University of California Davis and Databricks collaboration

Distributed Computing with Spark SQL

101 Spark SQL Environments and Queries

102 Spark SQL Core Concepts and Internal

Why Distributed Computing?

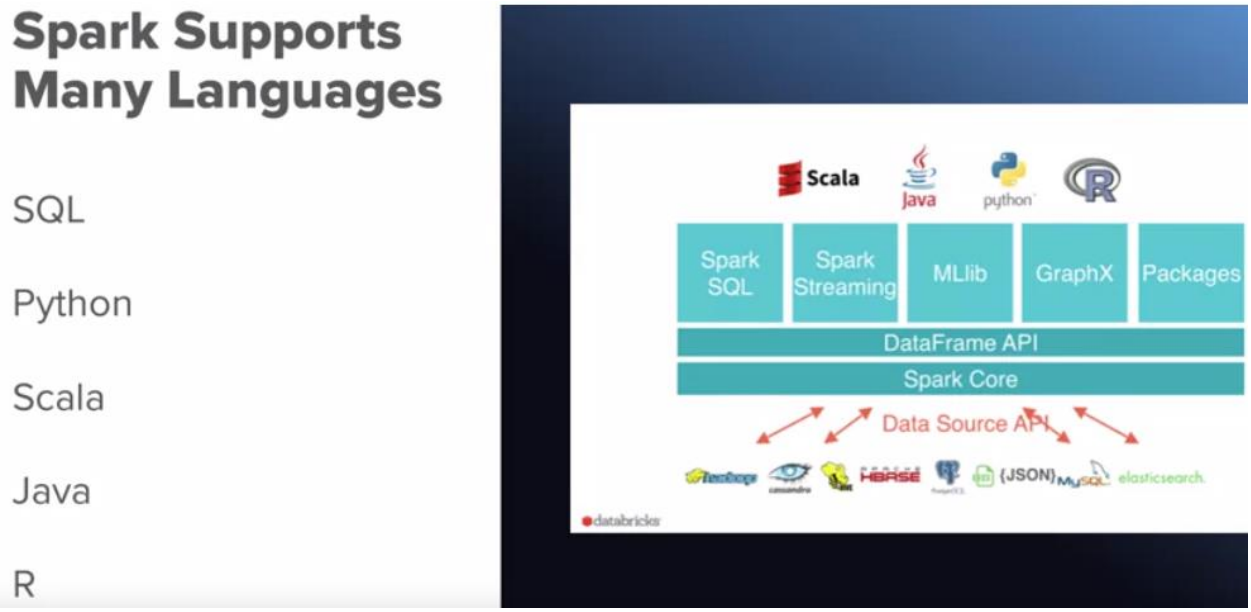
Qualities of Big Data:

Volume: The Amount of Data

Velocity: The Speed of Data

Variety: The Diversity of Data

Veracity: The Reliability of Data

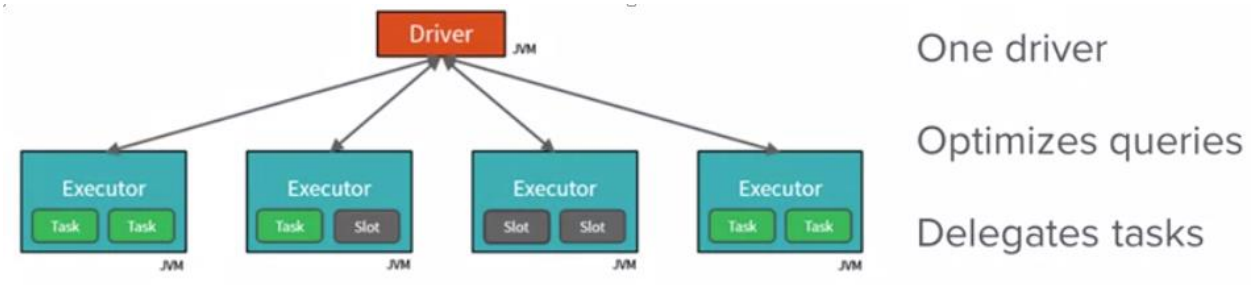


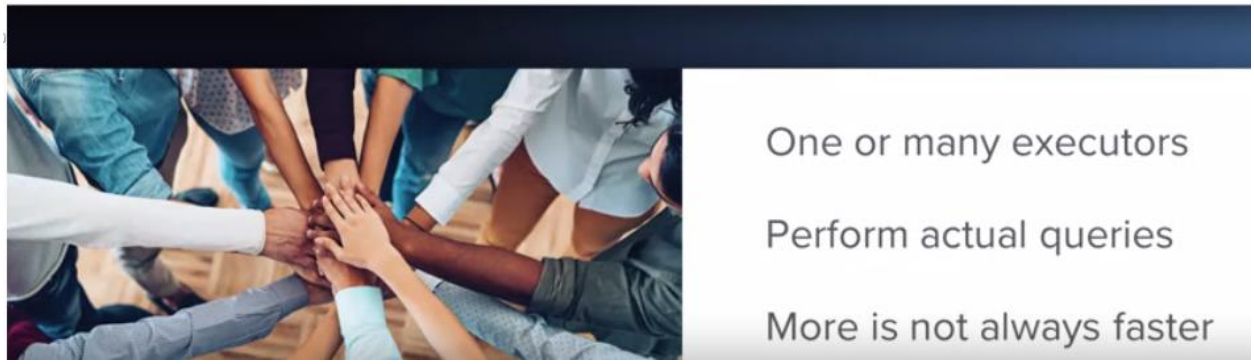
Why Spark is Popular?

Read and process data from many sources

Work with many file types

Solve many data problems faced by analysts



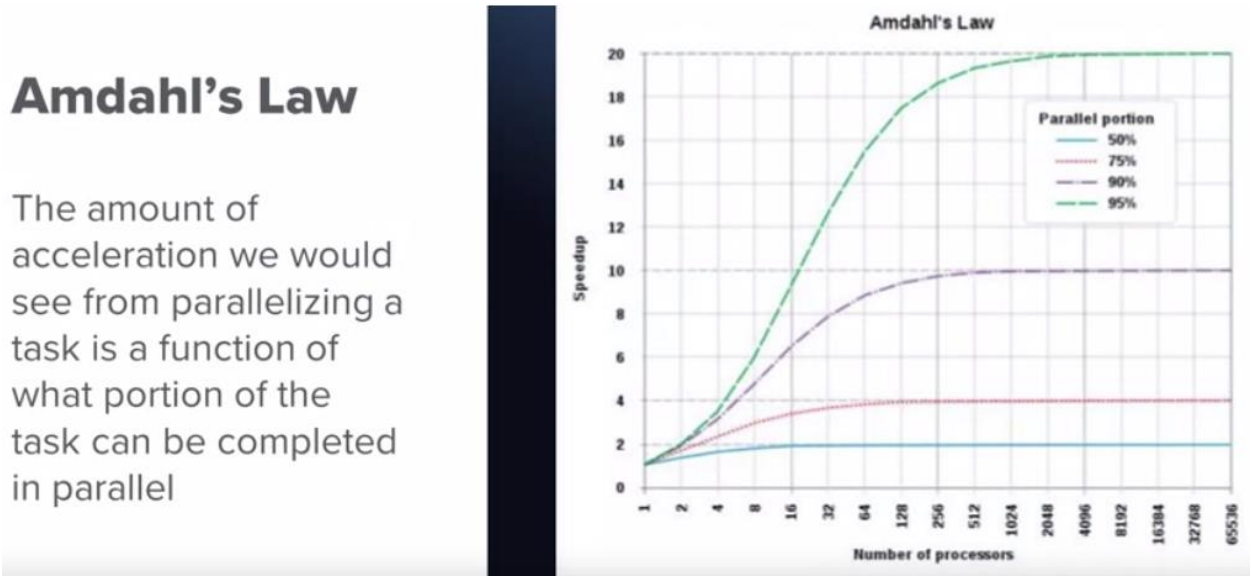


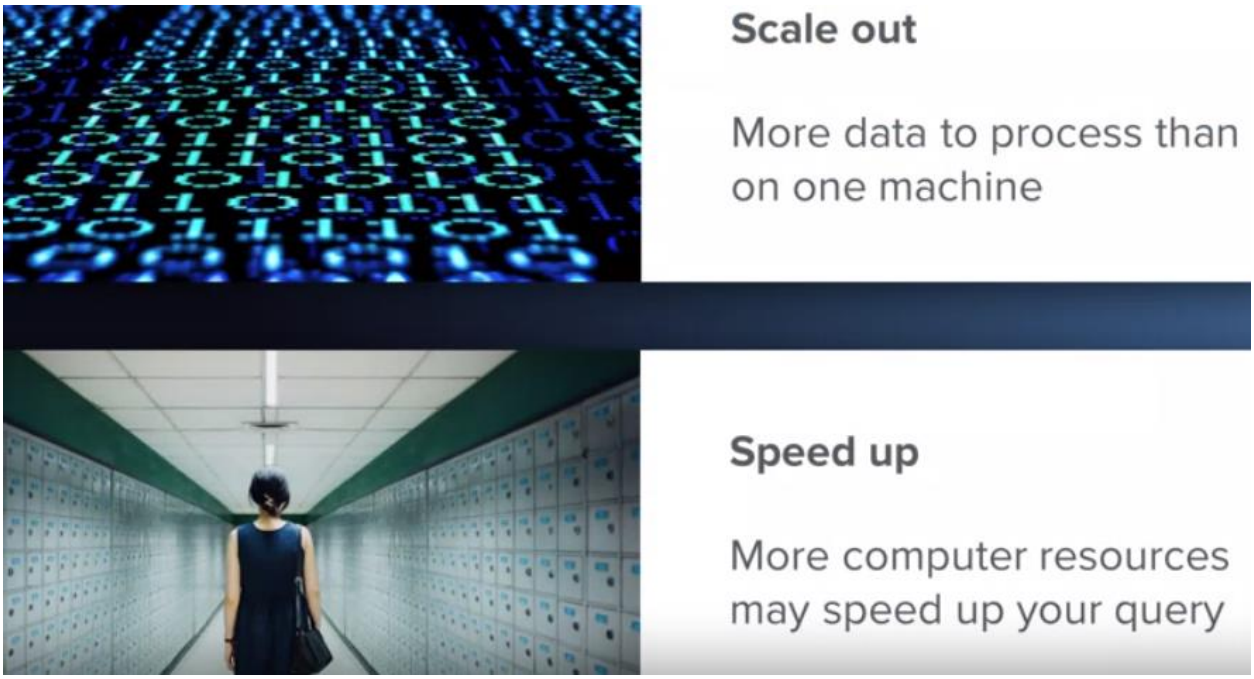
Why more computing power is not always faster?

Telling each person takes more time than doing the task

Distributing Computation is Parallelism

Scaling workloads to increasingly larger data set





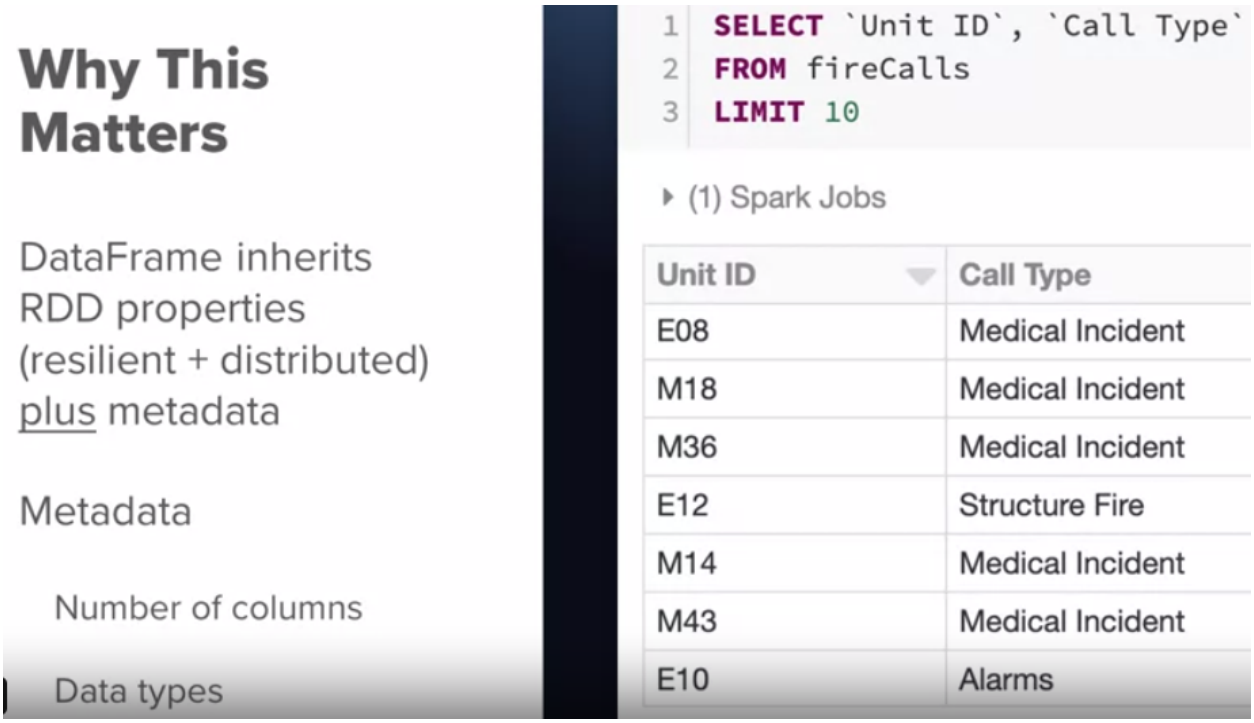
Resilient Distributed Dataset (RDD)

Resilient: Fault Tolerance

If you lose a worker, only recompute work that worker was responsible for

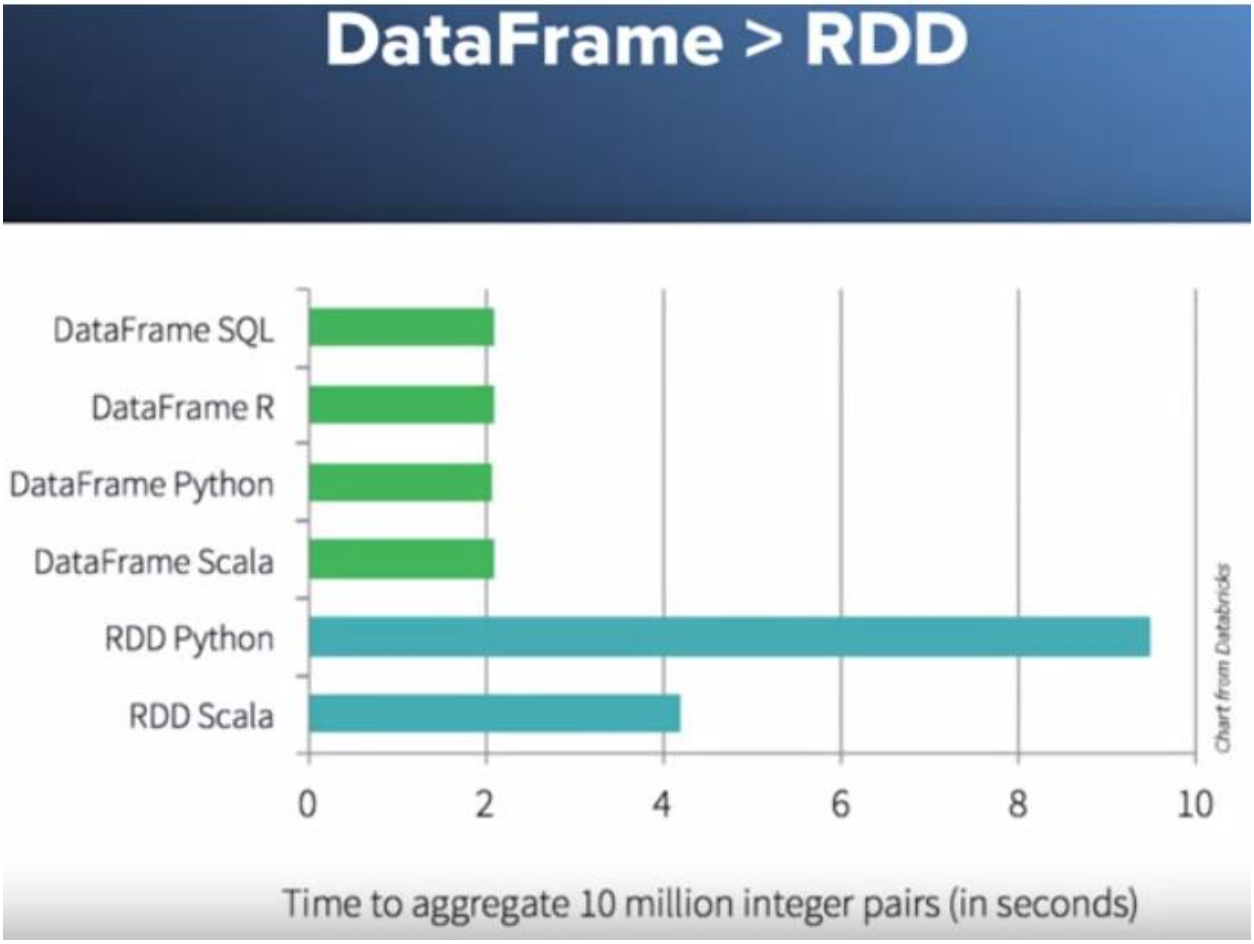
Distributed DataSet

Computed across multiple nodes. Results are aggregated by the driver.



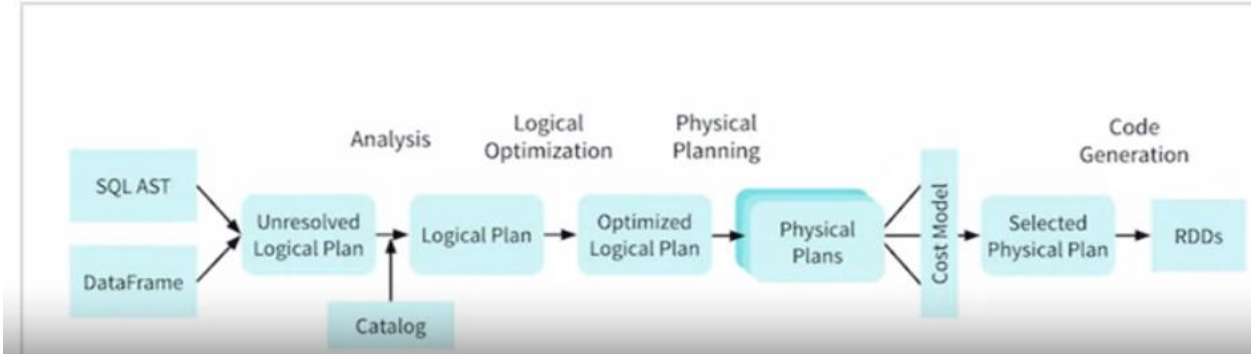
DataFrame

Highly optimized beyond RDDs. Spark SQL commands execute against DataFrame.

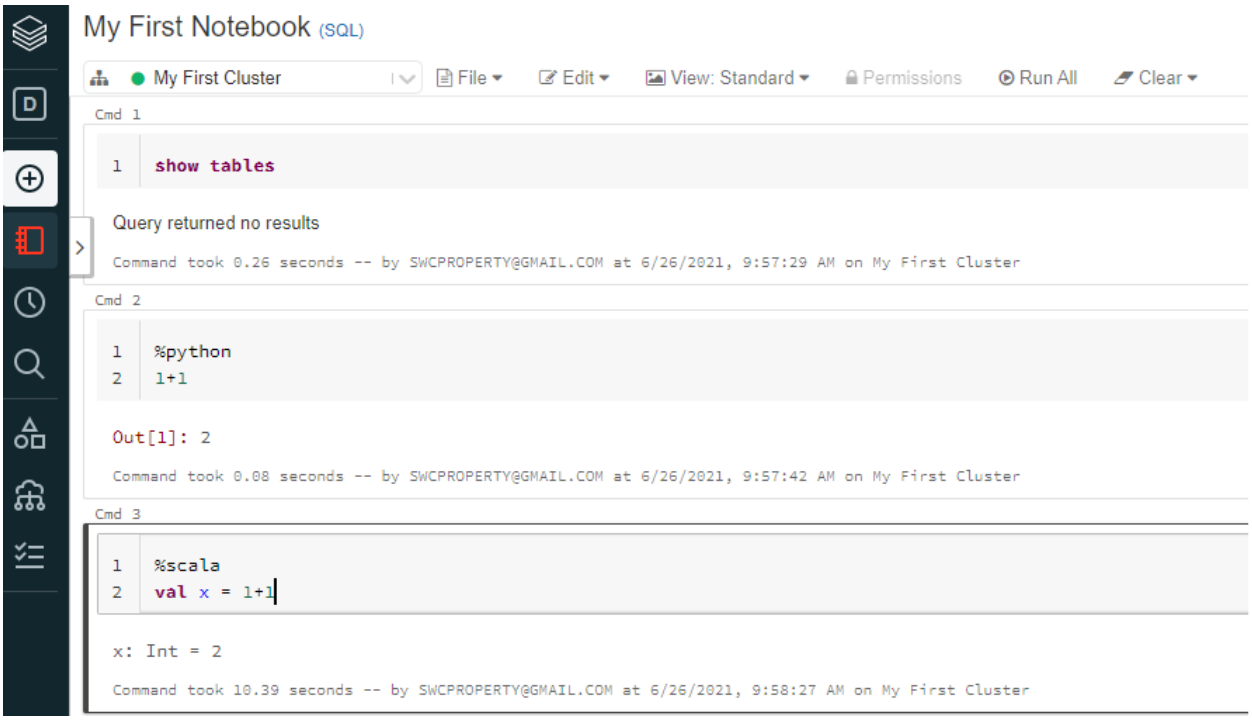


Spark DataFrame Execution

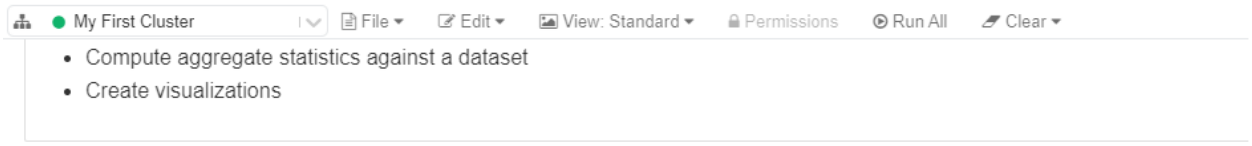
1. Unresolved logical plan before look-up in data catalog
2. Then Catalyst resolves them and creates a logical plan



Look at Tungsten which illustrates why DataFrames are more performant than RDDs



1.4-SQL-Notebook (SQL)



- Compute aggregate statistics against a dataset
- Create visualizations

Cmd 3

Mounting Data

We are going to run a Classroom Setup script to mount the data we will be using throughout the class.

A mount is a pointer to a remote storage location (typically AWS or Azure), so we can access that data from within Databricks.

Cmd 4

```
1  %run ../Includes/Classroom-Setup
```

```
Command took 50.69 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 10:19:03 AM on My First Cluster
```

OK

Cmd 5

☆ Create a Database and Table

Cmd 6

SF Fire Department Calls for Service

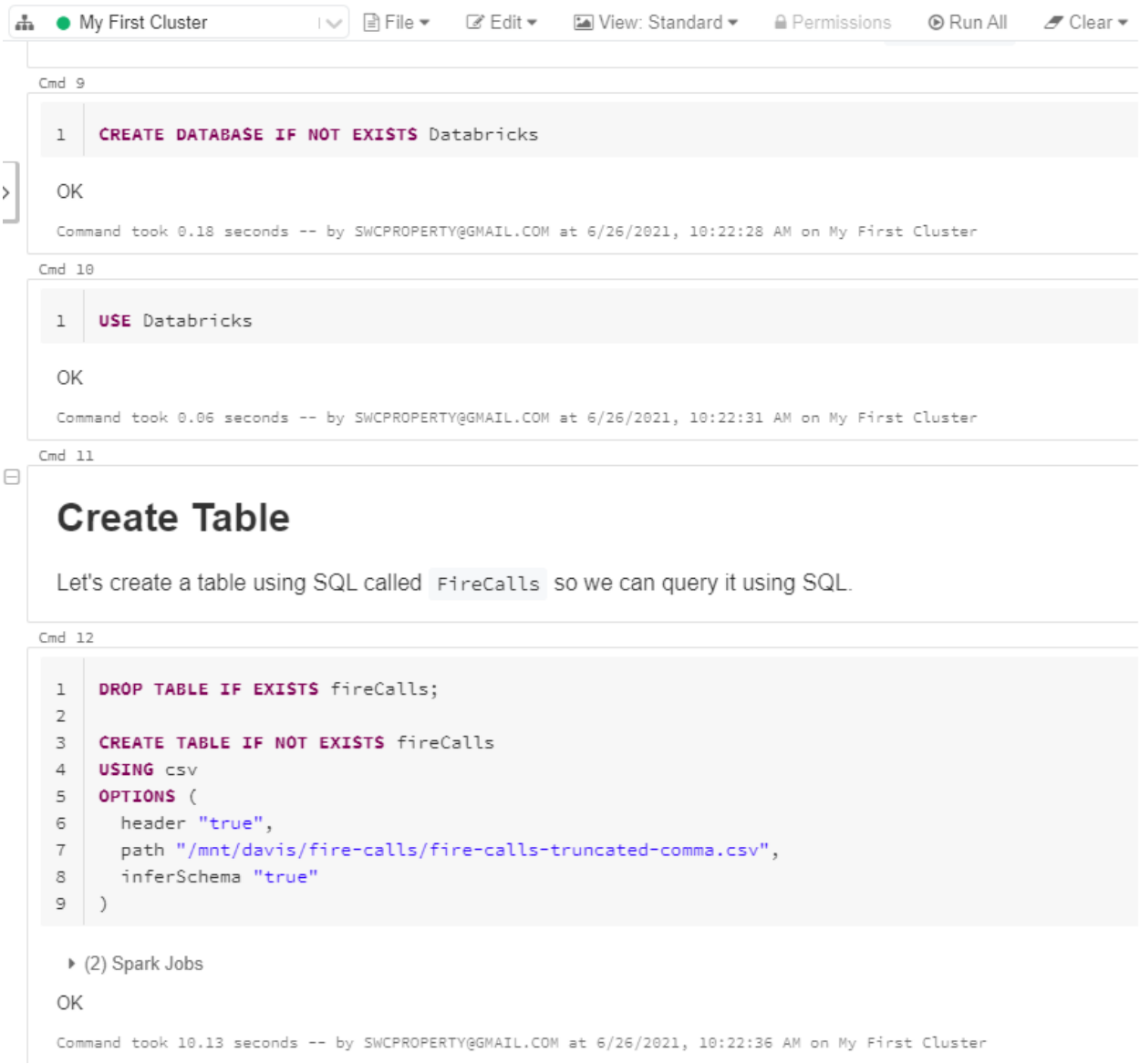
Let's take a look at the SF Fire Department Calls for Service dataset. This dataset includes all fire units responses to calls.

Cmd 7

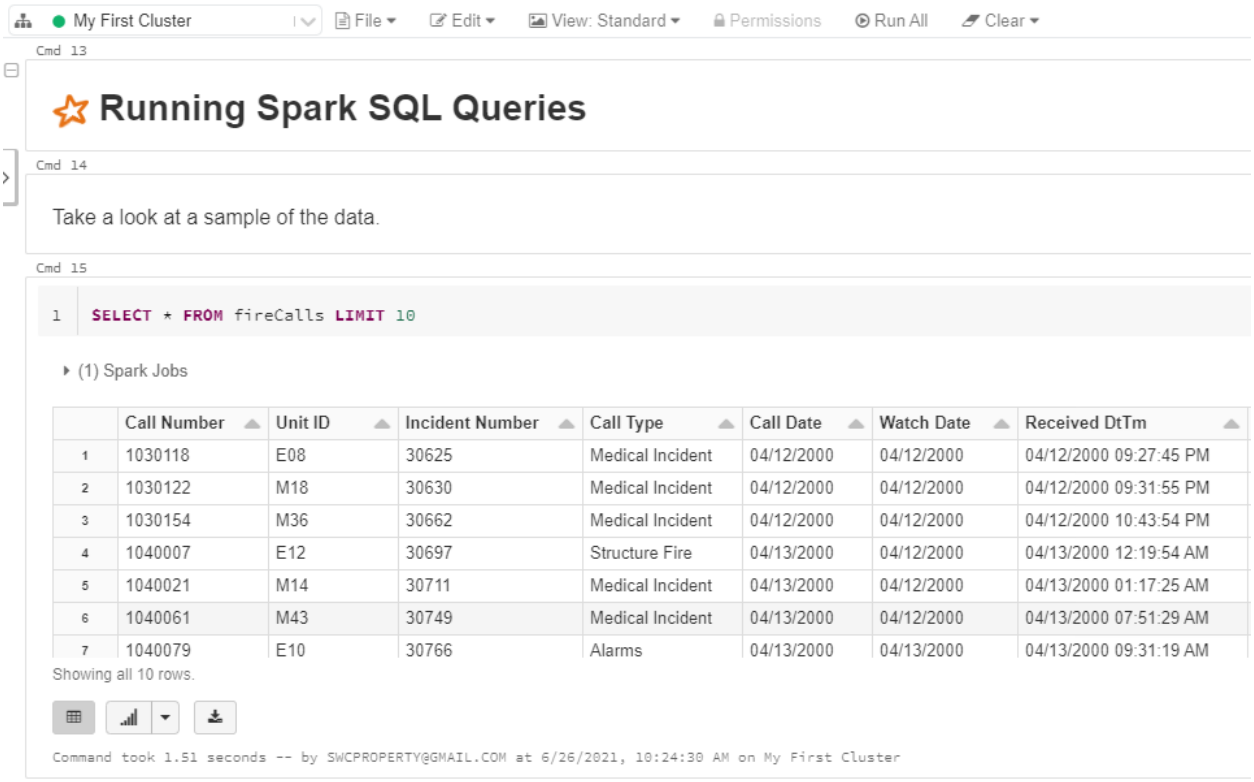
```
1  %fs head /mnt/davis/fire-calls/fire-calls-truncated-comma.csv
```

```
[Truncated to first 65536 bytes]
Call Number,Unit ID,Incident Number,Call Type,Call Date,Watch Date,Received DtTm,Entry DtTm,Dispatch DtTm,Response [
```

1.4-SQL-Notebook (SQL)



1.4-SQL-Notebook (SQL)



	Call Number	Unit ID	Incident Number	Call Type	Call Date	Watch Date	Received DtTm
1	1030118	E08	30625	Medical Incident	04/12/2000	04/12/2000	04/12/2000 09:27:45 PM
2	1030122	M18	30630	Medical Incident	04/12/2000	04/12/2000	04/12/2000 09:31:55 PM
3	1030154	M36	30662	Medical Incident	04/12/2000	04/12/2000	04/12/2000 10:43:54 PM
4	1040007	E12	30697	Structure Fire	04/13/2000	04/12/2000	04/13/2000 12:19:54 AM
5	1040021	M14	30711	Medical Incident	04/13/2000	04/12/2000	04/13/2000 01:17:25 AM
6	1040061	M43	30749	Medical Incident	04/13/2000	04/12/2000	04/13/2000 07:51:29 AM
7	1040079	E10	30766	Alarms	04/13/2000	04/13/2000	04/13/2000 09:31:19 AM

1.4-SQL-Notebook (SQL)

My First Cluster | File | Edit | View: Standard | Permissions | Run All | Clear

Command took 1.51 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 10:24:30 AM on My First Cluster

Cmd 16

Which neighborhoods have the most fire calls?

Cmd 17

```
1  SELECT `Neighborhooods - Analysis Boundaries` as neighborhood,
2    COUNT(`Neighborhooods - Analysis Boundaries`) as count
3  FROM FireCalls
4  GROUP BY `Neighborhooods - Analysis Boundaries`
5  ORDER BY count DESC
```

▸ (1) Spark Jobs

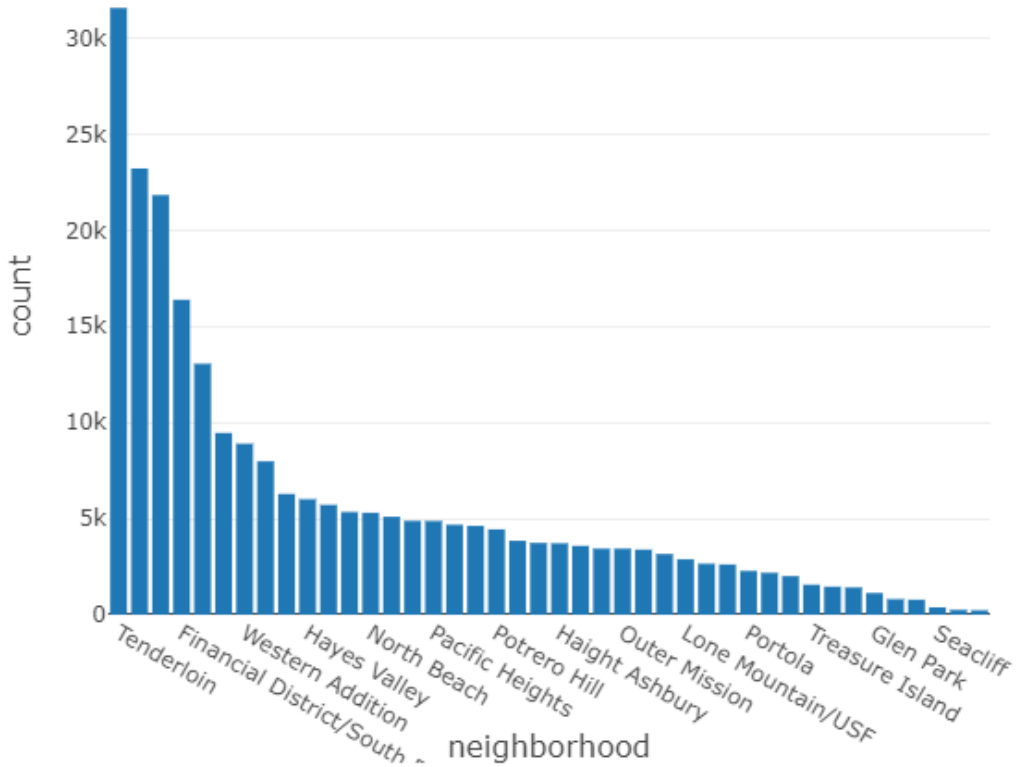
	neighborhood	count
1	Tenderloin	31564
2	South of Market	23212
3	Mission	21829
4	Financial District/South Beach	16384
5	Bayview Hunters Point	13057
6	Sunset/Parkside	9456
7	Western Addition	8899

Showing all 42 rows.

Command took 10.07 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 10:24:43 AM on My First Cluster

```
1  SELECT `Neighborhooods - Analysis Boundaries` as neighborhood,
2    COUNT(`Neighborhooods - Analysis Boundaries`) as count
3  FROM fireCalls
4  GROUP BY `Neighborhooods - Analysis Boundaries`
5  ORDER BY count DESC
```

▸ (1) Spark Jobs

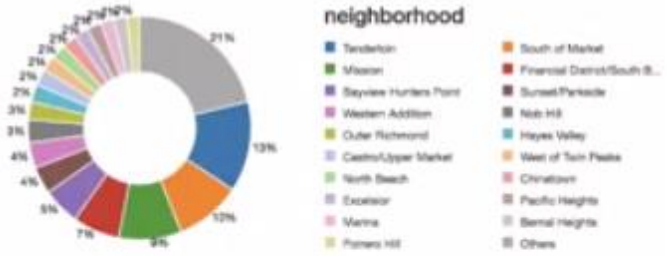


Plot Options...

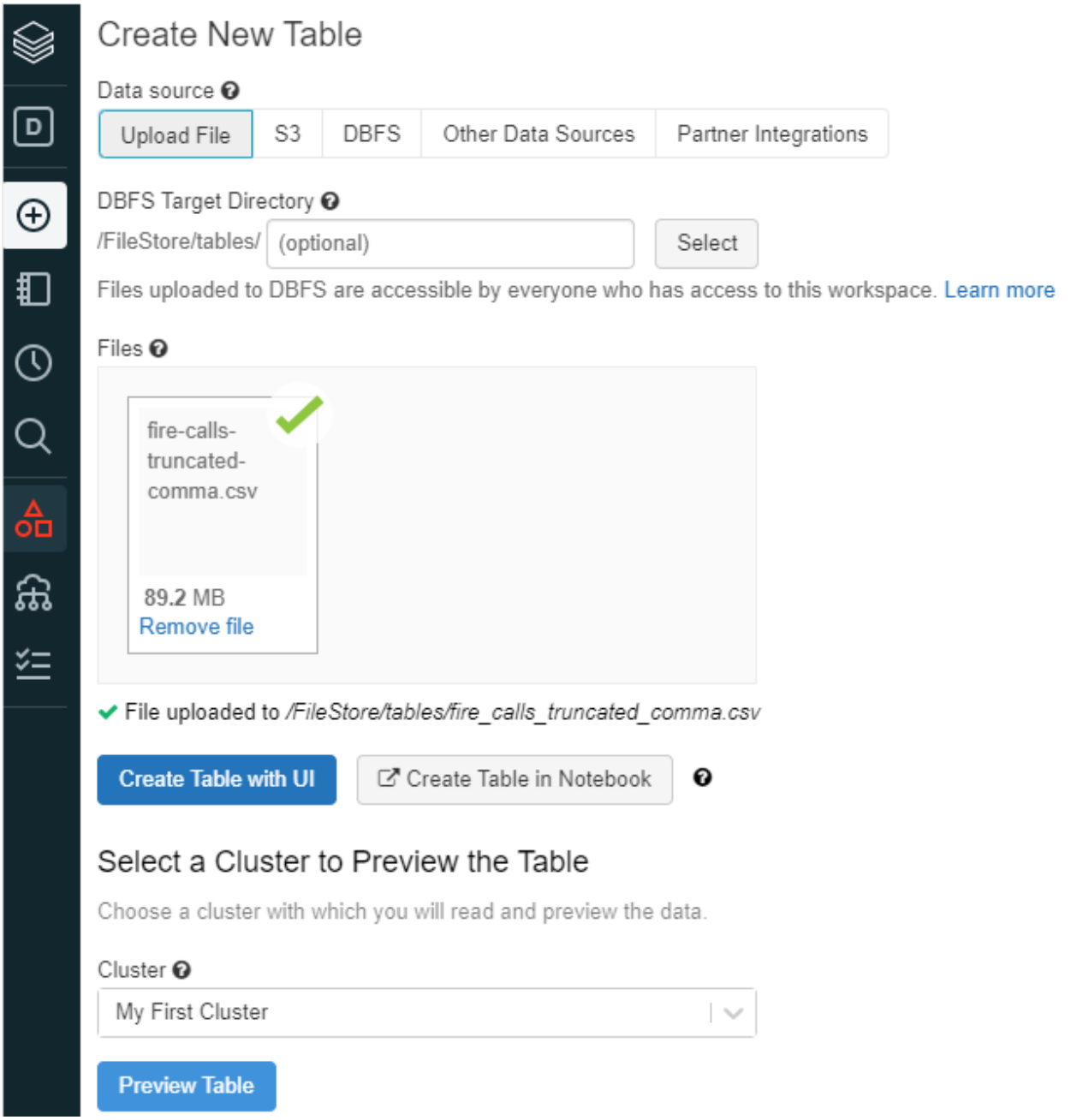
Command took 6.80 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 10:26:41 AM on My First Cluster

```
SELECT `Neighborhooods - Analysis Boundaries` as neighborhood,
  COUNT(`Neighborhooods - Analysis Boundaries`) as count
FROM fireCalls
GROUP BY `Neighborhooods - Analysis Boundaries`
ORDER BY count DESC
```

▸ (1) Spark Jobs



Plot Options...


Create New Table
Data source
Upload File
S3
DBFS
Other Data Sources
Partner Integrations
DBFS Target Directory
/FileStore/tables/
(optional)
Select
Files uploaded to DBFS are accessible by everyone who has access to this workspace. Learn more
Files
fire-calls-truncated-comma.csv
89.2 MB
Remove file
File uploaded to /FileStore/tables/fire_calls_truncated_comma.csv
Create Table with UI
Create Table in Notebook
Select a Cluster to Preview the Table
Choose a cluster with which you will read and preview the data.
Cluster
My First Cluster
Preview Table

Specify Table Attributes

Specify the Table Name, Database and Schema to add this to the data UI for other users to access

Table Name: firecalls_uploaded

Create in Database: default

File Type: CSV

Column Delimiter: ,

First row is header

Infer schema

Multi-line

Create Table

Create Table in Notebook

Table Preview

Call Number	Unit ID	Incident Number	Call Type	Call Date	Watch Date
INT	STRING	INT	STRING	STRING	STRING
1030118	E08	30625	Medical Incident	04/12/2000	04/12/2000
1030122	M18	30630	Medical Incident	04/12/2000	04/12/2000
1030154	M36	30662	Medical Incident	04/12/2000	04/12/2000
1040007	E12	30697	Structure Fire	04/13/2000	04/12/2000
1040021	M14	30711	Medical Incident	04/13/2000	04/12/2000
1040061	M43	30749	Medical Incident	04/13/2000	04/12/2000

default.firecalls_uploaded | Refresh

My First Cluster

Schema:

	col_name	data_type	comment
1	Call Number	int	null
2	Unit ID	string	null
3	Incident Number	int	null
4	Call Type	string	null
5	Call Date	string	null
6	Watch Date	string	null
7	Received DtTm	string	null

Showing all 34 rows.

Sample Data:

	Call Number	Unit ID	Incident Number	Call Type	Call Date	Watch Date	Received DtTm	Entry DtTm	Dispatch DtTm	Response DtTm
1	1030118	E08	30625	Medical Incident	04/12/2000	04/12/2000	04/12/2000 09:27:45 PM	04/12/2000 09:28:58 PM	04/12/2000 09:29:21 PM	04/12/2000 09:31:26 PM
2	1030122	M18	30630	Medical Incident	04/12/2000	04/12/2000	04/12/2000 09:31:55 PM	04/12/2000 09:33:48 PM	04/12/2000 09:34:10 PM	04/12/2000 09:35:59 PM
3	1030154	M36	30662	Medical Incident	04/12/2000	04/12/2000	04/12/2000 10:43:54 PM	04/12/2000 10:45:53 PM	04/12/2000 10:49:59 PM	04/12/2000 10:50:35 PM
4	1040007	E12	30697	Structure Fire	04/13/2000	04/12/2000	04/13/2000 12:19:54 AM	04/13/2000 12:29:24 AM	04/13/2000 12:29:35 AM	04/13/2000 12:31:25 AM
5	1040021	M14	30711	Medical Incident	04/13/2000	04/12/2000	04/13/2000 01:17:25 AM	04/13/2000 01:18:44 AM	04/13/2000 01:20:02 AM	04/13/2000 01:21:40 AM
6	1040061	M43	30749	Medical Incident	04/13/2000	04/12/2000	04/13/2000 07:51:29 AM	04/13/2000 07:55:35 AM	04/13/2000 07:55:54 AM	04/13/2000 07:59:58 AM
7	1040079	E10	30766	Alarms	04/13/2000	04/13/2000	04/13/2000 09:31:19 AM	04/13/2000 09:33:04 AM	04/13/2000 09:34:10 AM	04/13/2000 09:35:52 AM

Showing all 20 rows.

Querying Data

Now you can access the file you uploaded.

Cmd 7

Take a look at the first few lines of the data.

Cmd 8

```
1  SELECT * FROM firecalls_uploaded LIMIT 10
```

(1) Spark Jobs

	Call Number	Unit ID	Incident Number	Call Type	Call Date	Watch Date	Received DtTm	Entry DtTm	Dispatch DtTm	Response DtTm	On Scene
1	1030118	E08	30625	Medical Incident	04/12/2000	04/12/2000	04/12/2000 09:27:45 PM	04/12/2000 09:28:58 PM	04/12/2000 09:29:21 PM	04/12/2000 09:31:26 PM	04/12/200
2	1030122	M18	30630	Medical Incident	04/12/2000	04/12/2000	04/12/2000 09:31:55 PM	04/12/2000 09:33:48 PM	04/12/2000 09:34:10 PM	04/12/2000 09:35:59 PM	04/12/200
3	1030154	M36	30662	Medical Incident	04/12/2000	04/12/2000	04/12/2000 10:43:54 PM	04/12/2000 10:45:53 PM	04/12/2000 10:49:59 PM	04/12/2000 10:50:35 PM	04/12/200
4	1040007	E12	30697	Structure Fire	04/13/2000	04/12/2000	04/13/2000 12:19:54 AM	04/13/2000 12:29:24 AM	04/13/2000 12:29:35 AM	04/13/2000 12:31:25 AM	04/13/200
5	1040021	M14	30711	Medical Incident	04/13/2000	04/12/2000	04/13/2000 01:17:25 AM	04/13/2000 01:18:44 AM	04/13/2000 01:20:02 AM	04/13/2000 01:21:40 AM	04/13/200
6	1040061	M43	30749	Medical Incident	04/13/2000	04/12/2000	04/13/2000 07:51:29 AM	04/13/2000 07:55:35 AM	04/13/2000 07:55:54 AM	04/13/2000 07:59:58 AM	null
7	1040079	E10	30766	Alarms	04/13/2000	04/13/2000	04/13/2000 09:31:19 AM	04/13/2000 09:33:04 AM	04/13/2000 09:34:10 AM	04/13/2000 09:35:52 AM	04/13/200

Showing all 10 rows.

Command took 0.81 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 11:05:43 AM on My First Cluster

Look at the number of emergency calls by call type.

Cmd 10

```
1  SELECT `Call Type`, count(1) as count
2  FROM firecalls_uploaded
3  GROUP BY `Call Type`
4  ORDER BY count DESC
```

▸ (1) Spark Jobs

	Call Type	count
1	Medical Incident	156374
2	Structure Fire	31329
3	Alarms	26090
4	Traffic Collision	9749
5	Other	3799
6	Citizen Assist / Service Call	3600
7	Outside Fire	2940

Showing all 31 rows.

Command took 6.09 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 11:06:33 AM on My First Cluster

Cmd 11

Now look at calls by neighborhood.

Cmd 12

```
1  SELECT `Neighborhooods - Analysis Boundaries` as neighborhood, count(1) as count
2  FROM firecalls_uploaded
3  GROUP BY `Neighborhooods - Analysis Boundaries`
4  ORDER BY count DESC
5
```

▸ (1) Spark Jobs

	neighborhood	count
1	Tenderloin	31564
2	South of Market	23212
3	Mission	21829
4	Financial District/South Beach	16384
5	Bayview Hunters Point	13057
6	Sunset/Parkside	9456
7	Western Addition	8899

Showing all 42 rows.

Command took 5.47 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 11:07:41 AM on My First Cluster

Readings and Resources

Course Notebooks:

- https://files.training.databricks.com/courses/davis/Lessons.dbc

Readings

- Spark SQL and DataFrames and Datasets Guides
- SQL Guide from Databricks

Additional Resources

- Learning Spark, 2nd Edition (eBook compliments of Databricks).
- Introduction - The Internals of Spark SQL (free gitbook)

From <https://www.coursera.org/learn/spark-sql/supplement/l2Suj/readings-and-resources>

Queries in Spark SQL

Module 1 Assignment

☆ In this assignment you:

- Create a table
- Write SQL queries

For each **bold** question, input its answer in Coursera.

Cmd 3

```
%run ../Includes/Classroom-Setup
```

```
Command took 1.88 minutes -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 1:24:30 PM on My Cluster


WARNING: This curriculum was written for DBR 5.5 ML. Please create a new cluster that uses the runtime DBR 5.5 ML
```

OK

Cmd 4

Working with Incident Data

For this assignment, we'll be using a new dataset: the SF Fire Incident dataset. It has been mounted for you using the script above. The path to this dataset is as follows:

`/mnt/davis/fire-incidents/fire-incidents-2016.csv`

In this assignment, you will read the dataset and perform a number of different queries.

Cmd 5

☆ Create a Table

Create a new table called `fireIncidents` for this dataset. Be sure to use options to properly parse the data.

```
Create database if not exists Databricks;
Use Databricks;
Drop table if exists fireIncidents;
Create table if not exists fireIncidents
using csv
options (
 header "true",
 path "/mnt/davis/fire-incidents/fire-incidents-2016.csv",
 inferSchema "true"
 )
```

▸ (2) Spark Jobs

OK

```
Command took 3.53 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 1:36:29 PM on My Cluster
```

Cmd 7

Question 1

Return the first 10 lines of the data. On the Coursera platform, input the result to the following question:

What is the first value for "Incident Number"?

Cmd 8

```
select * from fireIncidents limit 10
```

▸ (1) Spark Jobs

	Incident Number	Exposure Number	Address	Incident Date	Call Number	Alarm DtTm	Arrival DtTm
1	16000003	0	Precita Av/florida Street	01/01/2016	160010015	01/01/2016 12:02:57 AM	01/01/2016 12:08:05 AM
2	16000004	0	1620 Eucalyptus Drive	01/01/2016	160010018	01/01/2016 12:03:02 AM	01/01/2016 12:09:32 AM
3	16000023	0	171 2nd Street	01/01/2016	160010157	01/01/2016 12:35:02 AM	01/01/2016 12:40:17 AM
4	16000034	0	535 Wisconsin Street	01/01/2016	160010210	01/01/2016 12:45:36 AM	01/01/2016 12:50:00 AM
5	16000051	0	El Camino Del Mar/seal Rock Drive	01/01/2016	160010302	01/01/2016 01:01:59 AM	01/01/2016 01:12:01 AM
6	16000053	0	443 Texas Street	01/01/2016	160010319	01/01/2016 01:09:40 AM	01/01/2016 01:15:14 AM
7	16000064	0	1217 Ralston	01/01/2016	160010349	01/01/2016 01:14:00 AM	01/01/2016 01:26:08 AM

Showing all 10 rows.

☆ WHERE Clauses

A `WHERE` clause is used to filter data that meets certain criteria, returning all values that evaluate to be true.

Cmd 10

Question 2

Return all incidents that occurred on Conor's birthday in 2016. For those of you who forgot his birthday, it's April 4th. On the Coursera platform, input the result to the following question:

What is the first value for "Incident Number" on April 4th, 2016?

Remember to use backticks (``) instead of single quotes (") for columns that have spaces in the name.

Cmd 11

```
1  select * from fireIncidents where `Incident Date` = '04/04/2016' limit 10
```

▸ (1) Spark Jobs

	Incident Number	Exposure Number	Address	Incident Date	Call Number	Alarm DtTm	Arrival DtTm	Close DtTm
1	16037478	0	Utah St/15th Street	04/04/2016	160950098	04/04/2016 12:59:29 AM	04/04/2016 01:26:18 AM	04/04/2016 01:26:23 AM
2	16037483	0	1755 Ofarrell Street	04/04/2016	160950150	04/04/2016 01:46:00 AM	04/04/2016 01:51:10 AM	04/04/2016 02:01:07 AM
3	16037484	0	Leavenworth St/eddy Street	04/04/2016	160950157	04/04/2016 01:54:18 AM	04/04/2016 01:57:36 AM	04/04/2016 02:19:33 AM
4	16037492	0	California St/davis Street	04/04/2016	160950202	04/04/2016 02:34:11 AM	04/04/2016 02:36:57 AM	04/04/2016 02:37:15 AM
5	16037503	0	765 Burnett Avenue	04/04/2016	160950309	04/04/2016 04:04:18 AM	04/04/2016 04:08:13 AM	04/04/2016 04:23:39 AM
6	16037505	0	90 Saturn Street	04/04/2016	160950317	04/04/2016 04:08:26 AM	04/04/2016 04:30:14 AM	04/04/2016 04:55:27 AM
7	16037508	0	24th St/folsom Street	04/04/2016	160950340	04/04/2016 04:34:35 AM	04/04/2016 04:37:41 AM	04/04/2016 04:38:40 AM
8	16037530	0	100 Pine Street	04/04/2016	160950579	04/04/2016 07:25:25 AM	04/04/2016 07:30:46 AM	04/04/2016 08:01:30 AM
9	16037540	0	301 Channel Street	04/04/2016	160950701	04/04/2016 08:15:45 AM	04/04/2016 08:20:18 AM	04/04/2016 08:40:45 AM
10	16037541	0	3rd St/carroll Avenue	04/04/2016	160950708	04/04/2016 08:16:45 AM	04/04/2016 08:21:45 AM	04/04/2016 08:37:36 AM

Question 3

Return all incidents that occurred on Conor's *or* Brooke's birthday. For those of you who forgot her birthday too, it's `9/27`.

Is the first fire call in this table on Brooke or Conor's birthday?

```
1  select * from fireIncidents where `Incident Date` IN ('09/27/2016') limit 10
```

▸ (2) Spark Jobs

	Incident Number	Exposure Number	Address	Incident Date	Call Number	Alarm DtTm	Arrival DtTm	Close DtTm	City	Zipcode
1	16107245	0	1 Sansome Street	09/27/2016	162710165	09/27/2016 01:33:15 AM	09/27/2016 01:36:51 AM	09/27/2016 01:41:03 AM	San Francisco	94104
2	16107249	0	1652 Eddy St 2	09/27/2016	162710179	09/27/2016 01:38:58 AM	09/27/2016 01:44:02 AM	09/27/2016 02:05:17 AM	San Francisco	94115
3	16107251	0	41 Dakota Street	09/27/2016	162710205	09/27/2016 02:01:50 AM	09/27/2016 02:07:26 AM	09/27/2016 02:32:47 AM	San Francisco	94107
4	16107261	0	Julian Av Street	09/27/2016	162710289	09/27/2016 03:15:06 AM	09/27/2016 03:21:50 AM	09/27/2016 03:34:39 AM	San Francisco	94103
5	16107280	0	25th St/florida Street	09/27/2016	162710510	09/27/2016 06:32:26 AM	09/27/2016 06:39:50 AM	09/27/2016 06:43:46 AM	San Francisco	94110
6	16107281	0	229 Haight Street	09/27/2016	162710530	09/27/2016 06:38:59 AM	09/27/2016 06:45:16 AM	09/27/2016 06:52:07 AM	San Francisco	94102
7	16107282	0	295 Precita Avenue	09/27/2016	162710533	09/27/2016 06:41:08 AM	09/27/2016 06:47:43 AM	09/27/2016 07:06:56 AM	San Francisco	94110

Showing all 10 rows.

Command took 1.26 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 1:46:46 PM on My Cluster

Cmd 14

Question 4

Return all incidents on either Conor or Brooke's birthday where the `Station Area` is greater than 20.

What is the "Station Area" for the first fire call in this table?

Cmd 15

☆ Aggregate Functions

Aggregate functions compute a single result value from a set of input values. Use the aggregate function `COUNT` to count the total records in the dataset.

```
1  select * from fireIncidents where `Incident Date` IN ('09/27/2016','04/04/2016') and `Station Area` > 20
```

▸ (2) Spark Jobs

	Incident Number	Exposure Number	Address	Incident Date	Call Number	Alarm DtTm	Arrival DtTm	Close DtTm	City	Zipcode
1	16037478	0	Utah St/15th Street	04/04/2016	160950098	04/04/2016 12:59:29 AM	04/04/2016 01:26:18 AM	04/04/2016 01:26:23 AM	San Francisco	94103
2	16037503	0	765 Burnett Avenue	04/04/2016	160950309	04/04/2016 04:04:18 AM	04/04/2016 04:08:13 AM	04/04/2016 04:23:39 AM	San Francisco	94131
3	16037543	0	41 Castle Street	04/04/2016	160950724	04/04/2016 08:21:41 AM	04/04/2016 08:27:11 AM	04/04/2016 08:33:06 AM	San Francisco	94133
4	16037566	0	4101 Noriega St 6 Street	04/04/2016	160951122	04/04/2016 10:02:36 AM	04/04/2016 10:06:35 AM	04/04/2016 10:40:17 AM	San Francisco	94122

Question 5

Count the incidents on Conor's birthday.

How many incidents were on Conor's birthday in 2016?

Cmd 18

```
1  select count(*) from fireIncidents where `Incident Date` IN ('04/04/2016')
```

▸ (2) Spark Jobs

	count(1)
1	80

Question 6

Return the total counts by `Ignition Cause` . Be sure to return the field `Ignition Cause` as well.

Hint: You'll have to use `GROUP BY` for this

How many fire calls had an "Ignition Cause" of "4 act of nature"?

Cmd 20

```
1  select `Ignition Cause`, count(*) from fireIncidents group by `Ignition Cause`
```

▸ (2) Spark Jobs

	Ignition Cause	count(1)
1	null	30690
2	3 failure of equipment or heat source	100
3	u cause undetermined after investigation	156
4	5 cause under investigation	35
5	2 unintentional	561
6	1 intentional	223
7	4 act of nature	5

Showing all 8 rows.

Command took 2.22 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 1:52:25 PM on My Cluster

Sorting

Question 7

Return the total counts by `Ignition Cause` sorted in ascending order.

Hint: You'll have to use `ORDER BY` for this.

What is the most common "Ignition Cause"? (Put the entire string)

Cmd 22

```
1  select `Ignition Cause`, count(*) from fireIncidents group by `Ignition Cause` order by count(*) asc
```

▸ (2) Spark Jobs

	Ignition Cause	count(1)
1	0 cause, other (only used for additional exposures)	1
2	4 act of nature	5
3	5 cause under investigation	35
4	3 failure of equipment or heat source	100
5	u cause undetermined after investigation	156
6	1 intentional	223
7	2 unintentional	561

Showing all 8 rows.

Command took 1.80 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 1:55:42 PM on My Cluster

Cmd 23

Return the total counts by `Ignition Cause` sorted in descending order.

Cmd 24

```
1  select `Ignition Cause`, count(*) from fireIncidents group by `Ignition Cause` order by count(*) desc
```

▸ (2) Spark Jobs

	Ignition Cause	count(1)
1	null	30690
2	2 unintentional	561
3	1 intentional	223
4	u cause undetermined after investigation	156
5	3 failure of equipment or heat source	100
6	5 cause under investigation	35
7	4 act of nature	5

Joins

Create the table `fireCalls` if it doesn't already exist. The path is as follows: `/mnt/davis/fire-calls/fire-calls-truncated-comma.csv`

Cmd 26

```
1  Create database if not exists Databricks;
2  Use Databricks;
3  Drop table if exists fireCalls;
4  Create table if not exists fireCalls
5  using csv
6  options (
7   header "true",
8   path "/mnt/davis/fire-calls/fire-calls-truncated-comma.csv",
9   inferSchema "true"
10  )
```

▸ (2) Spark Jobs

OK

Command took 8.53 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 1:57:40 PM on My Cluster

Cmd 27

Join the two tables on `Battalion` by performing an inner join.

Cmd 28

```
1  Select *
2  from fireIncidents as a
3  inner join fireCalls as b
4  on a.Battalion = b.Battalion
```

▸ (3) Spark Jobs

	Incident Number	Exposure Number	Address	Incident Date	Call Number	Alarm DtTm	Arrival DtTm	Close DtTm	City
1	16000199	0	17th St/sanchez Street	01/01/2016	160011095	01/01/2016 05:25:15 AM	01/01/2016 05:29:15 AM	01/01/2016 05:29:35 AM	San Francisco
2	16000199	0	17th St/sanchez Street	01/01/2016	160011095	01/01/2016 05:25:15 AM	01/01/2016 05:29:15 AM	01/01/2016 05:29:35 AM	San Francisco
3	16000199	0	17th St/sanchez Street	01/01/2016	160011095	01/01/2016 05:25:15 AM	01/01/2016 05:29:15 AM	01/01/2016 05:29:35 AM	San Francisco
4	16000199	0	17th St/sanchez Street	01/01/2016	160011095	01/01/2016 05:25:15 AM	01/01/2016 05:29:15 AM	01/01/2016 05:29:35 AM	San Francisco
5	16000199	0	17th St/sanchez Street	01/01/2016	160011095	01/01/2016 05:25:15 AM	01/01/2016 05:29:15 AM	01/01/2016 05:29:35 AM	San Francisco
6	16000199	0	17th St/sanchez Street	01/01/2016	160011095	01/01/2016 05:25:15 AM	01/01/2016 05:29:15 AM	01/01/2016 05:29:35 AM	San Francisco

Question 8

Count the total incidents from the two tables joined on `Battalion`.

What is the total incidents from the two joined tables?

Cmd 30

```
1  Select count(*)
2  from fireIncidents as a
3  inner join fireCalls as b
4  on a.Battalion = b.Battalion
```

▸ (2) Spark Jobs

	count(1)
1	847094402

Showing all 1 rows.

Command took 28.77 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 1:59:35 PM on My Cluster

Cmd 31

Congratulations! You made it to the end of the assignment!

Cmd 32

102 Spark SQL Core Concepts

Looking at Spark under the hood

Caching Data for increased performance

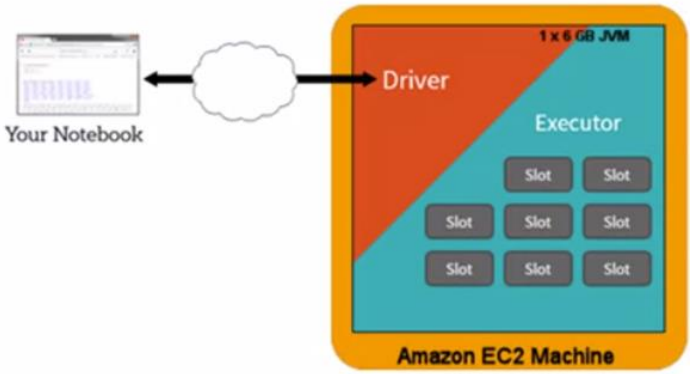
Partitioning: Modify configurations for decreased query time

Reading the Spark UI in order to analyze performance and identify bottlenecks

Spark Local Mode

Driver and executor are on the same physical machine

Same architecture for Databricks community edition



Units of Parallelism within a Spark Cluster

4 executors * 2 slots = 8 units of parallelism

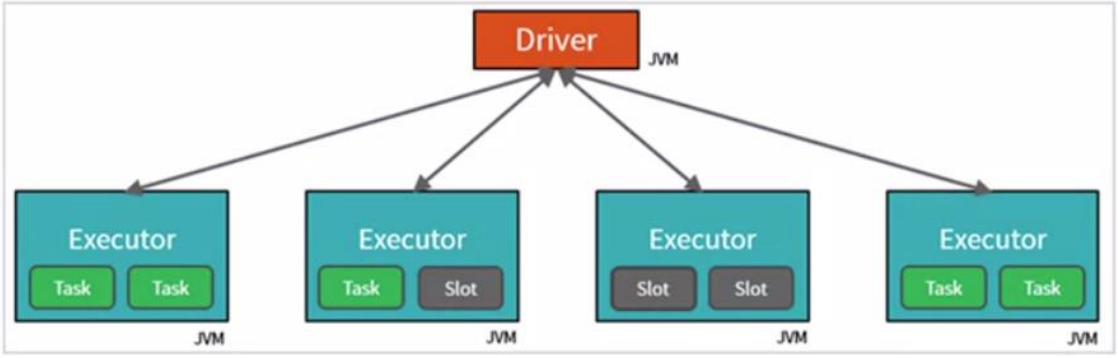
Units of Parallelism

Units of Parallelism Cluster Configuration

MCT: machines * cores * threads

Units of Parallelism for Data are called Partitions

Partitions are part of a large distributed dataset



Determining the number of partitions

Size of Dataset: the larger the dataset the more partitions

Grocery Example

10 friends are to go to the store and pick up 10 items each

Vs.

10 friends are to make 10 trips each to the store and pick up only 1 item each trip

Seeking balance between computation and communication

2.3 Caching

In this notebook you:

- Cache data for increased performance
- Read the Spark UI

Cmd 3

File Statistics

Let's see how large our file is on disk.

Cmd 4

```
1 %run ../Includes/Classroom-Setup
```

```
Command took 2.14 minutes -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 10:37:04 PM on My Cluster


WARNING: This curriculum was written for DBR 5.5 ML. Please create a new cluster that uses the runtime DBR 5.5 ML


OK
```

Cmd 5

```
1 %fs ls /mnt/davis/fire-calls/fire-calls-truncated-comma.csv
```

	path	name	size
1	dbfs:/mnt/davis/fire-calls/fire-calls-truncated-comma.csv	fire-calls-truncated-comma.csv	89222803

Cmd 6

Count

Let's see how long it takes to count all of the records in our dataset.

Cmd 7

```
1 SELECT count(*) FROM fireCalls
```

	count(1)
1	240613

Showing all 1 rows.

Command took 5.43 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 10:39:38 PM on My Cluster

Cmd 8

Cache Table

Cmd 9

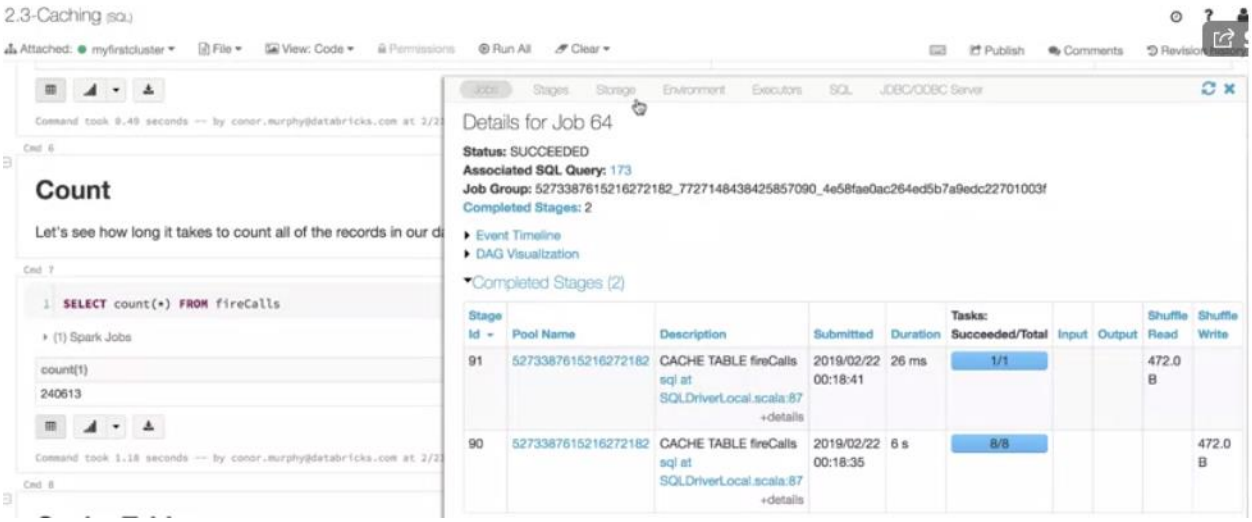
```
1  CACHE TABLE fireCalls
```

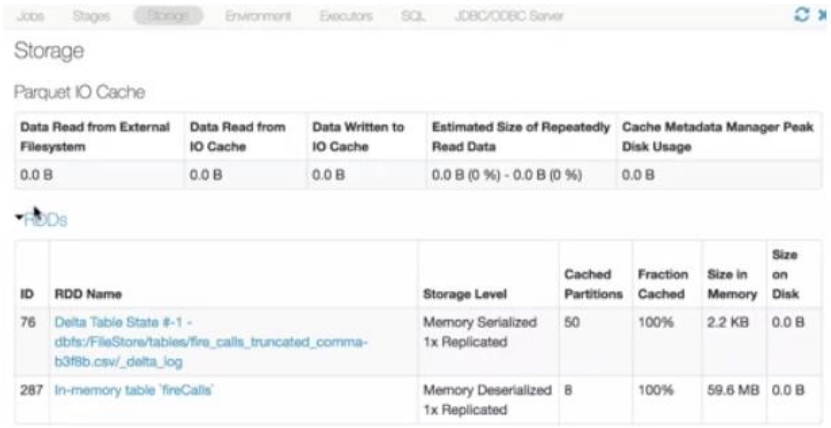
▾ (2) Spark Jobs

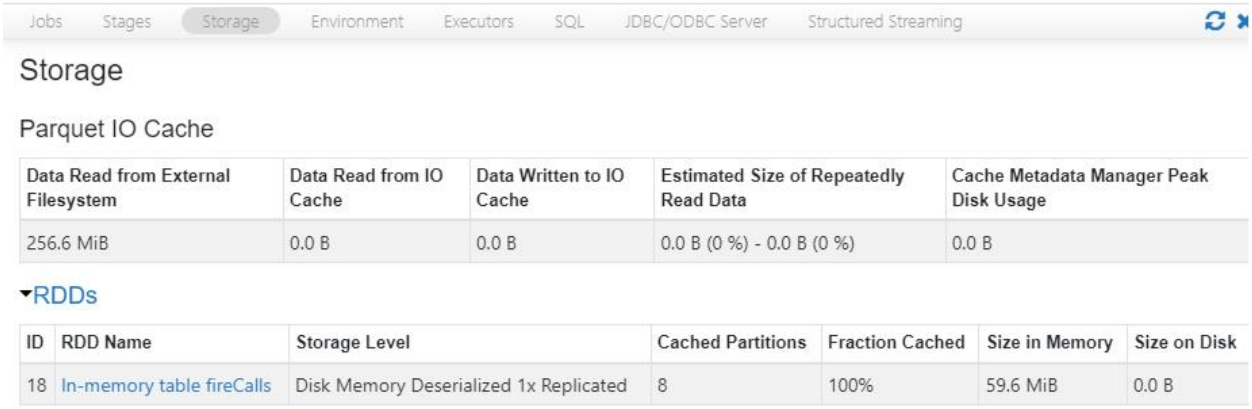
- ▸ Job 4 View (Stages: 1/1)
- ▸ Job 5 View (Stages: 1/1, 1 skipped)

OK

Command took 14.34 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 10:39:54 PM on My Cluster





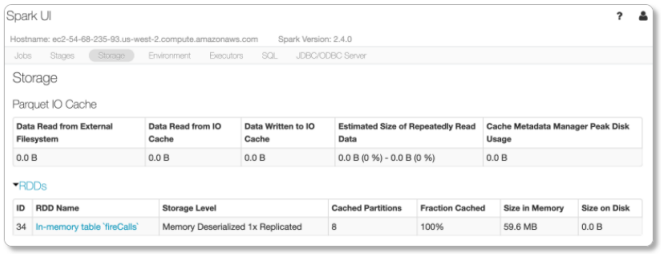


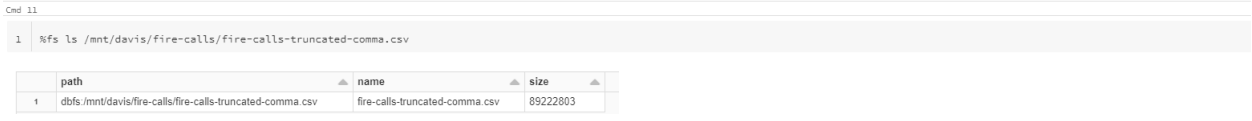
Spark UI

Wow! That took a long time to cache our data. Let's go ahead and take a look at it in the Spark UI.

You'll notice that our data when cached actually takes up less space than our file on disk! That is thanks to the Tungsten Optimizer. You can learn more about Tungsten from Josh Rosen's presentation at Spark Summit.

Our file in memory takes up ~59 MB, and on disk it takes up ~90 MB!





	path	name	size
1	dbfs:/mnt/davis/fire-calls/fire-calls-truncated-comma.csv	fire-calls-truncated-comma.csv	89222803

Count (Again)

Although it took a while to cache our data, every time we query our data, it should be lightning fast. See how long it takes to run the same query!

Cmd 13

```
1  SELECT count(*) FROM fireCalls
```

▸ (2) Spark Jobs

	count(1)
1	240613

Showing all 1 rows.

Command took 0.51 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 10:54:21 PM on My Cluster

Cmd 14

Uncache Table

Wow! That was a lot faster. Now, let's remove our table from the cache.

Cmd 15

```
1  UNCACHE TABLE fireCalls
```

OK

Command took 0.18 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 10:56:25 PM on My Cluster

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

Storage

Parquet IO Cache

Data Read from External Filesystem	Data Read from IO Cache	Data Written to IO Cache	Estimated Size of Repeatedly Read Data	Cache Metadata Manag
256.6 MiB	0.0 B	0.0 B	0.0 B (0 %) - 0.0 B (0 %)	0.0 B

```
1  CACHE LAZY TABLE fireCalls
```

OK

Command took 0.16 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 10:58:41 PM on My Cluster

Cmd 18

Small Query

This query was a lot faster to run. But, did it cache our entire dataset?

Cmd 19

```
1  SELECT * FROM fireCalls LIMIT 100
```

▾ (1) Spark Jobs

▸ Job 10 View (Stages: 1/1)

	Call Number	Unit ID	Incident Number	Call Type	Call Date	Watch Date	Receiv
1	1030118	E08	30625	Medical Incident	04/12/2000	04/12/2000	04/12/2
2	1030122	M18	30630	Medical Incident	04/12/2000	04/12/2000	04/12/2
3	1030154	M36	30662	Medical Incident	04/12/2000	04/12/2000	04/12/2
4	1040007	E12	30697	Structure Fire	04/13/2000	04/12/2000	04/13/2
5	1040021	M14	30711	Medical Incident	04/13/2000	04/12/2000	04/13/2
6	1040061	M43	30749	Medical Incident	04/13/2000	04/12/2000	04/13/2
7	1040079	E10	30766	Alarms	04/13/2000	04/13/2000	04/13/2

Showing all 100 rows.

Command took 2.60 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 10:58:44 PM on My Cluster

Jobs Stages Storage Environment Executors SQL JDBC/ODBC Server Structured St

Storage

Parquet IO Cache

Data Read from External Filesystem	Data Read from IO Cache	Data Written to IO Cache	Estimated Size of Repeatedly Read Data
267.8 MiB	0.0 B	0.0 B	0.0 B (0 %) - 0.0 B (0 %)

▾RDDs

ID	RDD Name	Storage Level	Cached Partitions	Fraction
43	In-memory table fireCalls	Disk Memory Deserialized 1x Replicated	1	13%

☆ Effects of Caching

Cmd 7

Count the number of records in our `fireCalls` table.

Question 1

How many fire calls are in our table?

Cmd 8

```
1  select count(*) from fireCalls
```

▸ (2) Spark Jobs

	count(1)
1	240613

Showing all 1 rows.

Command took 2.63 seconds -- by SWCPROPERTY@GMAIL.COM at 6/27/2021, 12:10:13 AM on My Cluster

Cmd 9

Now speed up your query by caching the data, then counting!

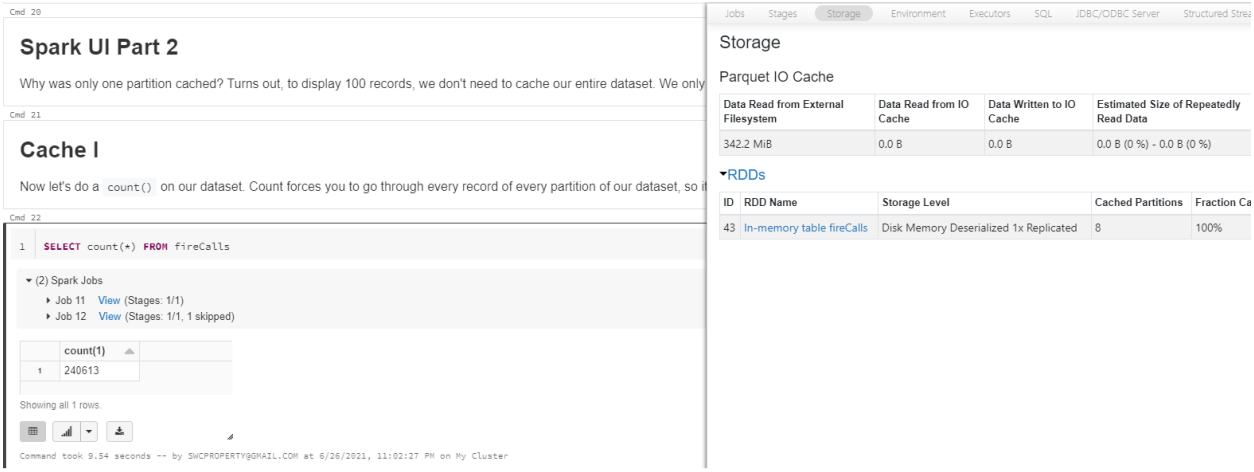
Cmd 10

```
1  cache table fireCalls
```

▾ (2) Spark Jobs
- ▸ Job 28 View (Stages: 1/1)
- ▸ Job 29 View (Stages: 1/1, 1 skipped)

OK

Command took 9.85 seconds -- by SWCPROPERTY@GMAIL.COM at 6/27/2021, 12:11:08 AM on My Cluster


Cmd 20
Spark UI Part 2
Why was only one partition cached? Turns out, to display 100 records, we don't need to cache our entire dataset. We only
Cmd 21
Cache I
Now let's do a count() on our dataset. Count forces you to go through every record of every partition of our dataset, so i
Cmd 22
1 SELECT count(*) FROM fireCalls
(2) Spark Jobs
Job 11 View (Stages: 1/1)
Job 12 View (Stages: 1/1, 1 skipped)
count(1)
1 240613
Showing all 1 rows.
Command took 9.54 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 11:02:27 PM on My Cluster
Jobs Stages Storage Environment Executors SQL JDBC/ODBC Server Structured Strea
Storage
Parquet IO Cache
Data Read from External Filesystem
Data Read from IO Cache
Data Written to IO Cache
Estimated Size of Repeatedly Read Data
342.2 MiB
0.0 B
0.0 B
0.0 B (0 %) - 0.0 B (0 %)
RDDs
ID
RDD Name
Storage Level
Cached Partitions
Fraction Ca
43
In-memory table fireCalls
Disk Memory Deserialized 1x Replicated
8
100%

Count

Look at how fast this call to `count()` is now!

Cmd 24

```
1  SELECT count(*) FROM fireCalls
```

▸ (2) Spark Jobs

	count(1)
1	240613

Showing all 1 rows.

Command took 0.64 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 11:03:50 PM on My Cluster

Cmd 25

Clear Cache

Let's remove any data that is currently cached.

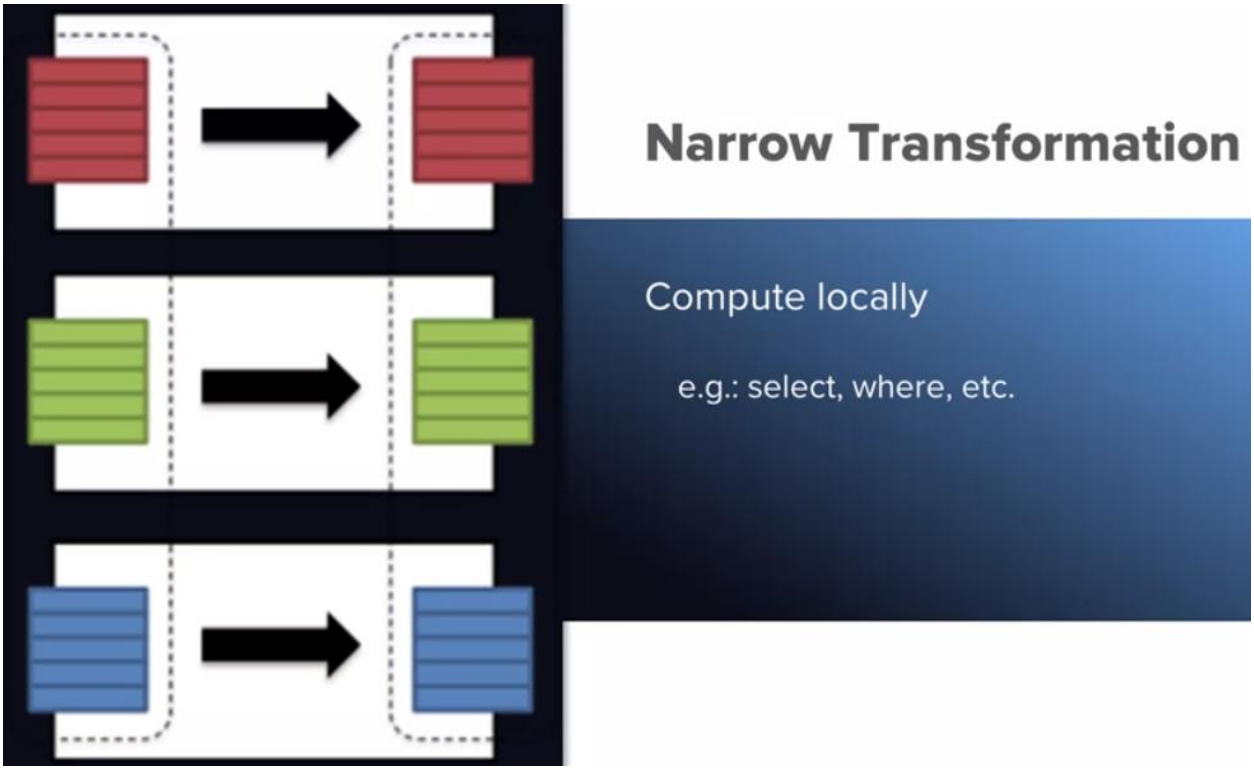
Cmd 26

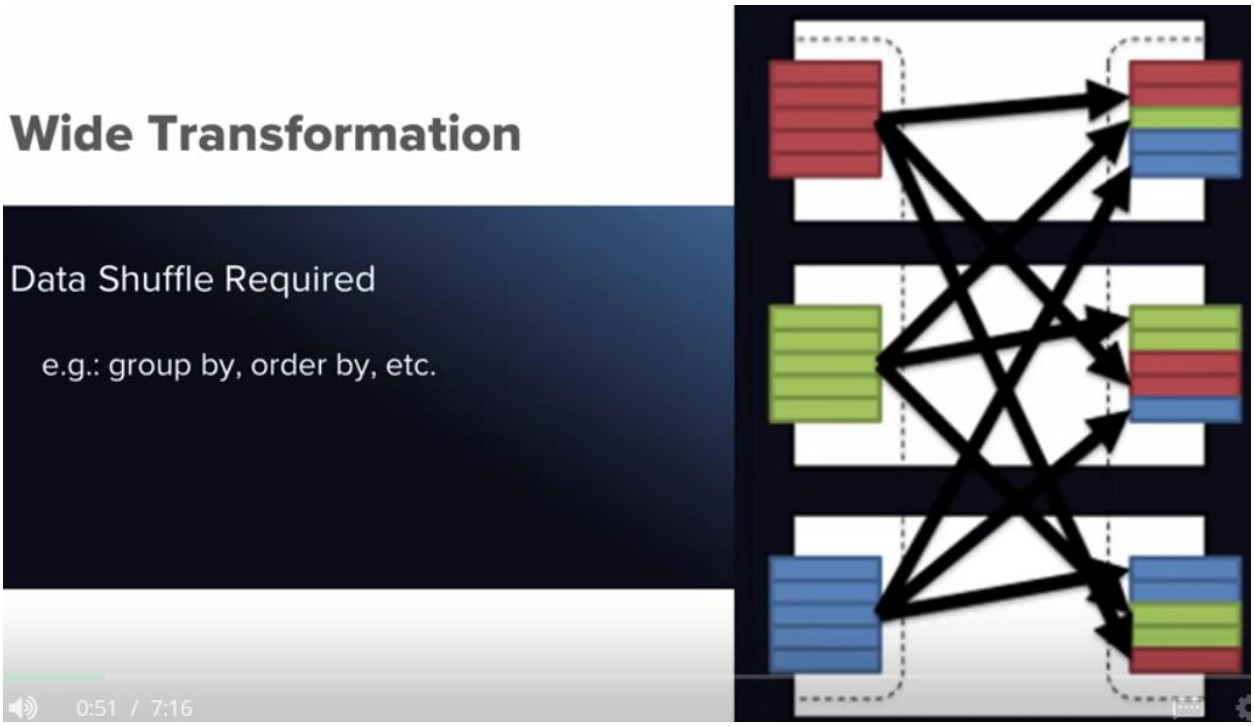
```
1  CLEAR CACHE
2
```

OK

Command took 0.10 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 11:03:59 PM on My Cluster

Cmd 27


Narrow Transformation
Compute locally
e.g.: select, where, etc.


Wide Transformation
Data Shuffle Required
e.g.: group by, order by, etc.
0:51 / 7:16

2.4-Shuffle-Partitions (SQL)

My Cluster | File | Edit | View: Standard | Permissions | Run All | Clear

2.4 Shuffle Partitions

In this notebook you:

- Understand the performance differences between wide and narrow transformations.
- Optimize Spark jobs by configuring Shuffle Partitions.

Cmd 3

Create the table if it doesn't exist.

Cmd 4

```
1  %run ../Includes/Classroom-Setup
```

```
Command took 8.04 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 11:13:17 PM on My Cluster


WARNING: This curriculum was written for DBR 5.5 ML. Please create a new cluster that uses the runtime DBR 5.5 ML


OK
```

Cmd 5

Let's see the most common call types in our dataset. Any guesses?

Cmd 6

```
1  SELECT `call type`, count(*) AS count
2  FROM firecalls
3  GROUP BY `call type`
4  ORDER BY count DESC
```

▾ (2) Spark Jobs

▾ Job 15 View (Stages: 1/1)

Stage 21: 8/8

▾ (2) Spark Jobs

▾ Job 15 View (Stages: 1/1)

Stage 21: 8/8

▸ Job 16 View (Stages: 1/1, 1 skipped)

	call type	count
1	Medical Incident	156374
2	Structure Fire	31329
3	Alarms	26090
4	Traffic Collision	9749
5	Other	3799
6	Citizen Assist / Service Call	3600
7	Outside Fire	2940

Showing all 31 rows.

Command took 7.31 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 11:13:32 PM on My Cluster

What is that 200/200?

Expand out the Spark job above. It should have:

- 1 stage with 2 tasks
- 1 stage with 200 tasks

The number assigned to the Job/Stage will depend on how many Spark jobs you have already executed on your cluster.

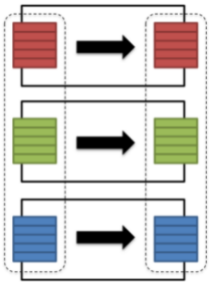
The Community Edition version of Databricks now provisions a single driver node with 2 cores for you. This change explains why your output might look different from the video.

Cmd 8

Narrow Transformations: The data required to compute the records in a single partition reside in at most one partition of the parent DataFrame.

Examples include:

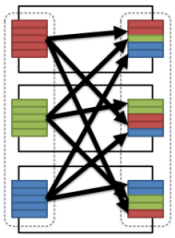
- `SELECT (columns)`
- `DROP (columns)`
- `WHERE`



Wide Transformations: The data required to compute the records in a single partition may reside in many partitions of the parent DataFrame.

Examples include:

- `DISTINCT`
- `GROUP BY`
- `ORDER BY`

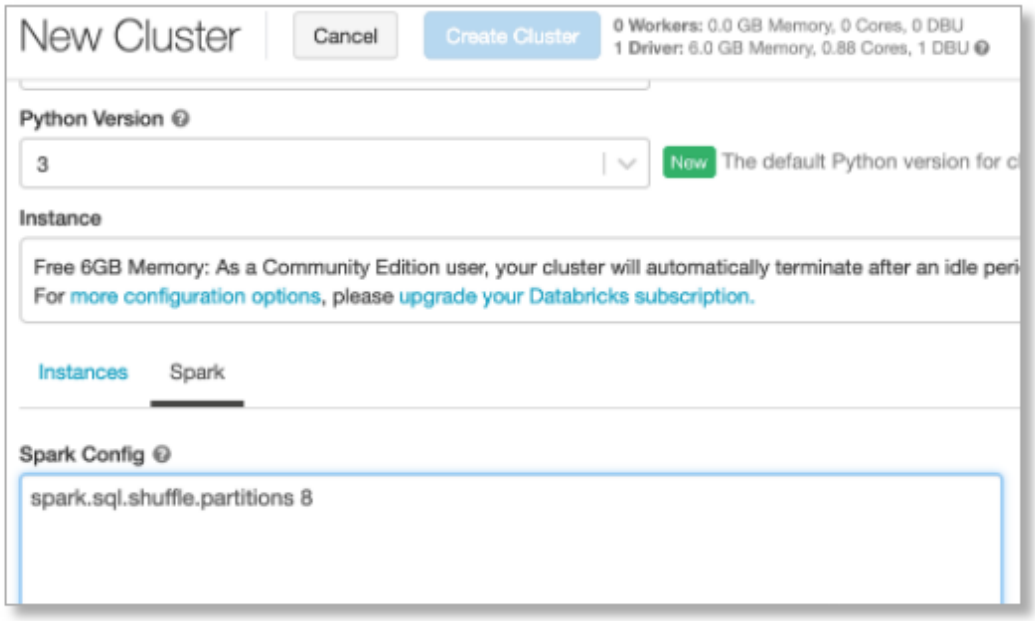


Shuffle Partitions

The `spark.sql.shuffle.partitions` parameter controls how many resulting partitions there are after a shuffle (wide transformation). By default, this value is 200 regardless of how large or small your dataset is, or your clu

Let's change this parameter to be 8 (default parallelism in Databricks Community edition is 2 because that is the number of threads we have available).

This configuration will only be changed for this notebook (it will default to 200 if you switch notebooks/detach from your cluster). If you want to set this parameter for all of your clusters, you can also set this configurati




```
md 10
```

```
1  SET spark.sql.shuffle.partitions=8
```

	key	value
1	spark.sql.shuffle.partitions	8

Showing all 1 rows.

```
Command took 0.30 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 11:28:13 PM on My Cluster
```

```
md 11
```

Let's try this again...

```
1  SELECT `call type`, count(*) AS count
2  FROM firecalls
3  GROUP BY `call type`
4  ORDER BY count DESC
```

▾ (2) Spark Jobs

- ▸ Job 17 View (Stages: 1/1)
- ▸ Job 18 View (Stages: 1/1, 1 skipped)

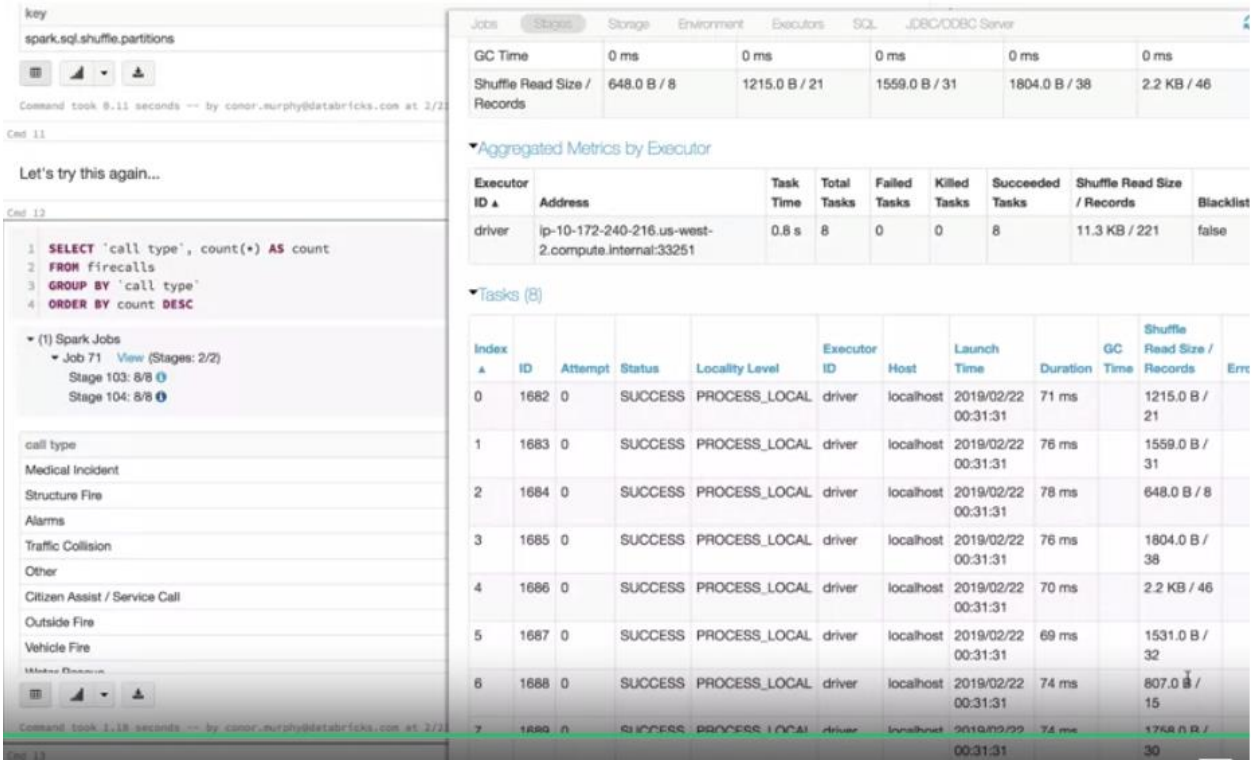
	call type	count
1	Medical Incident	156374
2	Structure Fire	31329
3	Alarms	26090
4	Traffic Collision	9749
5	Other	3799
6	Citizen Assist / Service Call	3600
7	Outside Fire	2940

Showing all 31 rows.

Command took 4.02 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 11:28:48 PM on My Cluster

Cmd 13

Wow! That was a bit faster, and we didn't have to change any of our SQL query code!


key
spark.sql.shuffle.partitions
Command took 0.11 seconds -- by conor.murphy@databricks.com at 2/21
Cmd 11
Let's try this again...
Cmd 12
SELECT `call type`, count(*) AS count
FROM firecalls
GROUP BY `call type`
ORDER BY count DESC
(1) Spark Jobs
Job 71 View (Stages: 2/2)
Stage 103: 8/8
Stage 104: 8/8
call type
Medical Incident
Structure Fire
Alarms
Traffic Collision
Other
Citizen Assist / Service Call
Outside Fire
Vehicle Fire
Command took 1.18 seconds -- by conor.murphy@databricks.com at 2/21
Cmd 13
Jobs Stages Storage Environment Executors SQL JDBC/ODBC Server
GC Time 0 ms 0 ms 0 ms 0 ms 0 ms
Shuffle Read Size / Records 648.0 B / 8 1215.0 B / 21 1559.0 B / 31 1804.0 B / 38 2.2 KB / 46
Aggregated Metrics by Executor
Executor ID Address Task Time Total Tasks Failed Tasks Killed Tasks Succeeded Tasks Shuffle Read Size / Records Blacklist
driver ip-10-172-240-216.us-west-2.compute.internal:33251 0.8 s 8 0 0 8 11.3 KB / 221 false
Tasks (8)
Index ID Attempt Status Locality Level Executor ID Host Launch Time Duration GC Time Shuffle Read Size / Records
0 1682 0 SUCCESS PROCESS_LOCAL driver localhost 2019/02/22 00:31:31 71 ms 1215.0 B / 21
1 1683 0 SUCCESS PROCESS_LOCAL driver localhost 2019/02/22 00:31:31 76 ms 1559.0 B / 31
2 1684 0 SUCCESS PROCESS_LOCAL driver localhost 2019/02/22 00:31:31 78 ms 648.0 B / 8
3 1685 0 SUCCESS PROCESS_LOCAL driver localhost 2019/02/22 00:31:31 76 ms 1804.0 B / 38
4 1686 0 SUCCESS PROCESS_LOCAL driver localhost 2019/02/22 00:31:31 70 ms 2.2 KB / 46
5 1687 0 SUCCESS PROCESS_LOCAL driver localhost 2019/02/22 00:31:31 69 ms 1531.0 B / 32
6 1688 0 SUCCESS PROCESS_LOCAL driver localhost 2019/02/22 00:31:31 74 ms 807.0 B / 15
7 1689 0 SUCCESS PROCESS_LOCAL driver localhost 2019/02/22 00:31:31 74 ms 1758.0 B / 30

Let's try this again...

Cmd 12

```
1  SELECT `call type`, count(*) AS count
2  FROM firecalls
3  GROUP BY `call type`
4  ORDER BY count DESC
```

▸ (2) Spark Jobs

	call type	count
1	Medical Incident	156374
2	Structure Fire	31329
3	Alarms	26090
4	Traffic Collision	9749
5	Other	3799
6	Citizen Assist / Service Call	3600
7	Outside Fire	2940

Showing all 31 rows.

Command took 4.02 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 11:28:48 PM on My Cluster

Cmd 13

Wow! That was a bit faster, and we didn't have to change any of our SQL query code!

Extension

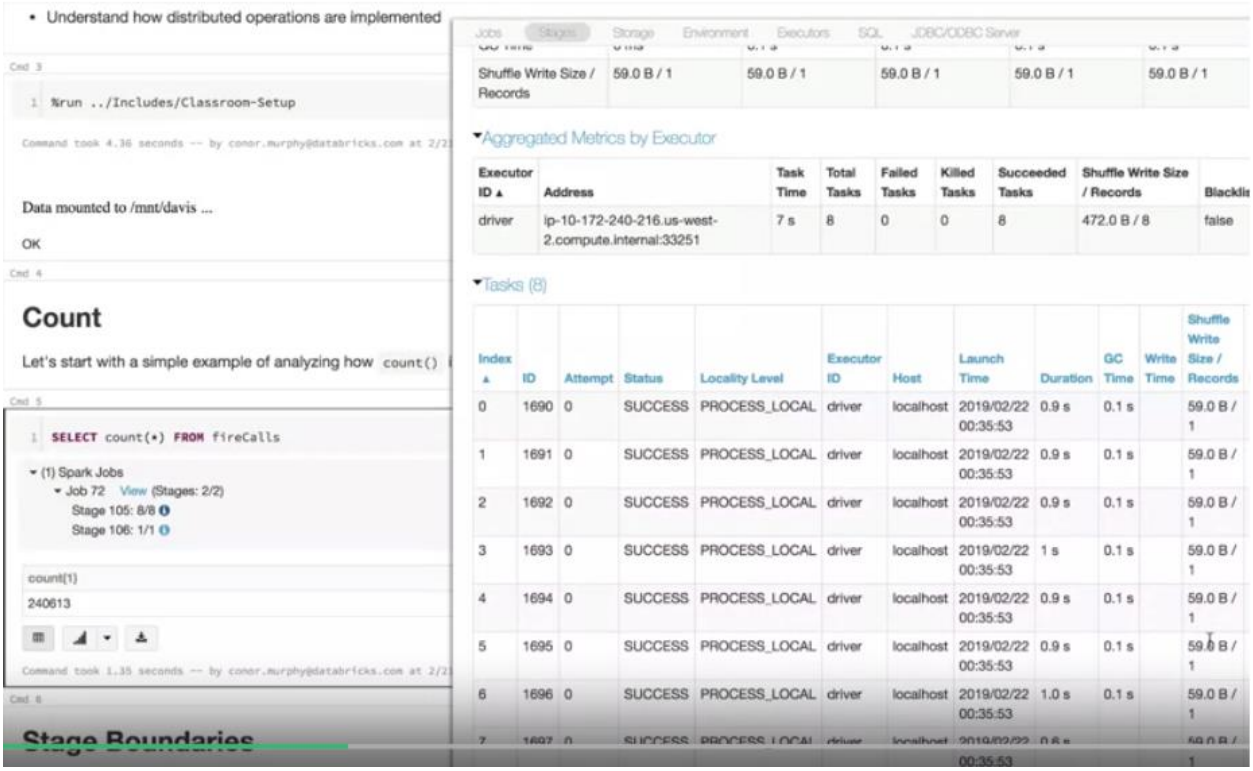
Try changing the shuffle partitions parameter to different values (e.g. 8, 64, 100, 400) and see how it impacts the performance.

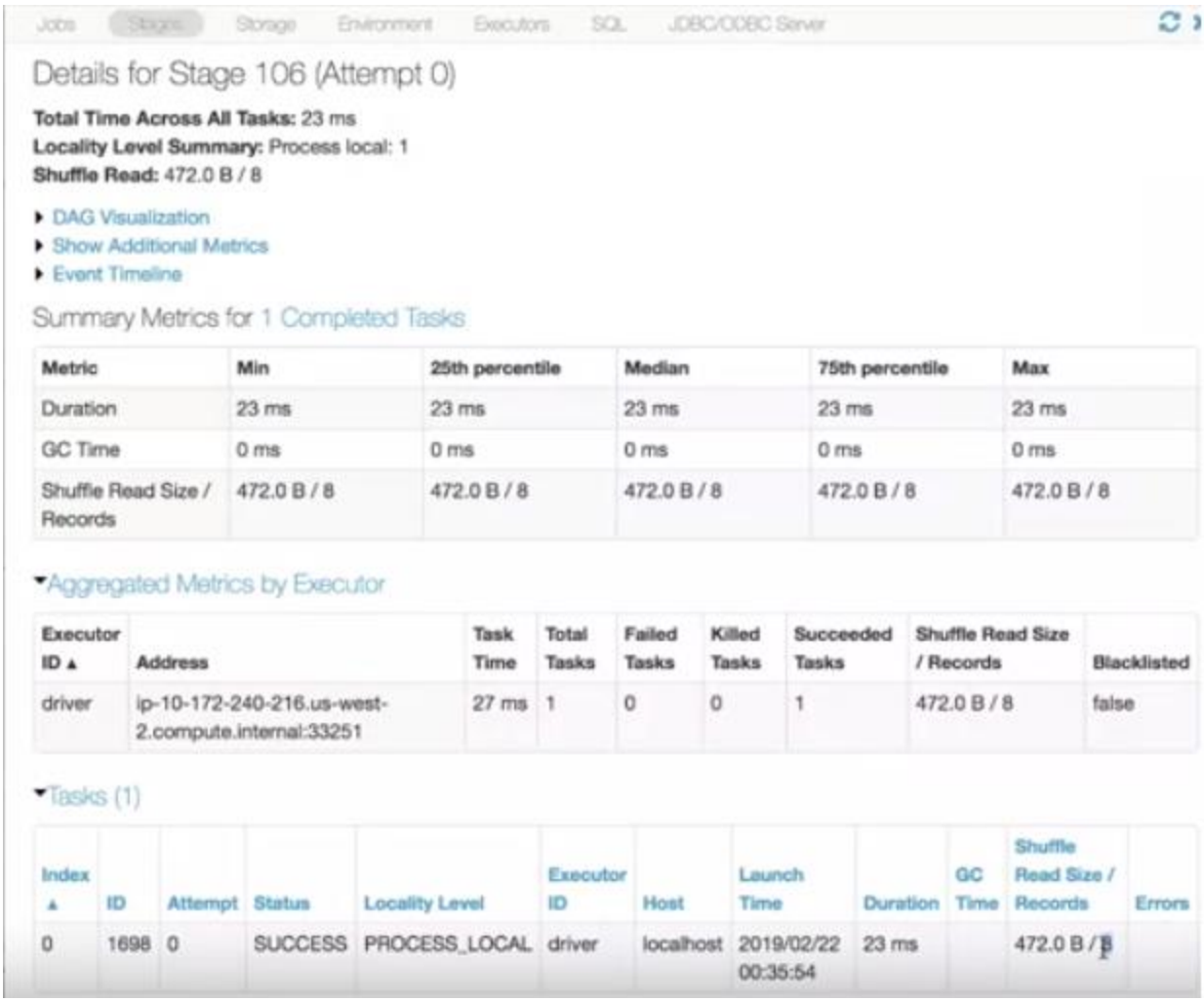
Cmd 15

How to detect bottlenecks in queries

Analyze Spark UI to examine:

- Shuffle reads
- Shuffle writes
- Detect data skew across partitions




Jobs
Stages
Storage
Environment
Executors
SQL
JDBC/ODBC Server
Details for Stage 106 (Attempt 0)
Total Time Across All Tasks: 23 ms
Locality Level Summary: Process local: 1
Shuffle Read: 472.0 B / 8
DAG Visualization
Show Additional Metrics
Event Timeline
Summary Metrics for 1 Completed Tasks
Metric | Min | 25th percentile | Median | 75th percentile | Max
Duration | 23 ms | 23 ms | 23 ms | 23 ms | 23 ms
GC Time | 0 ms | 0 ms | 0 ms | 0 ms | 0 ms
Shuffle Read Size / Records | 472.0 B / 8 | 472.0 B / 8 | 472.0 B / 8 | 472.0 B / 8 | 472.0 B / 8
Aggregated Metrics by Executor
Executor ID | Address | Task Time | Total Tasks | Failed Tasks | Killed Tasks | Succeeded Tasks | Shuffle Read Size / Records | Blacklisted
driver | ip-10-172-240-216.us-west-2.compute.internal:33251 | 27 ms | 1 | 0 | 0 | 1 | 472.0 B / 8 | false
Tasks (1)
Index | ID | Attempt | Status | Locality Level | Executor ID | Host | Launch Time | Duration | GC Time | Shuffle Read Size / Records | Errors
0 | 1698 | 0 | SUCCESS | PROCESS_LOCAL | driver | localhost | 2019/02/22 00:35:54 | 23 ms | | 472.0 B / 8

2.5 Spark UI

☆ In this notebook you:

- Analyze the Spark UI
- Understand how distributed operations are implemented

Cmd 3

```
%run ../Includes/Classroom-Setup
```

```
Command took 7.27 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 11:36:27 PM on My Cluster


WARNING: This curriculum was written for DBR 5.5 ML. Please create a new cluster that uses the runtime DBR 5.5


OK
```

Cmd 4

Count

Let's start with a simple example of analyzing how `count()` is implemented.

Cmd 5

```
SELECT count(*) FROM fireCalls
```

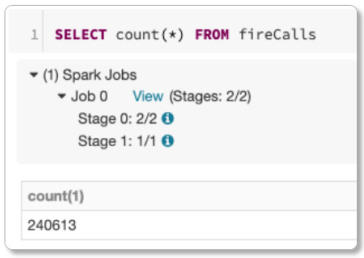
▾ (2) Spark Jobs
- ▸ Job 19 View (Stages: 1/1)
- ▸ Job 20 View (Stages: 1/1, 1 skipped)

	count(1)
1	240613

Showing all 1 rows.

Stage Boundaries

Start by expanding the Spark Job. You'll notice the job was broken into two stages.

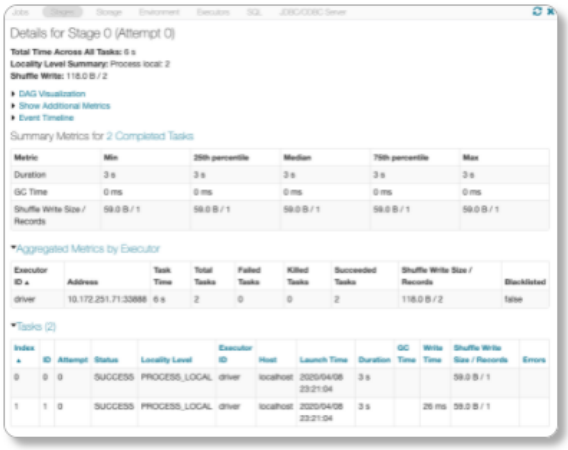


When we perform a count, each executor has to sum up their count locally. Only once all of them finish, then one slot is tasked with adding up all of the counts from the other executors. So it doesn't matter if you have bottleneck is the slowest executor :)

The Community Edition version of Databricks now provisions a single driver node with 2 cores for you. This change explains why your output might look different from the video.

Shuffle Write

When our executors locally count their number of records, they have to write it to their local disk for that one designated e



You'll notice the **2 shuffle records** that were written out (each of them is very small).

Shuffle Read

When that designated executor reads in the "shuffle files" from the other executors that is called a "shuffle read". Click on

Examine the Spark UI

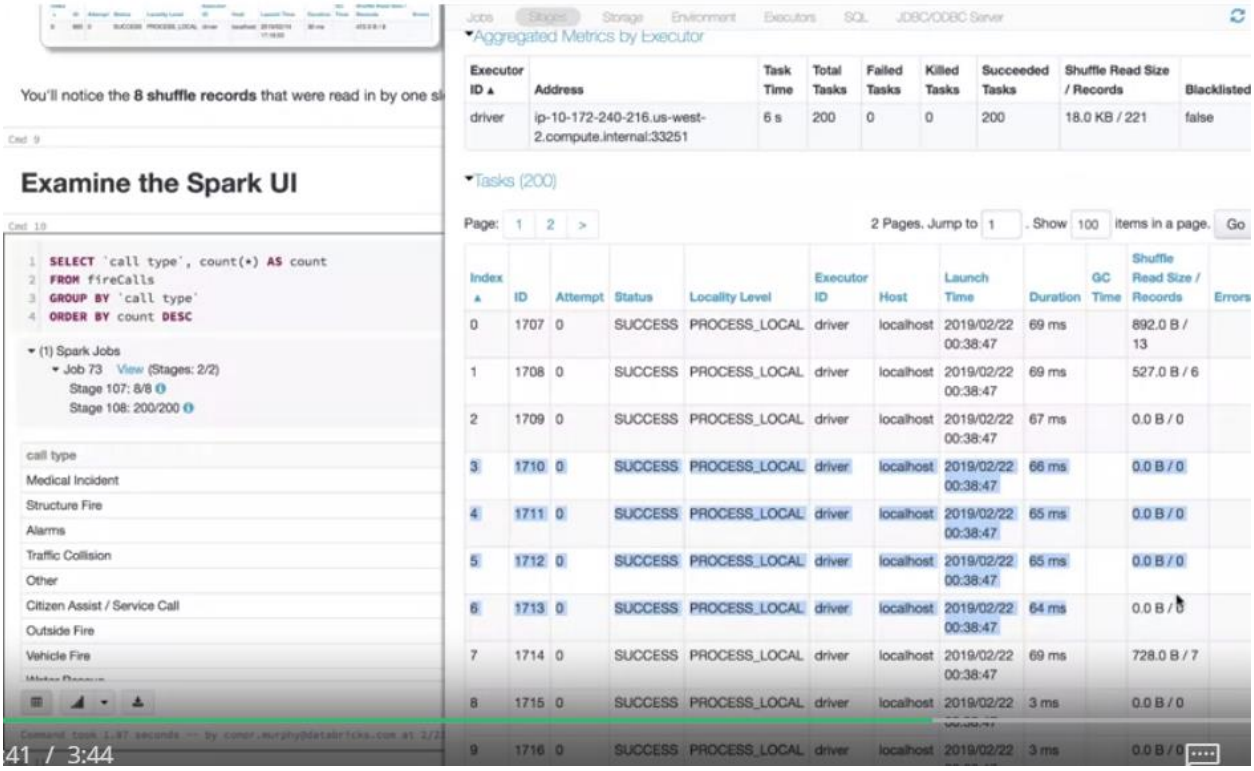
Cmd 10

```
1  SELECT `call type`, count(*) AS count
2  FROM fireCalls
3  GROUP BY `call type`
4  ORDER BY count DESC
```

▾ (2) Spark Jobs

▸ Job 21 View (Stages: 1/1)

▸ Job 22 View (Stages: 1/1, 1 skipped)



	call type	count
1	Medical Incident	156374
2	Structure Fire	31329
3	Alarms	26090
4	Traffic Collision	9749
5	Other	3799
6	Citizen Assist / Service Call	3600
7	Outside Fire	2940

Showing all 31 rows.

Command took 4.22 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 11:41:04 PM on My Cluster

d 11

Do you recall that 200/200 parameter? Because of it, we have to have 200 resulting partitions,

d 12

```
1  SET spark.sql.shuffle.partitions=8
```

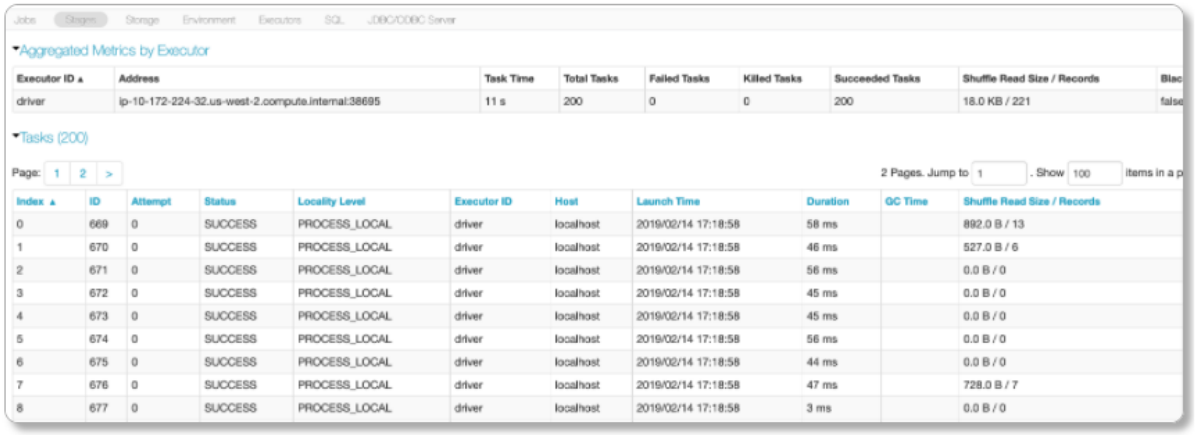
	key	value
1	spark.sql.shuffle.partitions	8

Showing all 1 rows.

Command took 0.12 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 11:42:21 PM on My Cluster

Skewed Partitions

If you look at the Spark UI, you'll notice there is significant skew in our data. Most partitions don't actually have any records!



Jobs Stages Storage Environment Executors SQL JDBC/ODBC Server

Aggregated Metrics by Executor

Executor ID ▲	Address	Task Time	Total Tasks	Failed Tasks	Killed Tasks	Succeeded Tasks	Shuffle Read Size / Records	Blac
driver	ip-10-172-224-32.us-west-2.compute.internal:38695	11 s	200	0	0	200	18.0 KB / 221	false

Tasks (200)

Page: 1 2 > 2 Pages. Jump to 1 . Show 100 items in a p

Index ▲	ID	Attempt	Status	Locality Level	Executor ID	Host	Launch Time	Duration	GC Time	Shuffle Read Size / Records
0	669	0	SUCCESS	PROCESS_LOCAL	driver	localhost	2019/02/14 17:18:58	58 ms		892.0 B / 13
1	670	0	SUCCESS	PROCESS_LOCAL	driver	localhost	2019/02/14 17:18:58	46 ms		527.0 B / 6
2	671	0	SUCCESS	PROCESS_LOCAL	driver	localhost	2019/02/14 17:18:58	56 ms		0.0 B / 0
3	672	0	SUCCESS	PROCESS_LOCAL	driver	localhost	2019/02/14 17:18:58	45 ms		0.0 B / 0
4	673	0	SUCCESS	PROCESS_LOCAL	driver	localhost	2019/02/14 17:18:58	45 ms		0.0 B / 0
5	674	0	SUCCESS	PROCESS_LOCAL	driver	localhost	2019/02/14 17:18:58	56 ms		0.0 B / 0
6	675	0	SUCCESS	PROCESS_LOCAL	driver	localhost	2019/02/14 17:18:58	44 ms		0.0 B / 0
7	676	0	SUCCESS	PROCESS_LOCAL	driver	localhost	2019/02/14 17:18:58	47 ms		728.0 B / 7
8	677	0	SUCCESS	PROCESS_LOCAL	driver	localhost	2019/02/14 17:18:58	3 ms		0.0 B / 0

Cmd 14

Re-run Query

Let's try re-running that query.

Cmd 15

```
1  SELECT `call type`, count(*) AS count
2  FROM fireCalls
3  GROUP BY `call type`
4  ORDER BY count DESC
```

▾ (2) Spark Jobs

▸ Job 23 View (Stages: 1/1)

▸ Job 24 View (Stages: 1/1, 1 skipped)

Let's try re-running that query.

Cmd 15

```
1  SELECT `call type`, count(*) AS count
2  FROM fireCalls
3  GROUP BY `call type`
4  ORDER BY count DESC
```

▾ (2) Spark Jobs

▸ Job 23 View (Stages: 1/1)

▸ Job 24 View (Stages: 1/1, 1 skipped)

	call type	count
1	Medical Incident	156374
2	Structure Fire	31329
3	Alarms	26090
4	Traffic Collision	9749
5	Other	3799
6	Citizen Assist / Service Call	3600
7	Outside Fire	2940

Showing all 31 rows.

Cmd 16

Great! You'll notice that the resulting partitions have a greater number of records and there is less skew across partitions.

▾Aggregated Metrics by Executor

Executor ID ▲	Address	Task Time	Total Tasks	Failed Tasks	Killed Tasks	Succeeded Tasks	Shuffle Read Size / Records	Bl
driver	ip-10-172-224-32.us-west-2.compute.internal:38695	0.4 s	8	0	0	8	11.3 KB / 221	fal

▾Tasks (8)

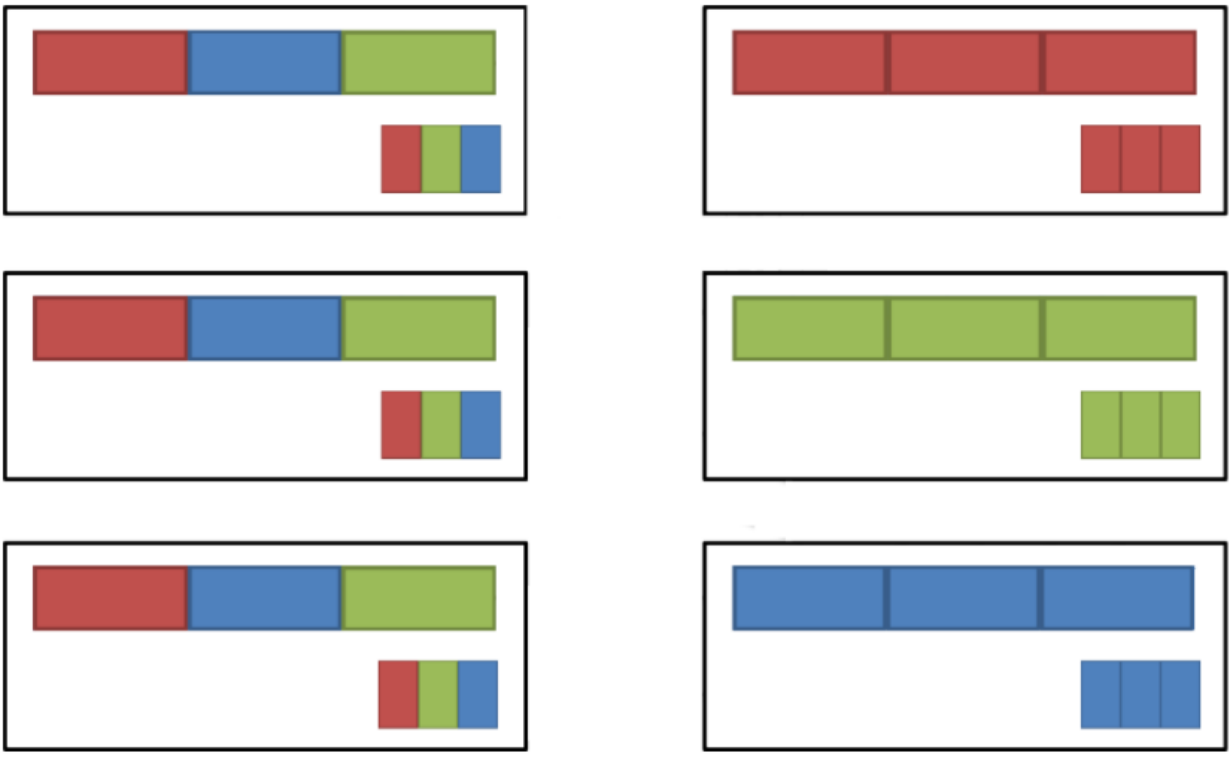
Index ▲	ID	Attempt	Status	Locality Level	Executor ID	Host	Launch Time	Duration	GC Time	Shuffle Read Size / Records
0	901	0	SUCCESS	PROCESS_LOCAL	driver	localhost	2019/02/14 17:23:49	5 ms		1215.0 B / 21
1	902	0	SUCCESS	PROCESS_LOCAL	driver	localhost	2019/02/14 17:23:49	5 ms		1559.0 B / 31
2	903	0	SUCCESS	PROCESS_LOCAL	driver	localhost	2019/02/14 17:23:49	9 ms		648.0 B / 8
3	904	0	SUCCESS	PROCESS_LOCAL	driver	localhost	2019/02/14 17:23:49	9 ms		1804.0 B / 38
4	905	0	SUCCESS	PROCESS_LOCAL	driver	localhost	2019/02/14 17:23:49	6 ms		2.2 KB / 46
5	906	0	SUCCESS	PROCESS_LOCAL	driver	localhost	2019/02/14 17:23:49	4 ms		1531.0 B / 32
6	907	0	SUCCESS	PROCESS_LOCAL	driver	localhost	2019/02/14 17:23:49	4 ms		807.0 B / 15
7	908	0	SUCCESS	PROCESS_LOCAL	driver	localhost	2019/02/14 17:23:49	4 ms		1758.0 B / 30

Cmd 17

Broadcast and Shuffle Joins

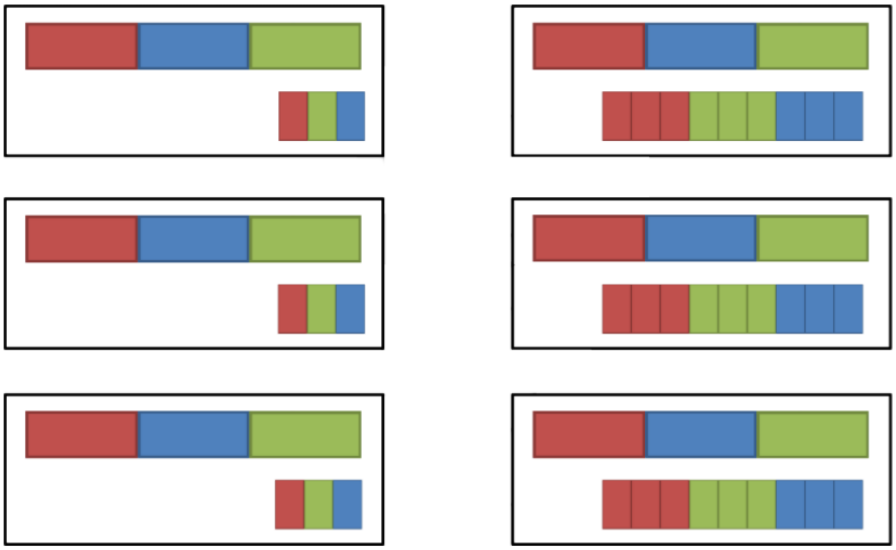
Standard Join

- In a standard join, **ALL** the data is shuffled.
- This can be really expensive.



Broadcast Join

- In a Broadcast Join, only the "small" data is moved.
- It duplicates the "small" data across all executors.
- But the "big" data is left untouched.
- If the "small" data is small enough, this can be **VERY** efficient.



:md 7

In brief, joins are *very* expensive operations. This becomes increasingly apparent in big data environments where joins involve transferring data across a network.

☆ Examining Physical Plans

Cmd 9

Let's make sure our data is accessible.

Cmd 10

```
1  USE databricks;
2
3  DESCRIBE fireCalls
```

	col_name	data_type	comment
1	Call Number	int	null
2	Unit ID	string	null
3	Incident Number	int	null
4	Call Type	string	null
5	Call Date	string	null
6	Watch Date	string	null
7	Received DtTm	string	null

Showing all 34 rows.

Command took 0.44 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 11:54:33 PM on My Cluster

Cmd 11

Now create the table `fireCallsParquet`.

Cmd 12

```
1  CREATE OR REPLACE TEMPORARY VIEW fireCallsParquet
2  USING Parquet
3  OPTIONS (
4      path "/mnt/davis/fire-calls/fire-calls-8p.parquet"
5    )
```

▸ (1) Spark Jobs

OK

We can join these two datasets and examine the physical plan (how data is physically affected) using `EXPLAIN`.

Cmd 14

```
EXPLAIN
  SELECT *
  FROM fireCalls
  JOIN fireCallsParquet on fireCalls.`Call Number` = fireCallsParquet.`Call_Number`
```

	plan
1	== Physical Plan == AdaptiveSparkPlan isFinalPlan=false +- SortMergeJoin [Call Number#6264], [Call_Number#6188], Inner :- Sort [Call Number#6264 ASC NULLS FIRST], false, 0 : +- Exchange hashpartitioning(Call Number#6264, 200), ENSURE_REQUIREMENTS, [id=#1162] : +- Filter isnotnull(Call Number#6264) : +- FileScan csv databricks.firecalls[Call Number#6264,Unit ID#6265,Incident Number#6266,Call Type#6267,Call Date#6268,Watch Date#6269,Received DtTm#6270,Entry DtTm#6271,Dispat...

Showing all 1 rows.

Command took 0.23 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 11:55:52 PM on My Cluster

Cmd 15

Automatic and Manual broadcasting

- Depending on size of the data that is being loaded into Spark, Spark uses internal heuristics to decide how to join that data to other data.
- Automatic broadcast depends on `spark.sql.autoBroadcastJoinThreshold`
  - The setting configures the **maximum size in bytes** for a table that will be broadcast to all worker nodes when performing a join
  - Default is 10MB
- A `broadcast` function can be used in Spark to instruct Catalyst that it should probably broadcast one of the tables that is being joined.

If the `broadcast` hint isn't used, but one side of the join is small enough (i.e., its size is below the threshold), that data source will be read into the Driver and broadcast to all Executors.

Using BroadcastHashJoin

Cmd 16

```
%python
spark.conf.get("spark.sql.autoBroadcastJoinThreshold")
```

```
Out[1]: '10485760b'
```

Command took 0.10 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 11:58:12 PM on My Cluster

Now take a look at the physical plan when we broadcast one of the datasets. The broadcast join hint is going to operate like a SQL hint, but Spark will still parse this even though it is commented out.

Cmd 18

```
EXPLAIN
  SELECT /*+ BROADCAST(fireCalls) */ *
  FROM fireCalls
  JOIN fireCallsParquet on fireCalls.`Call Number` = fireCallsParquet.`Call_Number`

```

	plan
1	== Physical Plan == AdaptiveSparkPlan isFinalPlan=false +- BroadcastHashJoin [Call Number#6264], [Call_Number#6188], Inner, BuildLeft, false :- BroadcastExchange HashedRelationBroadcastMode(List(cast(input[0, int, false] as bigint)),false), [id=#1252] : +- Filter isnotnull(Call Number#6264) : +- FileScan csv databricks.firecalls[Call Number#6264,Unit ID#6265,Incident Number#6266,Call Type#6267,Call Date#6268,Watch Date#6269,Received DtTm#6270,Entry DtTm#6271,Dispatch DtTm#6272,Resp...

Showing all 1 rows.

Command took 0.25 seconds -- by SWCPROPERTY@GMAIL.COM at 6/26/2021, 11:58:18 PM on My Cluster

Cmd 19

Spark Internals

Module 2 Assignment

☆ In this assignment you:

- Analyze the effects of caching data
- Speed up Spark queries by changing default configurations

For each **bold** question, input its answer in Coursera.

Cmd 3

```
1  %run ../Includes/Classroom-Setup
```

```
Command took 6.76 seconds -- by SWCPROPERTY@GMAIL.COM at 6/27/2021, 12:08:56 AM on My Cluster


WARNING: This curriculum was written for DBR 5.5 ML. Please create a new cluster that uses the runtime DBR 5.5 ML


OK
```

Cmd 4

Make sure nothing is already cached by clearing the cache (**NOTE**: This will affect all users on this cluster).

Cmd 5

```
1  CLEAR CACHE
```

OK

```
Command took 0.05 seconds -- by SWCPROPERTY@GMAIL.COM at 6/27/2021, 12:09:06 AM on My Cluster
```

☆ Effects of Caching

md 7

Count the number of records in our `fireCalls` table.

Question 1

How many fire calls are in our table?

Cmd 8

```
1  select count(*) from fireCalls
```

▸ (2) Spark Jobs

	count(1)
1	240613

Showing all 1 rows.

Command took 2.63 seconds -- by SWCPROPERTY@GMAIL.COM at 6/27/2021, 12:10:13 AM on My Cluster

Cmd 9

Now speed up your query by caching the data, then counting!

Cmd 10

```
1  cache table fireCalls
```

▾ (2) Spark Jobs
- ▸ Job 28 View (Stages: 1/1)
- ▸ Job 29 View (Stages: 1/1, 1 skipped)

OK

Command took 9.85 seconds -- by SWCPROPERTY@GMAIL.COM at 6/27/2021, 12:11:08 AM on My Cluster

Look in the Spark UI, how many partitions is our data?

Cmd 12

Changing Spark Configurations

We cached the data to speed up our computation, but let's see if we can get it even faster by changing some default Spark Configurations.

Let's count the number of each type of unit. Group by the `Unit Type`, count the number of calls for each type, and display the unit types with the most calls first.

Question 2

Which "Unit Type" is the most common?

Cmd 14

```
1  select `Unit Type`, count(*) as count
2  from fireCalls
3  group by `Unit Type`
4  order by count desc
```

▸ (2) Spark Jobs

	Unit Type	count
1	ENGINE	92828
2	MEDIC	74219
3	TRUCK	25995
4	CHIEF	17757
5	PRIVATE	14793
6	RESCUE CAPTAIN	8259
7	RESCUE SQUAD	4413

Showing all 10 rows.

Command took 1.69 seconds -- by SWCPROPERTY@GMAIL.COM at 6/27/2021, 12:16:35 AM on My Cluster

Question 3

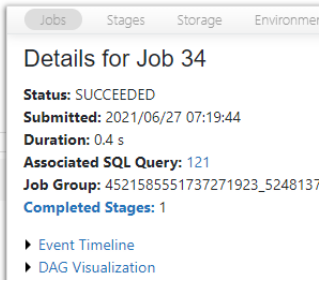
What type of transformation, wide or narrow, did the `GROUP BY` and `ORDER BY` queries result in?

Let's change the `spark.sql.shuffle.partitions` configuration from its default value of `200` and set it to `2`.

Cmd 16

```
1  SET spark.sql.shuffle.partitions=2
```

	key	value
1	spark.sql.shuffle.partitions	2



```
1  select `Unit Type`, count(*) as count
2  from fireCalls
3  group by `Unit Type`
4  order by count desc
```

▾ (2) Spark Jobs
- ▸ Job 34 View (Stages: 1/1)
- ▸ Job 35 View (Stages: 1/1, 1 skipped)

	Unit Type	count
1	ENGINE	92828
2	MEDIC	74219
3	TRUCK	25995
4	CHIEF	17757
5	PRIVATE	14793
6	RESCUE CAPTAIN	8259

Question 3

What type of transformation, wide or narrow, did the `GROUP BY` and `ORDER BY` queries result in?

Let's change the `spark.sql.shuffle.partitions` configuration from its default value of `200` and set it to `2`.

md 16

```
1  SET spark.sql.shuffle.partitions=2
```

	key	value
1	spark.sql.shuffle.partitions	2

Showing all 1 rows.

Command took 0.08 seconds -- by SWCPROPERTY@GMAIL.COM at 6/27/2021, 12:19:35 AM on My Cluster

md 17

Copy and run the code from earlier to get the unit type counts from earlier. Now how long does this query take?

```
1  select `Unit Type`, count(*) as count
2  from fireCalls
3  group by `Unit Type`
4  order by count desc
```

▾ (2) Spark Jobs

- ▸ Job 34 View (Stages: 1/1)
- ▸ Job 35 View (Stages: 1/1, 1 skipped)

	Unit Type	count
1	ENGINE	92828
2	MEDIC	74219
3	TRUCK	25995
4	CHIEF	17757
5	PRIVATE	14793
6	RESCUE CAPTAIN	8259