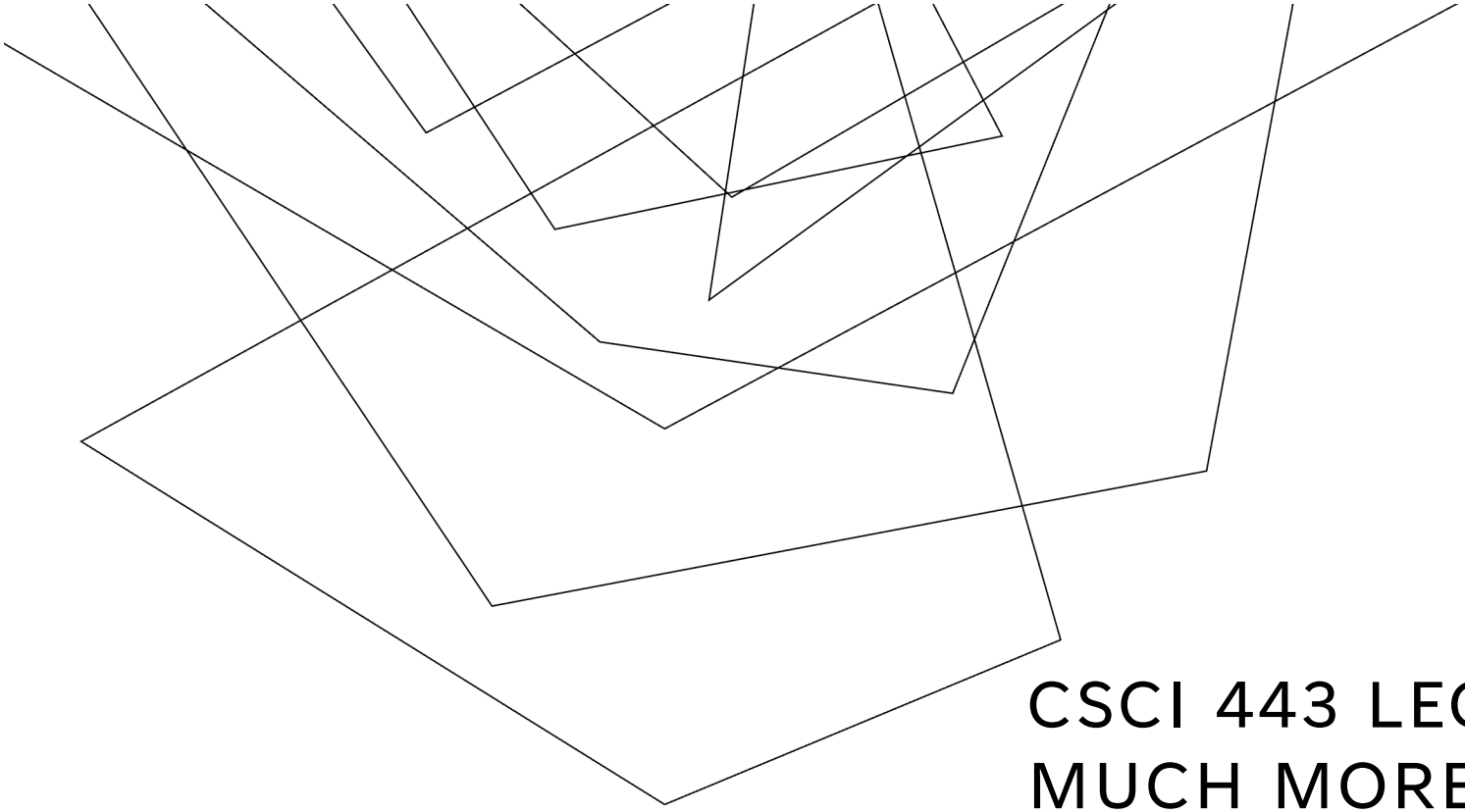
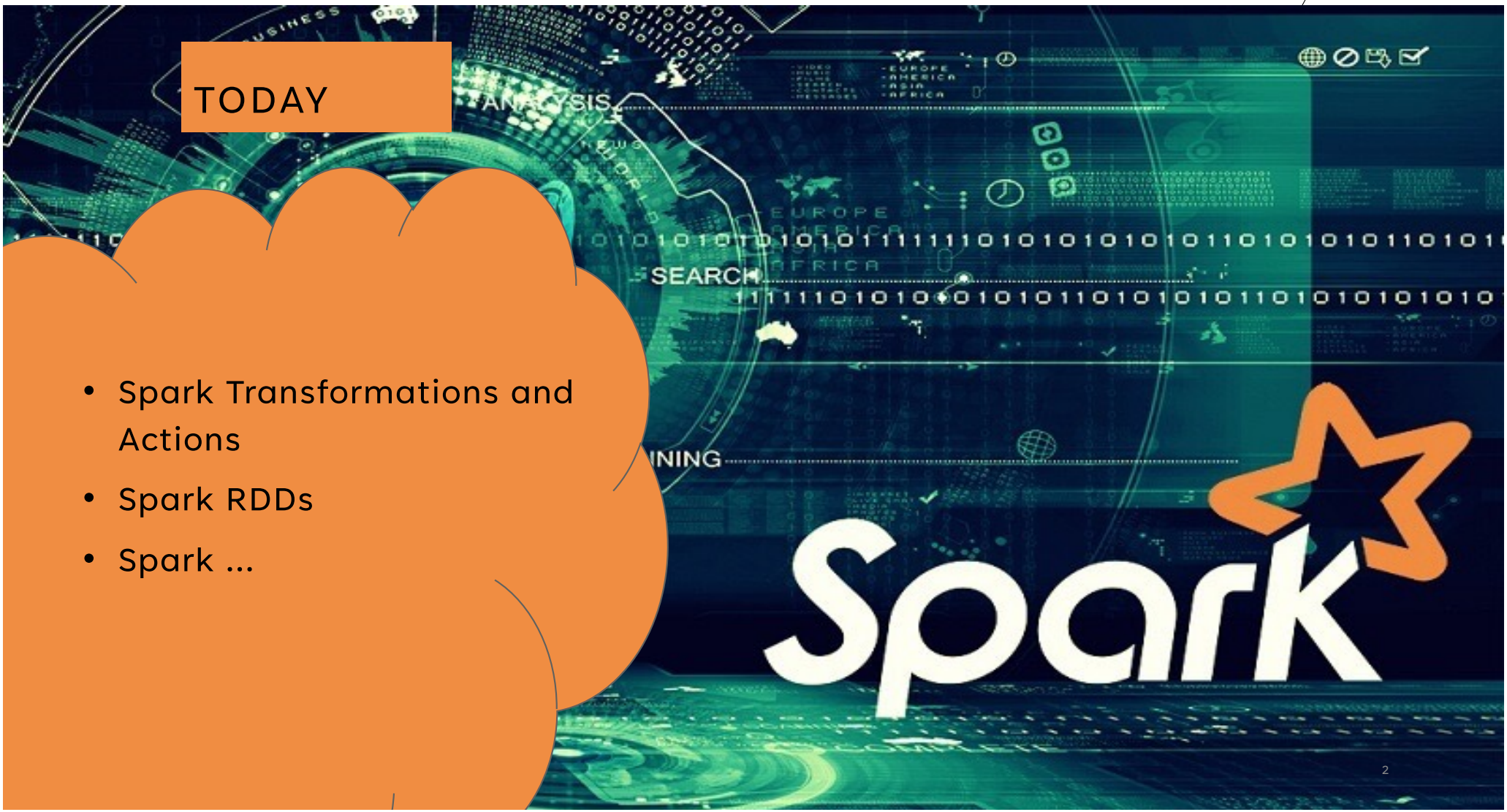


CSCI 443 LECTURE 5: MUCH MORE ABOUT SPARK

Professor David Harrison


TODAY
• Spark Transformations and Actions
• Spark RDDs
• Spark ...
SEARCH
Spark

HOMEWORK 1

Due Tonight.

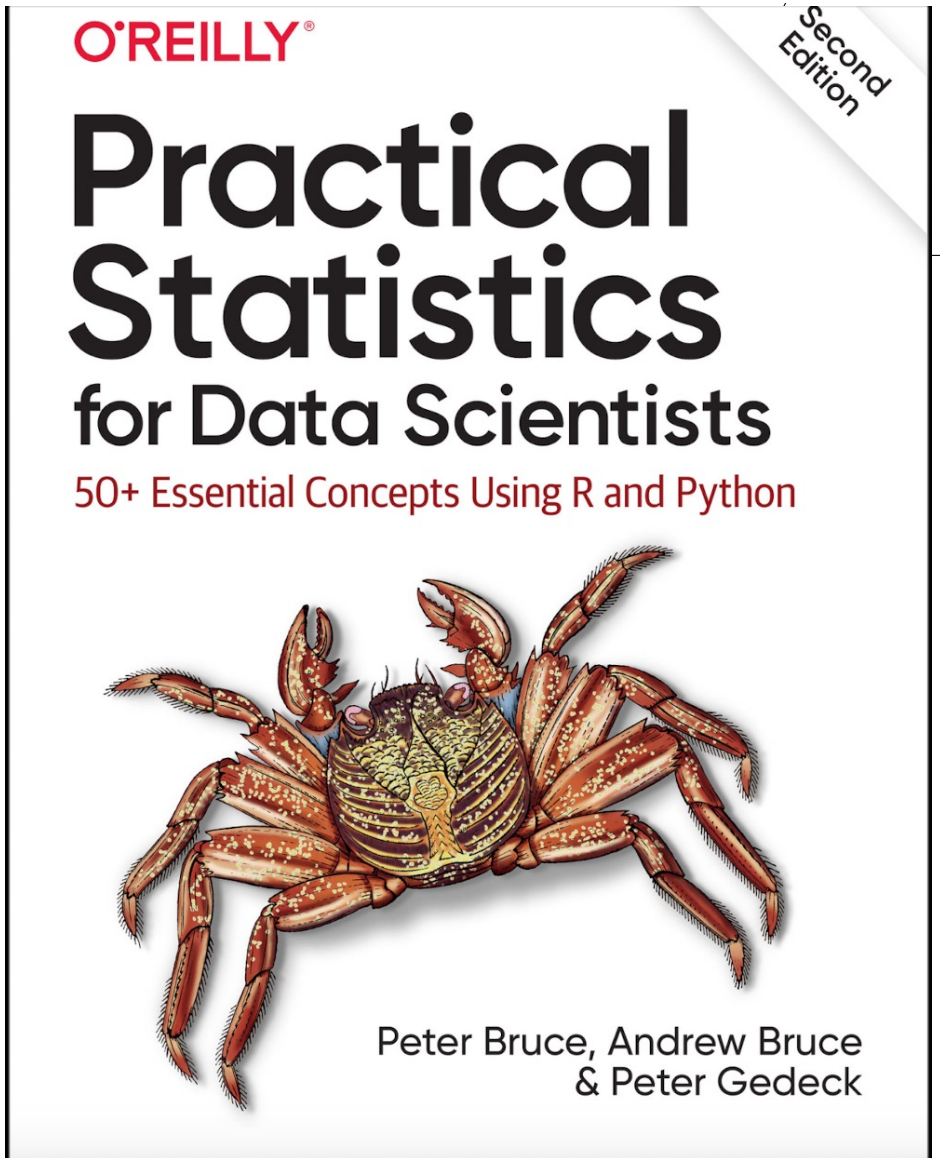
11 pm.

Submission:

- Submit archived Databricks Notebook to Blackboard.
- NOTE: Submission only needs to be the notebook. No README is necessary.

READING!

- Read Chapter 1: Exploratory Data Analysis.



OFFICE HOURS

Due to scheduling conflict, office hours updated

Monday	1:00-2:00 PM
Tuesday	4-5 PM

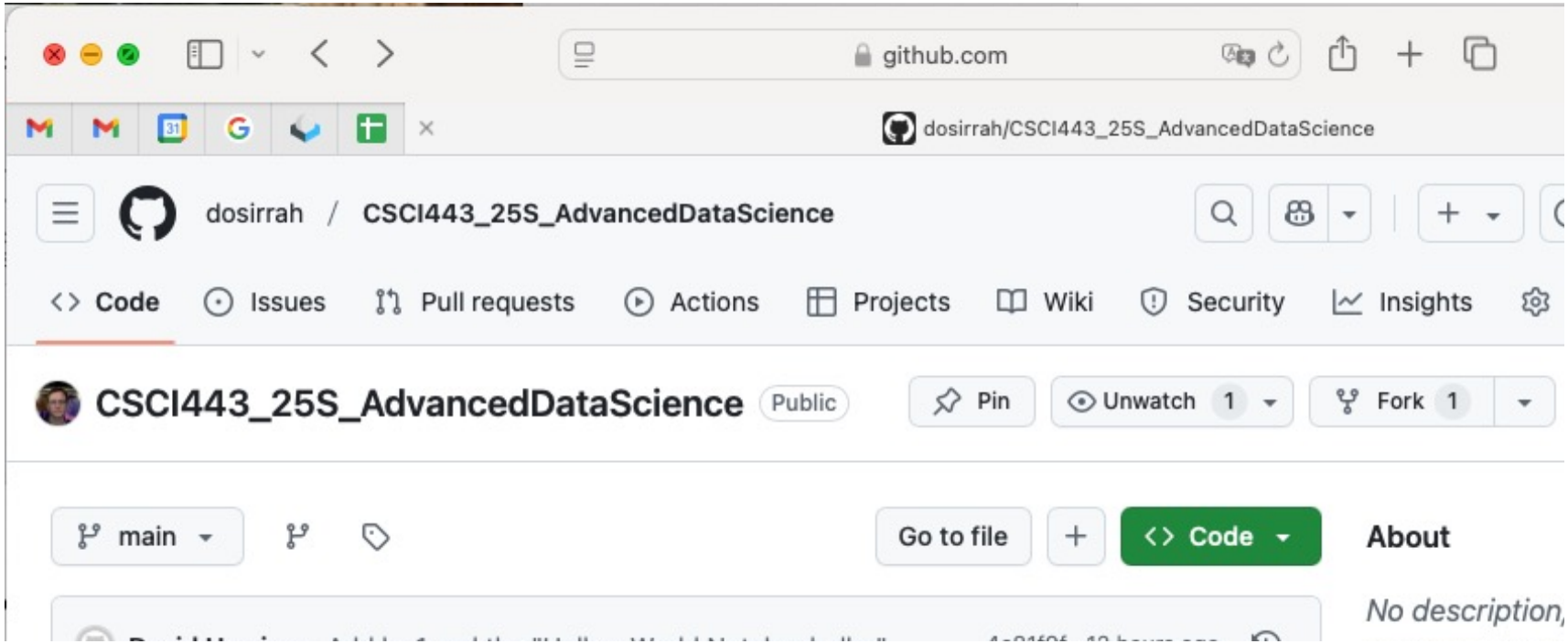
.

GITHUB

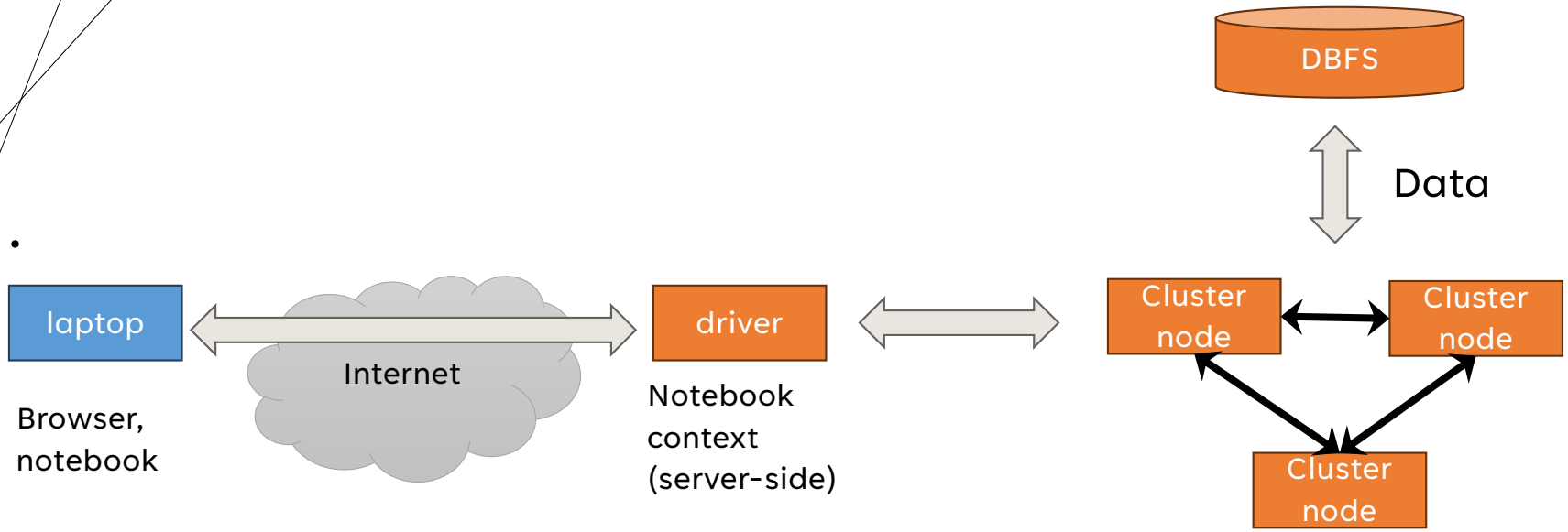
Lecture slides and examples have been committed to GitHub for lectures 1 through 4.

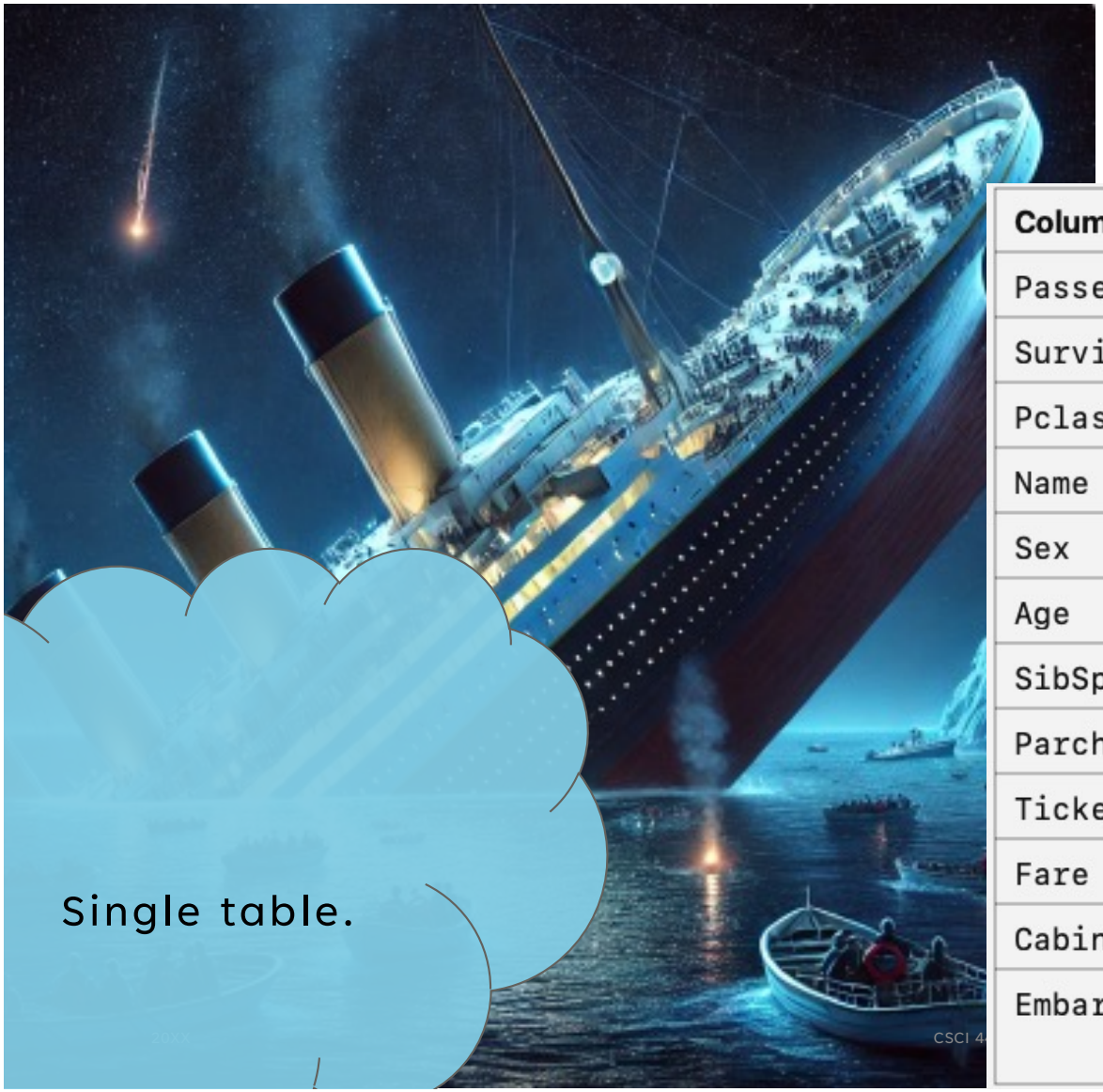
The project is at

https://github.com/dosirrah/CSCI443_25S_AdvancedDataScience



SPARK ARCHITECTURE (1000 FT)





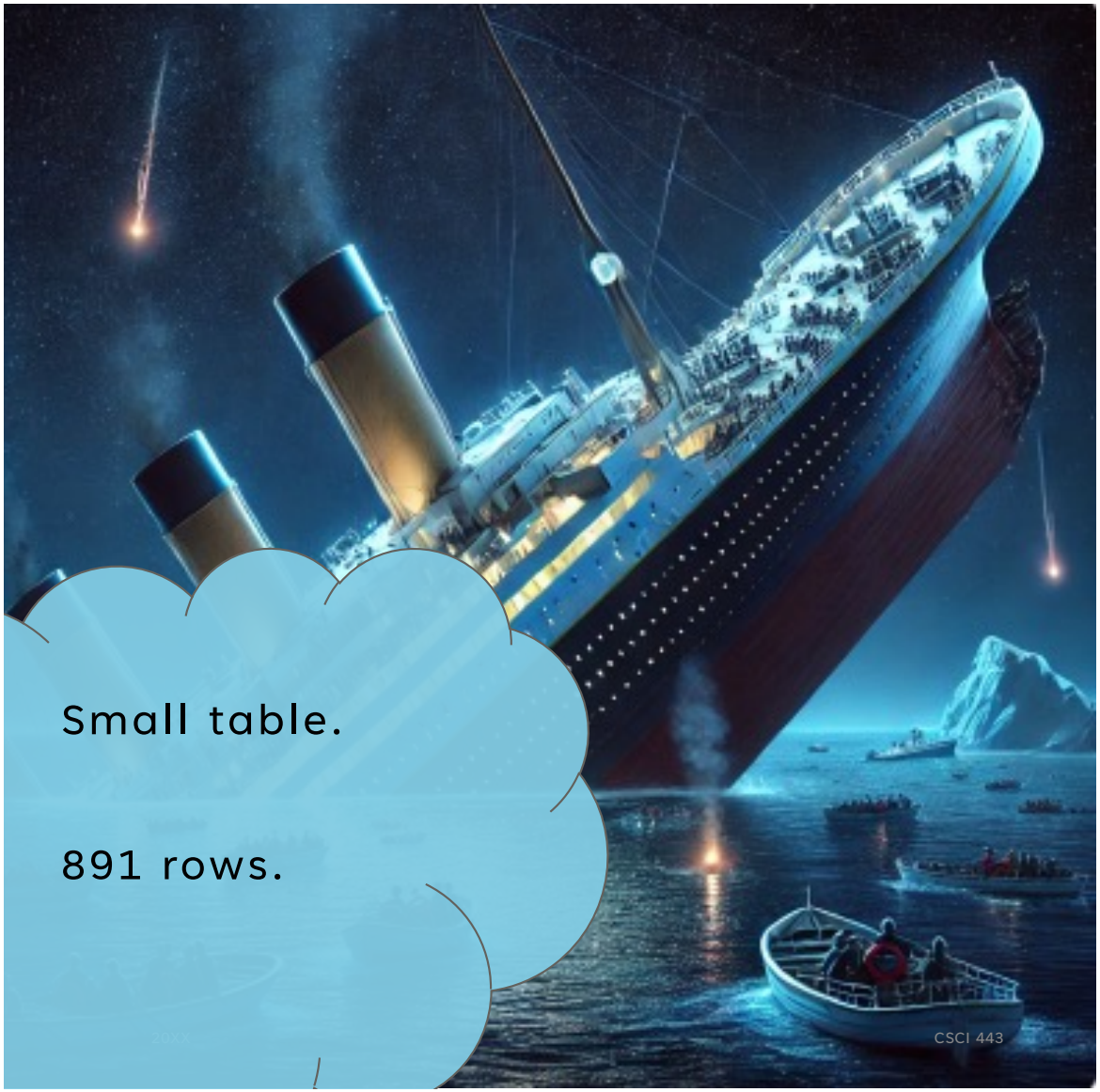
TITANIC DATASET

Column Name	Description
PassengerId	Unique identifier for each passenger
Survived	Survival (0 = No, 1 = Yes)
Pclass	Passenger class (1 = 1st, 2 = 2nd, 3 = 3rd)
Name	Passenger's full name
Sex	Passenger's sex (male/female)
Age	Passenger's age (in years)
SibSp	Number of siblings/spouses aboard the Titanic
Parch	Number of parents/children aboard the Titanic
Ticket	Ticket number
Fare	Fare price paid for the ticket
Cabin	Cabin number (if assigned)
Embarked	Port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)

TITANIC DATASET


```
Just now (1s)

df.count()

(2) Spark Jobs

Out[14]: 891
```

TITANIC DATASET



Small table.

891 rows.

```
Just now (1s)

df.count()

(2) Spark Jobs

Out[14]: 891
```

TITANIC DATASET

Today we avoid using Pandas or pyspark.pandas.

Only Spark DataFrames

By default Spark does not infer a schema.

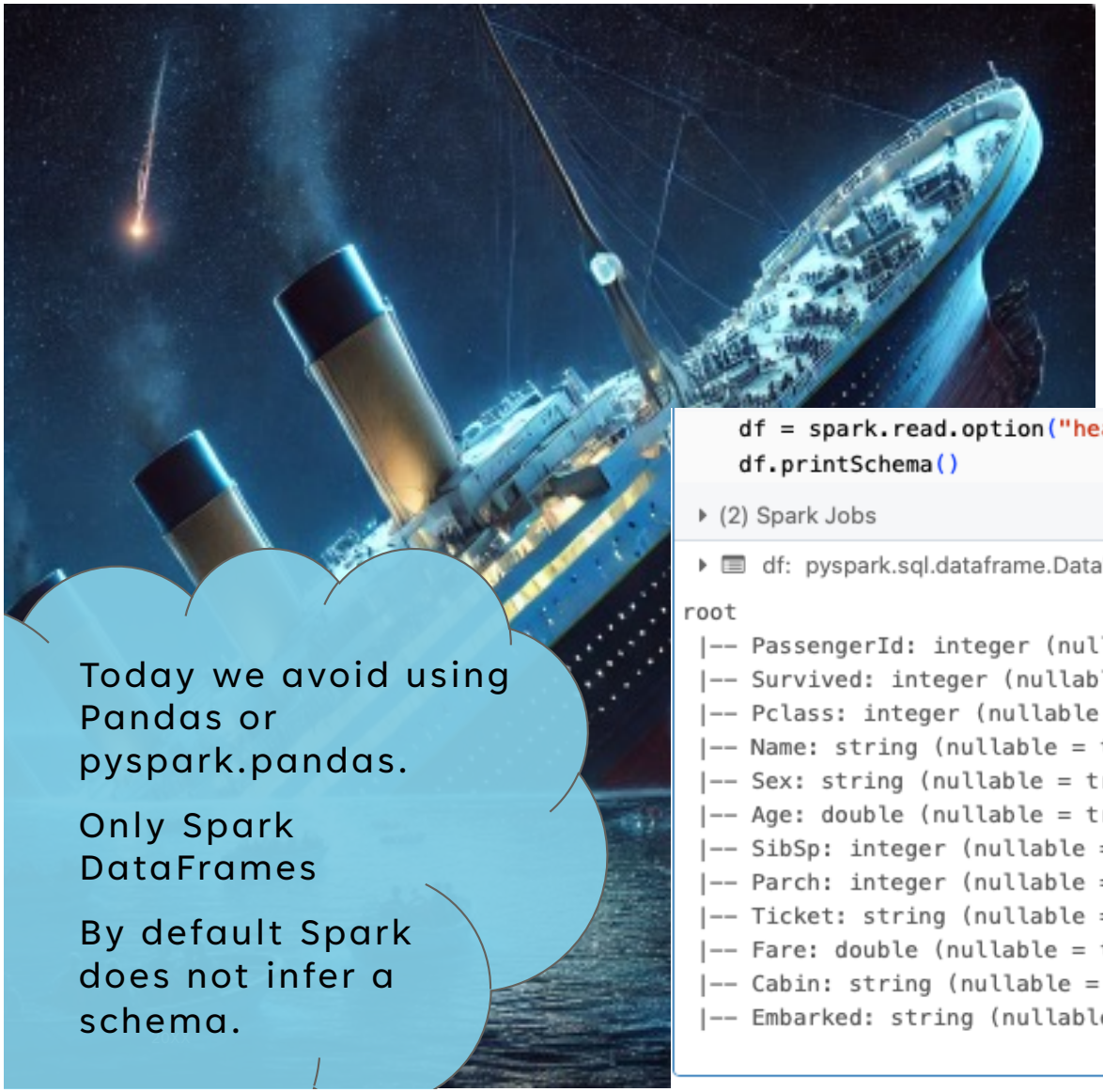
```
07:35 PM (2s)

import pyspark.pandas as ps

file_path = "dbfs:/FileStore/shared_uploads/daharri6@olemiss.edu

# Load a Spark DataFrame
df = spark.read.option("header", "true").csv(file_path)

# Print out the first 10 rows
df.show(10)
```

(2) Spark Jobs

df: pyspark.sql.dataframe.DataFrame

```
PassengerId: string
Survived: string
Pclass: string
Name: string
Sex: string
Age: string
SibSp: string
Parch: string
Ticket: string
Fare: string
Cabin: string
Embarked: string
```

TITANIC DATASET



Today we avoid using Pandas or pyspark.pandas.

Only Spark DataFrames

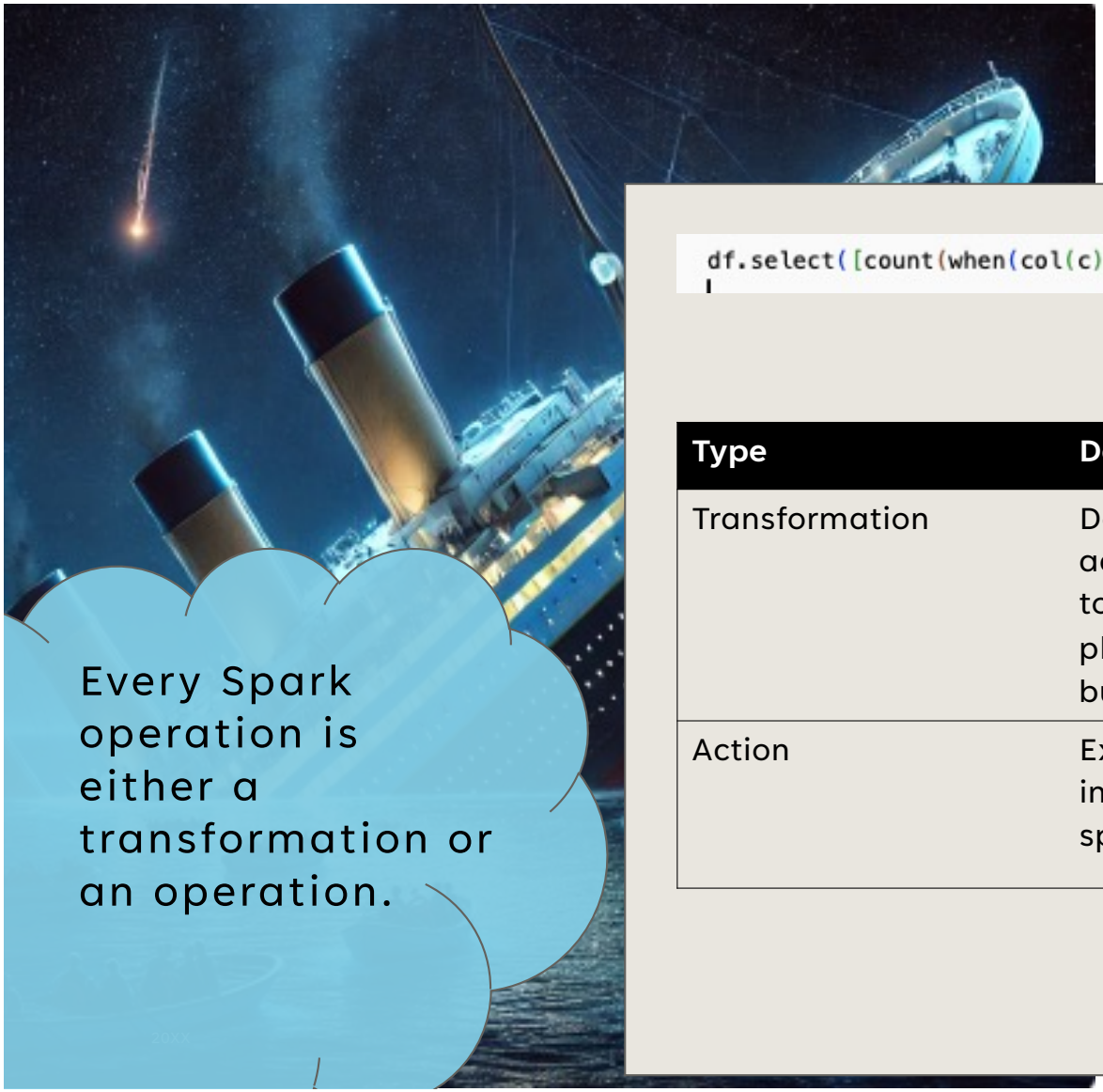
By default Spark does not infer a schema.

```
df = spark.read.option("header", "true").option("inferSchema", "true").csv(file_path)
df.printSchema()
```

▸ (2) Spark Jobs

▸ df: pyspark.sql.dataframe.DataFrame = [PassengerId: integer, Survived: integer ... 10 more fields]

```
root
 |-- PassengerId: integer (nullable = true)
 |-- Survived: integer (nullable = true)
 |-- Pclass: integer (nullable = true)
 |-- Name: string (nullable = true)
 |-- Sex: string (nullable = true)
 |-- Age: double (nullable = true)
 |-- SibSp: integer (nullable = true)
 |-- Parch: integer (nullable = true)
 |-- Ticket: string (nullable = true)
 |-- Fare: double (nullable = true)
 |-- Cabin: string (nullable = true)
 |-- Embarked: string (nullable = true)
```

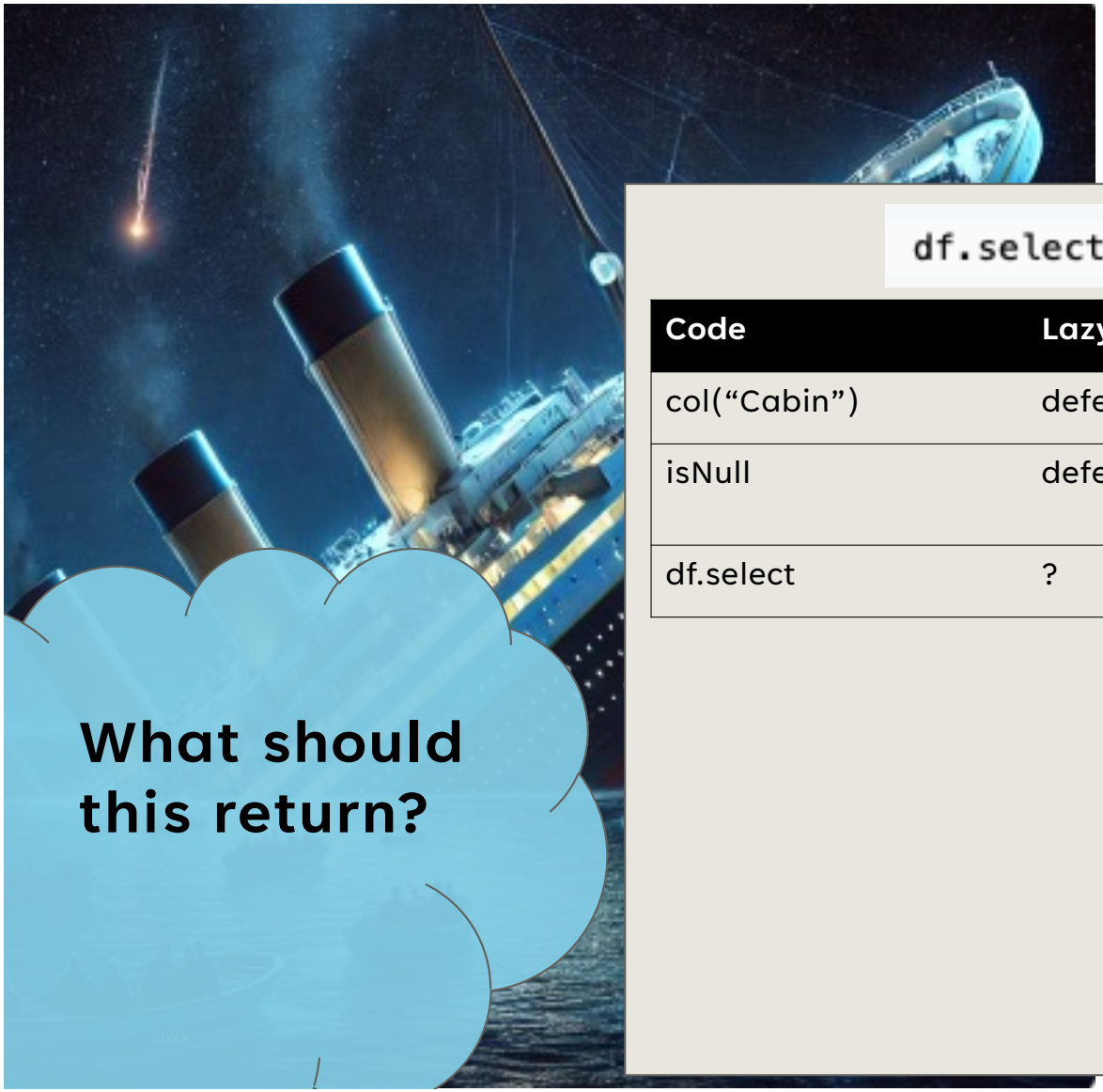


Every Spark operation is either a transformation or an operation.

TITANIC DATASET

```
df.select([count(when(col(c).isNull(), c)).alias(c) for c in df.columns]).show()
```

Type	Description	Execution time
Transformation	Defines an operation that is added to the plan. No need to specify on what data the plan will be executed while building the plan.	Deferred
Action	Executes all transformations in the the plan on the specified data.	Immediate



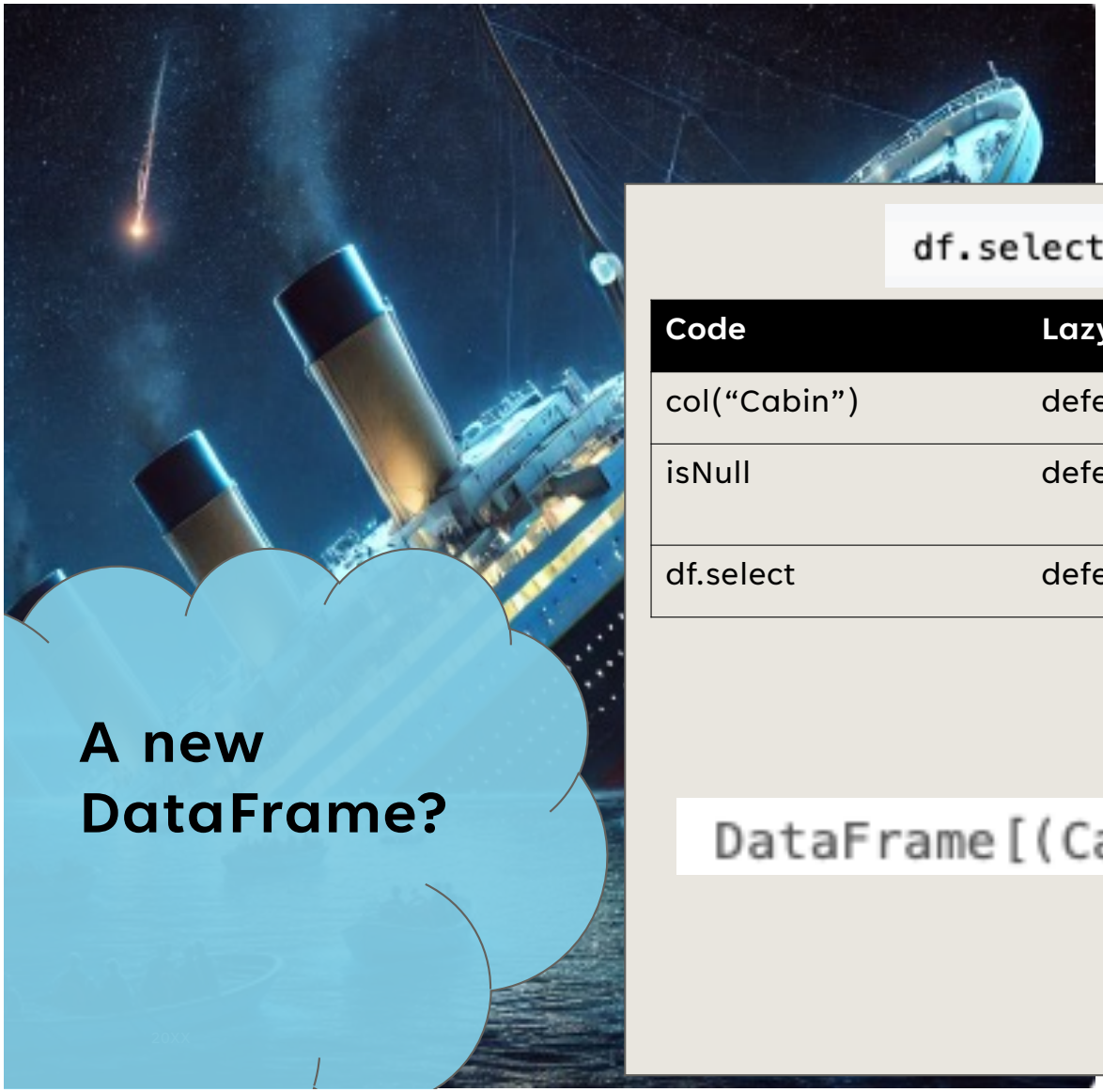
What should this return?

TITANIC DATASET

```
df.select(col("Cabin").isNull())
```

Code	Lazy?	Description
col(“Cabin”)	deferred	Will select column *c* when executed
isNull	deferred	Transformation to return true for each field that is null.
df.select	?	Selectsa column on Dataframe *df*

?



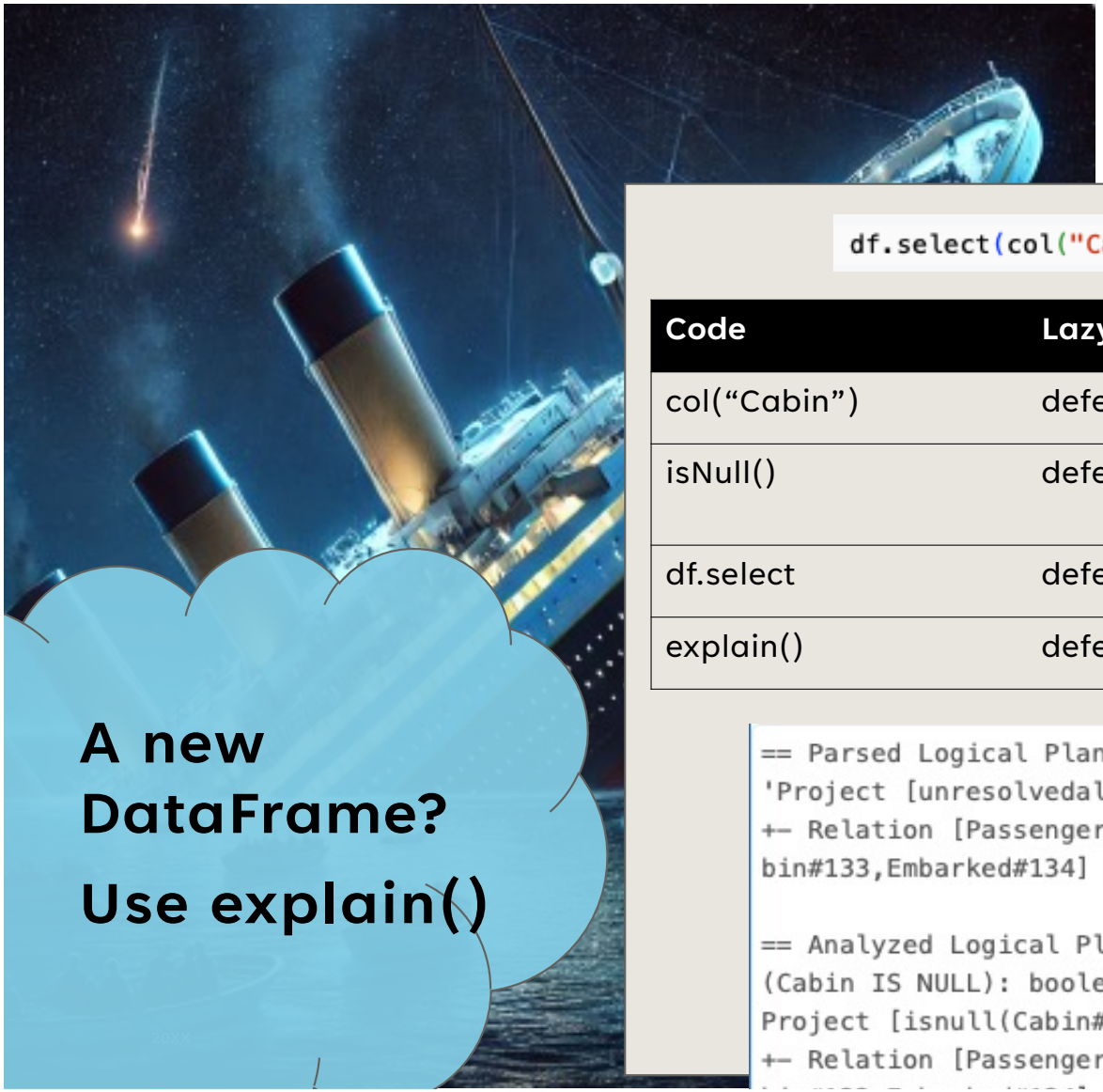
A new DataFrame?

TITANIC DATASET

```
df.select(col("Cabin").isNull())
```

Code	Lazy?	Description
col(“Cabin”)	deferred	Will select column *c* when executed
isNull	deferred	Transformation to return true for each field that is null.
df.select	deferred	Associates plan with Dataframe *df*

```
DataFrame[(Cabin IS NULL): boolean]
```



TITANIC DATASET

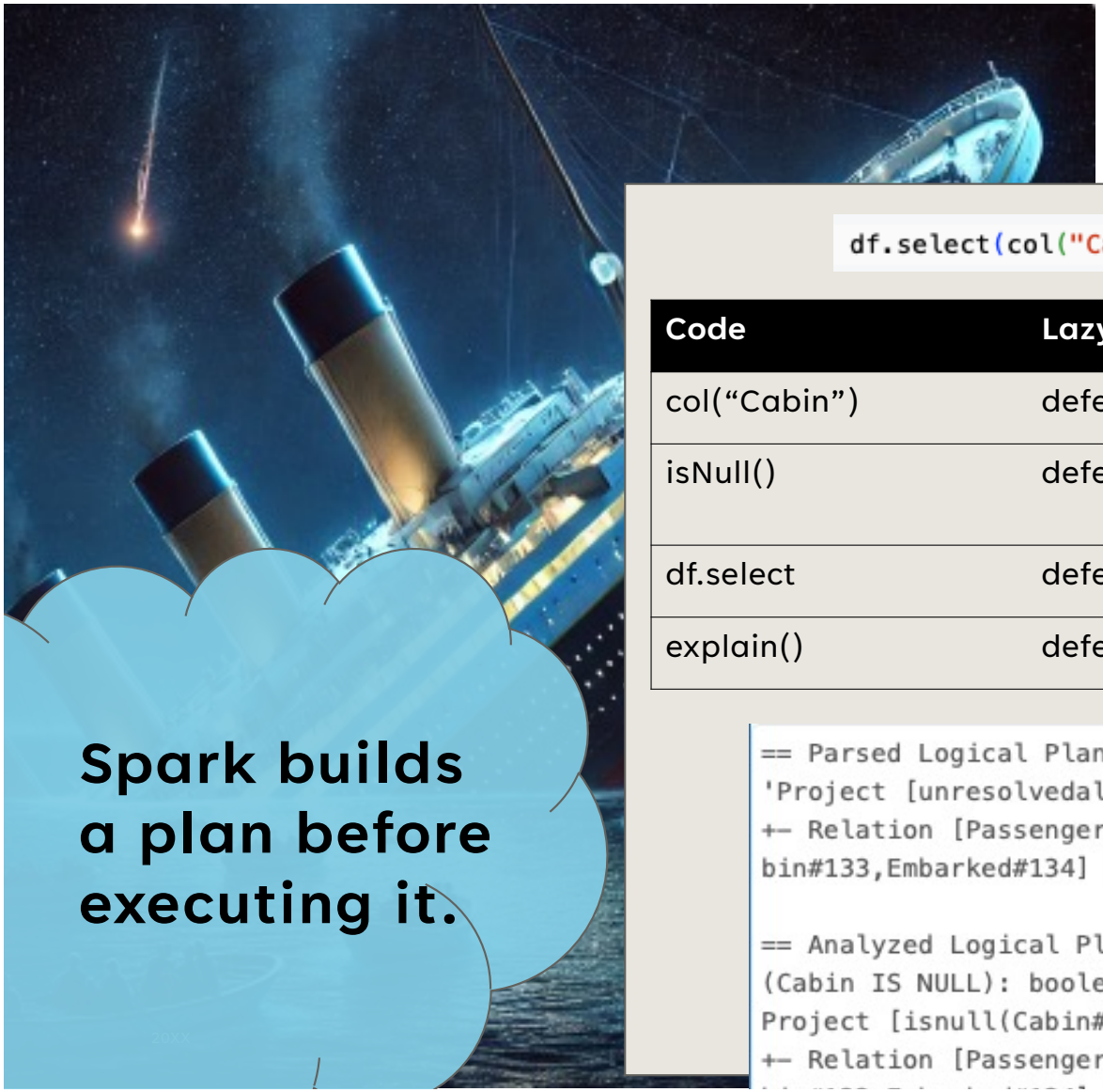
A new DataFrame?

Use explain()

```
df.select(col("Cabin").isNull()).explain(extended=True)
```

Code	Lazy?	Description
col(“Cabin”)	deferred	Will select column *c* when executed
isNull()	deferred	Transformation to return true for each field that is null.
df.select	deferred	Associates plan with Dataframe *df*
explain()	deferred	*Describes the plan.*

```
== Parsed Logical Plan ==
'Project [unresolvedalias(isnull('Cabin), Some(org.apache.spark
+- Relation [PassengerId#123,Survived#124,Pclass#125,Name#126,S
bin#133,Embarked#134] csv

== Analyzed Logical Plan ==
(Cabin IS NULL): boolean
Project [isnull(Cabin#133) AS (Cabin IS NULL)#1686]
+- Relation [PassengerId#123,Survived#124,Pclass#125,Name#126,S
```



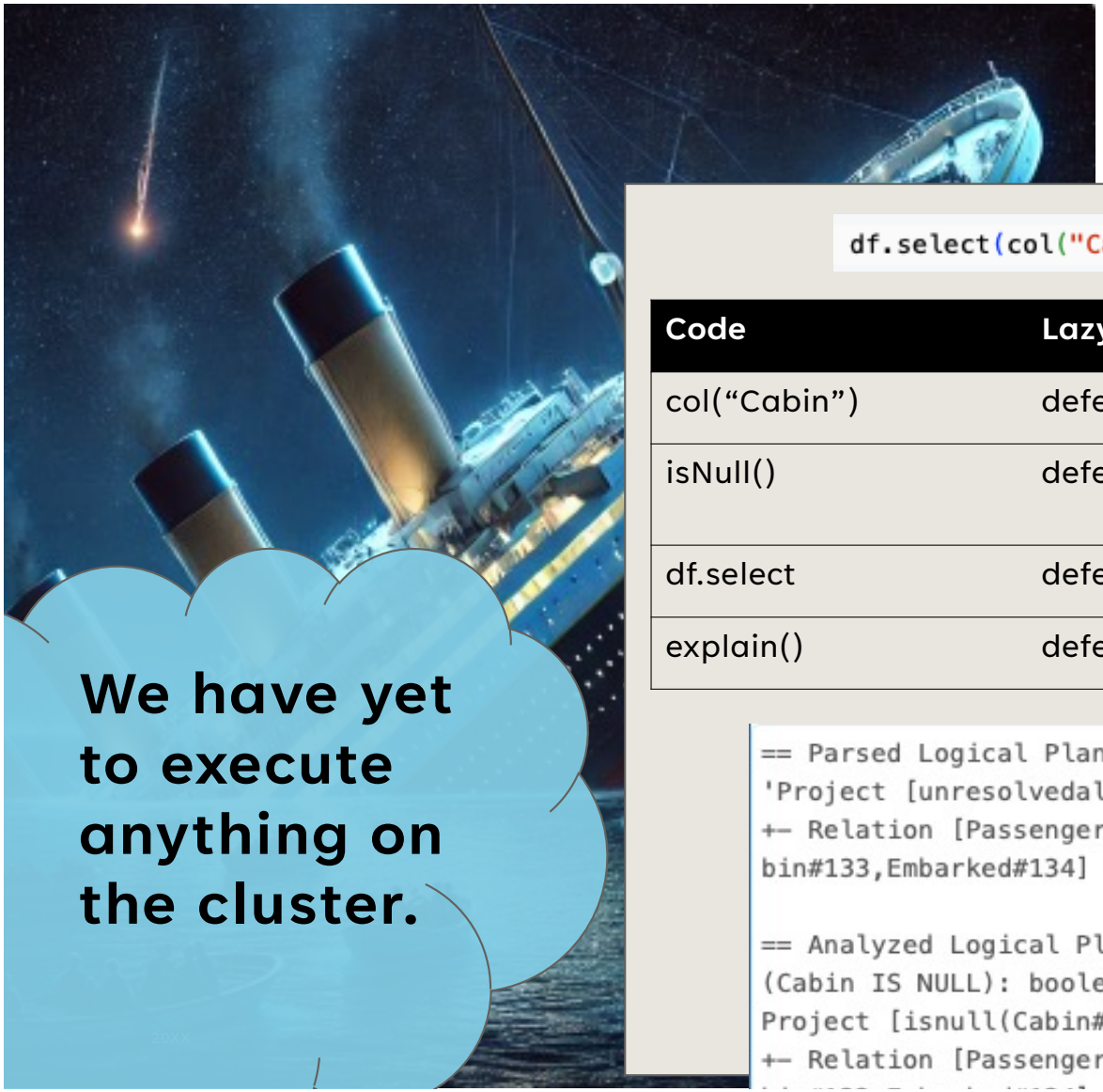
TITANIC DATASET

```
df.select(col("Cabin").isNull()).explain(extended=True)
```

Code	Lazy?	Description
col(“Cabin”)	deferred	Will select column *c* when executed
isNull()	deferred	Transformation to return true for each field that is null.
df.select	deferred	Associates plan with Dataframe *df*
explain()	deferred	*Describes the plan.*

```
== Parsed Logical Plan ==
'Project [unresolvedalias(isnull('Cabin), Some(org.apache.spark
+- Relation [PassengerId#123,Survived#124,Pclass#125,Name#126,S
bin#133,Embarked#134] csv

== Analyzed Logical Plan ==
(Cabin IS NULL): boolean
Project [isnull(Cabin#133) AS (Cabin IS NULL)#1686]
+- Relation [PassengerId#123,Survived#124,Pclass#125,Name#126,S
```

Spark builds a plan before executing it.



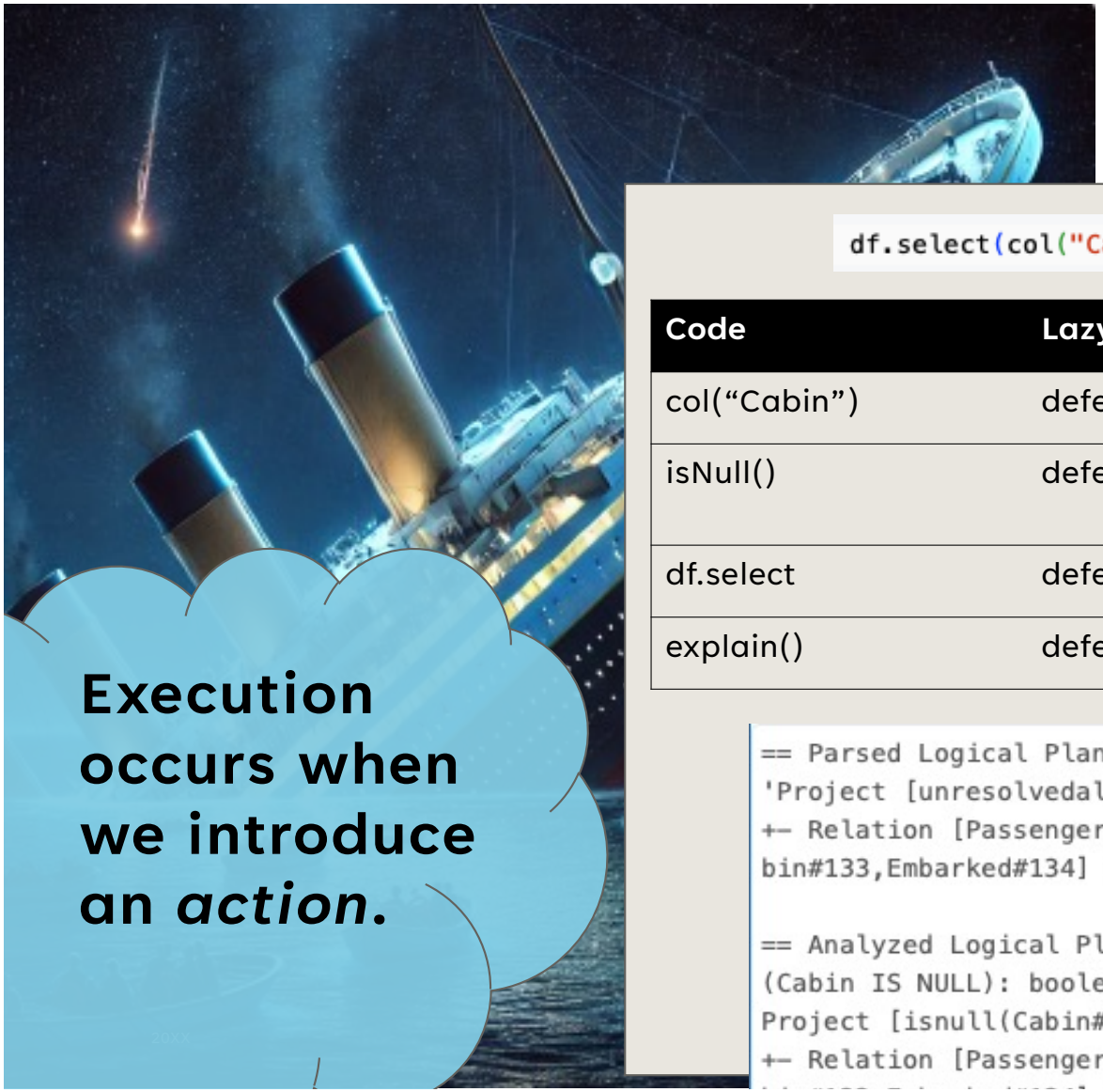
We have yet to execute anything on the cluster.

TITANIC DATASET

```
df.select(col("Cabin").isNull()).explain(extended=True)
```

Code	Lazy?	Description
col(“Cabin”)	deferred	Will select column *c* when executed
isNull()	deferred	Transformation to return true for each field that is null.
df.select	deferred	Associates plan with Dataframe *df*
explain()	deferred	*Describes the plan.*

```
== Parsed Logical Plan ==
'Project [unresolvedalias(isnull('Cabin), Some(org.apache.spark
+- Relation [PassengerId#123,Survived#124,Pclass#125,Name#126,S
bin#133,Embarked#134] csv

== Analyzed Logical Plan ==
(Cabin IS NULL): boolean
Project [isnull(Cabin#133) AS (Cabin IS NULL)#1686]
+- Relation [PassengerId#123,Survived#124,Pclass#125,Name#126,S
```



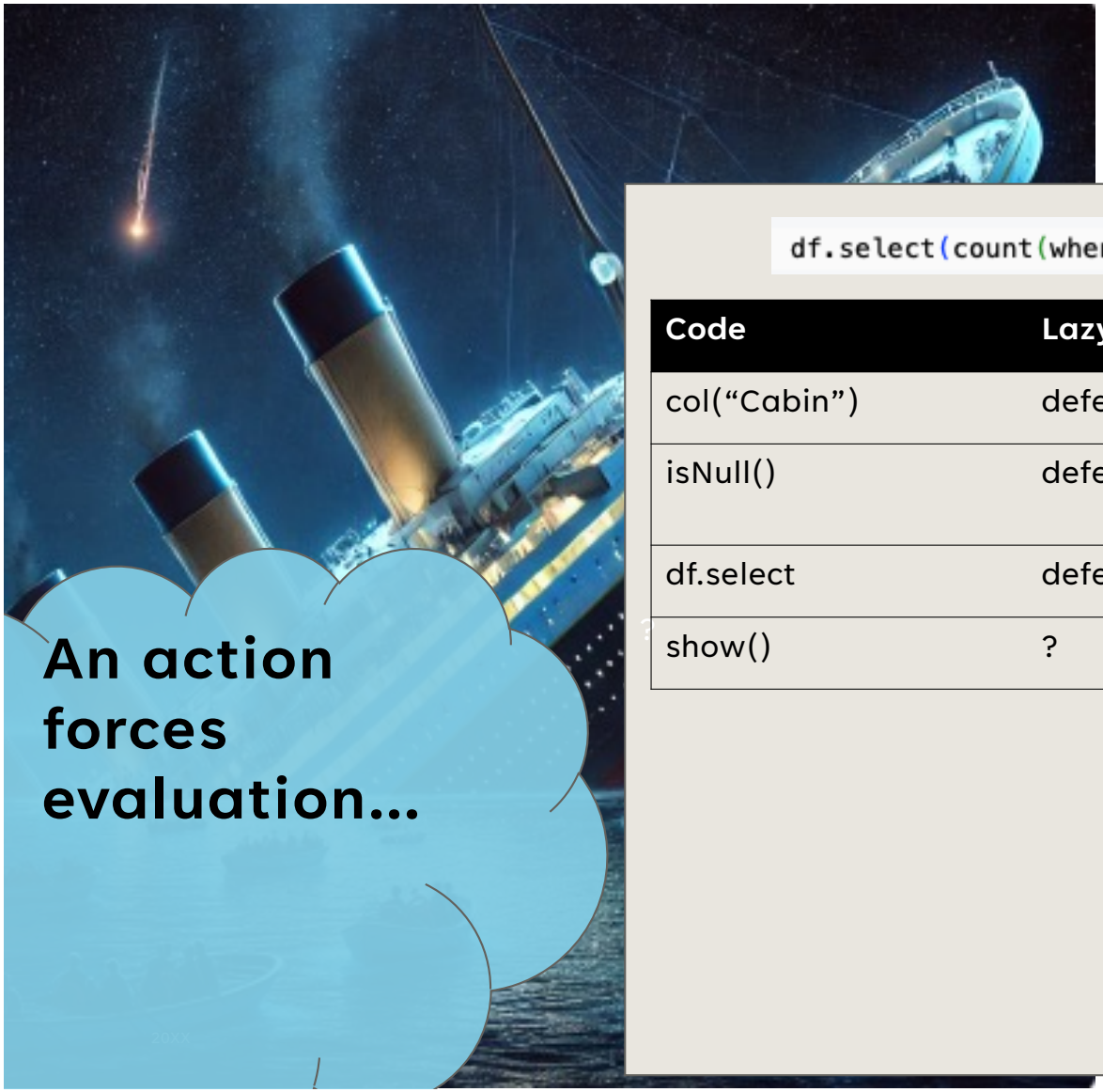
Execution occurs when we introduce an *action*.

TITANIC DATASET

```
df.select(col("Cabin").isNull()).explain(extended=True)
```

Code	Lazy?	Description
col(“Cabin”)	deferred	Will select column *c* when executed
isNull()	deferred	Transformation to return true for each field that is null.
df.select	deferred	Associates plan with Dataframe *df*
explain()	deferred	*Describes the plan.*

```
== Parsed Logical Plan ==
'Project [unresolvedalias(isnull('Cabin), Some(org.apache.spark
+- Relation [PassengerId#123,Survived#124,Pclass#125,Name#126,S
bin#133,Embarked#134] csv

== Analyzed Logical Plan ==
(Cabin IS NULL): boolean
Project [isnull(Cabin#133) AS (Cabin IS NULL)#1686]
+- Relation [PassengerId#123,Survived#124,Pclass#125,Name#126,S
```



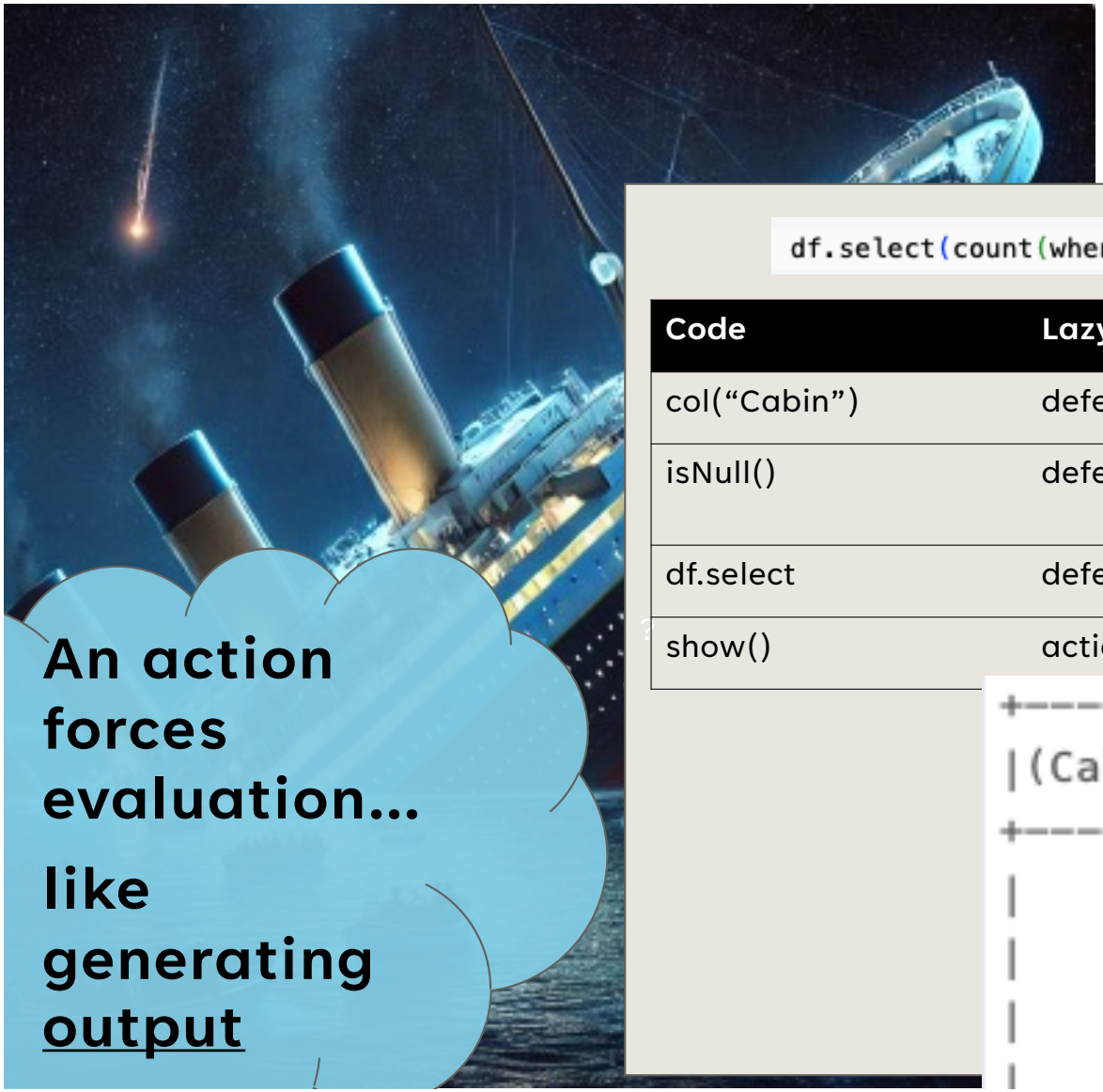
An action forces evaluation...

TITANIC DATASET

```
df.select(count(when(col("Cabin").isNull(), "Cabin"))).show()
```

Code	Lazy?	Description
col(“Cabin”)	deferred	Will select column *c* when executed
isNull()	deferred	Transformation to return true for each field that is null.
df.select	deferred	Associates plan with Dataframe *df*
show()	?	*Show* us the result.

?



TITANIC DATASET

```
df.select(count(when(col("Cabin").isNull(), "Cabin"))).show()
```

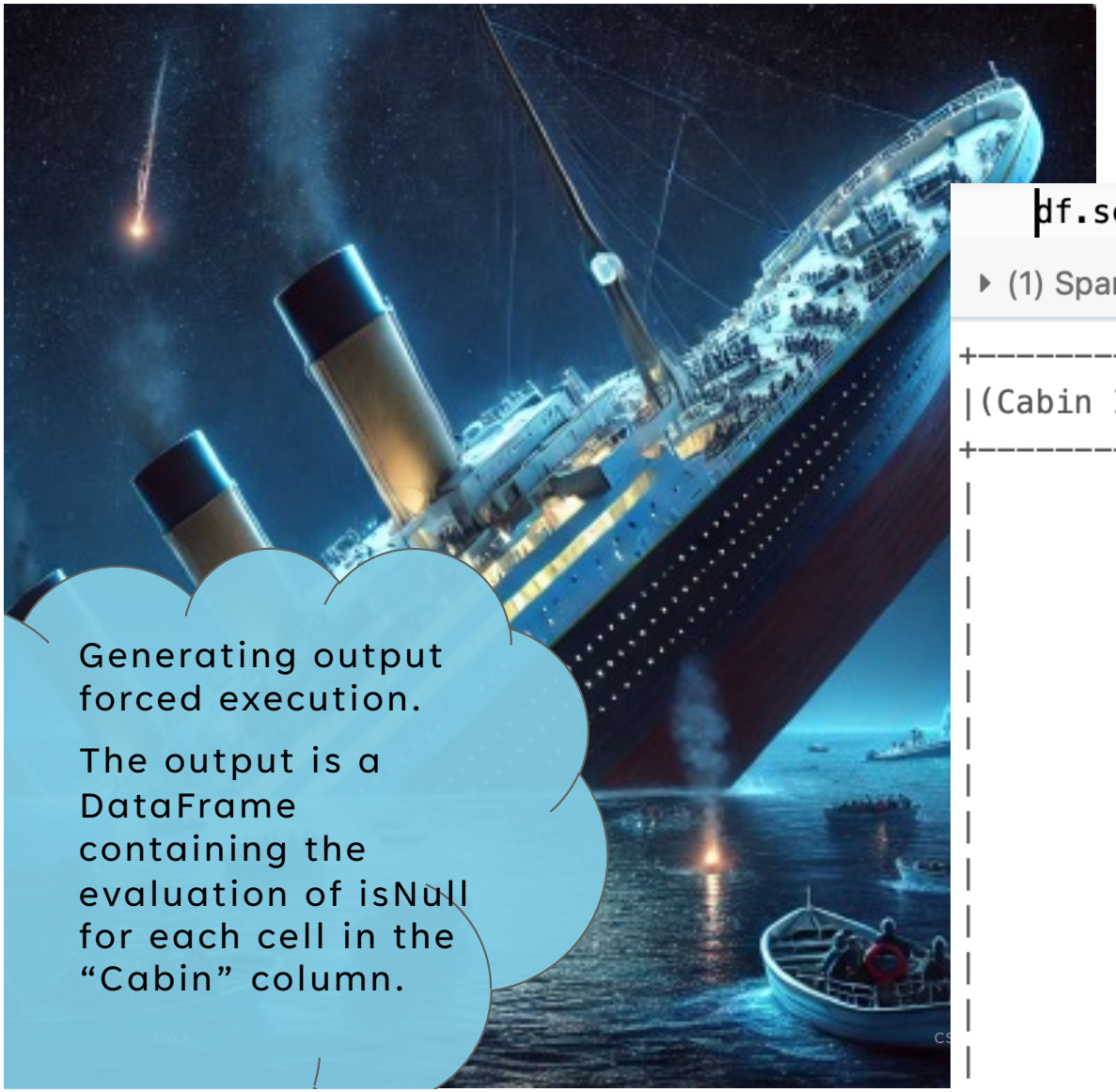
Code	Lazy?	Description
col(“Cabin”)	deferred	Will select column *c* when executed
isNull()	deferred	Transformation to return true for each field that is null.
df.select	deferred	Associates plan with Dataframe *df*
show()	action	*Show* us the result.

```
+---------------+
|(Cabin IS NULL)|
+---------------+
|           true|
|          false|
|           true|
|          false|
```

An action forces evaluation...

like generating <u>output</u>

TITANIC DATASET



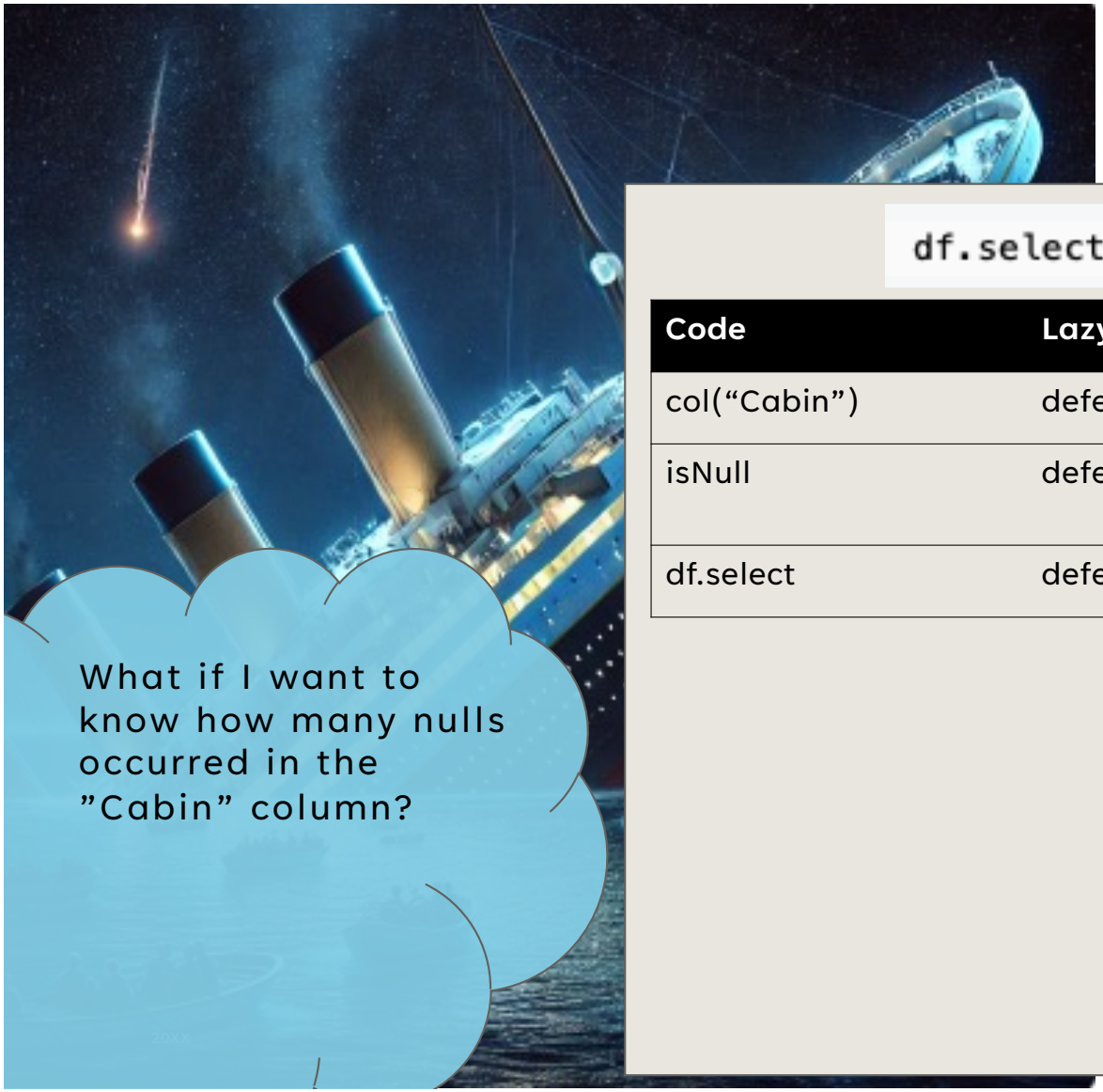
Generating output forced execution.

The output is a DataFrame containing the evaluation of isNull for each cell in the “Cabin” column.

```
df.select(col("Cabin").isNull()).show()
```

▸ (1) Spark Jobs

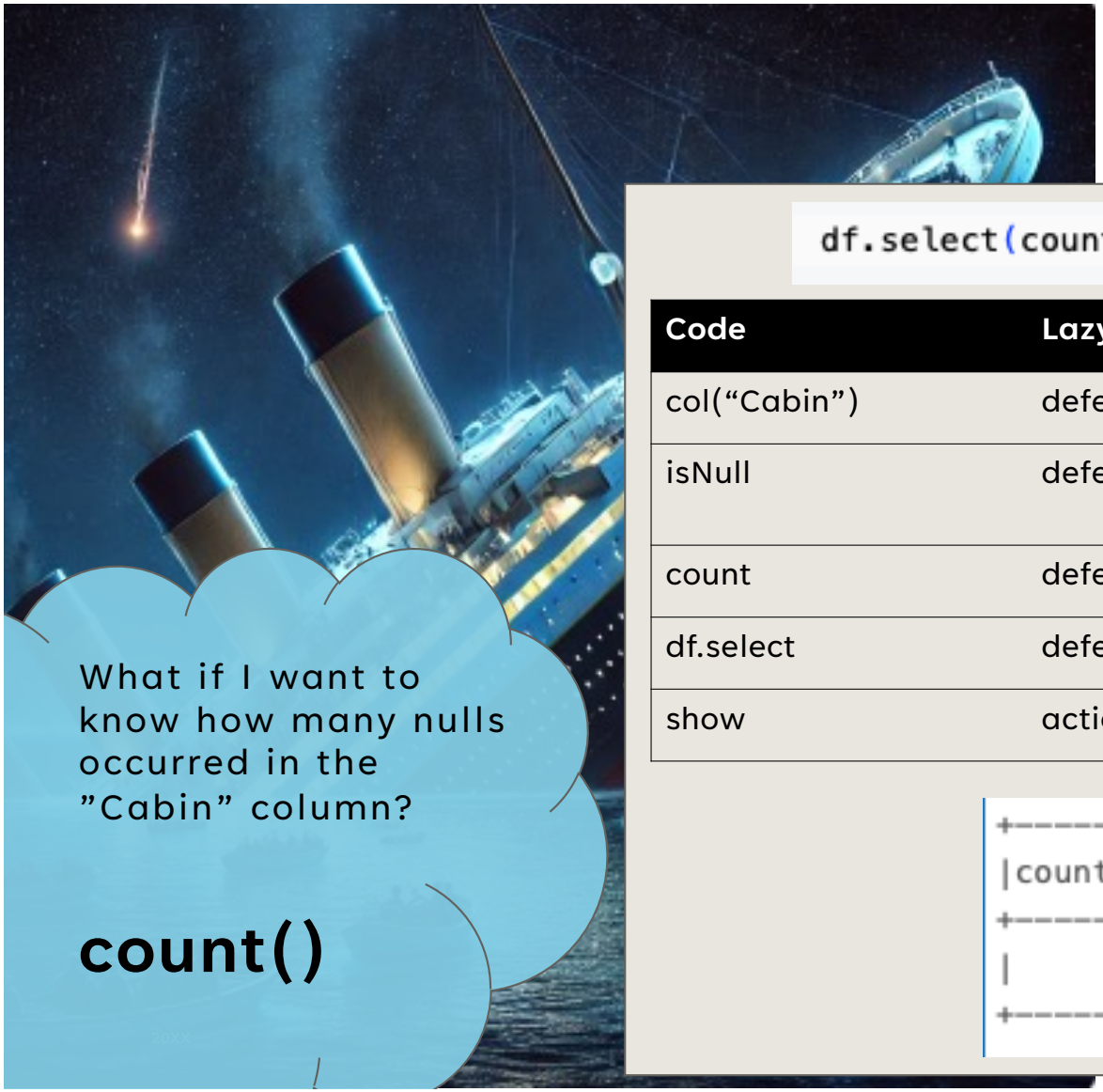
```
+---------------+
|(Cabin IS NULL)|
+---------------+
|           true|
|          false|
|           true|
|          false|
|           true|
|           true|
|          false|
|           true|
|           true|
|           true|
|          false|
|          false|
|           true|
```



TITANIC DATASET

```
df.select(col("Cabin").isNull())
```

Code	Lazy?	Description
col(“Cabin”)	deferred	Will select column *c* when executed
isNull	deferred	Transformation to return true for each field that is null.
df.select	deferred	Associates the plan with Dataframe *df*



What if I want to know how many nulls occurred in the ”Cabin” column?

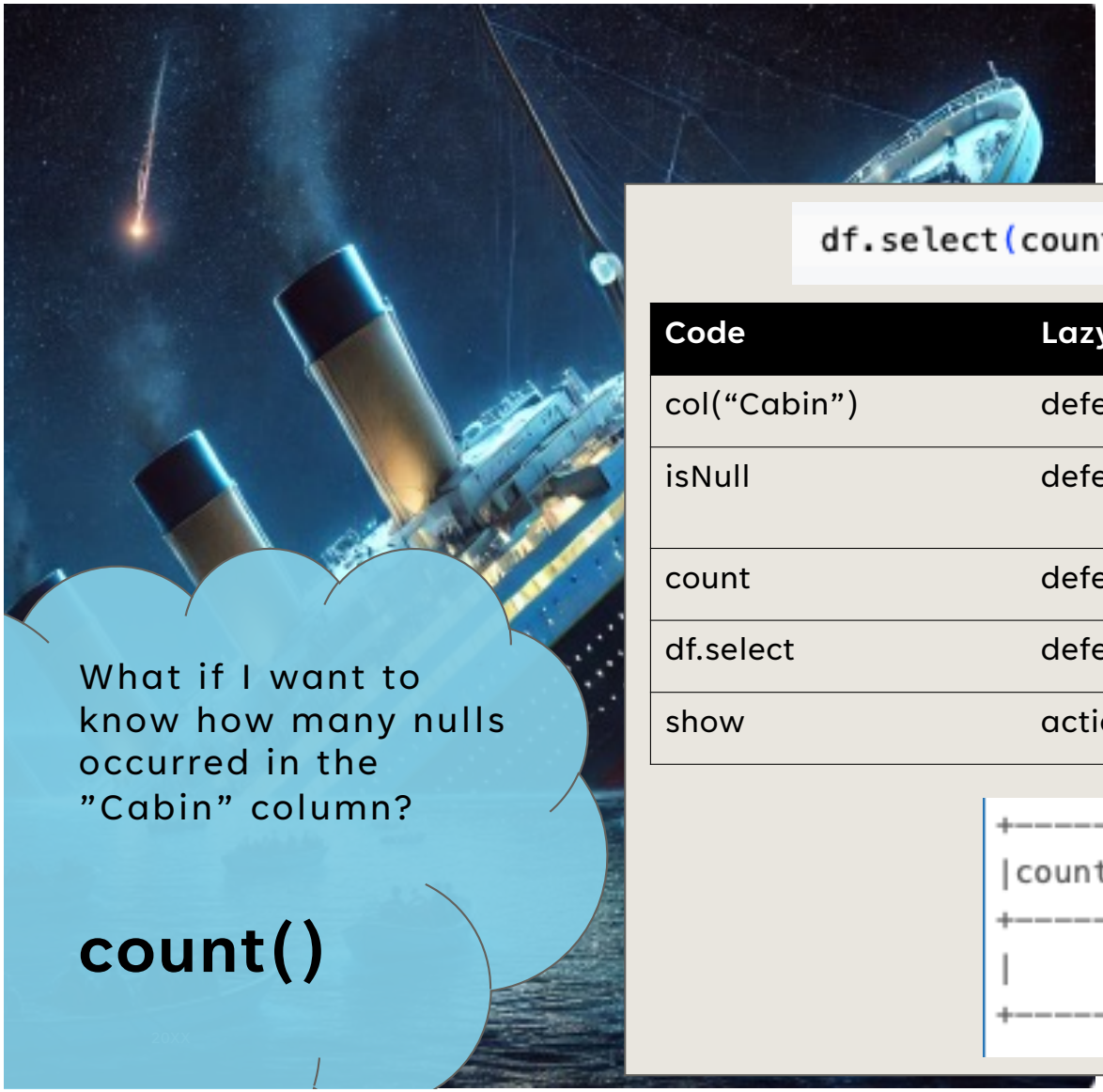
count()

TITANIC DATASET

```
df.select(count(col("Cabin").isNull())).show()
```

Code	Lazy?	Description
col(“Cabin”)	deferred	Will select column *c* when executed
isNull	deferred	Transformation to return true for each field that is null.
count	deferred	Add counting transformation to plan.
df.select	deferred	Associates the plan with Dataframe *df*
show	action	Show the result

```
+-----------------------+
|count((Cabin IS NULL))|
+-----------------------+
|                    891|
+-----------------------+
```



What if I want to know how many nulls occurred in the ”Cabin” column?

count()

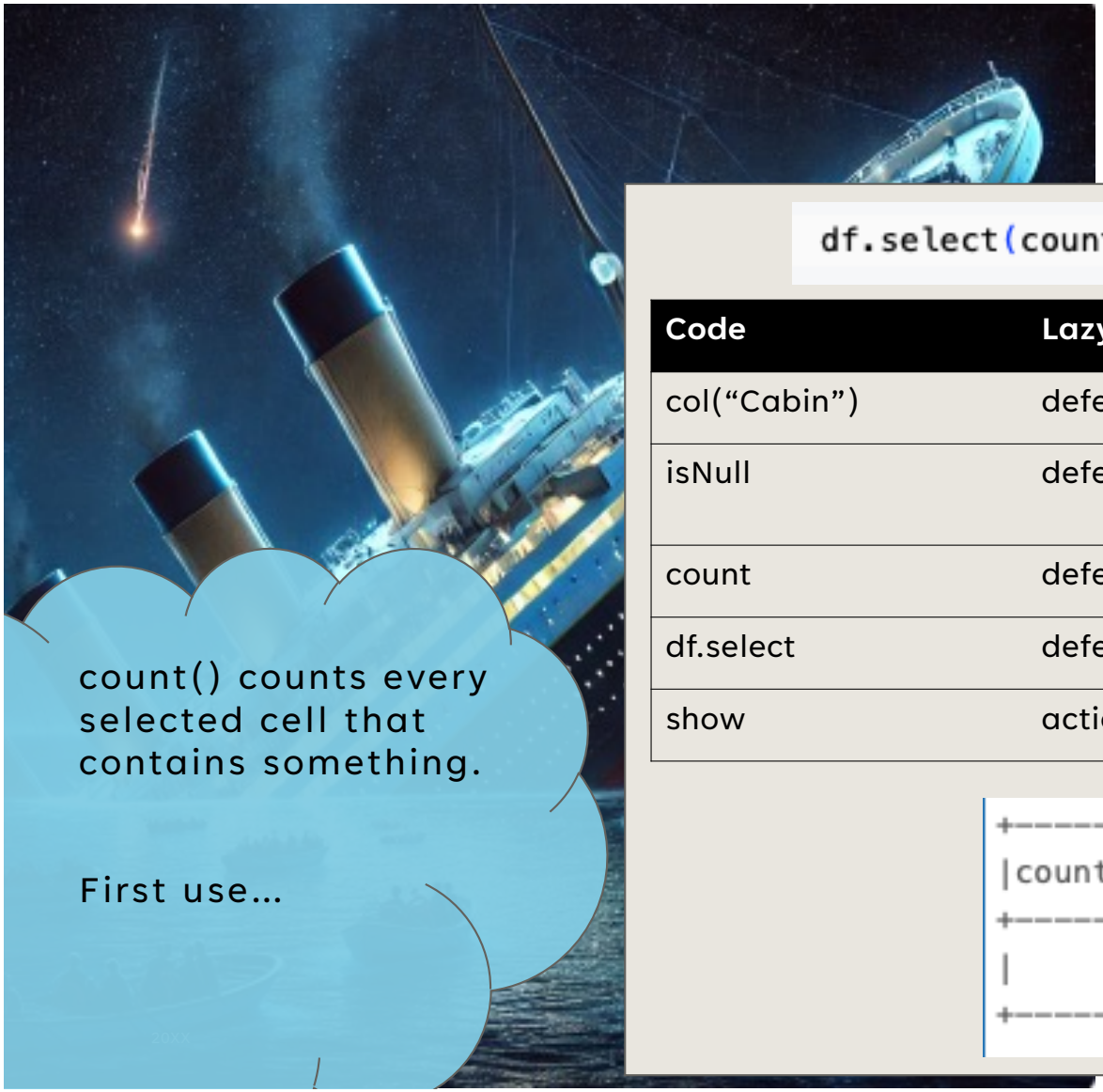
TITANIC DATASET

```
df.select(count(col("Cabin").isNull())).show()
```

Code	Lazy?	Description
col(“Cabin”)	deferred	Will select column *c* when executed
isNull	deferred	Transformation to return true for each field that is null.
count	deferred	Add counting transformation to plan.
df.select	deferred	Associates the plan with Dataframe *df*
show	action	Show the result

```
+-------------------------+
|count((Cabin IS NULL))|
+-------------------------+
|                      891|
+-------------------------+
```

✗ No



count() counts every selected cell that contains something.

First use...

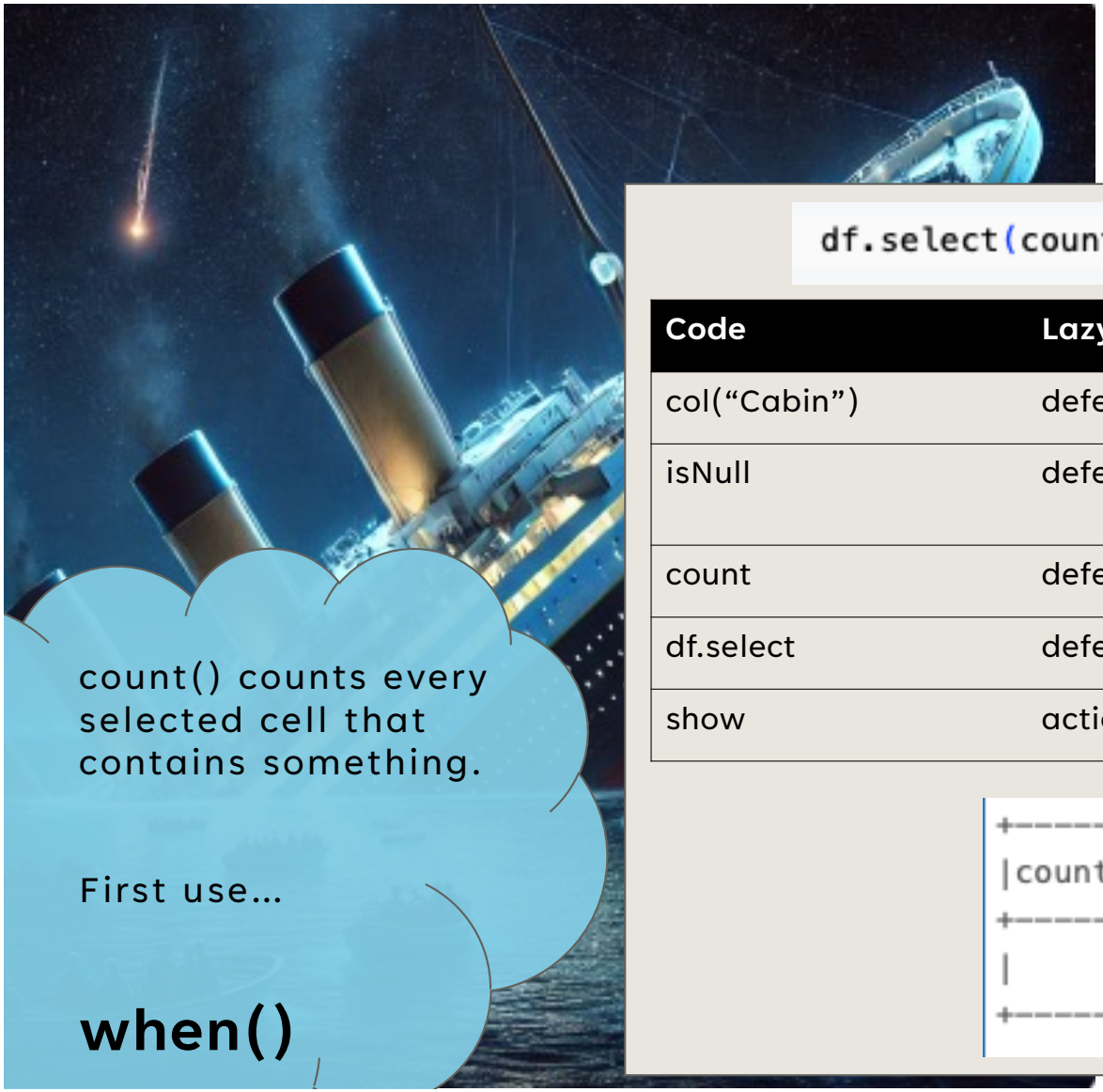
TITANIC DATASET

```
df.select(count(col("Cabin").isNull())).show()
```

Code	Lazy?	Description
col("Cabin")	deferred	Will select column *c* when executed
isNull	deferred	Transformation to return true for each field that is null.
count	deferred	Add counting transformation to plan.
df.select	deferred	Associates the plan with Dataframe *df*
show	action	Show the result

```
+-----------------------+
|count((Cabin IS NULL))|
+-----------------------+
|                    891|
+-----------------------+
```

✗ No



count() counts every selected cell that contains something.

First use...

when()

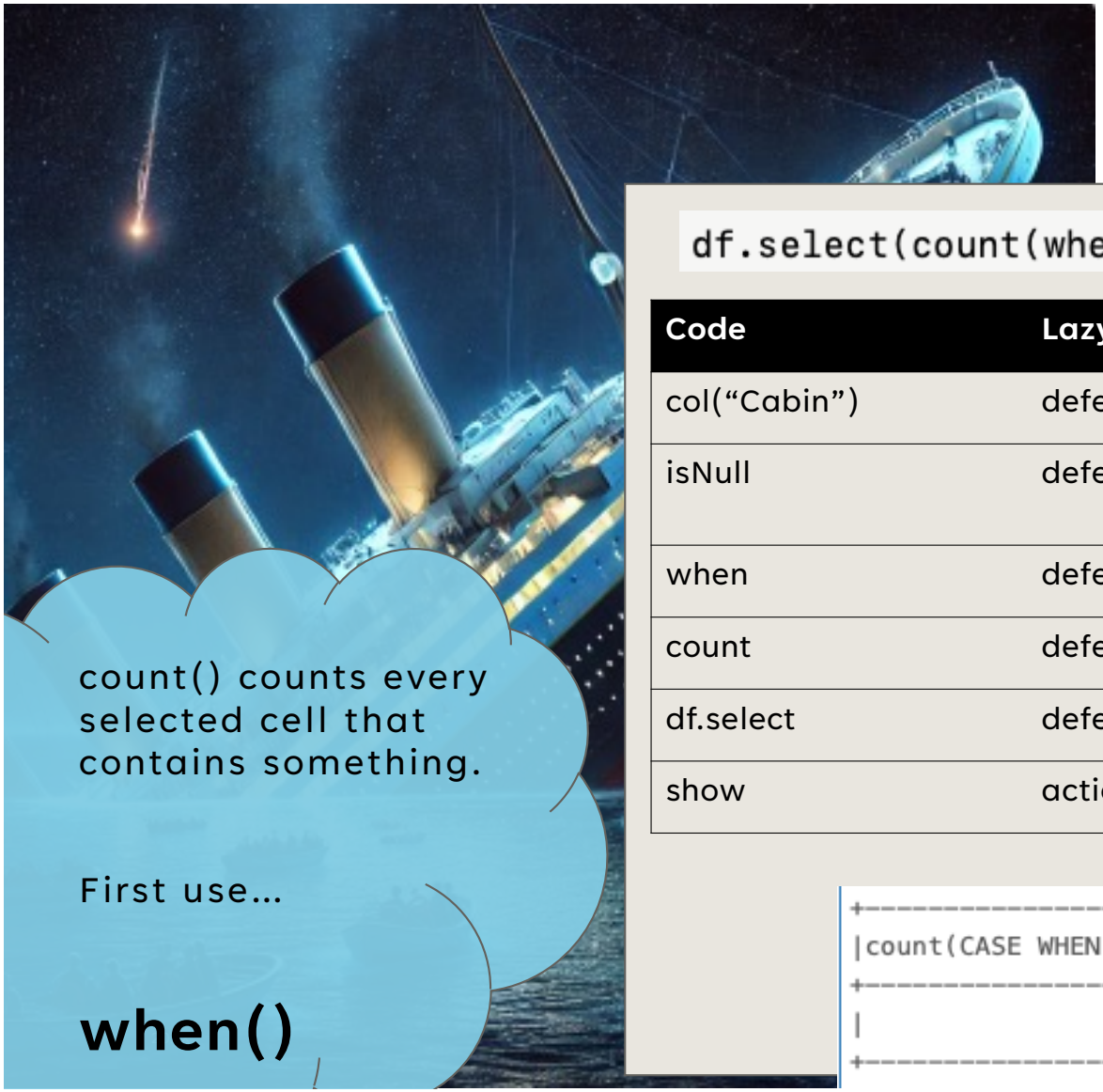
TITANIC DATASET

```
df.select(count(col("Cabin").isNull())).show()
```

Code	Lazy?	Description
col("Cabin")	deferred	Will select column *c* when executed
isNull	deferred	Transformation to return true for each field that is null.
count	deferred	Add counting transformation to plan.
df.select	deferred	Associates the plan with Dataframe *df*
show	action	Show the result

```
+-----------------------+
|count((Cabin IS NULL))|
+-----------------------+
|                    891|
+-----------------------+
```

✗ No



count() counts every selected cell that contains something.

First use...

when()

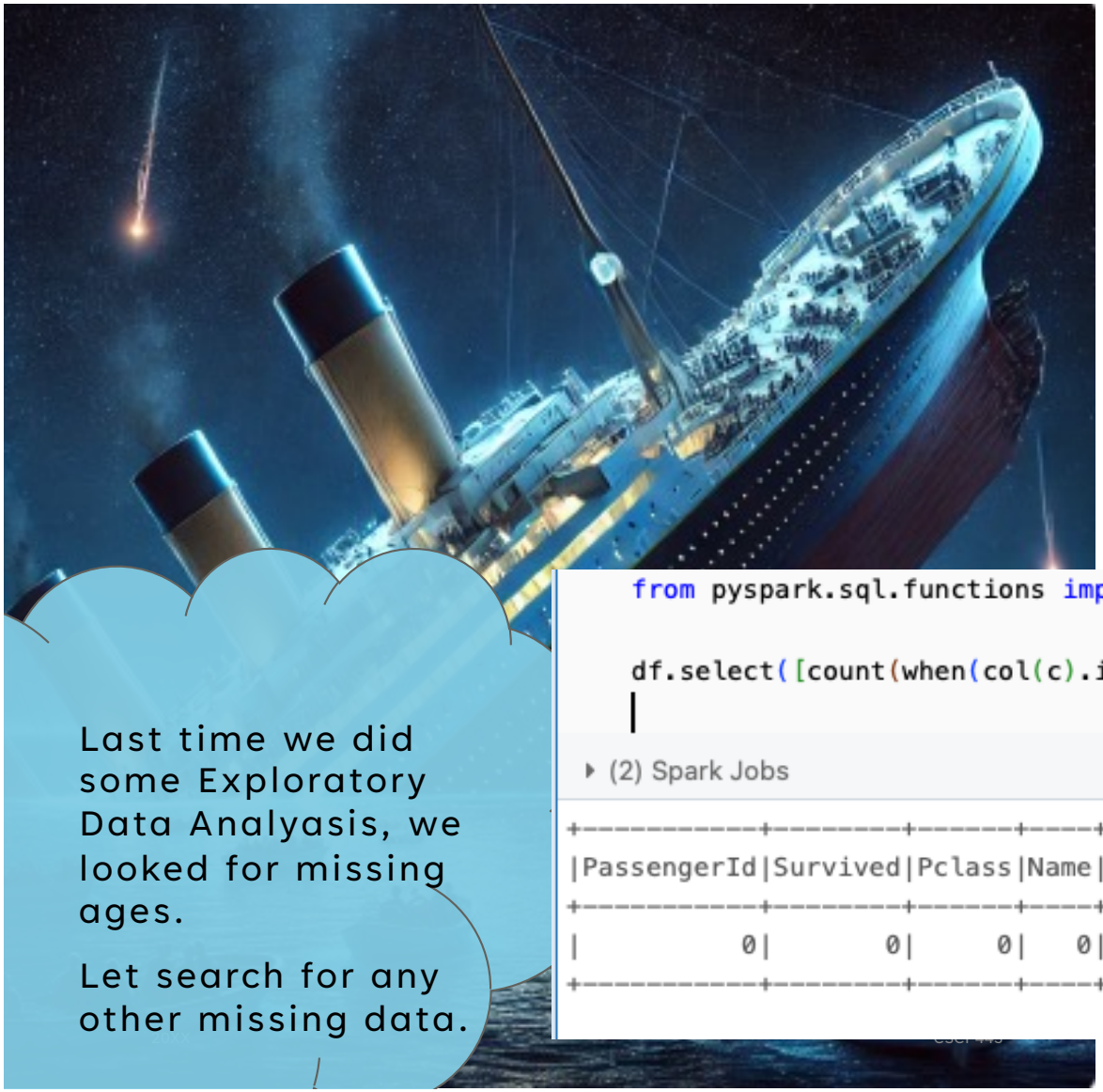
TITANIC DATASET

```
df.select(count(when(col("Cabin").isNull(), 1))).show()
```

Code	Lazy?	Description
col("Cabin")	deferred	Will select column *c* when executed
isNull	deferred	Transformation to return true for each field that is null.
when	deferred	Replace nulls with ones.
count	deferred	Add counting transformation to plan.
df.select	deferred	Associates the plan with Dataframe *df*
show	action	Show the result

```
+-----------------------------------------------------+
|count(CASE WHEN (Cabin IS NULL) THEN Cabin END)|
+-----------------------------------------------------+
|                                                  687|
+-----------------------------------------------------+
```

TITANIC DATASET



Last time we did some Exploratory Data Analysis, we looked for missing ages.

Let search for any other missing data.

```
from pyspark.sql.functions import col, isnan, when, count

df.select([count(when(col(c).isNull(), c)).alias(c) for c in df.columns]).show()
```

▸ (2) Spark Jobs

```
+-----------+--------+------+----+---+---+-----+-----+------+----+-----+--------+
|PassengerId|Survived|Pclass|Name|Sex|Age|SibSp|Parch|Ticket|Fare|Cabin|Embarked|
+-----------+--------+------+----+---+---+-----+-----+------+----+-----+--------+
|          0|       0|     0|   0|  0|177|    0|    0|     0|   0|  687|       2|
+-----------+--------+------+----+---+---+-----+-----+------+----+-----+--------+
```

TRANSFORMATIONS

Added to the plan.

Transformation	Description	Example
`select()`	Select specific columns	`df.select("Name", "Age")`
`filter()/where()`	Filter rows based on a condition	`df.filter(df.Age > 30)`
`withColumn()`	Add or modify a column	`df.withColumn("AgePlusOne", df.Age + 1)`
`drop()`	Remove a column	`df.drop("Ticket")`
`orderBy()`	Sort rows by a column	`df.orderBy("Fare", ascending=False)`
`limit()`	Take the first N rows (still lazy)	`df.limit(10)`
`distinct()`	Remove duplicate rows	`df.select("Pclass").distinct()`

ACTIONS

Execute the plan to generate output.

Action	Description	Example
`show()`	Displays results	`df.show(5)`
`collect()`	Brings all data to the driver (⚠️ not recommended for large data)	`df.collect()`
`count()`	Returns the total number of rows	`df.count()`
`first()`	Returns the first row	`df.first()`
`take(n)`	Returns the first n rows	`df.take(5)`
`describe()`	Computes summary statistics	`df.describe().show()`
`summary()`	More detailed statistics than `describe()`	`df.summary().show()`

RESILIENT DISTRIBUTED DATASETS (RDD)

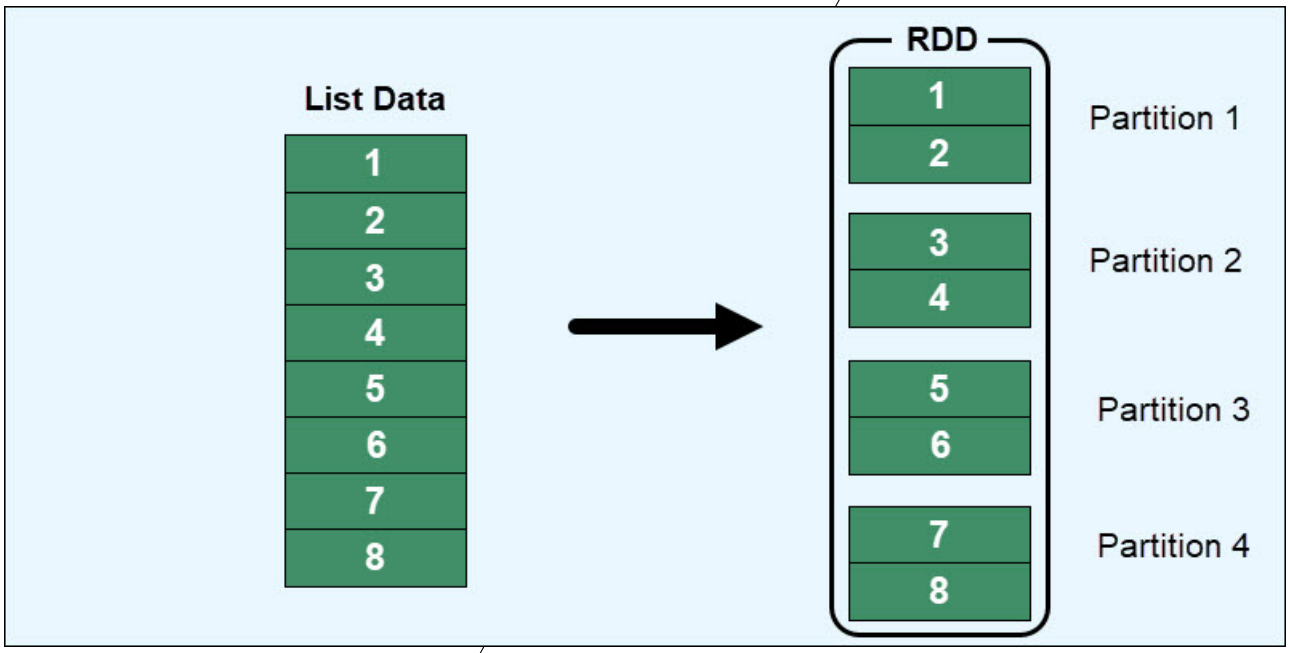
Each DataFrame has an underlying RDD.

Each Row in the RDD holds data for a record.

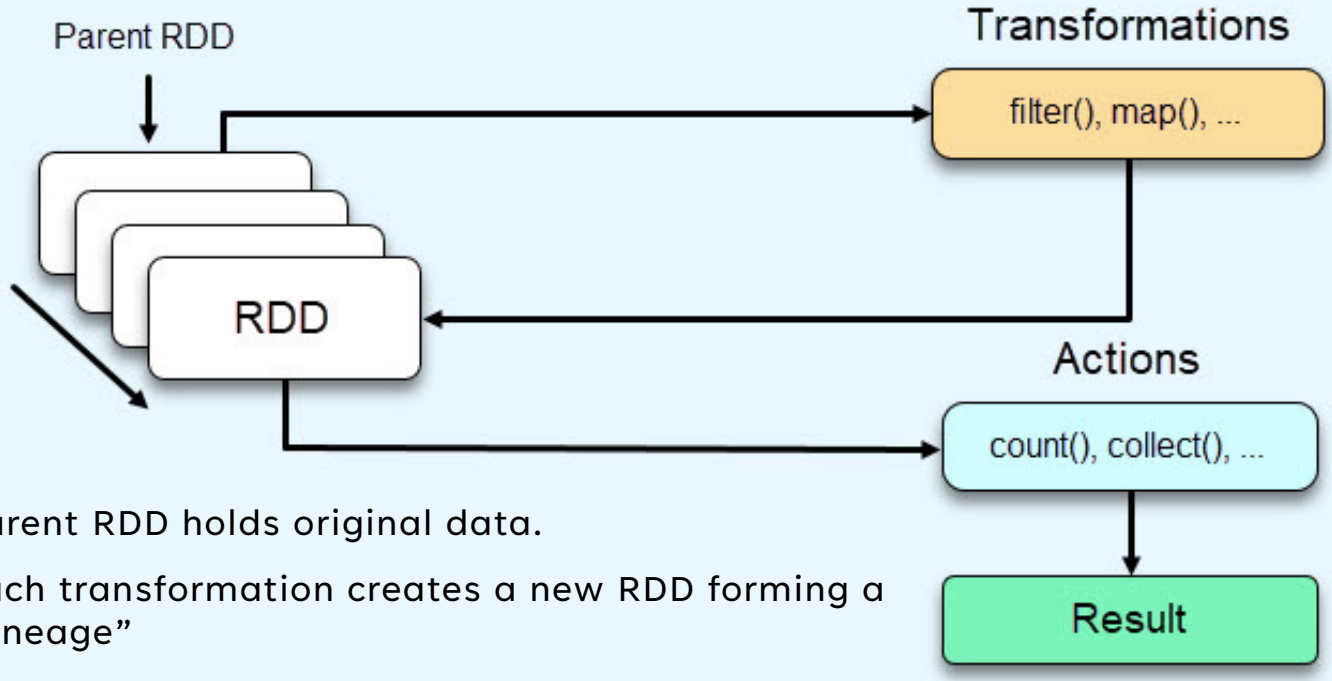
A Row is like an array where each field corresponds to a column in the DataFrame.

RDD partitioned across the cluster.

- Set of rows in each partition.



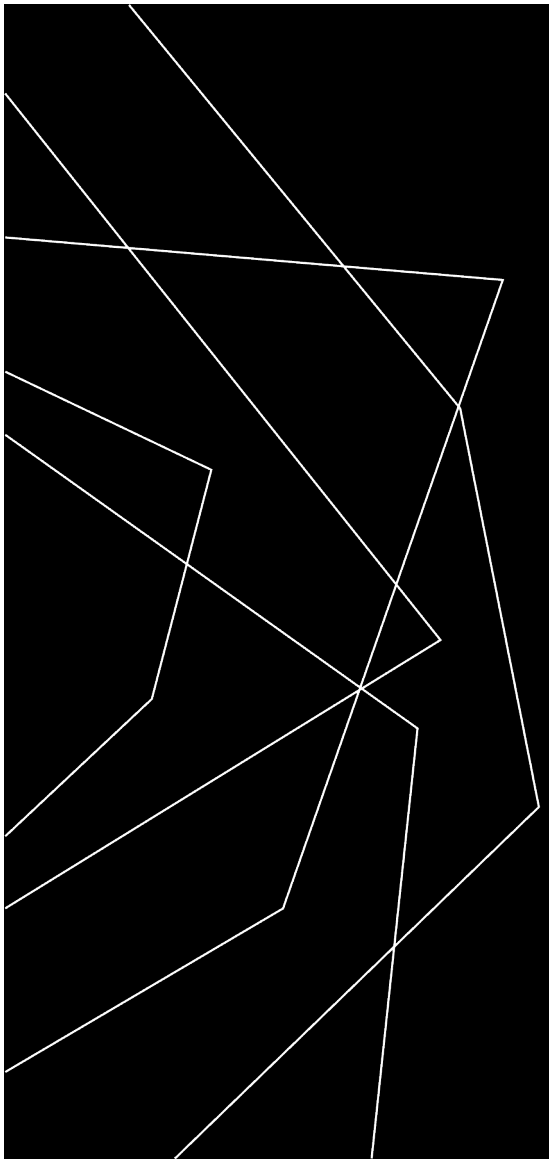
SEQUENCE OF TRANSFORMATIONS



Parent RDD holds original data.

Each transformation creates a new RDD forming a "lineage"

Actions trigger computation.



THANK YOU

David Harrison

Harrison@cs.olemiss.edu