DataStax Developer Days

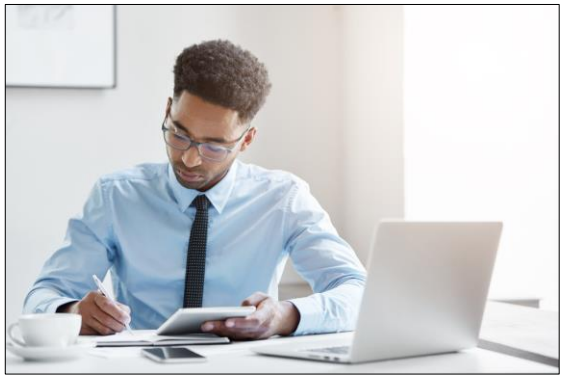
DATASTAX ACADEMY

DataStax Developer Days

Analytics

About You

- How much Apache Cassandra Experience?
- How's your SQL?
- How did you hear about this event?



DATASTAX ACADEMY

What is your data telling you?

Top Uploaders

DATASTAX ACADEMY

```
from cassandra.cluster import Cluster

cluster = Cluster(['172.31.28.68'])
session = cluster.connect("killrvideo")

d = session.execute("select * from user_videos")

counts = {}

for row in d:
  if(row.userid not in counts):
    counts[row.userid] = 1
  else:
    counts[row.userid] += 1

largestCount = 0
largestKey = ""
for key in counts:
  if(counts[key] > largestCount):
    largestCount = counts[key]
    largestKey = key

print("Largest count: " + str(largestCount))
print("Largest user id: " + str(largestKey))
user = session.execute("select firstname, lastname from users where userid = " + str(largestKey))[0]
print("Largest user name: " + user.firstname + " " + user.lastname)
```


WHAT
DA?

Can’t immediately tell what it’s doing

Pulls data from the cluster to a client machine

Execution on one core

Pain to write, pain to read

Think of the children!

SQL…yes… S.Q.L.

```
SELECT firstname, lastname, count(*) AS uploadCount
FROM users NATURAL JOIN user_videos
GROUP BY userid, firstname, lastname
ORDER BY uploadCount DESC
LIMIT 10;
```

OLTP vs. OLAP

I want it, and
I want it now!

Hmmmmmm…

Cassandra

DSE Analytics

DATASTAX ACADEMY

Transactional

- Fast
- Short transactions
- Many of them
- INSERT, UPDATE, DELETE
- Real time

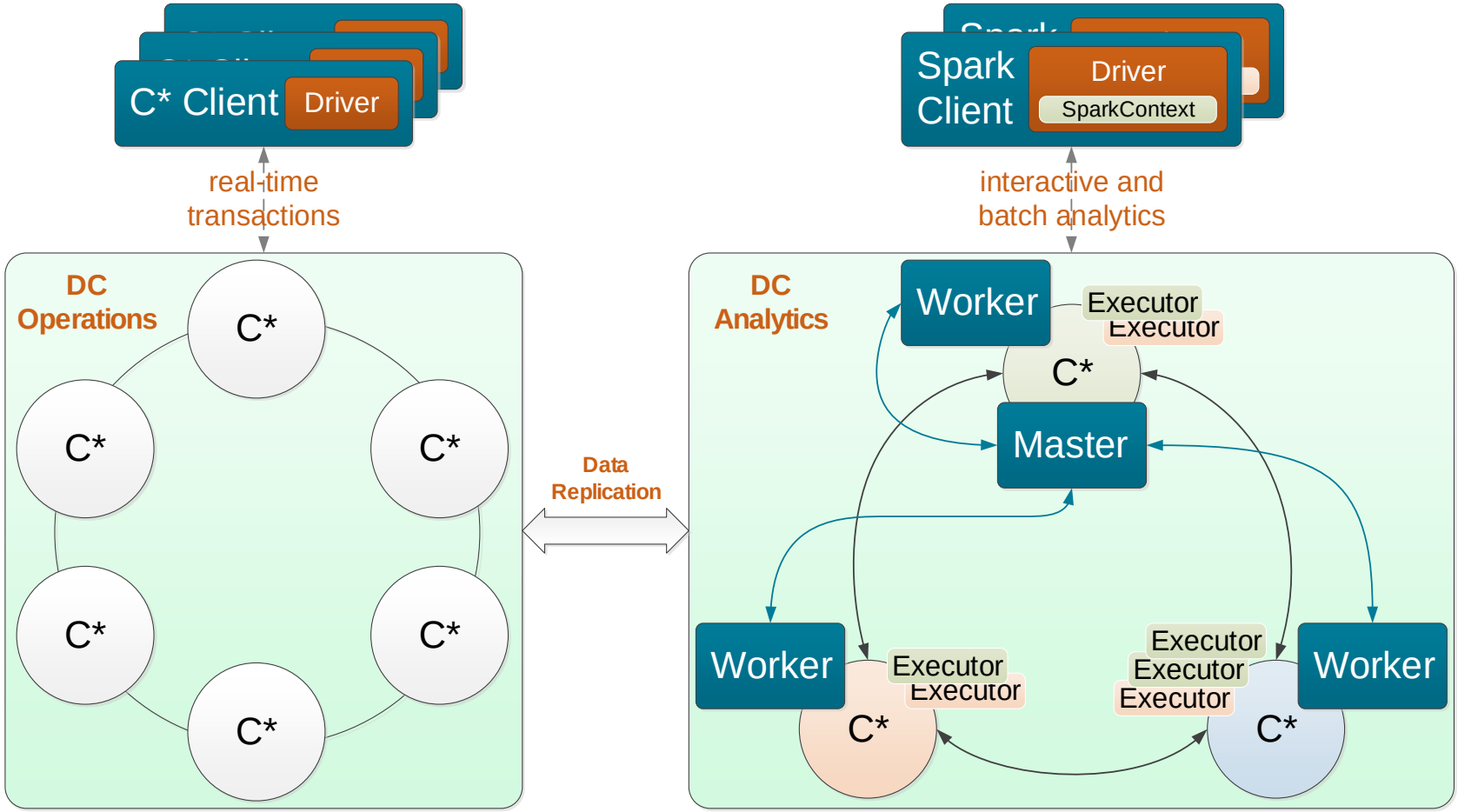
Analytical

- Slow
- Aggregations
- Few of them
- INSERT, UPDATE, DELETE
- Seconds, minutes, hours

“Let’s shrink Big Data into Small Data ...
and hope it magically becomes Great Data.”

DataStax Enterprise Analytics

- Apache Spark™
- Integrated
- Start your cluster with -k
- Or set it in the configuration file


C* Client
Driver
real-time
transactions
Spark
Client
Driver
SparkContext
interactive and
batch analytics
DC
Operations
C*
C*
C*
C*
C*
C*
Data
Replication
DC
Analytics
Worker
Executor
Executor
C*
Master
Worker
Executor
Executor
C*
Executor
Executor
Executor
Worker
C*

DATASTAX ACADEMY

“Apache Cassandra™ is a data store.”

dse spark-sql

```
ubuntu@dse-node:~$ dse spark-sql
The log file is at /home/ubuntu/.spark-sql-shell.log
spark-sql>
```

DATASTAX ACADEMY

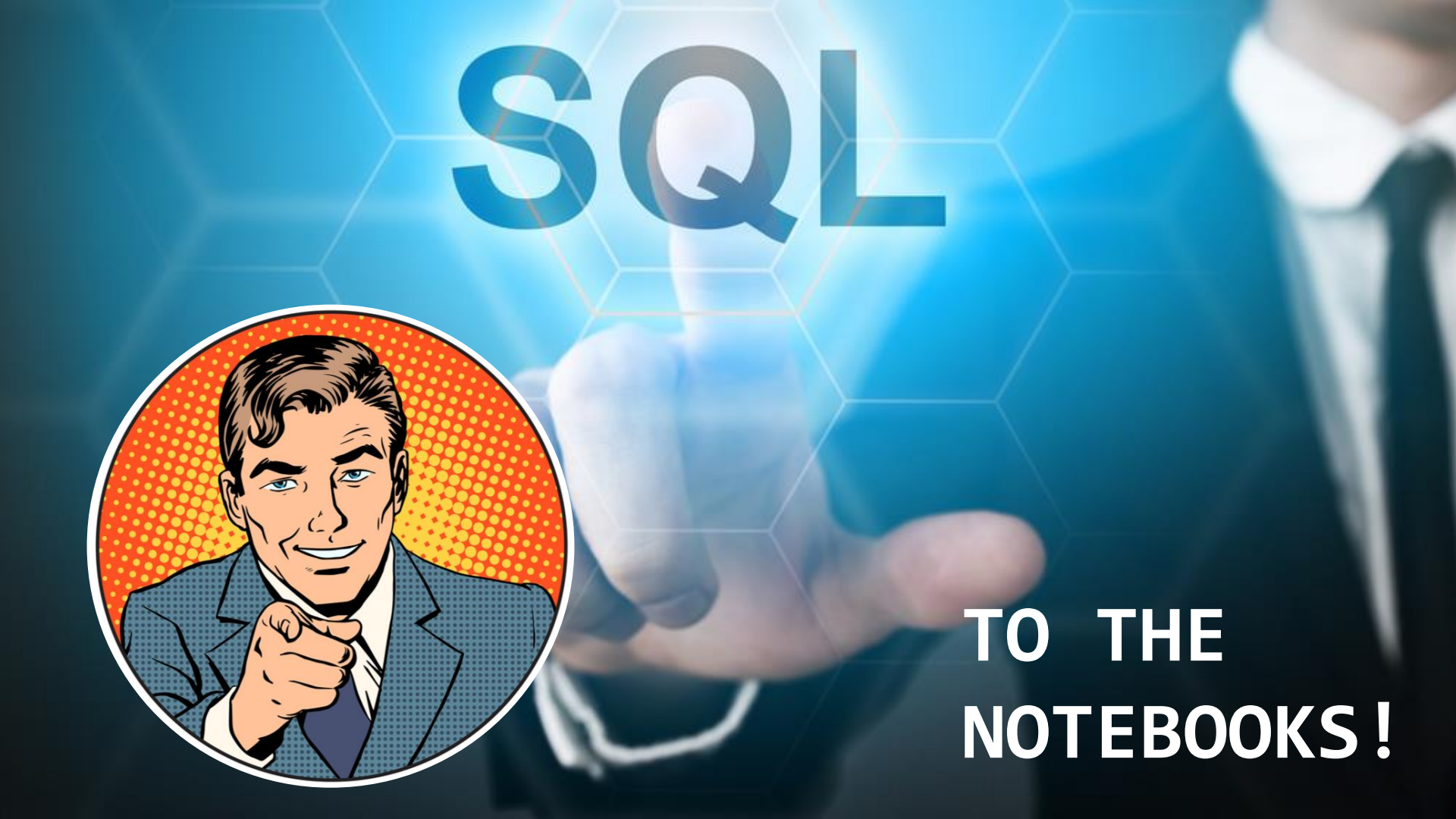
Try it out!

- SSH into your cluster
  - (Or use your browser)
- Start up dse spark-sql
- SELECT some data from a KillrVideo table

ssh? Huh?

- Download the pem file we sent via email
- Open a terminal

```
ssh -i yourpemfilename.pem ubuntu@yournodeipaddress
```


SQL
TO THE
NOTEBOOKS!

Write SQL Queries to Find the Following

- How many user ratings are there?
- What `userid` has the largest number of ratings?
- What are each user's average rating?
- What `userids` are the negative nelly's, that is, the users who consistently rate videos poorly? Consistently means at least eight movie ratings, and their average rating is less than 2.5.
- What are the actual names of the users that are negative nellies?

How many user ratings are there?

```
SELECT count(*)
FROM killrvideo.video_ratings_by_user;
```

What userid has the largest number of ratings?

DATASTAX ACADEMY

```
SELECT userid, count(*) as numRatings
FROM killrvideo.video_ratings_by_user
GROUP BY userid
ORDER BY numRatings DESC
LIMIT 1;
```

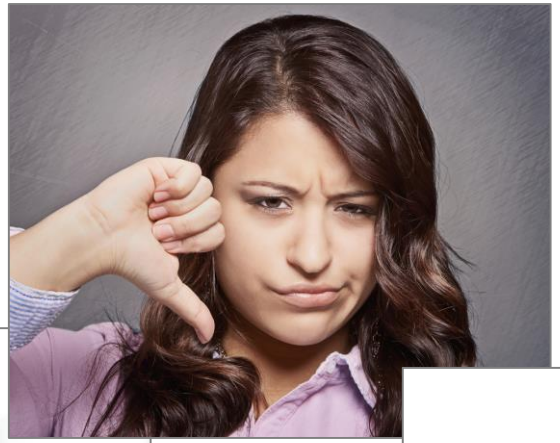
What are each user's average rating?

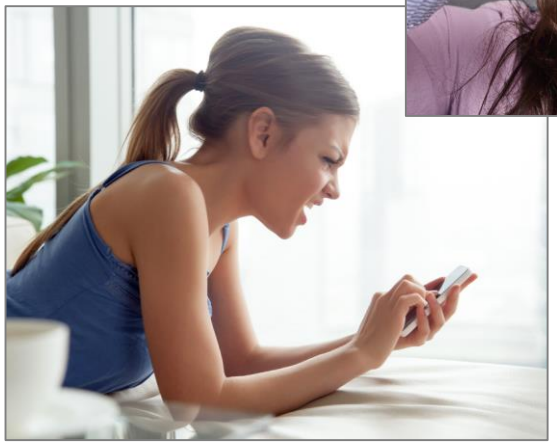
```
SELECT userid, count(*) as numRatings, avg(rating)
FROM killrvideo.video_ratings_by_user
GROUP BY userid
ORDER BY numRatings DESC
LIMIT 10;
```

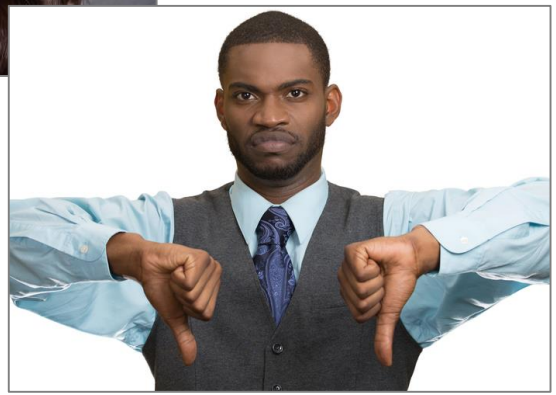
What userids are the negative nelly's?

```
SELECT userid, count(rating) as theCount, AVG(rating) as theAvg
FROM killrvideo.video_ratings_by_user
GROUP BY userid
HAVING theCount > 7 AND theAvg < 2.5
ORDER BY theCount DESC, theAvg ASC;
```

Nagative Nelly – Douggie Downers







What are the names?

```
SELECT firstname, lastname, count(rating) as theCount, AVG(rating) as theAvg
FROM killrvideo.users NATURAL JOIN killrvideo.video_ratings_by_user
GROUP BY userid, firstname, lastname
HAVING count(rating) > 7 AND AVG(rating) < 2.5;
```

DATASTAX ACADEMY

Apache Spark™ REPL

Read-Eval-Print-Loop

REPL

- Read-Evaluate-Print Loop
- Terminal for Apache Spark™ commands
- Variations
  - Spark SQL
  - Scala
  - Python

dse spark

DATASTAX ACADEMY

```
ubuntu@developer-day-node1:~$ dse spark
The log file is at /home/ubuntu/.spark-shell.log
Creating a new Spark Session
Spark context Web UI available at http://13.57.198.178:4041
Spark Context available as 'sc' (master = dse://?, app id = app-20180410160751-0001).
Spark Session available as 'spark'.
Spark SqlContext (Deprecated use Spark Session instead) available as 'sqlContext'
Welcome to

      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.2.0.11
      /_/

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_161)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Scala

- Built on top of Java
- More terse
- Functional

DATASTAX ACADEMY

```
scala> println("Hello World")

Hello World
```

SparkSession

- REPL automatically sets up a variable named `spark`
  - Instance of a `SparkSession`
- `SparkSession` is your entry point for all things Spark

Spark SQL

DATASTAX ACADEMY

- Run SQL commands via `spark.sql()`
- SQL statement doesn't need semicolon because no longer in Spark SQL shell
- Returns a DataFrame

```
spark.sql("""SELECT firstname, lastname, count(rating) as theCount,
  AVG(rating) as theAvg
  FROM killrvideo.users NATURAL JOIN killrvideo.video_ratings_by_user
  GROUP BY userid, firstname, lastname
  HAVING count(rating) > 7 AND AVG(rating) < 2.5""")
```

Multi-Line Entry

- Sometimes the Scala REPL struggles interpreting multi-line statements
- Type `:paste` then hit enter
  - Puts the REPL into paste mode
- Paste your multi-line expression, then type CTRL-D to get out of paste mode

- ***Be sure your cursor is on a new line before you type CTRL-D***

vars and vals

- vars are variable (value can change)
- vals are constant (immutable)

```
scala> val cowName = "Betsy"
cowName: String = Betsy

scala> cowName = "Bessie"
<console>:12: error: reassignment to val
       cowName = "Bessie"
               ^

scala> var dogName = "Sally"
dogName: String = Sally

scala> dogName = "Tex"
dogName: String = Tex
```

DATASTAX ACADEMY

DataFrames

Open the tutorial notebook and learn up on DataFrames

DataFrame Summary

- Schema
- Rows
- Columns
- Queries
- `spark.sql()`

DATASTAX ACADEMY

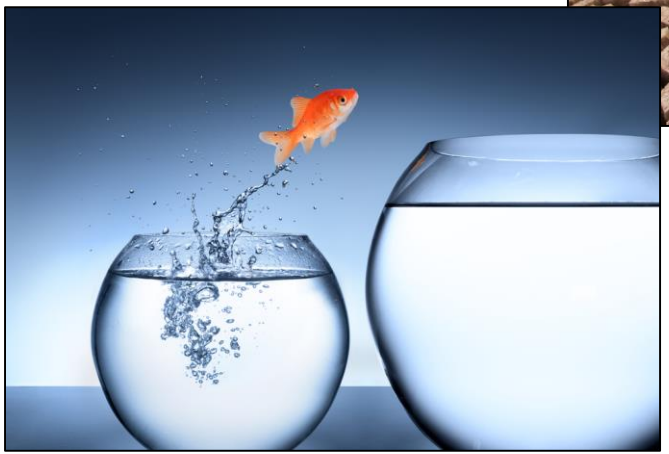
ETL

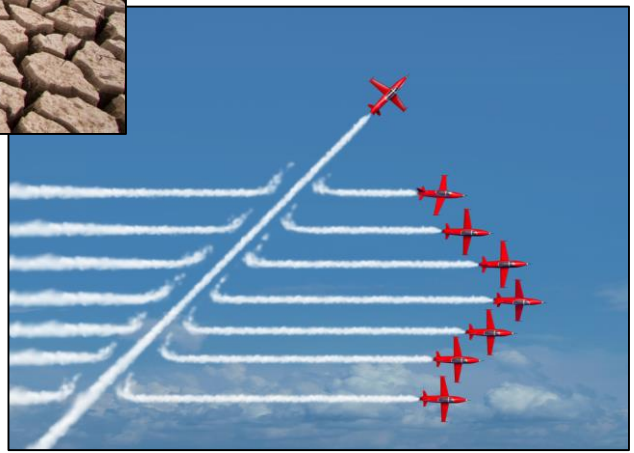
Extract Transform Load

ETL

EXTRACT
TRANSFORM
LOAD

Requirements Change





“Data modelling for Cassandra is so…waterfall.”

“You weren’t kidding about getting the primary key correct!”

Schema Evolution with DSE Analytics

1. Pull your data
2. Make a new table
3. Save
4. Party!

THE GOAL

```
CREATE TABLE killrvideo.comments_by_user_video (
    userid uuid,
    commentid timeuuid,
    comment text,
    videoid uuid,
    PRIMARY KEY ((userid, videoid), commentid)
)
```

Pull Your Data

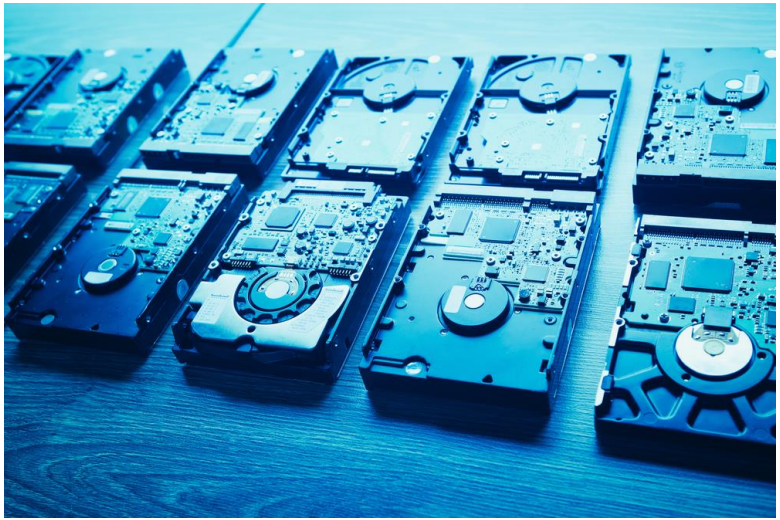
```
val pulled = spark.sql("SELECT * FROM killrvideo.comments_by_user")
```

Make a New Table

```
pulled.createCassandraTable(
  "killrvideo",
  "comments_by_user_video",
  partitionKeyColumns = Some(Seq("userid", "videoid")),
  clusteringKeyColumns = Some(Seq("commentid"))
)
```

Save

```
pulled
  .write
  .cassandraFormat("comments_by_user_video", "killrvideo")
  .save()
```




PARTY!

Pulling Your Data

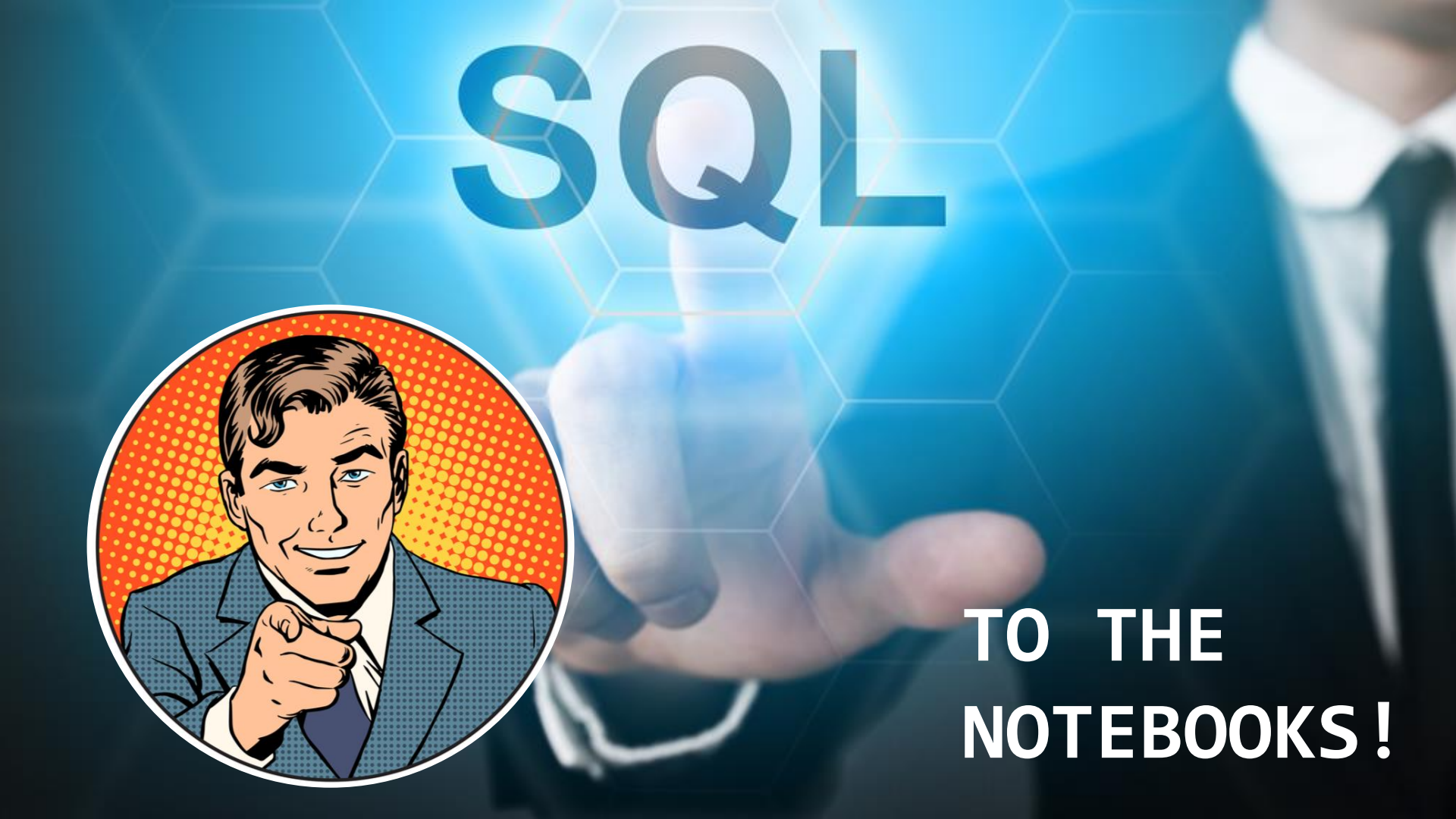
- Transform your query however you like in your SQL statement
- You don't actually pull until you `save()`
- And the *cluster* then does all the work!

- yeah, that's cool

Likely Story

You boss comes to you and asks, “Hey Jimmy, I need you to add a report where I can view how many times a person has rated our videos and what their average rating is. I need to view this data on a daily basis, that is, what are their numbers for any given day.”

Whatcha gonna do?


SQL
TO THE
NOTEBOOKS!

Write a query to pull the userid, count(), and avg() rating of each user:

DATASTAX ACADEMY

```
SELECT userid, count(rating) as theCount, AVG(rating) as theAvg
FROM killrvideo.video_ratings_by_user
GROUP BY userid
```

Join your last query to a new query that adds in the user's firstname and lastname.

```
SELECT userid, firstname, lastname, count(rating) as theCount,
AVG(rating) as theAvg
FROM killrvideo.users NATURAL JOIN
     killrvideo.video_ratings_by_user
GROUP BY userid, firstname, lastname
```

Add a column to bucket all existing values into a single day.

DATASTAX ACADEMY

```
val bucket = spark.sql("""
  SELECT userid, firstname, lastname,
         count(rating) as theCount, AVG(rating) as theAvg
  FROM killrvideo.users NATURAL JOIN
       killrvideo.video_ratings_by_user
  GROUP BY userid, firstname, lastname""")
  .withColumn("days_since_epoch", expr(daysSinceEpoch().toString))
```

Use createCassandraTable() to create your table in Cassandra.

```
bucket.createCassandraTable(
  "killrvideo",
  "video_ratings_by_day",
  partitionKeyColumns = Some(Seq("days_since_epoch")),
  clusteringKeyColumns = Some(Seq("userid"))
)
```

Save your results to the database.

```
bucket
  .write
  .cassandraFormat("video_ratings_by_day", "killrvideo")
  .save()
```


Be a
hero!