Initial Plan ==
   SortMergeJoin [key#13], [a#23], Inner
   :- Sort [key#13 ASC NULLS FIRST], false, 0
   :  +- Exchange hashpartitioning(key#13, 5), true, [id=#117]
   :        ...
```




```
spark.conf.set("spark.sql.adaptive.enabled", "true")
```

Query Hints

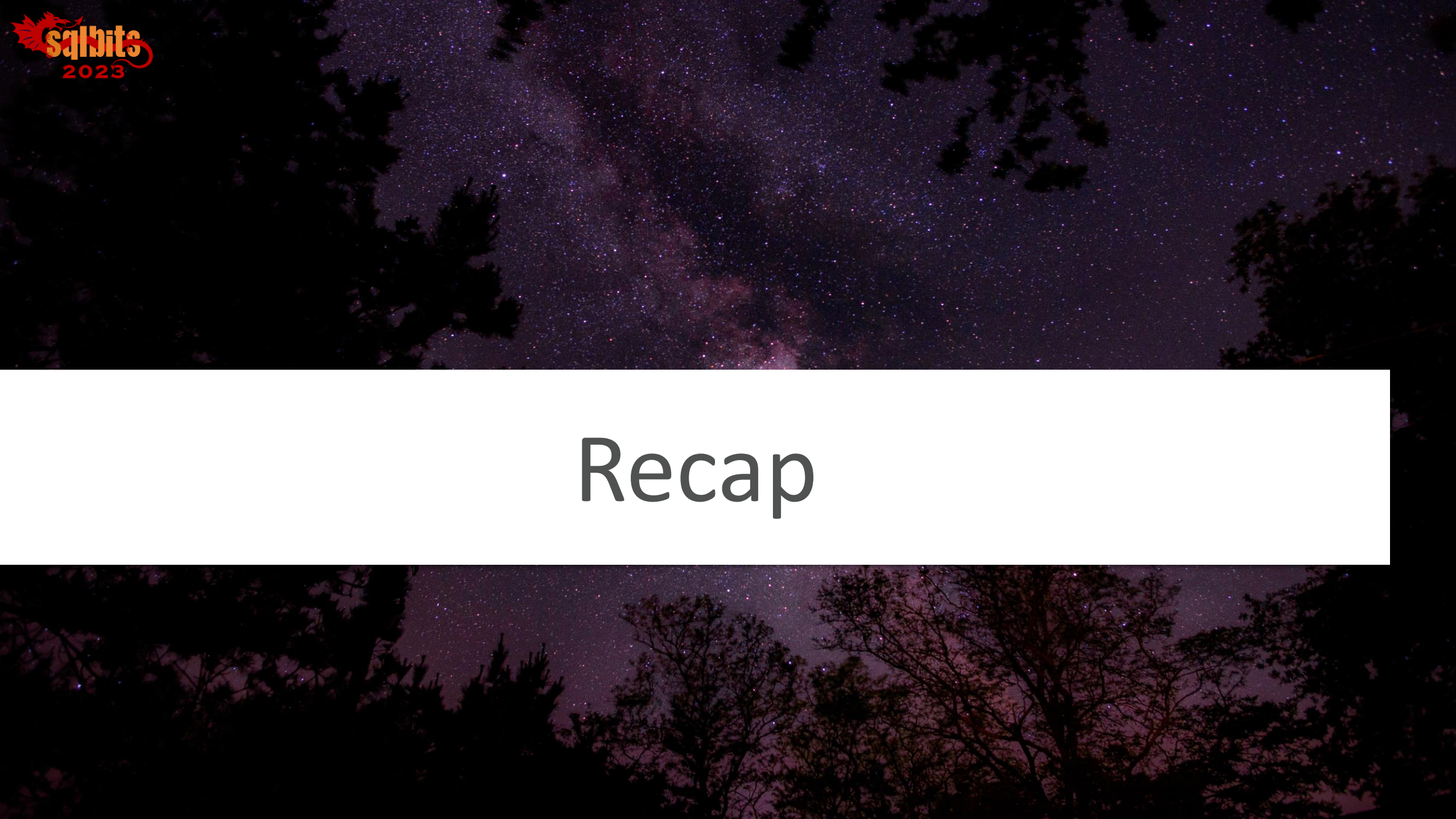
Query Hints

- Specify the approach
- Partitioning Hints
  - COALESCE, REPARTITION, REPARTITION_BY_RANGE, REBALANCE

```
SELECT /*+ [COALESCE | REPARTITION | REPARTITION_BY_RANGE | REBALANCE](n) */
    <columnName>
FROM <t1>
```

- Join Hints
  - BROADCAST, MERGE, SHUFFLE_HASH, SHUFFLE_REPLICATE_NL

```
SELECT /*+ [BROADCAST | MERGE | SHUFFLE_HASH | SHUFFLE_HASH](t1) */
    <columnName>
FROM <t1>
INNER JOIN <t2> ON t1.key = t2.key;
```



Recap

Recap

- Execution Plans
- Logical Plan and Physical Plan
- Additional parameters
- Spark UI
- Adaptive Query Execution (AQE)


```
.explain() or EXPLAIN
```

Check out our YouTube channel and Blogs


sqlbits
2023
Thank You
falek@advancinganalytics.co.uk
@falekmiah
falekmiah.com
FalekMiah01
advancinganalytics.co.uk