CSCI 443: LECTURE 22 PYSPARK AND E-COMMERCE

Professor David Harrison

DATES OF INTEREST

April 22	HW5 handed out
May 2	HW5 due (Thursday)

worth 6% of your grade which is under the 10% limit for dead week.

May 6-10	Finals week (M-F)
May 7	Final (Tuesday, 4:00pm)

OFFICE HOURS

Tuesday	4:00-5:00 PM
Wednesday	12:30-2:30 PM

BLACKBOARD & GITHUB

Slides and a jupyter notebook for lectures 20 and 21 are on blackboard and in GitHub.

The project is at

https://github.com/dosirrah/CSCI443_AdvancedDataScience

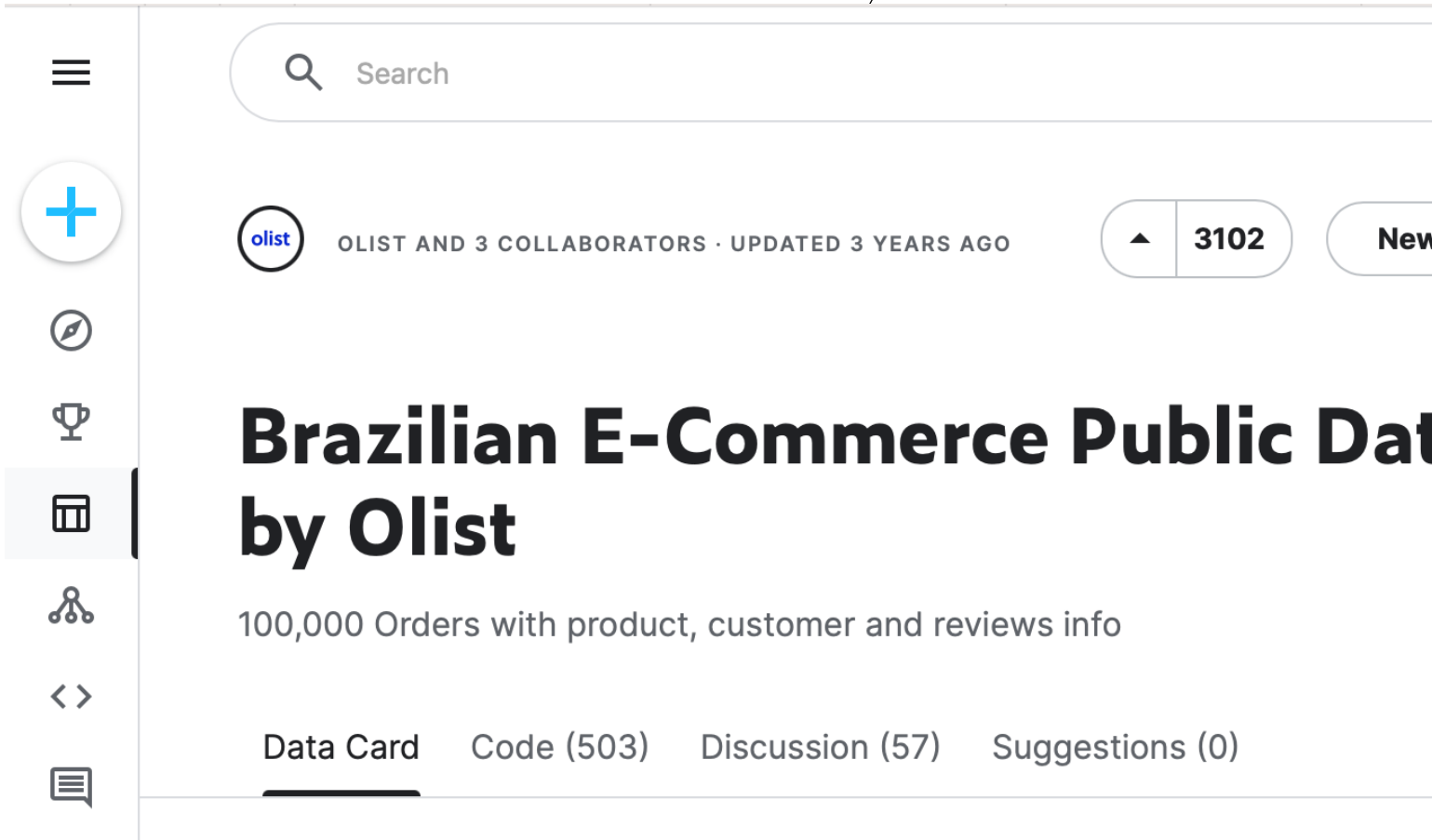
TODAY

- E-commerce Analytics with Olist Dataset
- Databricks
- Spark DataFrames
- RDD
- Catalyst Optimizer

OLIST: BRAZILIAN E-COMMERCE PLATFORM

OLIST

- Dataset for homework 5.
- We will create a data pipeline for creating periodic reports and for anomaly detection



OLIST

https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce

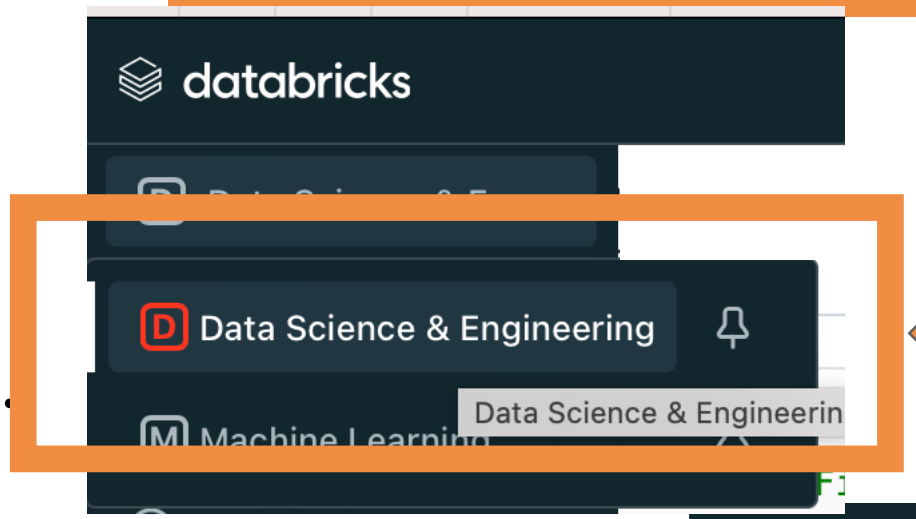
- The dataset is not too large, but large enough to exercise some of the capabilities of Spark.
- The dataset has been committed to github using git's large-file-storage (git-lfs).
- If you pull the class repository it will download a copy of the dataset to your local system. (see hw5/archive)

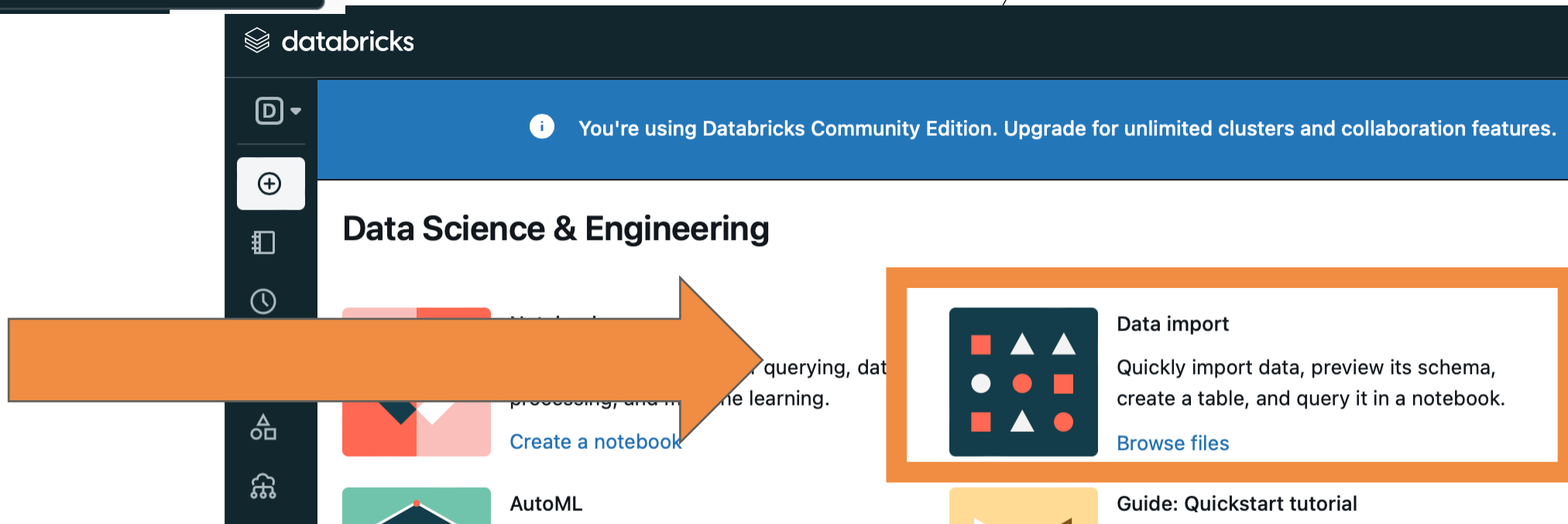
  OR

- Or you can download the dataset from Kaggle.

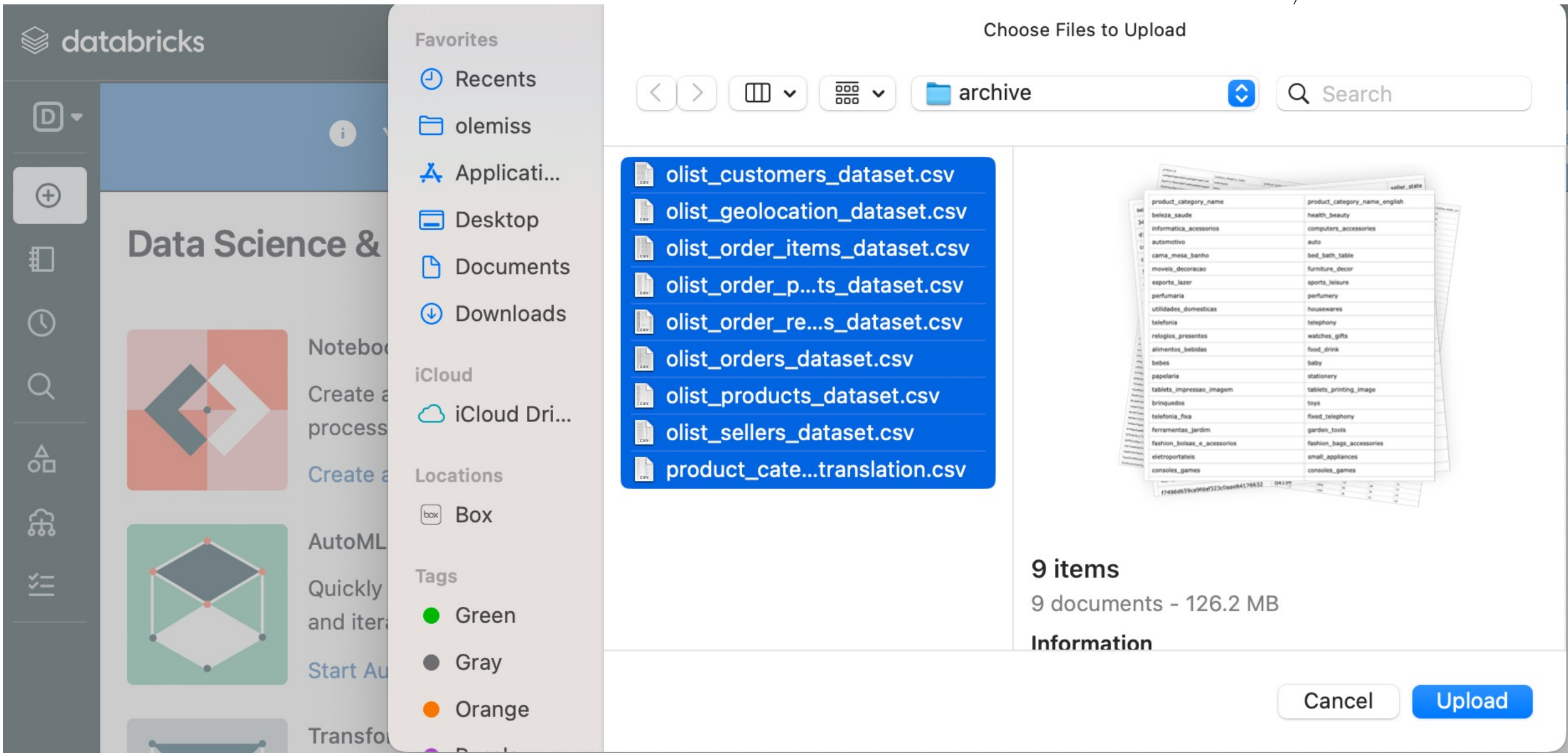
```
dave@FogelmauashsMBP archive % ls -1
olist_customers_dataset.csv
olist_geolocation_dataset.csv
olist_order_items_dataset.csv
olist_order_payments_dataset.csv
olist_order_reviews_dataset.csv
olist_orders_dataset.csv
olist_products_dataset.csv
olist_sellers_dataset.csv
product_category_name_translation.csv
dave@FogelmauashsMBP archive %
```

UPLOADING DATA





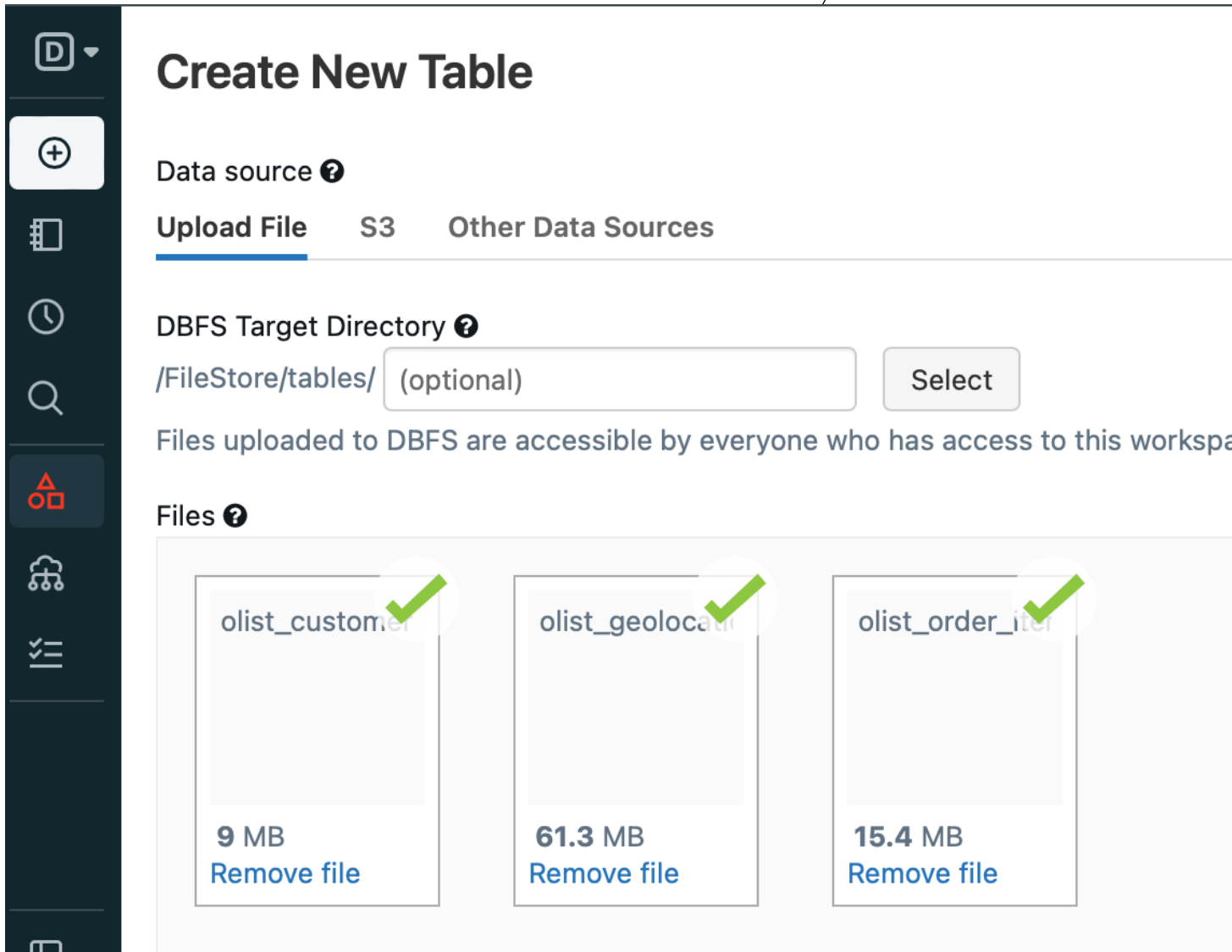
UPLOADING DATA



UPLOADING DATA

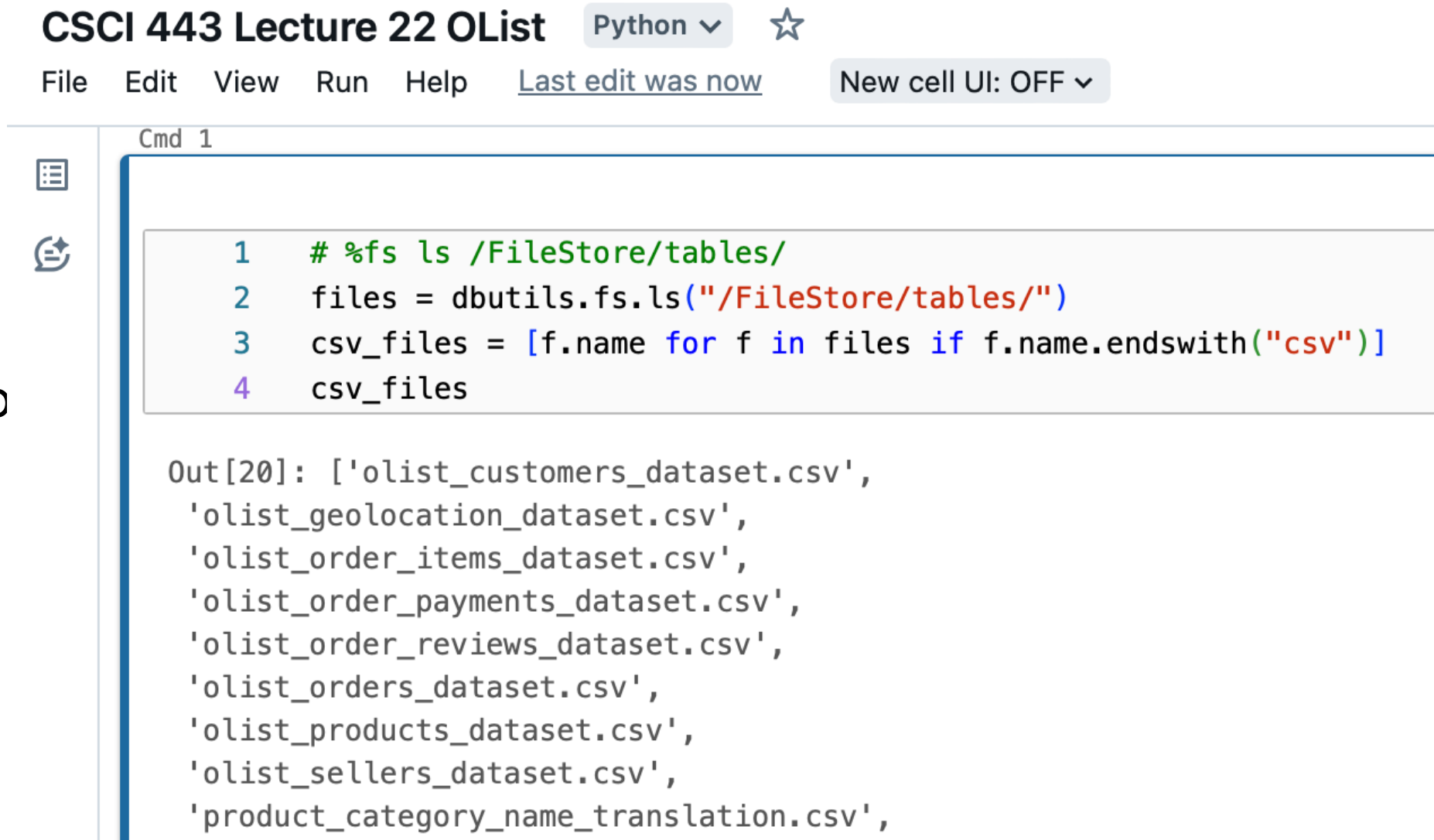
It assumes we are creating a table, but for now I only want to upload the files.

I put them in the default location: /FileStore/tables/



CONFIRMING DATA

- Create a new notebook.
- Use dbutils to list files.



USE A SPARK DATAFRAME

Pandas DataFrame

- Operates in-memory on a single machine.
- Fast for small datasets
- Struggles with large datasets
- Primarily uses Python
- DataFrames are mutable.
  - Can change after creation
  - No Fault Tolerance
  - Data is lost if notebook crashes or interrupted.
- Does not natively support concurrency

Spark DataFrame

- Operates on a distributed system.
  - Data partitioned across multiple machines.
- Handles large datasets
  - Processes data in partitions allowing parallelism across machines.
- Uses lazy evaluation
  - Plans tasks
  - Executes them when an action is triggered.
- Provides APIs in Python, Scala, Java, and R
- DataFrames are immutable
  - Provides fault tolerance

USE A KOALAS TO INTERFACE TO SPARK

- Koalas provides a Pandas-like interface to Spark
- Koalas uses Spark DataFrames under the hood.
- Koalas is as scalable as Spark DataFrames.

USE A KOALAS TO INTERFACE TO SPARK

- Koalas provides a Pandas interface to Spark DataFrames

```
1  import databricks.koalas as ks
```

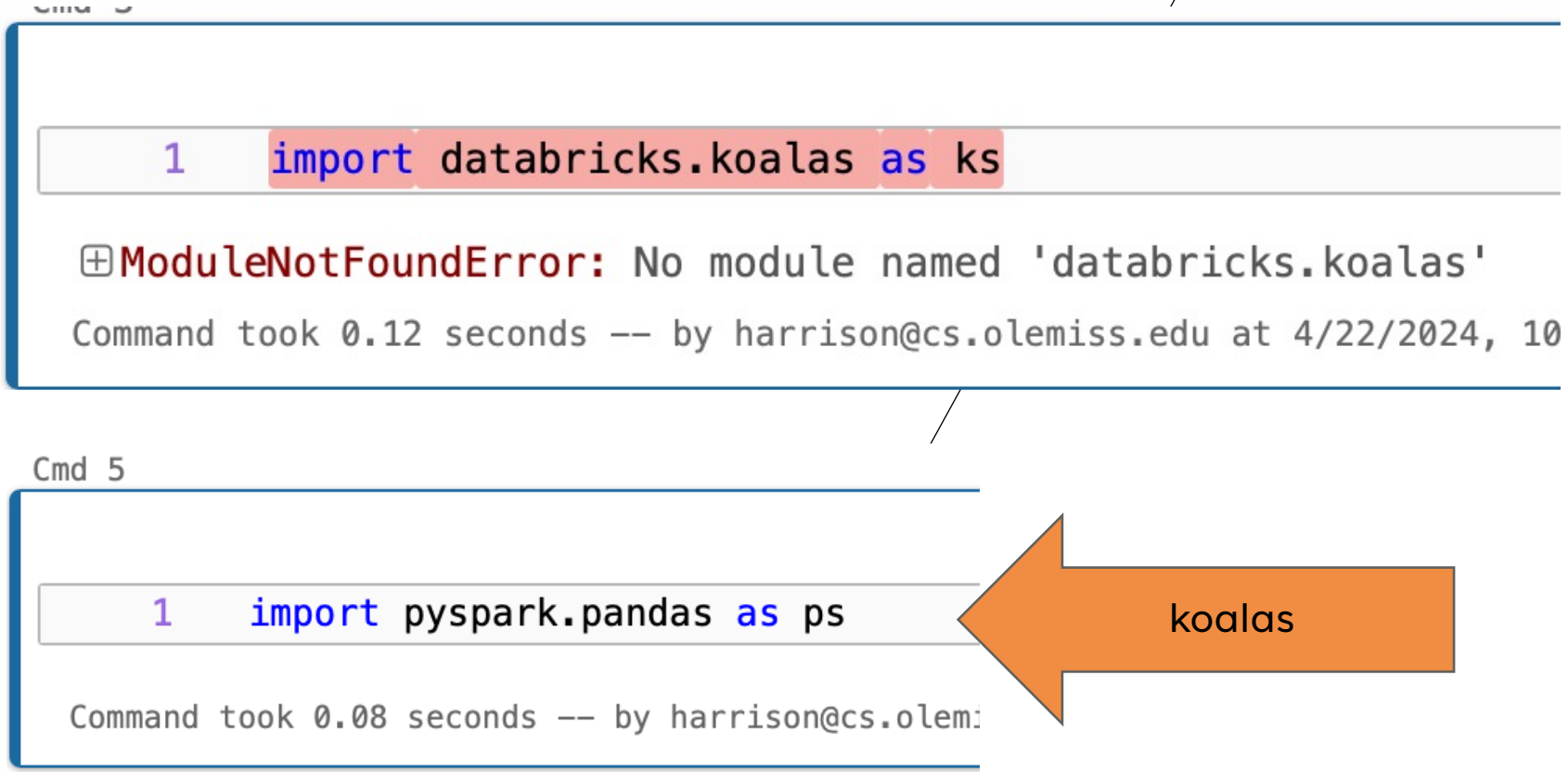
```
ModuleNotFoundError: No module named 'databricks.koalas'
Command took 0.12 seconds -- by harrison@cs.olemiss.edu at 4/22/2024, 10
```

```
1  %pip install koalas
2  import databricks.koalas as ks
```

```
Python interpreter will be restarted.
Requirement already satisfied: koalas in /local_disk0/.ephemeral_nfs/envs
b/python3.9/site-packages (1.8.2)
Requirement already satisfied: pandas>=0.23.2 in /databricks/python3/lib/
Requirement already satisfied: numpy>=1.14 in /databricks/python3/lib/pyt
Requirement already satisfied: pyarrow>=0.10 in /databricks/python3/lib/p
Requirement already satisfied: python-dateutil>=2.8.1 in /databricks/pyth
2->koalas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /databricks/python3/lib/py
(2021.3)
Requirement already satisfied: six>=1.5 in /databricks/python3/lib/python
das>=0.23.2->koalas) (1.16.0)
Python interpreter will be restarted.
```

NEWER KOALAS INTERFACE

- Koalas provides a Pandas interface to Spark DataFrames



1. LOAD THE DATA

- Use pandas read_csv to load csv files from Databricks FileStore

Cmd 5

Python

```
1  import pyspark.pandas as ps
2
3  # Reading a CSV file into a Koalas DataFrame
4  df = ps.read_csv('/FileStore/tables/olist_customers_dataset.csv')
5
6  # Now you can use Pandas-like operations
7  print(df.head())
```

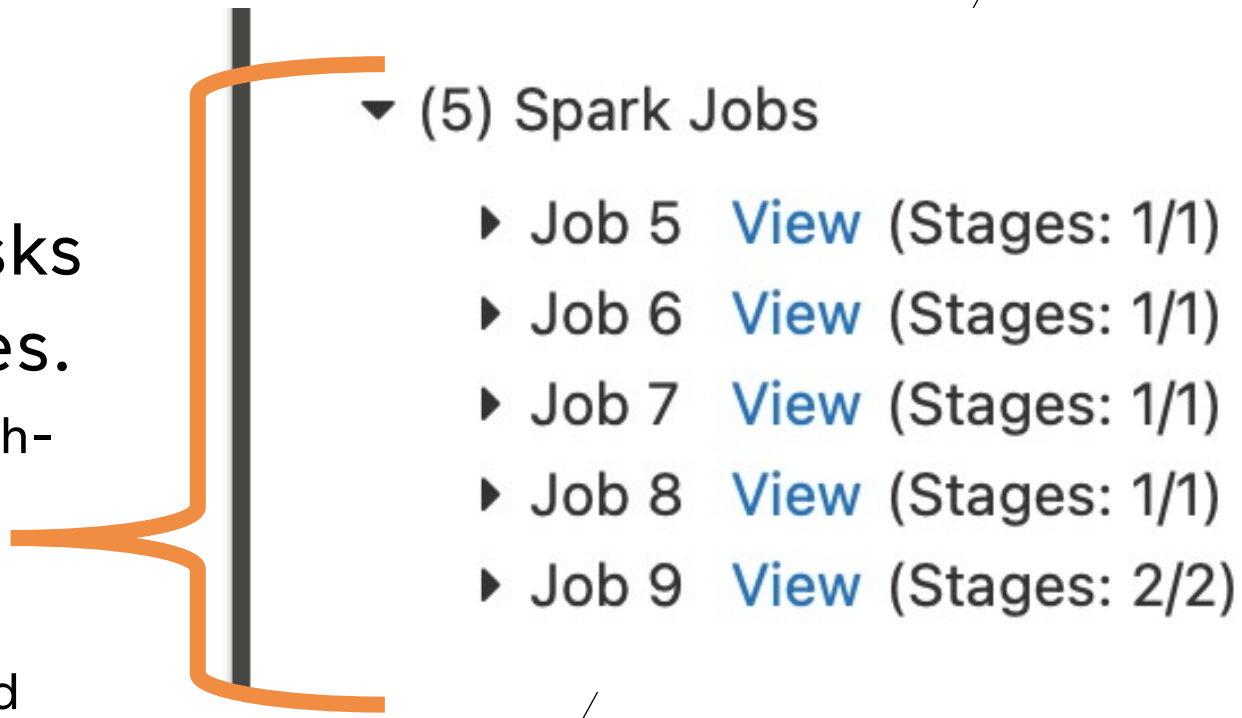
(5) Spark Jobs

- Job 5 View (Stages: 1/1)
- Job 6 View (Stages: 1/1)
- Job 7 View (Stages: 1/1)
- Job 8 View (Stages: 1/1)
- Job 9 View (Stages: 2/2)

```
                        customer_id                customer_unique_id  customer_zip_cod
e_prefix          customer_city customer_state
0  06b8999e2fba1a1fbc88172c00ba8bc7  861eff4711a542e4b93843c6dd7febb0
14409                  franca             SP
1  18955e83d337fd6b2def6b18a428ac77  290c77bc529b7ac935b93aa66c333dc3
```

1. LOAD THE DATA (CONT.)

- Shows Spark running jobs that distribute tasks across the cluster nodes.
  - Each job corresponds to a high-level action (like reading, collecting, outputing)
  - Each stage involves shuffling data (repartitioning as caused by SQL joins, groupBy, ...)



2. DATA EXPLORATION

Python

```
1  customers_df.columns
```

```
Out[9]: Index(['customer_id', 'customer_unique_id', 'customer_zip_code_prefix',
       'customer_city', 'customer_state'],
      dtype='object')
```

Command took 0.05 seconds -- by harrison@cs.olemiss.edu at 4/23/2024, 12:13:29 AM on Olist Cluster

Is there missing data?

```
1  customers_df.isnull().sum()
2
```

▸ (3) Spark Jobs

```
Out[10]: customer_id                0
customer_unique_id         0
customer_zip_code_prefix   0
customer_city              0
customer_state             0
dtype: int64
```

```
1  customers_df.describe()
2
```

▸ (2) Spark Jobs

	customer_zip_code_prefix
count	99441.000000
mean	35137.474583
std	29797.938996
min	1003.000000
25%	11346.000000
50%	24415.000000
75%	58884.000000
max	99990.000000

3. DATA CLEANING

Is there missing data?

- If yes, we need to either using imputing to fill-in missing data

OR

- We remove the records missing data.

NO missing data in this case.

```
1   customers_df.isnull().sum()
2
```

▸ (3) Spark Jobs

```
Out[10]: customer_id                 0
customer_unique_id          0
customer_zip_code_prefix    0
customer_city               0
customer_state              0
dtype: int64
```

4. ANALYZE THE DATA

Let’s perform some

- Simple transformations
- Basic filtering
- Aggregation

Python

```
1  customers_df.columns
```

```
Out[9]: Index(['customer_id', 'customer_unique_id', 'customer_zip_code_prefix',
       'customer_city', 'customer_state'],
      dtype='object')
```

Command took 0.05 seconds -- by harrison@cs.olemiss.edu at 4/23/2024, 12:13:29 AM on Olist Cluster

TRANSFORMATION

Create a new column that categorizes customers by what part of the country they come from based on state.

1. There are 26 states and a federal district.
2. South Region: Paraná, Santa Catarina, and Rio Grande do Sul
3. Southeast Region: Espírito Santo, Minas Gerais, Rio de Janeiro, and São Paulo
4. North Region: Acre, Amapá, Amazonas, Pará, Rondônia, Roraima, and Tocantins
5. Northeast Region: Alagoas, Bahia, Ceará, Maranhão, Paraíba, Pernambuco, Piauí, Rio Grande do Norte, and Sergipe
6. Central-West Region: Goiás, Mato Grosso, Mato Grosso do Sul, and the Federal District

1. **Acre (AC)**
2. **Alagoas (AL)**
3. **Amapá (AP)**
4. **Amazonas (AM)**
5. **Bahia (BA)**
6. **Ceará (CE)**
7. **Distrito Federal (DF)**
8. **Espírito Santo (ES)**
9. **Goiás (GO)**
10. **Maranhão (MA)**
11. **Mato Grosso (MT)**
12. **Mato Grosso do Sul (MS)**
13. **Minas Gerais (MG)**
14. **Pará (PA)**
15. **Paraíba (PB)**
16. **Paraná (PR)**
17. **Pernambuco (PE)**
18. **Piauí (PI)**
19. **Rio de Janeiro (RJ)**
20. **Rio Grande do Norte (RN)**
21. **Rio Grande do Sul (RS)**
22. **Rondônia (RO)**
23. **Roraima (RR)**
24. **Santa Catarina (SC)**
25. **São Paulo (SP)**
26. **Sergipe (SE)**
27. **Tocantins (TO)**

ASIDE: EXAMPLE DIFFERENCE BETWEEN PANDAS AND KOALAS

Pandas DataFrame

- Obtains list of ALL customer states
- Slices first 10,
  - i.e., truncates to the first 10.
- Resonably fast only because Pandas works entirely in memory.
- Although would still be slow for large in-memory data frames.

Pandas on Spark DataFrame (Koalas)

- Lazy evaluates.
  - The slice [:10] doesn't trigger evaluation, buts adds to an exrecution plan.
    collect()
- Doesn't actually execute any queries at this point.

ASIDE: EXAMPLE DIFFERENCE BETWEEN PANDAS AND KOALAS

Pandas on Spark DataFrame

- Steps added to execution plan.
- Note the absence of any “Spark jobs”
- 0.21 second execution time.

ASIDE: EXAMPLE DIFFERENCE BETWEEN PANDAS AND KOALAS

Pandas on Spark DataFrame

- Steps added to execution plan.
- Note the absence of any “Spark jobs”
- 0.21 second execution time.

Spark jobs!!!!

- print() forces execution of the execution plan.
- 2.32 second execution time.

```
states = customers_df["customer_state"][:10]
```

```
Command took 0.21 seconds -- by harrison@cs.olemiss.edu at
```

```
Cmd 12
```

```
print(states)
```

```
▸ (3) Spark Jobs

0    SP
1    SP
2    SP
3    SP
4    SP
5    SC
6    SP
7    MG
8    PR
9    MG
Name: customer_state, dtype: object

Command took 2.32 seconds -- by harrison@cs.olemiss.edu at
```

ASIDE: MORE CONVENTIONAL WAY OF OBTAINING FIRST 10

Pandas on Spark DataFrame

- head(10) added to execution plan.
- Note the absence of any “Spark jobs”
- 0.23 second execution time.

Spark jobs!!!!

- print() forces execution of the execution plan.
- 1.59 second execution time.

```
states = customers_df["customer_state"].head(10)

Command took 0.23 seconds -- by harrison@cs.olemiss.edu at 4/23/20

Cmd 16

print(states)

▸ (3) Spark Jobs

0    SP
1    SP
2    SP
3    SP
4    SP
5    SC
6    SP
7    MG
8    PR
9    MG
Name: customer_state, dtype: object

Command took 1.59 seconds -- by harrison@cs.olemiss.edu at 4/23/20
```

TRANSFORMATION

Create a new column that categorizes customers by what part of the country they come from based on state.

1. The code checks the two letter state code and returns a region name.
2. apply() calls categorize_state for each row in the column “customer_state” resulting in a “transformation” of the data into a new set of data.

```python
def categorize_state(state):
    # Southeast Region: Espírito Santo, Minas Gerais, Rio de Janeiro, and São Paulo
    if state in ['SP', 'RJ', 'ES', 'MG']:
        return 'Southeast'
    # South Region: Paraná, Santa Catarina, and Rio Grande do Sul
    elif state in ['RS', 'SC', 'PR']:
        return 'South'
    # North Region: Acre, Amapá, Amazonas, Pará, Rondônia, Roraima, and Tocantins
    elif state in ['AC', 'AP', 'AM', 'PA', 'RO', 'RR', 'TO']:
        return 'North':
    # Northeast Region: Alagoas, Bahia, Ceará, Maranhão, Paraíba, Pernambuco,
    # Piauí, Rio Grande do Norte, and Sergipe
    elif state in ['AL', 'BA', 'CE', 'MA', 'PB', 'PE', 'PI', 'RN', 'SE']
        return 'Northeast':
    #Central-West Region: Goiás, Mato Grosso, Mato Grosso do Sul, and the Federal District
    else:
        return "Central-West"

customers_df['region'] = customers_df['customer_state'].apply(categorize_state)
```

- Any call that creates new data is called a *transformation*.
- Transformations create new data but cannot modify the underlying Spark DataFrames.
- In Spark, DataFrames are immutable.
- Pandas-on-Spark wraps a Spark DataFrame and maintains a “lineage” of transformations to that Spark DataFrame.

TRANSFORMATION

Lazy operation again?

apply() adds transformations to the execution plan.

It isn't always clear when an operation triggers an evaluation.

```python
def categorize_state(state):
    # Southeast Region: Espírito Santo, Minas Gerais, Rio de Janeiro, and São Paulo
    if state in ['SP', 'RJ', 'ES', 'MG']:
        return 'Southeast'
    # South Region: Paraná, Santa Catarina, and Rio Grande do Sul
    elif state in ['RS', 'SC', 'PR']:
        return 'South'
    # North Region: Acre, Amapá, Amazonas, Pará, Rondônia, Roraima, and Tocantins
    elif state in ['AC', 'AP', 'AM', 'PA', 'RO', 'RR', 'TO']:
        return 'North':
    # Northeast Region: Alagoas, Bahia, Ceará, Maranhão, Paraíba, Pernambuco,
    # Piauí, Rio Grande do Norte, and Sergipe
    elif state in ['AL', 'BA', 'CE', 'MA', 'PB', 'PE', 'PI', 'RN', 'SE']
        return 'Northeast':
    #Central-West Region: Goiás, Mato Grosso, Mato Grosso do Sul, and the Federal District
    else:
        return "Central-West"

customers_df['region'] = customers_df['customer_state'].apply(categorize_state)
```

TRANSFORMATION

Lazy operation again?

apply() adds transformations to the execution plan.

It isn't always clear when an operation triggers an evaluation.

```
def categorize_state(state):
    # Southeast Region: Espírito Santo, Minas Gerais, Rio de Janeiro, and São Paulo
    if state in ['SP', 'RJ', 'ES', 'MG']:
        return 'Southeast'
    # South Region: Paraná, Santa Catarina, and Rio Grande do Sul
    elif state in ['RS', 'SC', 'PR']:
        return 'South'
    # North Region: Acre, Amapá, Amazonas, Pará, Rondônia, Roraima, and Tocantins
    elif state in ['AC', 'AP', 'AM', 'PA', 'RO', 'RR', 'TO']:
        return 'North'
    # Northeast Region: Alagoas, Bahia, Ceará, Maranhão, Paraíba, Pernambuco,
    # Piauí, Rio Grande do Norte, and Sergipe
    elif state in ['AL', 'BA', 'CE', 'MA', 'PB', 'PE', 'PI', 'RN', 'SE']:
        return 'Northeast'
    #Central-West Region: Goiás, Mato Grosso, Mato Grosso do Sul, and the Federal District
    else:
        return "Central-West"

customers_df['region'] = customers_df['customer_state'].apply(categorize_state)
```

Spark jobs!!!!

▸ (3) Spark Jobs

Command took 2.24 seconds -- by harrison@cs.olemiss.edu at 4/23/2024, 12:35:39 PM on Olist Cluster C

FILTERING

Filter to obtain only customers in the Souteast region.

Cmd 23

```
southeast_customers = customers_df[customers_df['region'] == 'Southeast']
```

Command took 0.27 seconds -- by harrison@cs.olemiss.edu at 4/23/2024, 12:51:01 PM on Olist Cluster C

Cmd 24

```
southeast_customers.head()
```

▸ (3) Spark Jobs

Spark jobs!!!!

	customer_id	customer_unique_id	customer_zip_code_prefix	customer_city	customer_state	region
0	06b8999e2fba1a1fbc88172c00ba8bc7	861eff4711a542e4b93843c6dd7febb0	14409	franca	SP	Southeast
1	18955e83d337fd6b2def6b18a428ac77	290c77bc529b7ac935b93aa66c333dc3	9790	sao bernardo do campo	SP	Southeast
2	4e7b3e00288586ebd08712fdd0374a03	060e732b5b29e8181a18229c7b0b2b5e	1151	sao paulo	SP	Southeast
3	b2b6027bc5c5109e529d4dc6358b12c3	259dac757896d24d7702b9acbbff3f3c	8775	mogi das cruzes	SP	Southeast
4	4f2d8ab171c80ec8364f7c12e35b23ad	345ecd01c38d18a9036ed96c73b8d066	13056	campinas	SP	Southeast

Command took 6.12 seconds -- by harrison@cs.olemiss.edu at 4/23/2024, 12:51:04 PM on Olist Cluster C

AGGREGATION

groupby adds transformation to execution plan.

print(sizes) triggers groupby to shuffle data between partitions.

Spark jobs!!!!

Cmd 20

```
sizes = customers_df.groupby('region').size()
```

Command took 0.16 seconds -- by harrison@cs.olemiss.edu a

Cmd 21

```
print(sizes)
```

▸ (4) Spark Jobs

```
region
South            14148
Southeast        68266
Central-West      5782
North             1851
Northeast         9394
dtype: int64
```

Command took 4.29 seconds -- by harrison@cs.olemiss.edu a

VISUALIZATION

Matplotplib doesn't always play well with Pandas-on-spark.

```
plt.figure(figsize=(8, 5))
plt.bar(region_counts.index, region_counts.values)
plt.show()
```

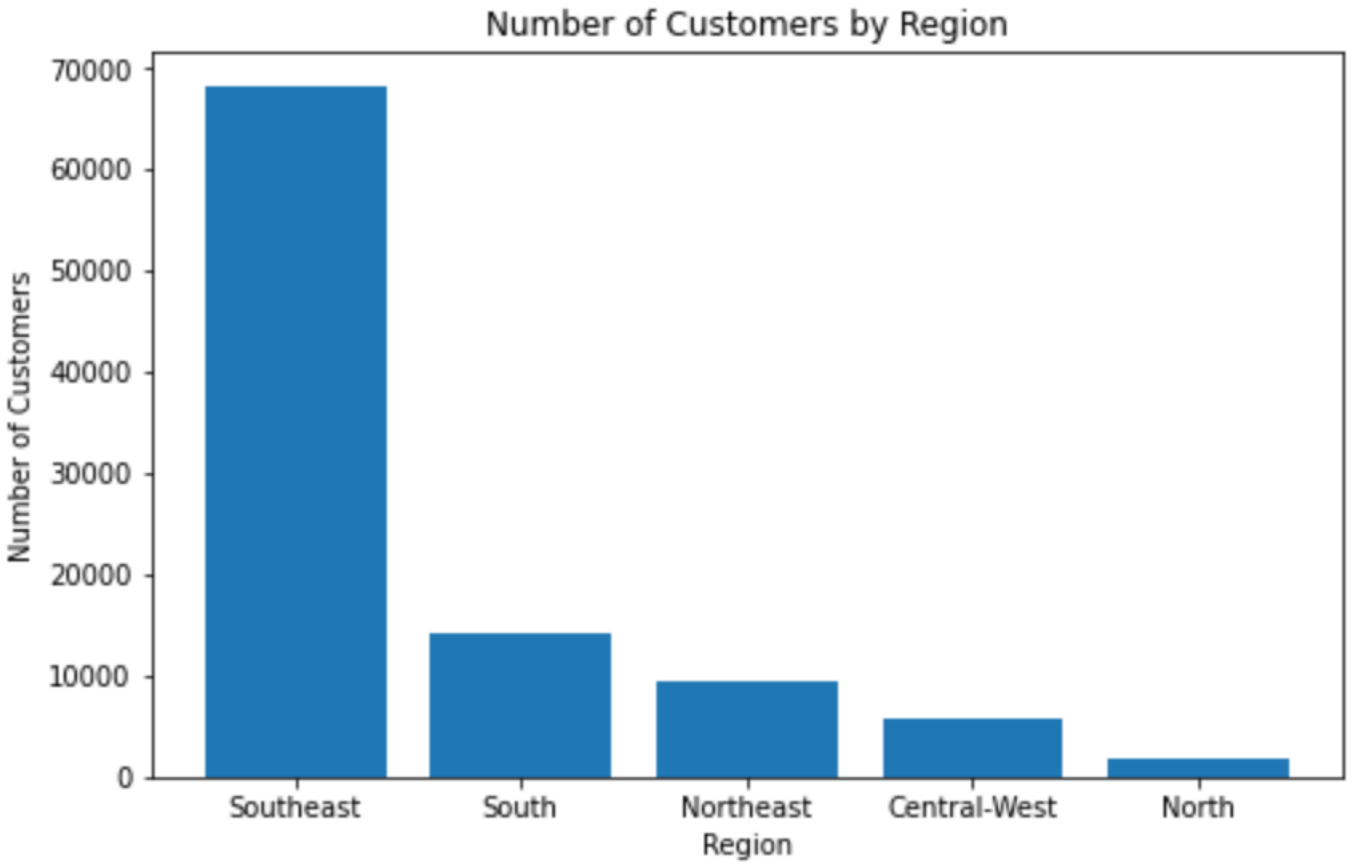
```
...
  File "/databricks/python/lib/python3.9/site-packages/executing/executing.py"
    raise NotOneValueFound('Expected one value, found 0')
executing.executing.NotOneValueFound: Expected one value, found 0
```

VISUALIZATION

Matplotplib doesn't always play well with Pandas-on-spark.

Converting Series index to numpy array, solved the problem.

```
labels = region_counts.index.to_numpy()
plt.figure(figsize=(8, 5))
plt.bar(labels, region_counts.values)
plt.title('Number of Customers by Region')
plt.xlabel('Region')
plt.ylabel('Number of Customers')
plt.show()
```



RESILIENT DISTRIBUTED DATASETS (RDD)

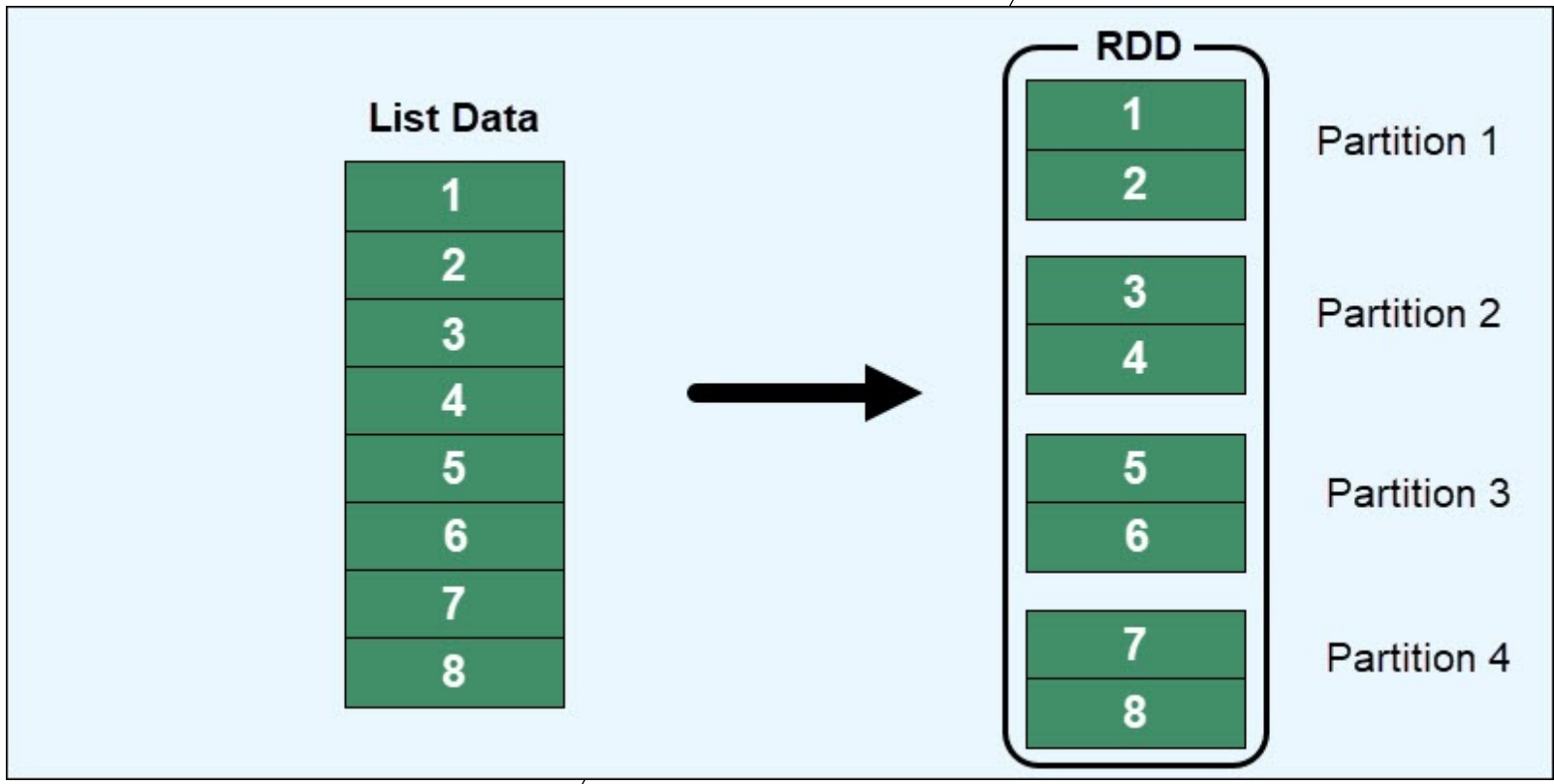
Each DataFrame has an underlying RDD.

Each Row in the RDD holds data for a record.

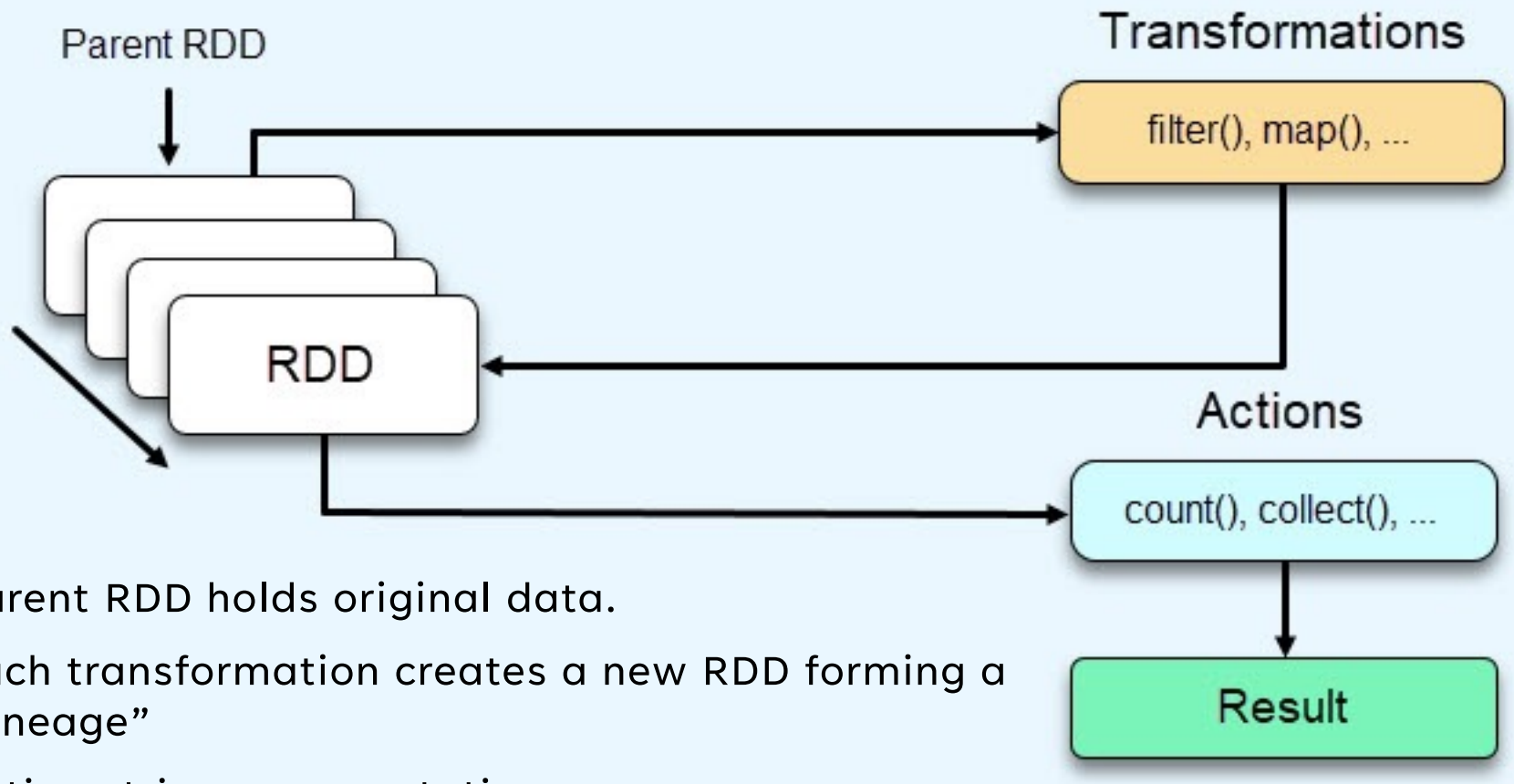
A Row is like an array where each field corresponds to a column in the DataFrame.

RDD partitioned across the cluster.

- Set of rows in each partition.



SEQUENCE OF TRANSFORMATIONS

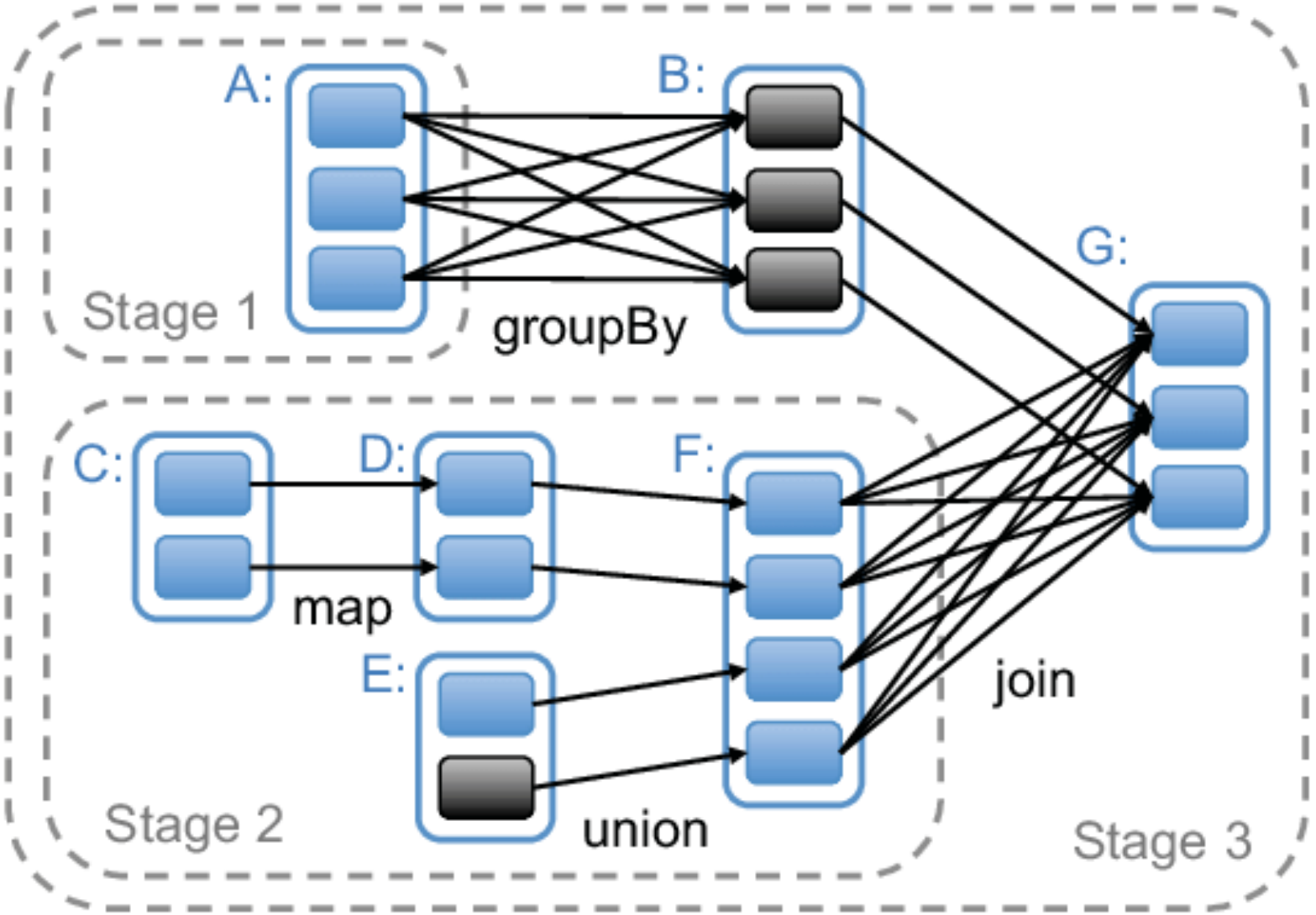


Parent RDD holds original data.

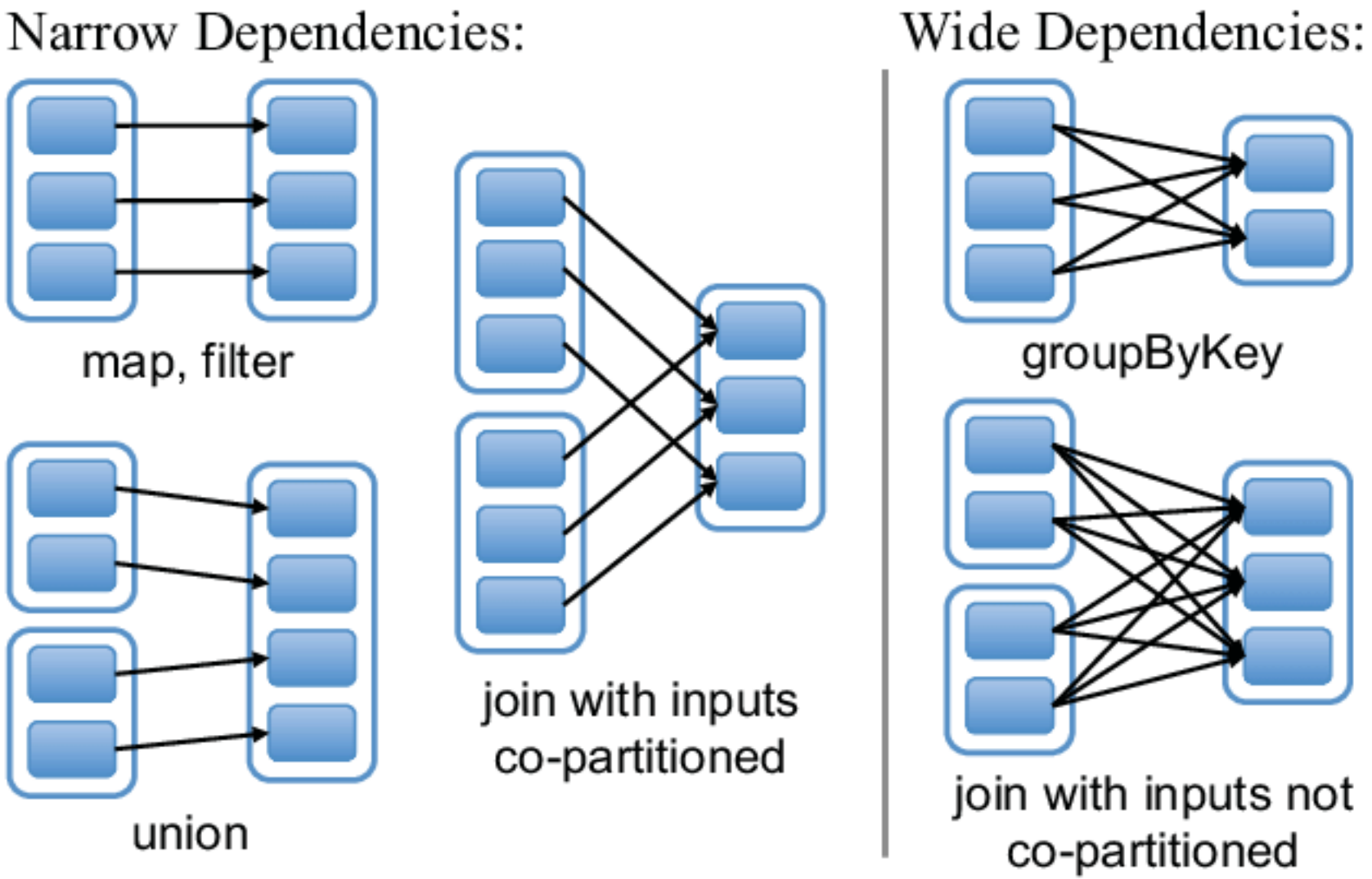
Each transformation creates a new RDD forming a "lineage"

Actions trigger computation.

TRANSFORMATIONS FORM A DAG

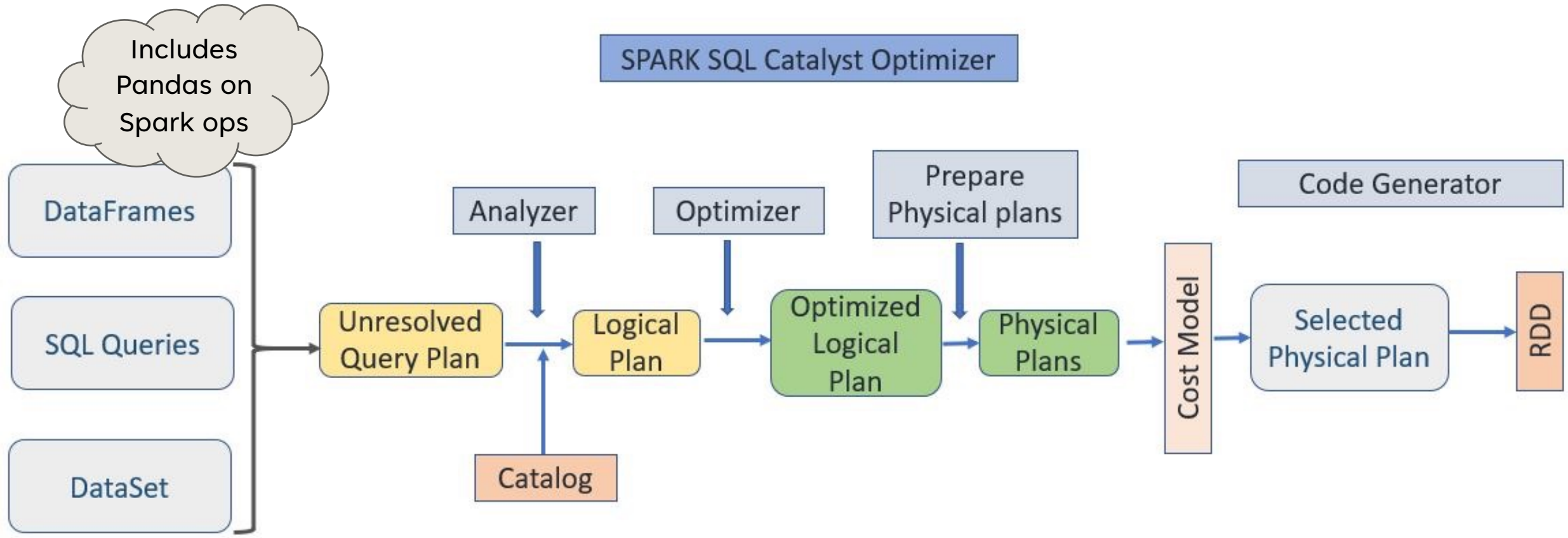


DIFFERENT TRANSFORMATIONS, DIFFERENT DEPENDENCIES

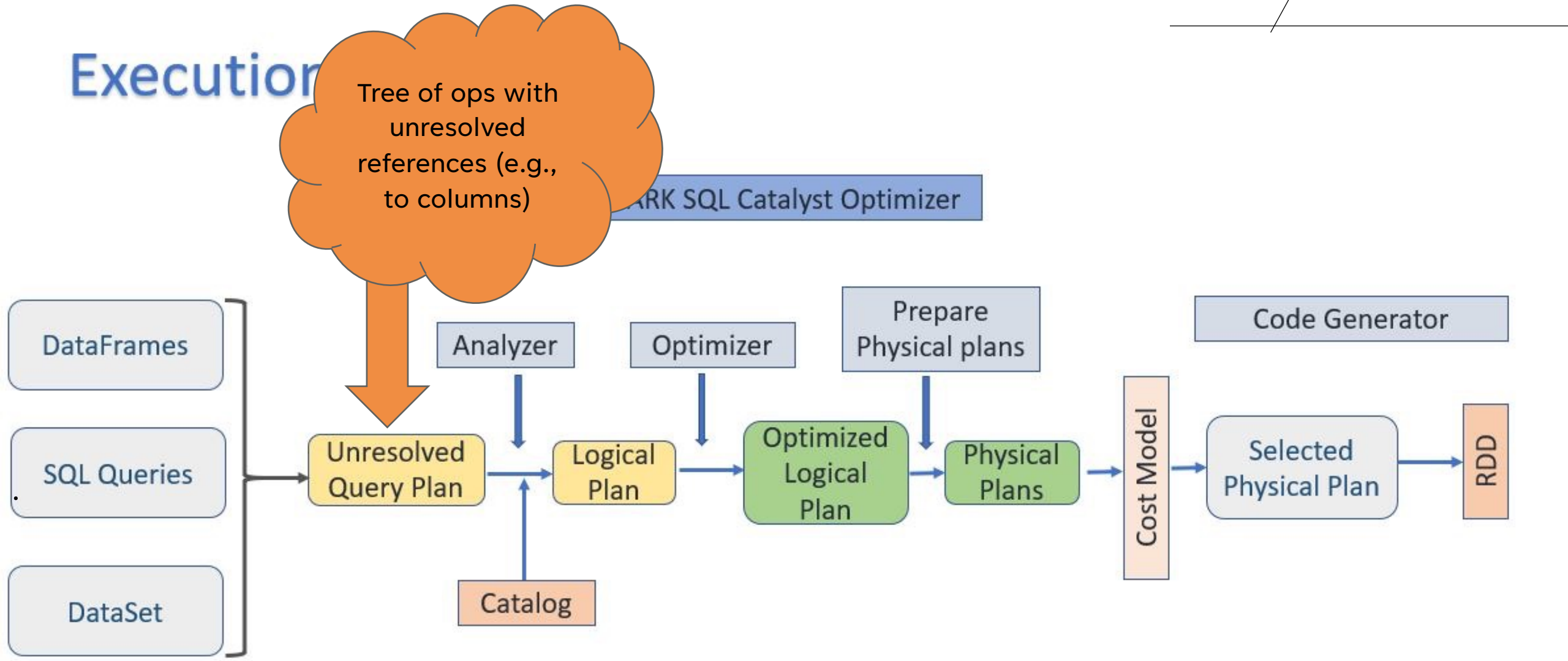


CATALYST OPTIMIZER

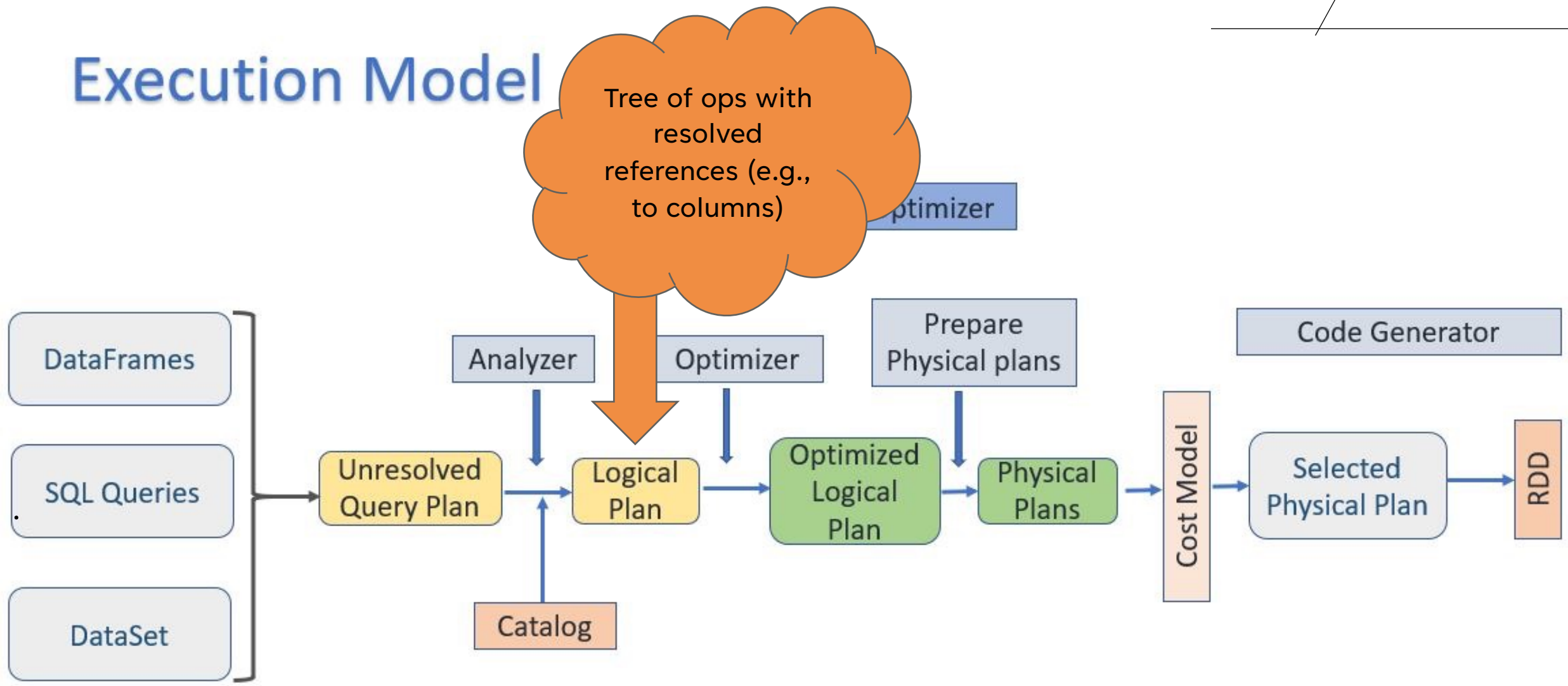
Execution Model



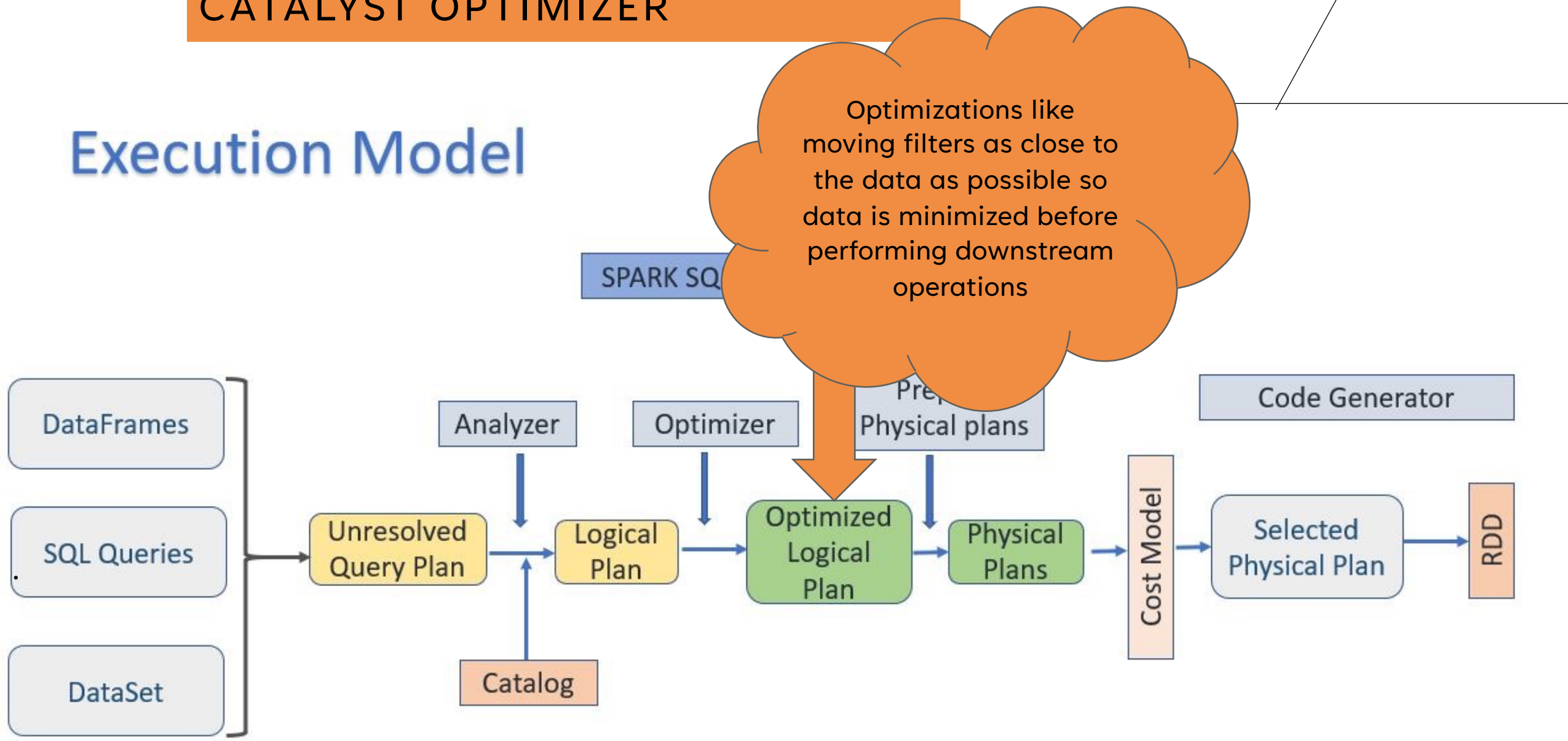
CATALYST OPTIMIZER



CATALYST OPTIMIZER

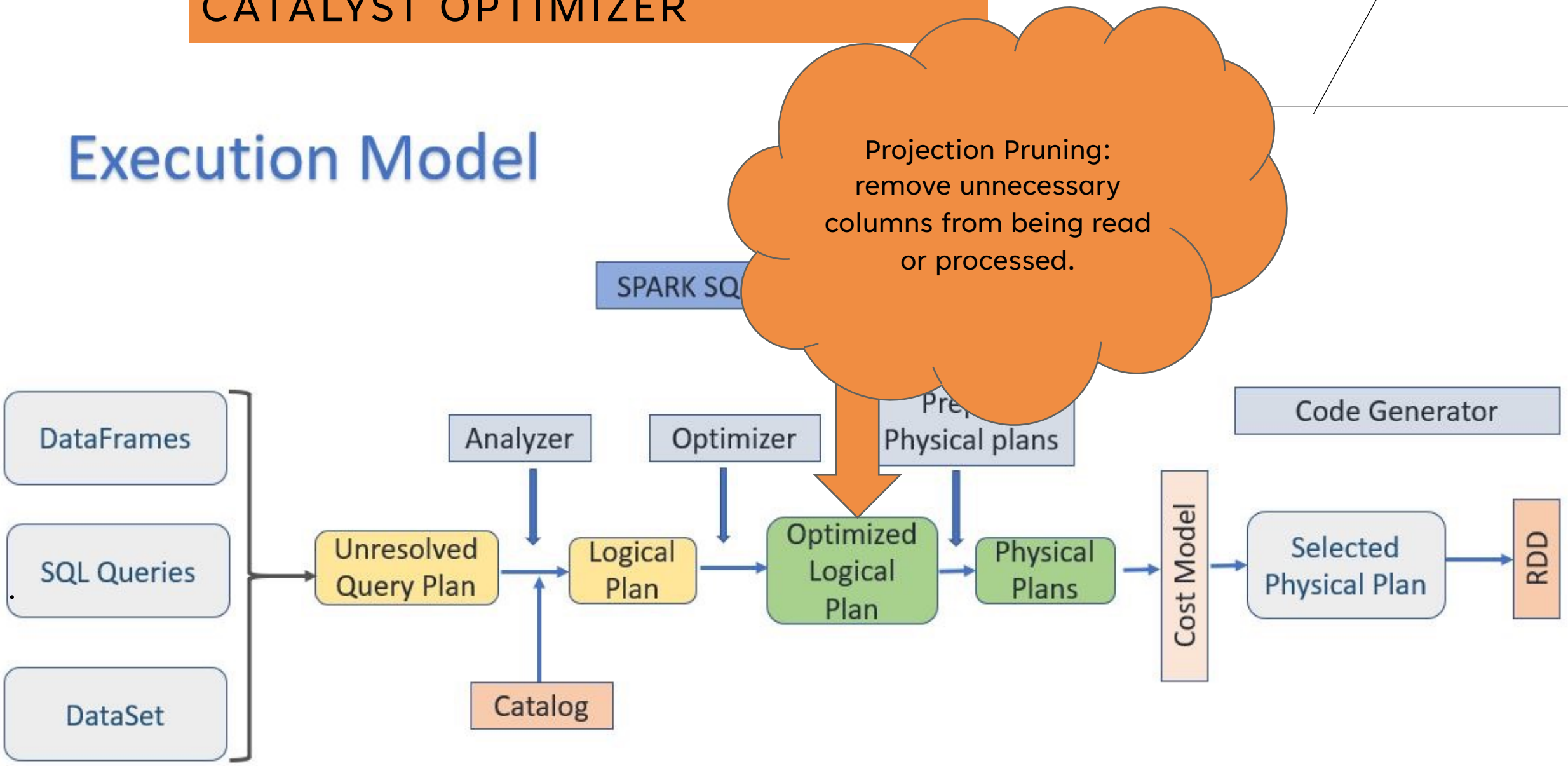


CATALYST OPTIMIZER



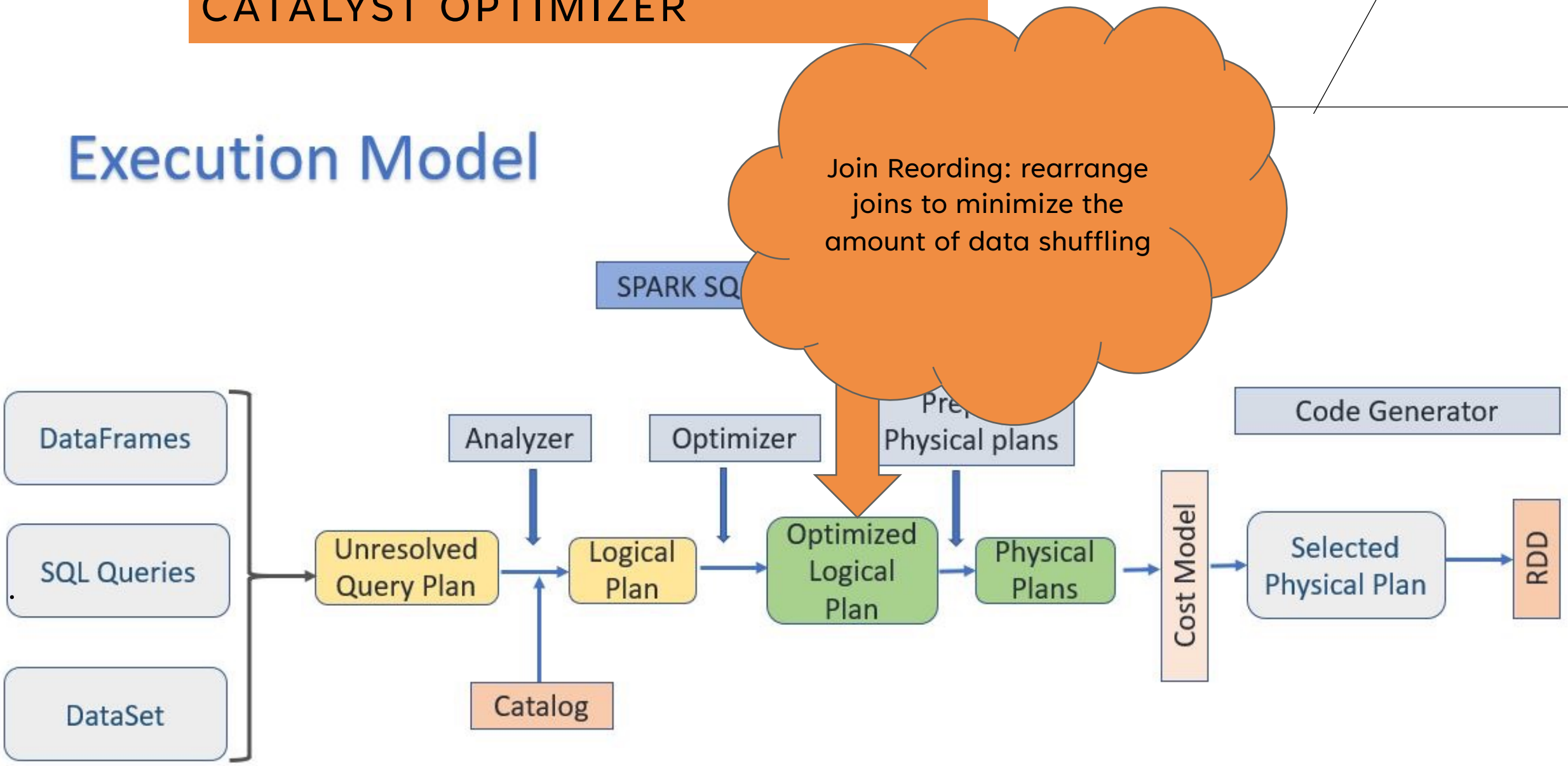
CATALYST OPTIMIZER

Execution Model

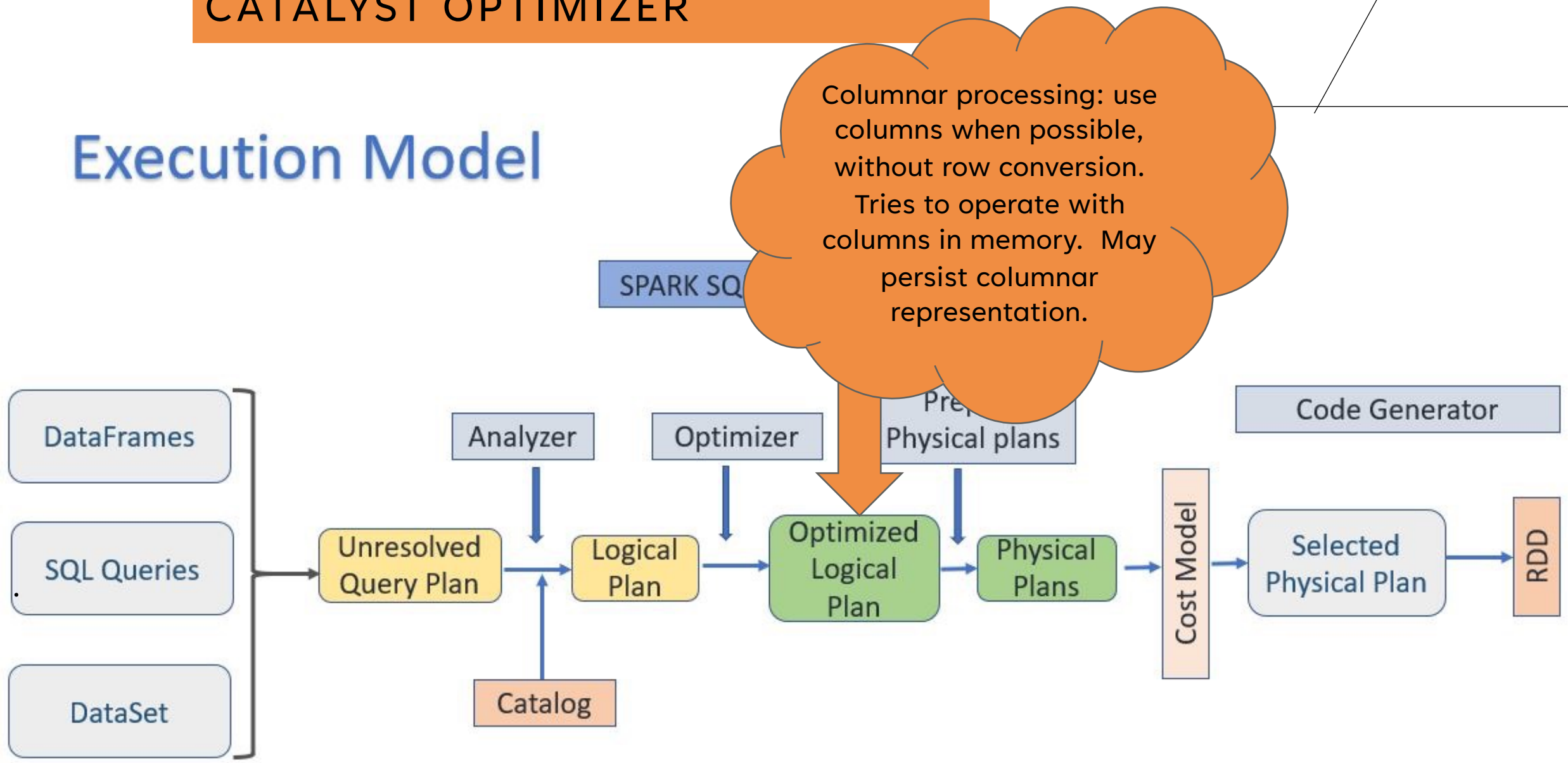


CATALYST OPTIMIZER

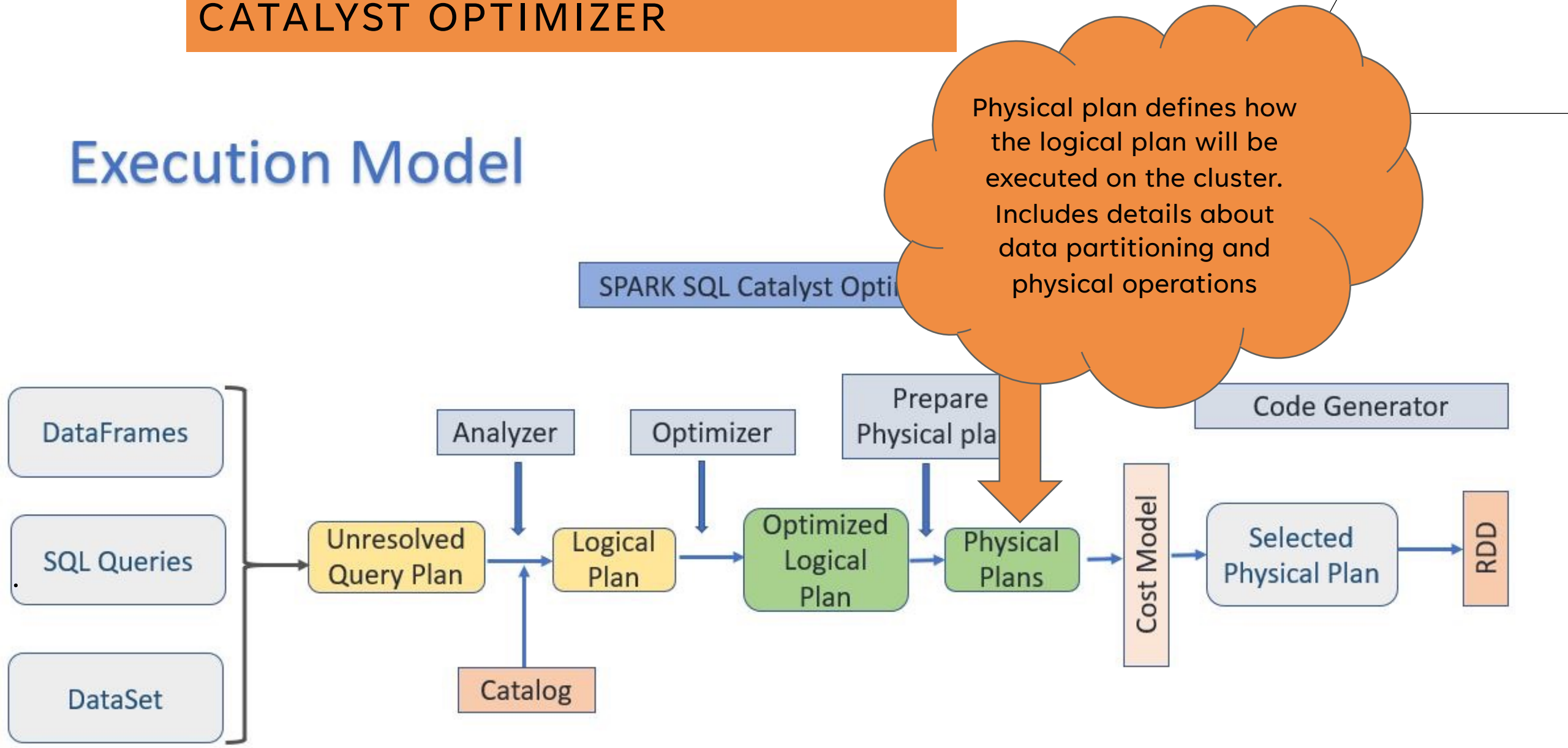
Execution Model



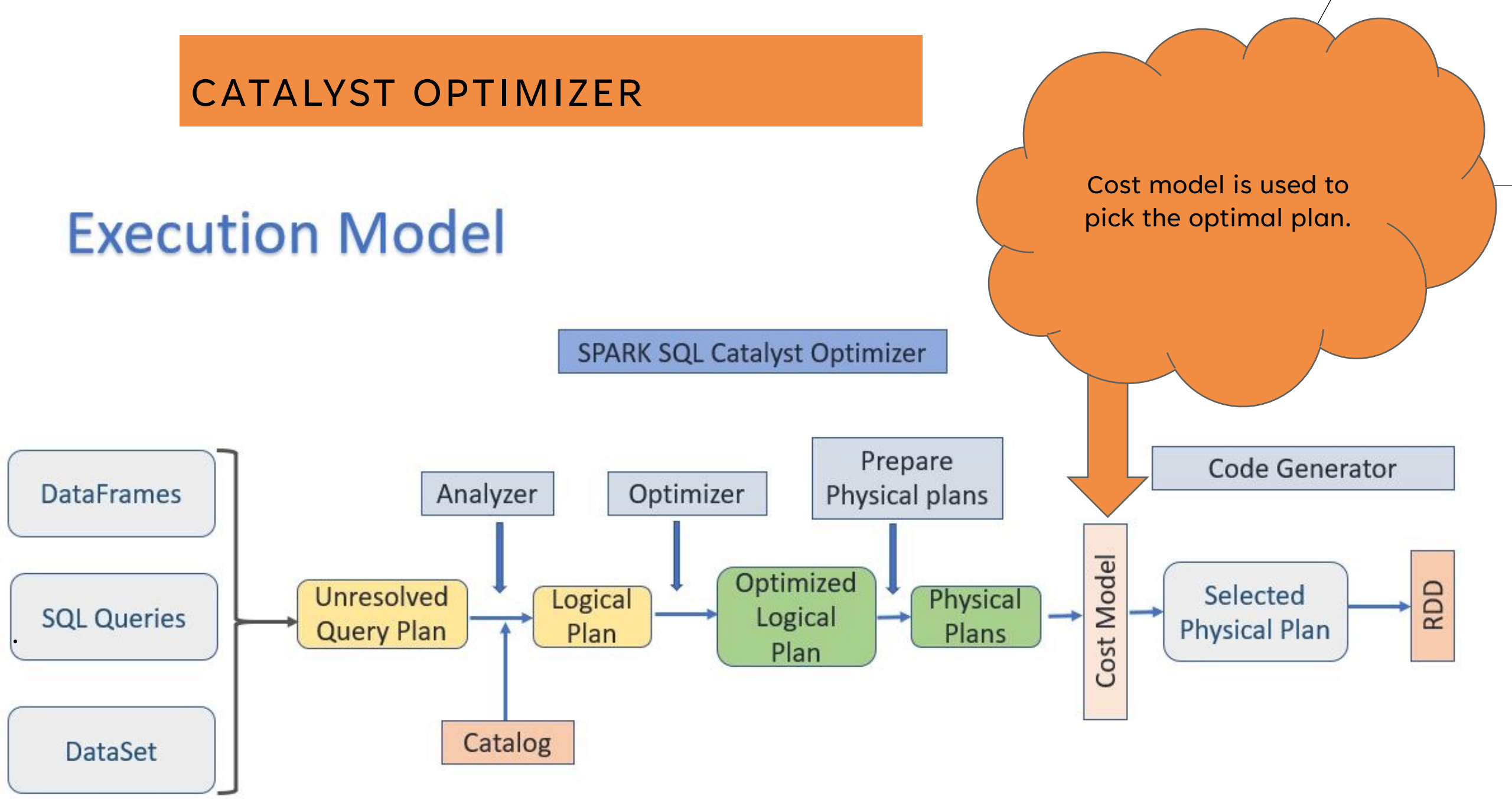
CATALYST OPTIMIZER



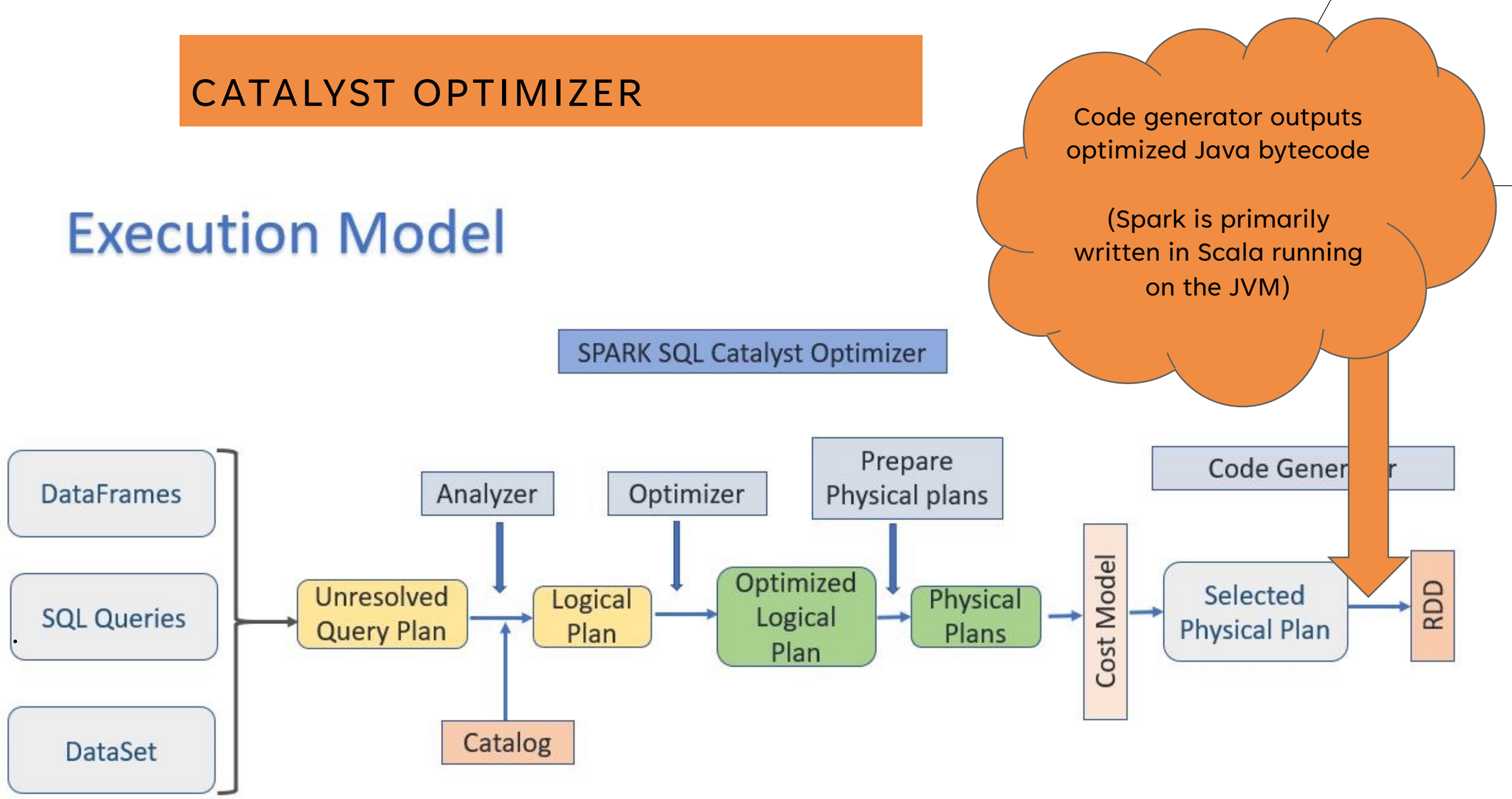
CATALYST OPTIMIZER

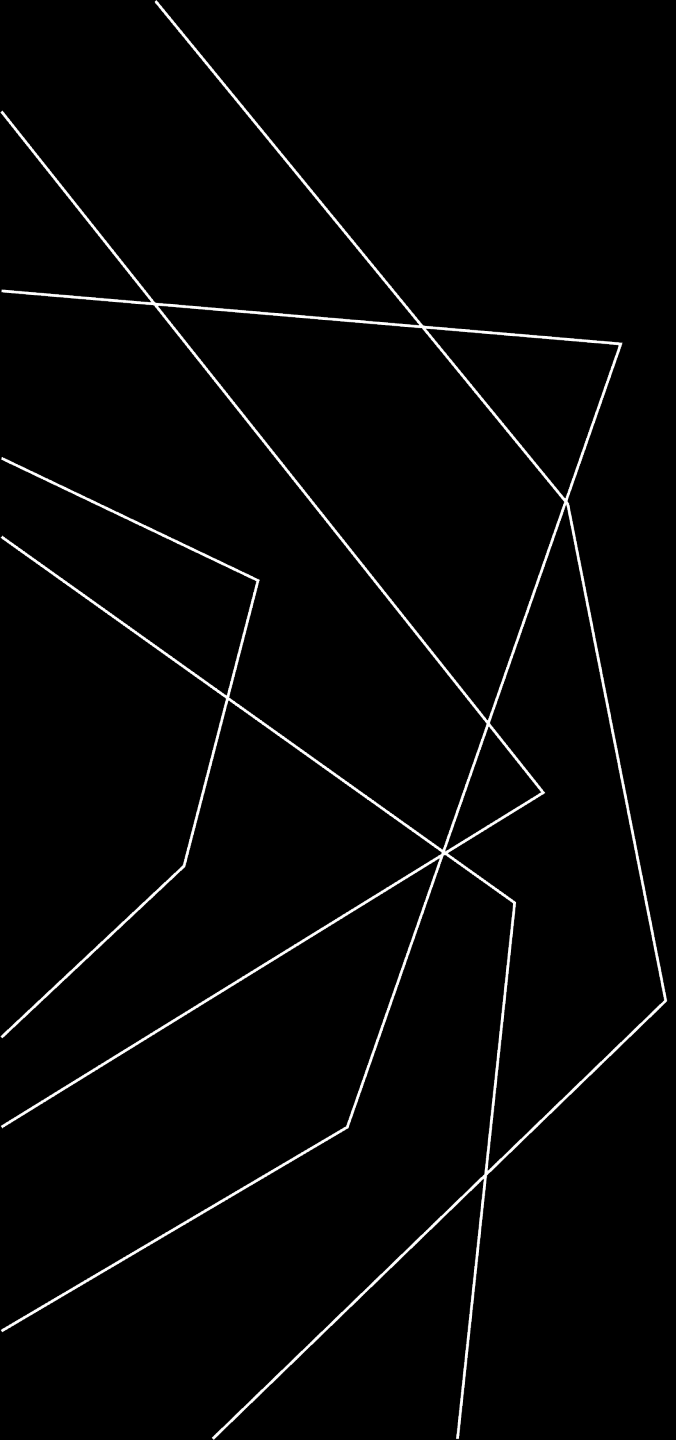


CATALYST OPTIMIZER



CATALYST OPTIMIZER





THANK YOU

David Harrison

Harrison@cs.olemiss.edu