Data Exploration with Apache Drill - Day 1

Charles S. Givre
@cgivre
thedataist.com
linkedin.com/in/cgivre

Expectations for this class:

- Please participate and **ask questions.** You can use the slack channel, or email me questions at cgivre@thedataist.com.
- Please follow along and TRY OUT the examples yourself during the class
- All the answers are in the slide decks, but please try to complete the exercises without looking at the answers.
- Have fun!

Conventions for this class:

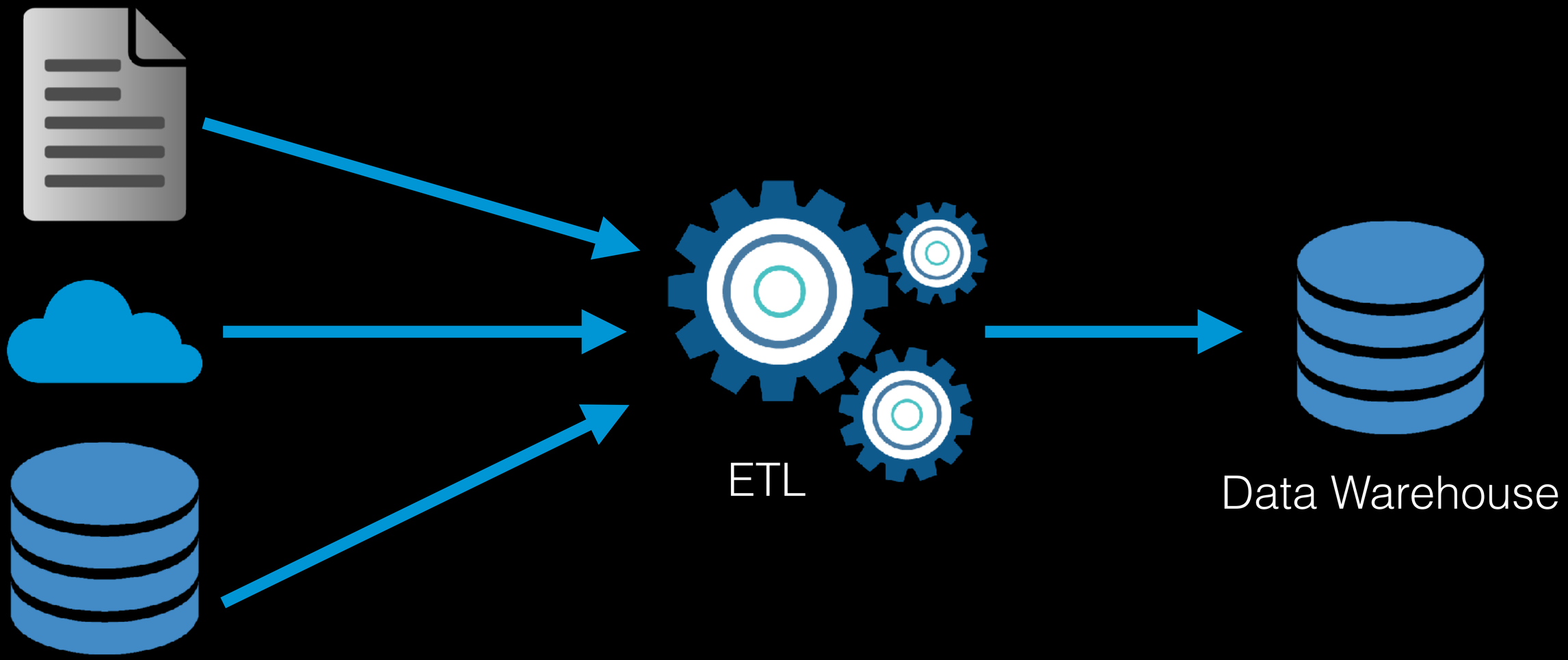
- SQL Commands and Keywords will be written in ALL CAPS
- Variable names will use underscores and be completely in lowercase
- File names will be as they are in the file system
- User provided input will be enclosed in <input>

The problems

We want SQL and BI support
without compromising flexibility
and ability of NoSchema datastores.

Data is not arranged in an optimal way for ad-hoc analysis

Data is not arranged in an optimal way for ad-hoc analysis

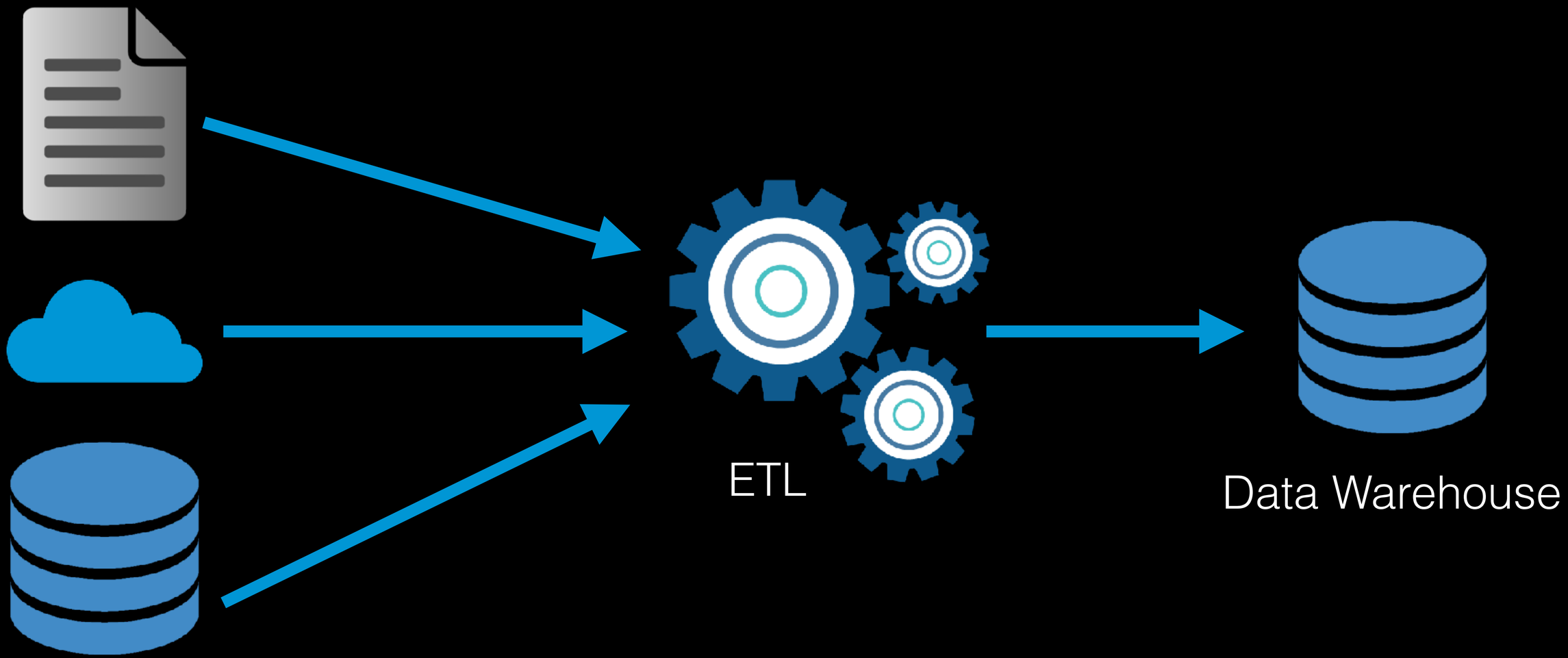


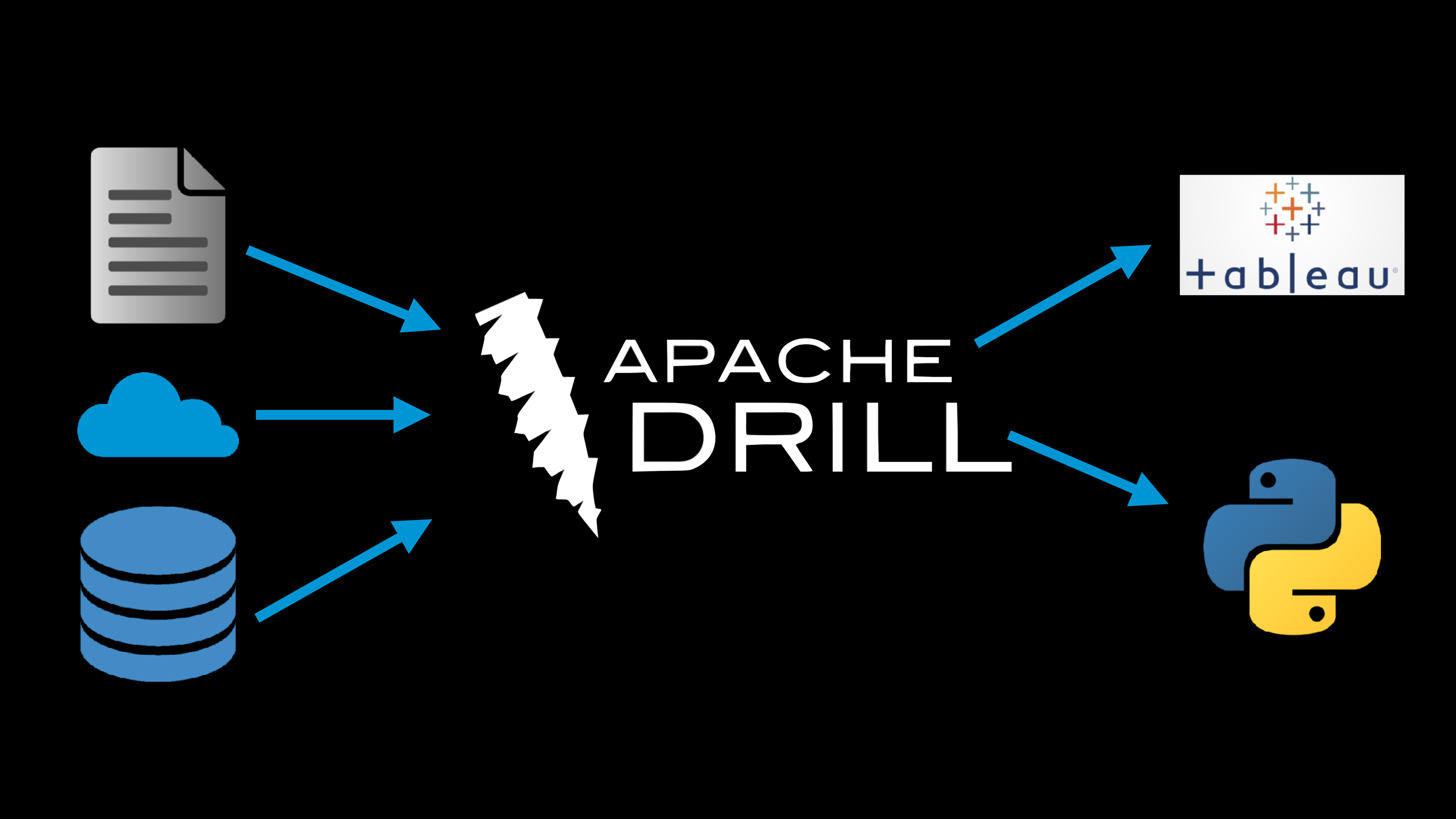
Analytics teams spend between 50%-90% of their time preparing their data.

76% of Data Scientist say this is the least enjoyable part of their job.

http://visit.crowdflower.com/rs/416-ZBE-142/images/CrowdFlower_DataScienceReport_2016.pdf

The ETL Process **consumes the most time** and **contributes almost no value** to the end product.


ETL
Data Warehouse


APACHE
DRILL
+ableau

You just query the data...
no schema

Drill is NOT just SQL on Hadoop

Drill scales

Drill is open source

Download Drill at: drill.apache.org

Why should you use Drill?

Why should you use Drill?
Drill is **easy to use**

Drill is **easy to use**

Drill uses **standard ANSI SQL**

Drill is **FAST!!**

CASE-1: Aggregation Query: Total number of records that have a rating of "3" or above	
Spark	spark-submit --class AggQuery --conf "spark.driver.memory=3g" sql1/target/scala-2.11/sparksqldemo1-assembly-1.0.jar
	real 0m52.631s user 1m31.097s sys 0m1.002s
Drill	drill-embedded -f sql1.sql (ALTER SYSTEM SET `store.json.read_numbers_as_double` = true; SELECT COUNT(MOVIE) FROM dfs.`ratings.json` WHERE RATE > 3.0;)
	real 0m23.917s user 0m26.819s sys 0m0.703s

https://www.mapr.com/blog/comparing-sql-functions-and-performance-apache-spark-and-apache-drill

CASE-2: Join Query: The user and movie names were joined, and the top 10 female users who most rated movies were extracted	
Spark	spark-submit --class topTenWomen --conf "spark.driver.memory=3g" sql2-1/target/scala-2.11/sparksqldemo2-1-assembly-1.0.jar
	real 0m57.671s user 1m40.031s sys 0m1.152s
Drill	drill-embedded -f sql2-1.sql (ALTER SYSTEM SET `store.json.read_numbers_as_double` = true; SELECT RATtbl.UID, COUNT(RATtbl.UID) as NUMEVALS FROM dfs.`ratings.json` as RATtbl JOIN dfs.`users.json` as USRtbl ON RATtbl.UID = USRtbl.UID WHERE USRtbl.GENDER = 'F' GROUP BY RATtbl.UID ORDER BY COUNT(RATtbl.UID) DESC LIMIT 10;)
	real 0m27.576s user 0m28.370s sys 0m0.803s

https://www.mapr.com/blog/comparing-sql-functions-and-performance-apache-spark-and-apache-drill

CASE-3: Join Query: The user and movie names were joined, and the top 10 names of highly rated movies were extracted	
Spark	spark-submit --class topTenMoviename --conf "spark.driver.memory=3g" sql2-2/target/scala-2.11/sparksqldemo2-2-assembly-1.0.jar
	real 0m57.982s user 1m41.480s sys 0m1.246s
Drill	drill-embedded -f sql2-2.sql (ALTER SYSTEM SET `store.json.read_numbers_as_double` = true; SELECT TMP2.MOVIE, TMP2.NUMRAT, TITLE FROM (SELECT MOVIE, COUNT(MOVIE) as NUMRAT FROM dfs.`ratings.json` GROUP BY MOVIE) TMP2 JOIN dfs.`movies.json` AS MOVtbl ON TMP2.MOVIE = MOVtbl.MOVIE ORDER BY TMP2.NUMRAT DESC LIMIT 10;)
	real 0m28.217s user 0m29.263s sys 0m0.658s

https://www.mapr.com/blog/comparing-sql-functions-and-performance-apache-spark-and-apache-drill

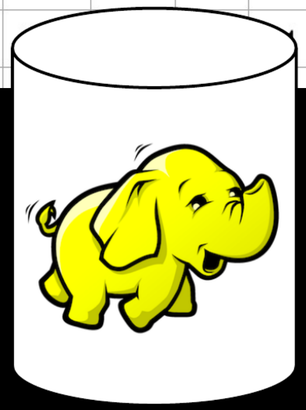
Quick Demo

Thank you Jair Aguirre!!

Quick Demo

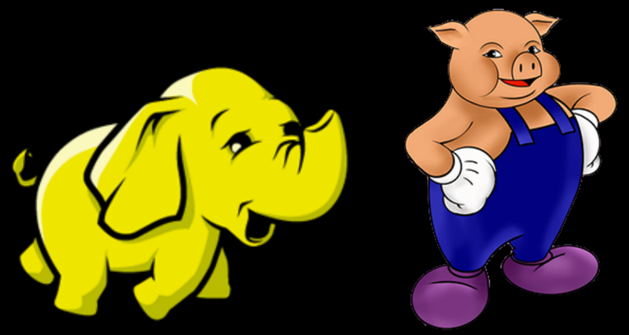
yearID	lgID	teamID	franchID	divID	Rank	G	Ghome	W	L	DivWin	WCWin	LgWin	WSWin	R	AB	H	2B	3B	HR	E
1871	NA	BS1	BNA		3	31		20	10			N		401	1372	426	70	37	3	
1871	NA	CH1	CNA		2	28		19	9			N		302	1196	323	52	21	10	
1871	NA	CL1	CFC		8	29		10	19			N		249	1186	328	35	40	7	
1871	NA	FW1	KEK		7	19		7	12			N		137	746	178	19	8	2	
1871	NA	NY2	NNA		5	33		16	17			N		302					1	

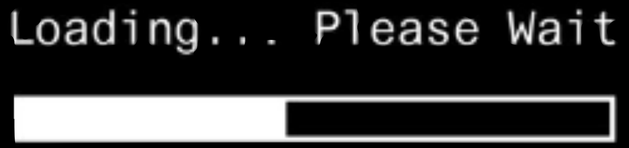
anlahman.com/baseball-archive/statistics



Quick Demo

```
data = load '/user/cloudera/data/baseball_csv/Teams.csv' using PigStorage(',');
filtered = filter data by ($0 == '1988');
tm_hr = foreach filtered generate (chararray) $40 as team, (int) $19 as hrs;
ordered = order tm_hr by hrs desc;
dump ordered;
```





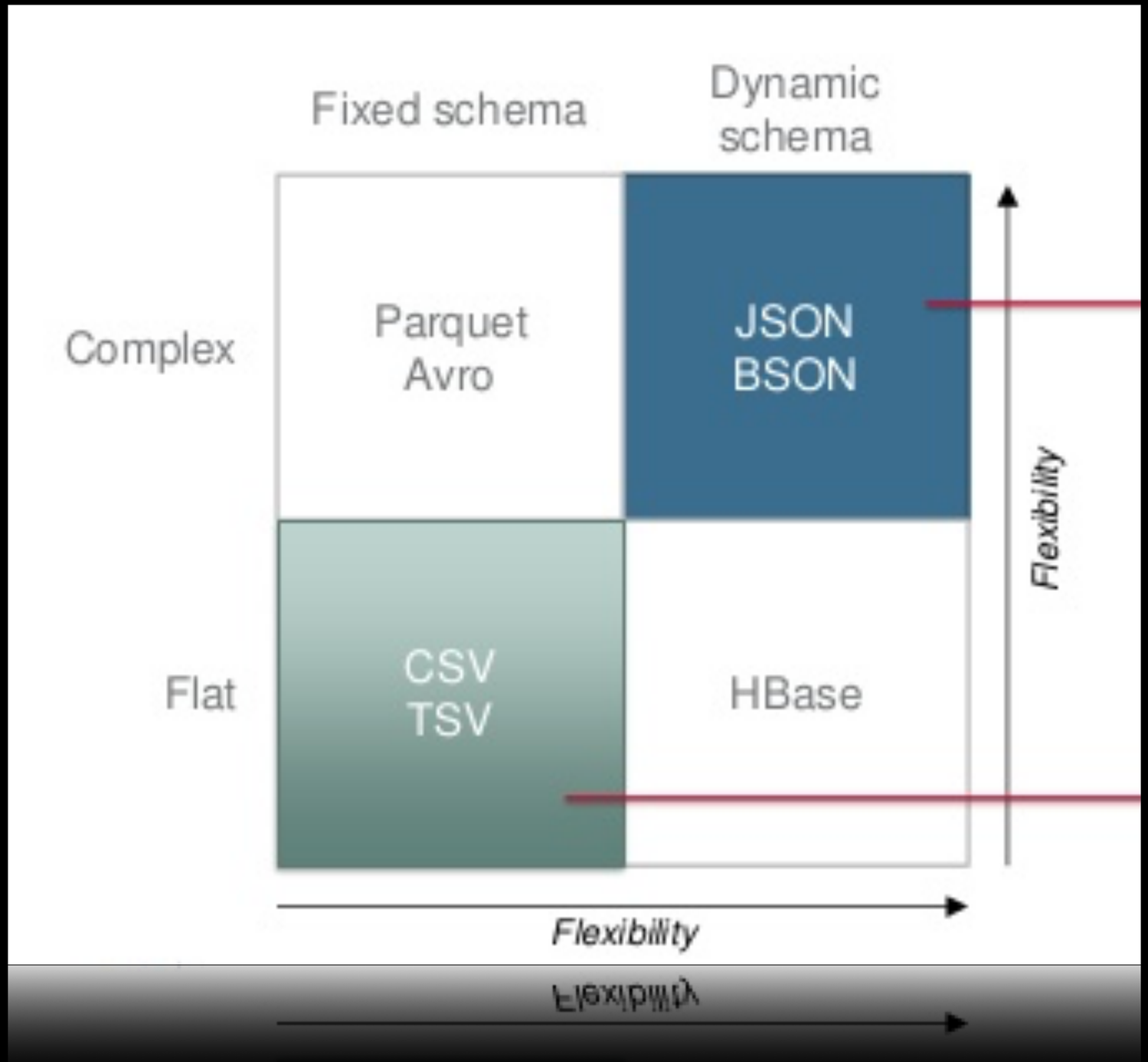
Execution Time:
1 minute, 38 seconds

Quick Demo

```
SELECT columns[40], cast(columns[19] as int) AS HR
FROM `baseball_csv/Teams.csv`
WHERE columns[0] = '1988'
ORDER BY HR desc;
```

Execution Time:
0232 seconds!!

Drill is Versatile


Fixed schema
Dynamic schema
Complex
Flat
Parquet
Avro
JSON
BSON
CSV
TSV
HBase
Flexibility
Flexibility

NoSQL, No Problem

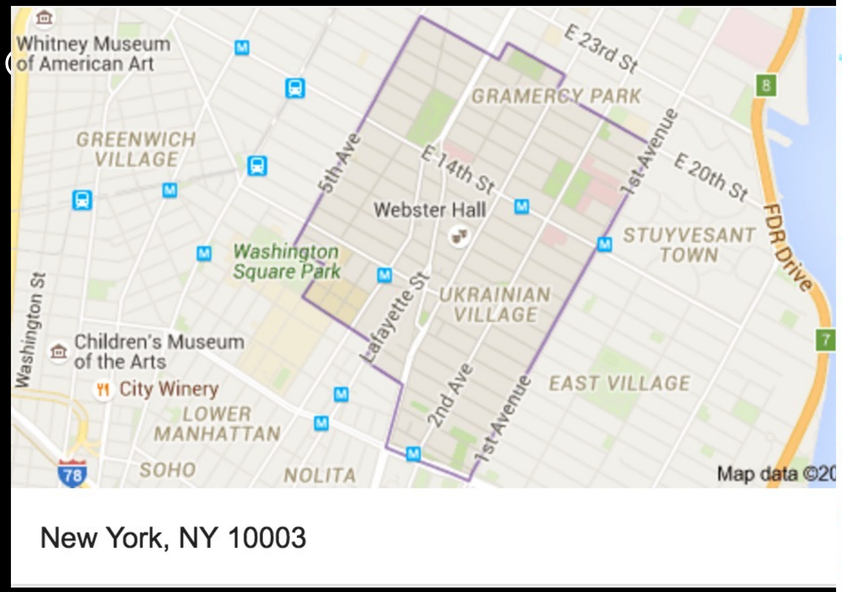
NoSQL, No Problem


```
{
  "address": {
     "building": "1007",
     "coord": [ -73.856077, 40.848447 ],
     "street": "Morris Park Ave",
     "zipcode": "10462"
  },
  "borough": "Bronx",
  "cuisine": "Bakery",
  "grades": [
     { "date": { "$date": 1393804800000 }, "grade": "A", "score": 2 },
     { "date": { "$date": 1378857600000 }, "grade": "A", "score": 6 },
     { "date": { "$date": 1358985600000 }, "grade": "A", "score": 10 },
     { "date": { "$date": 1322006400000 }, "grade": "A", "score": 9 },
     { "date": { "$date": 1299715200000 }, "grade": "B", "score": 14 }
  ],
  "name": "Morris Park Bake Shop",
  "restaurant_id": "30075445"
}
```


https://raw.githubusercontent.com/mongodb/docs-assets/primer-dataset/primer-dataset.json

NoSQL, No Problem

```
SELECT t.address.zipcode AS zip, count(name) AS rests
FROM `restaurants` t
GROUP BY t.address.zipc
ORDER BY rests DESC
LIMIT 10;
```

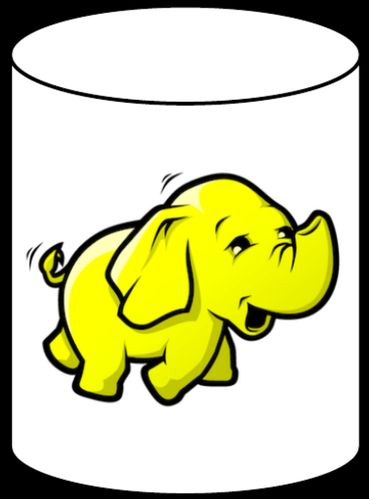


zip	rests
10003	686
10019	675
10036	611
10001	520
10022	485
10013	480
10002	471
10011	467
10016	433
10014	428

https://raw.githubusercontent.com/mongodb/docs-assets/primer-dataset/primer-dataset.json

Querying Across Silos

Querying Across Silos



Farmers Market Data

Restaurant Data

Querying Across Silos

```
SELECT t1.Borough, t1.markets, t2.rests, cast(t1.markets AS
FLOAT)/ cast(t2.rests AS FLOAT) AS ratio
FROM (
 SELECT Borough, count(`Farmers Markets Name`) AS markets
 FROM `farmers_markets.csv`
 GROUP BY Borough ) t1
 JOIN (
   SELECT borough, count(name) AS rests
   FROM mongo.test.`restaurants`
   GROUP BY borough
   ) t2
ON t1.Borough=t2.borough
ORDER BY ratio DESC;
```

Querying Across Silos

Borough	markets	rests	ratio
Bronx	18	2338	0.007698888
Brooklyn	34	6086	0.005586592
Manhattan	36	10259	0.003509114
Queens	12	5656	0.0021216408
Staten Island	1	969	0.0010319918

Execution Time: 0.502 Seconds

Why **aren’t** you using Drill?

Installing & Configuring Drill

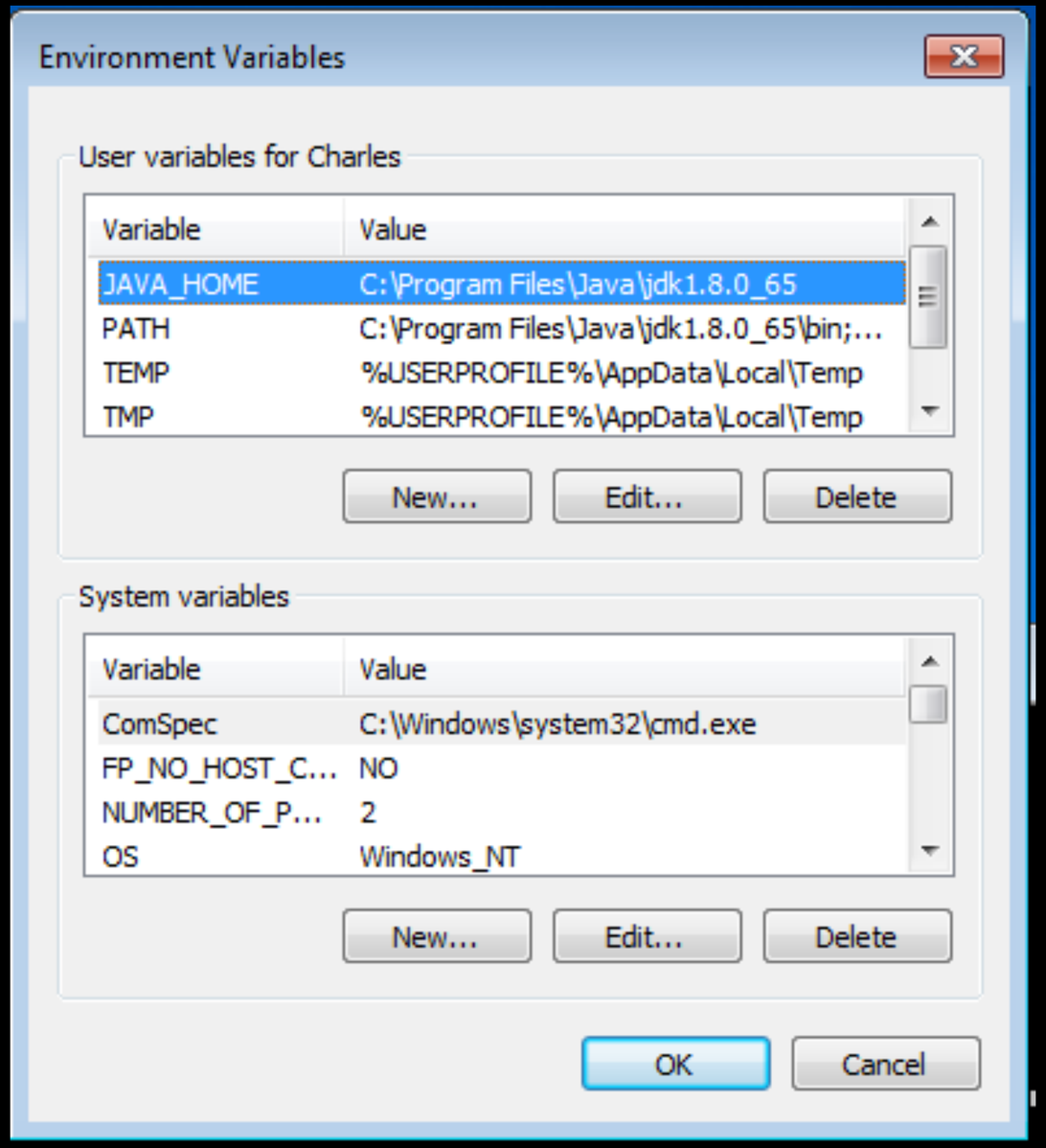
Embedded

Distributed

Step 1: Download Drill:
drill.apache.org/download/

Drill Requirements

- Oracle Java SE Development Kit (JDK 7) or higher. (Verify this by opening a command prompt and typing `java -version`)
- On Windows machines, you will need to set the JAVA_HOME and PATH variables.


Environment Variables
User variables for Charles
Variable
Value
JAVA_HOME
C:\Program Files\Java\jdk1.8.0_65
PATH
C:\Program Files\Java\jdk1.8.0_65\bin;...
TEMP
%USERPROFILE%\AppData\Local\Temp
TMP
%USERPROFILE%\AppData\Local\Temp
New...
Edit...
Delete
System variables
Variable
Value
ComSpec
C:\Windows\system32\cmd.exe
FP_NO_HOST_C...
NO
NUMBER_OF_P...
2
OS
Windows_NT
New...
Edit...
Delete
OK
Cancel

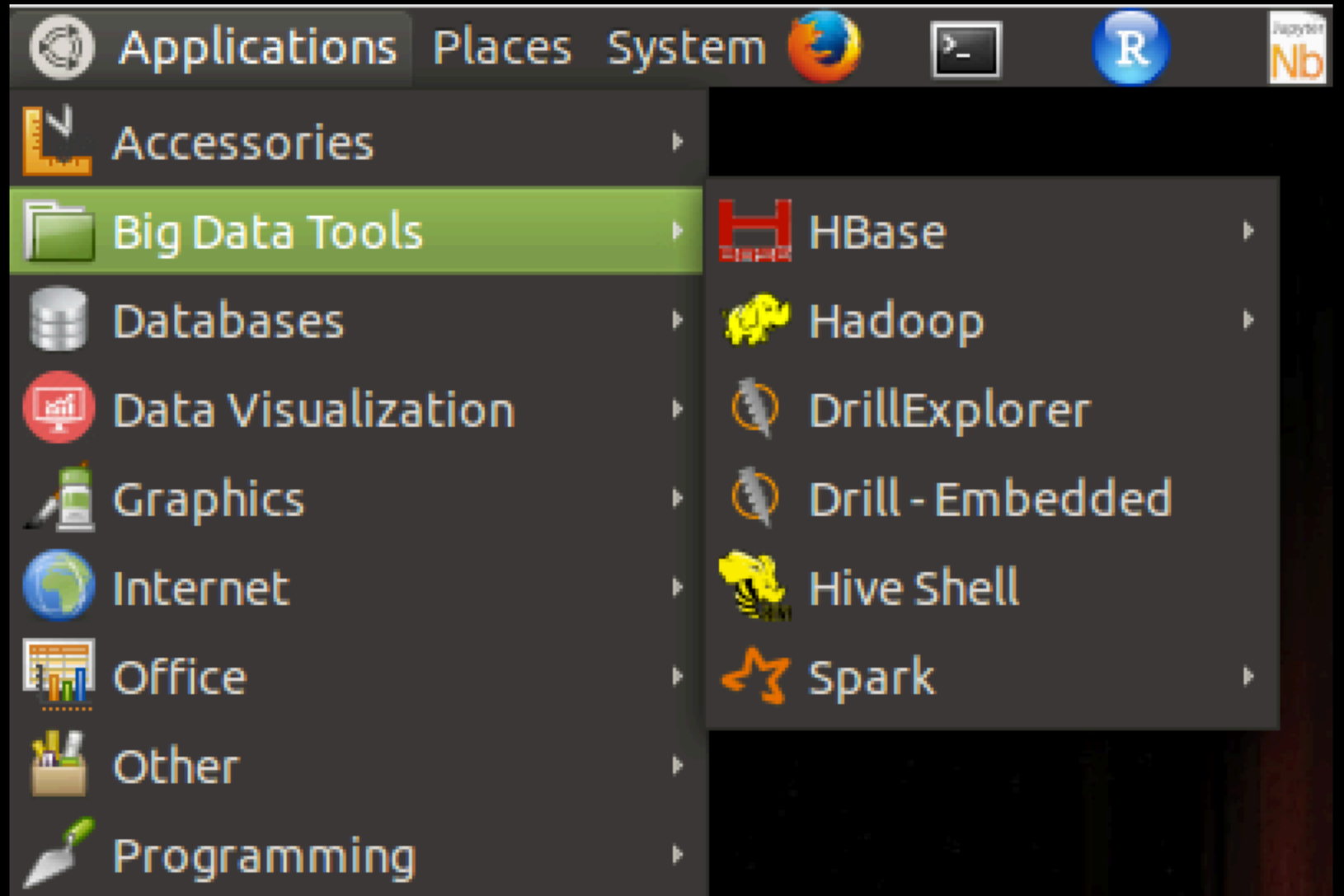
Starting Drill

Embedded Mode: For use on a standalone system

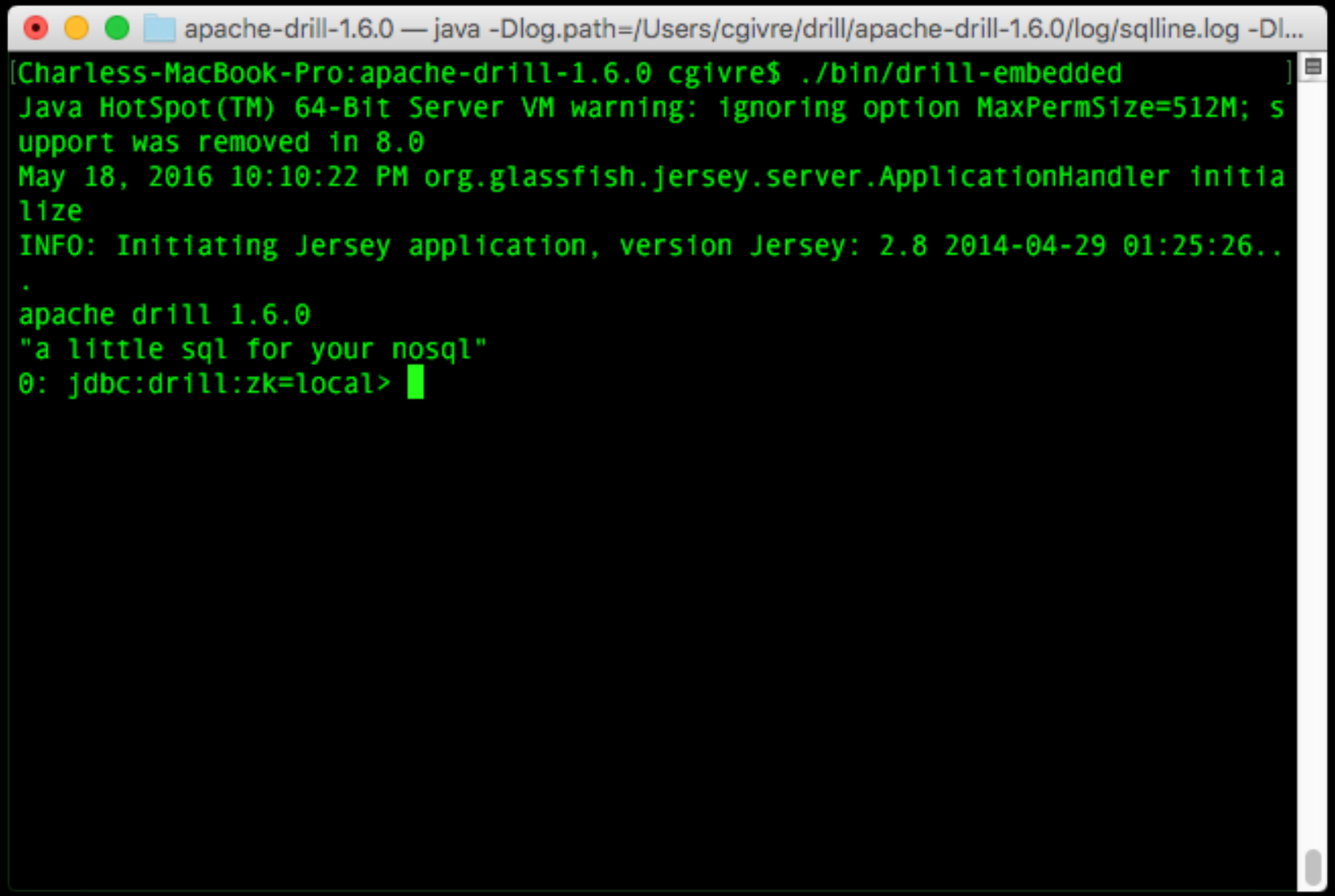
```
$./bin/drill-embedded
```


```
sqlline.bat -u "jdbc:drill:zk=local"
```

Starting Drill

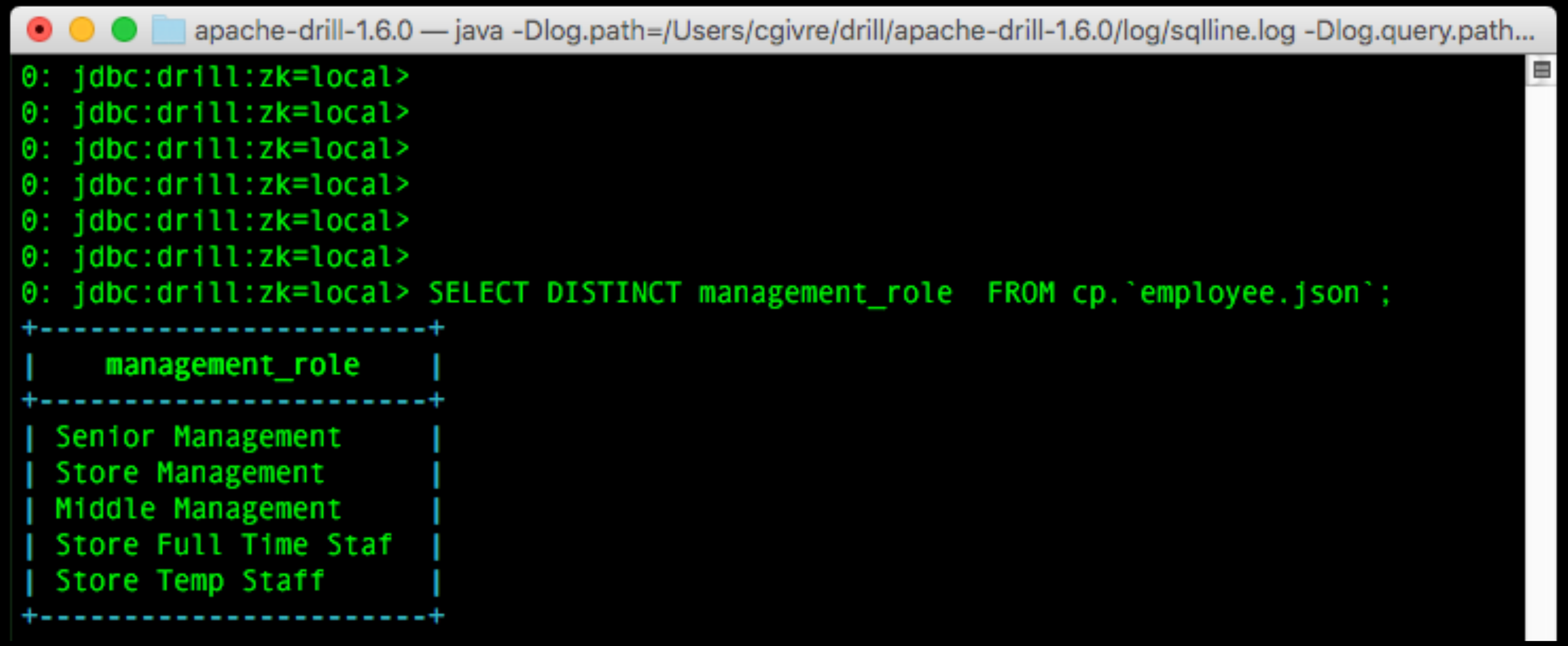


Drill's Command Line Interface



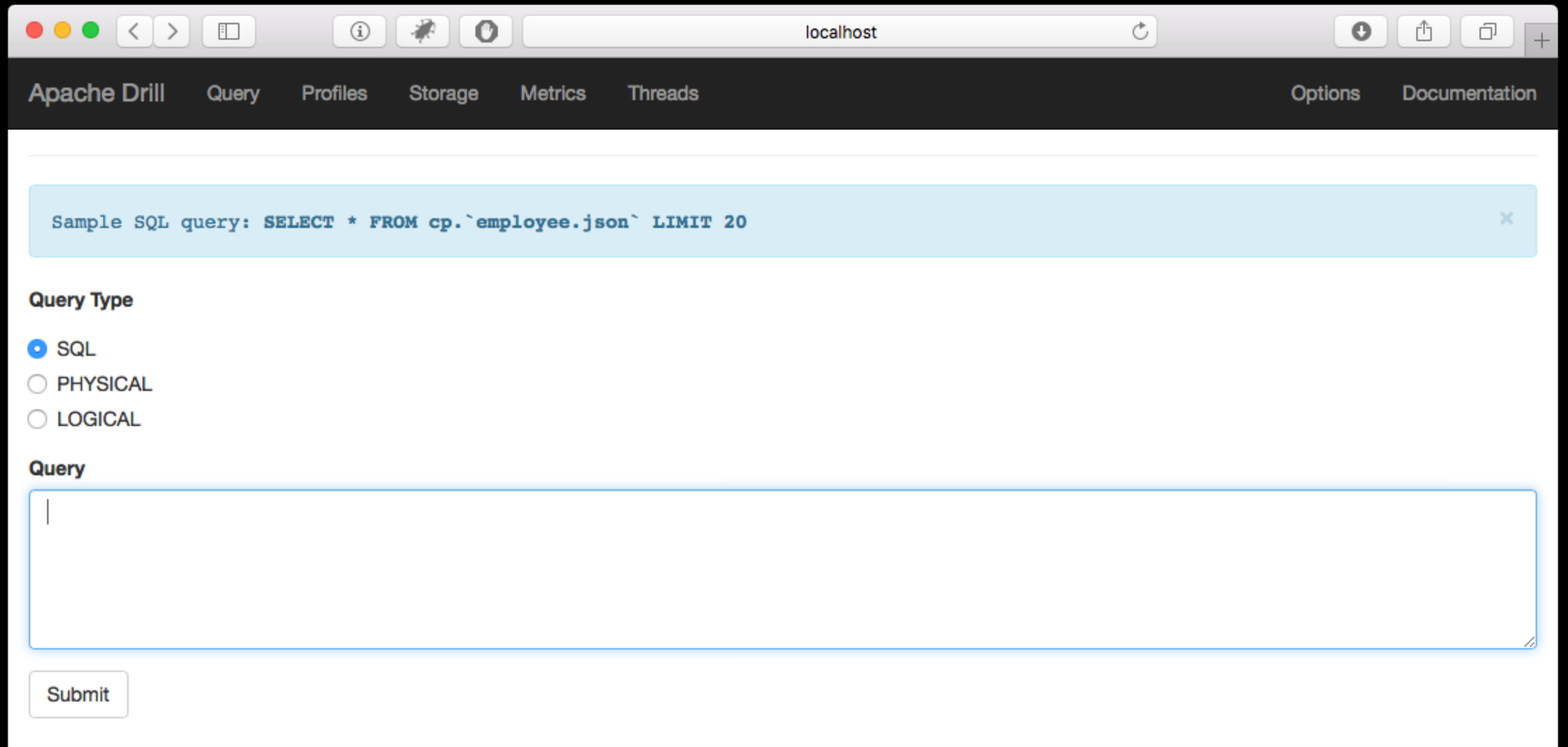
Drill's Command Line Interface

```
SELECT DISTINCT management_role FROM cp.`employee.json`;
```



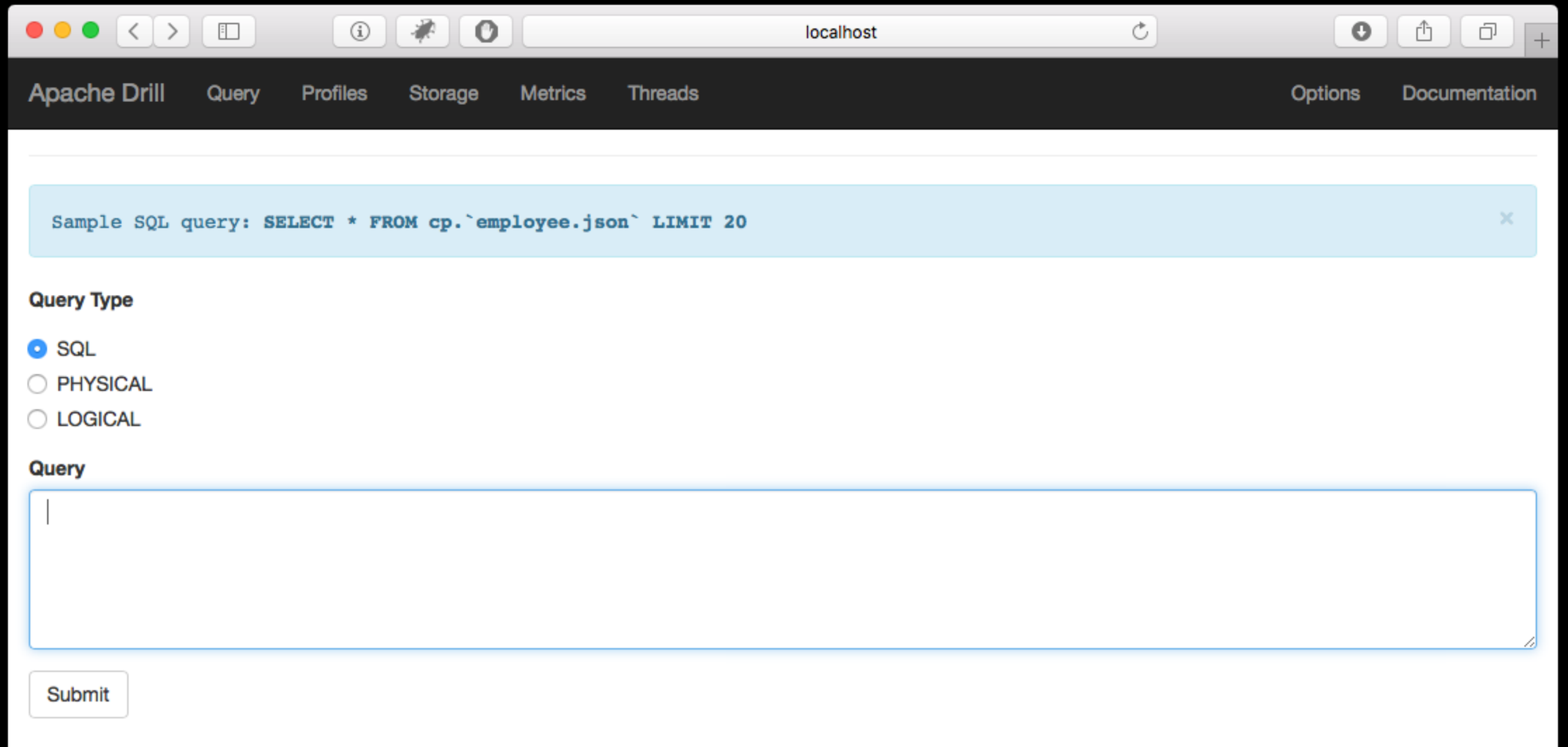
Drill Web UI

http://localhost:8047



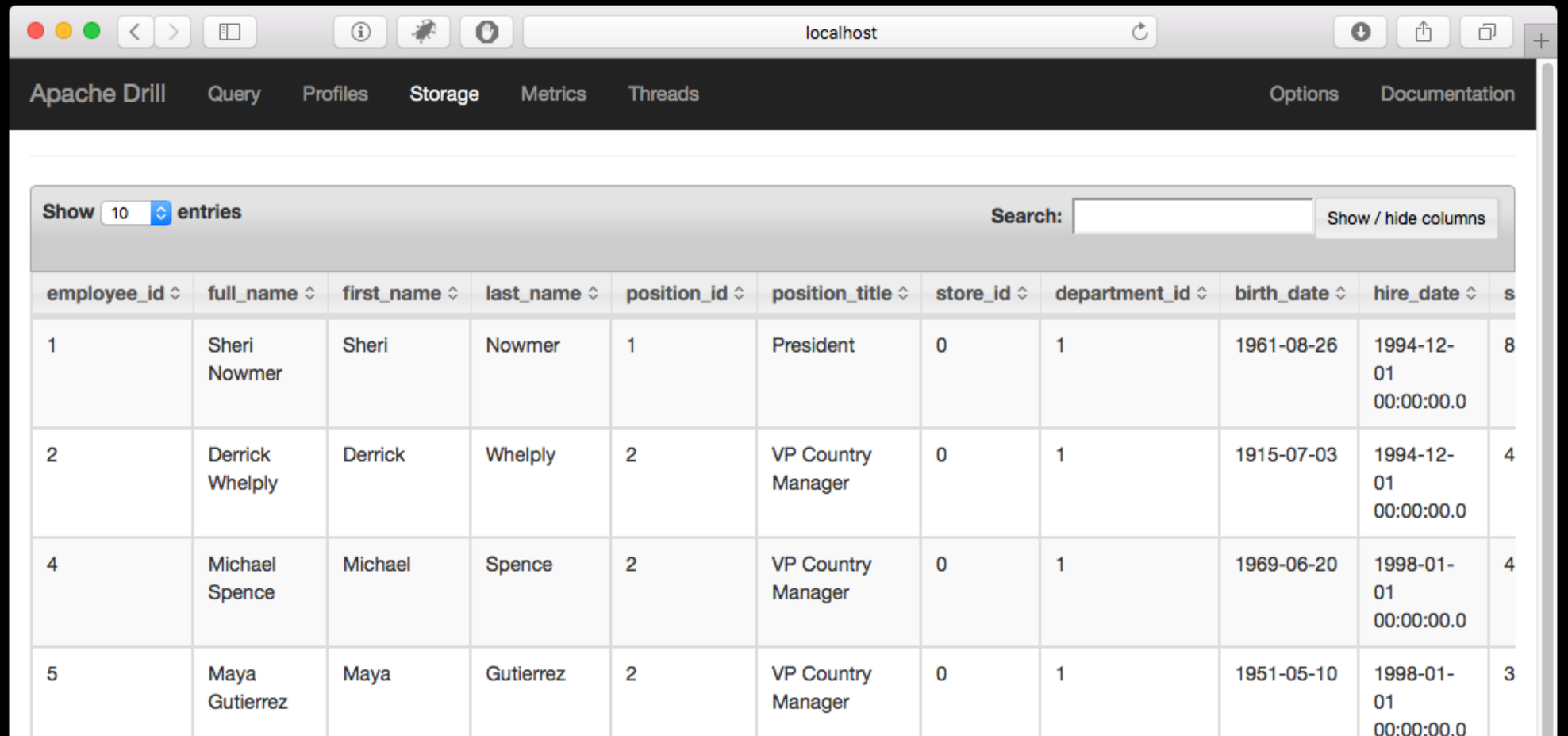
Drill Web UI

```
SELECT * FROM cp.`employee.json` LIMIT 20
```



Drill Web UI

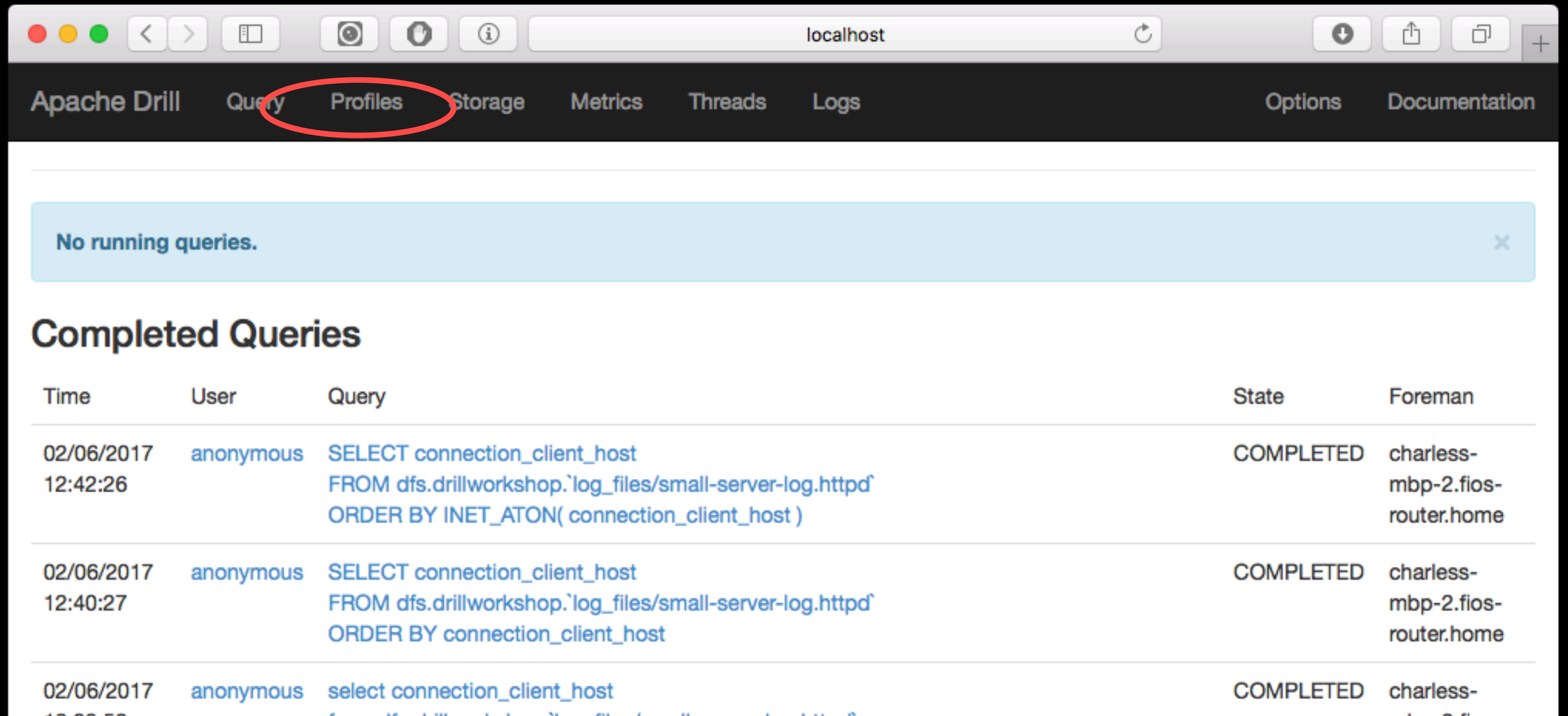
```
SELECT * FROM cp.`employee.json` LIMIT 20
```



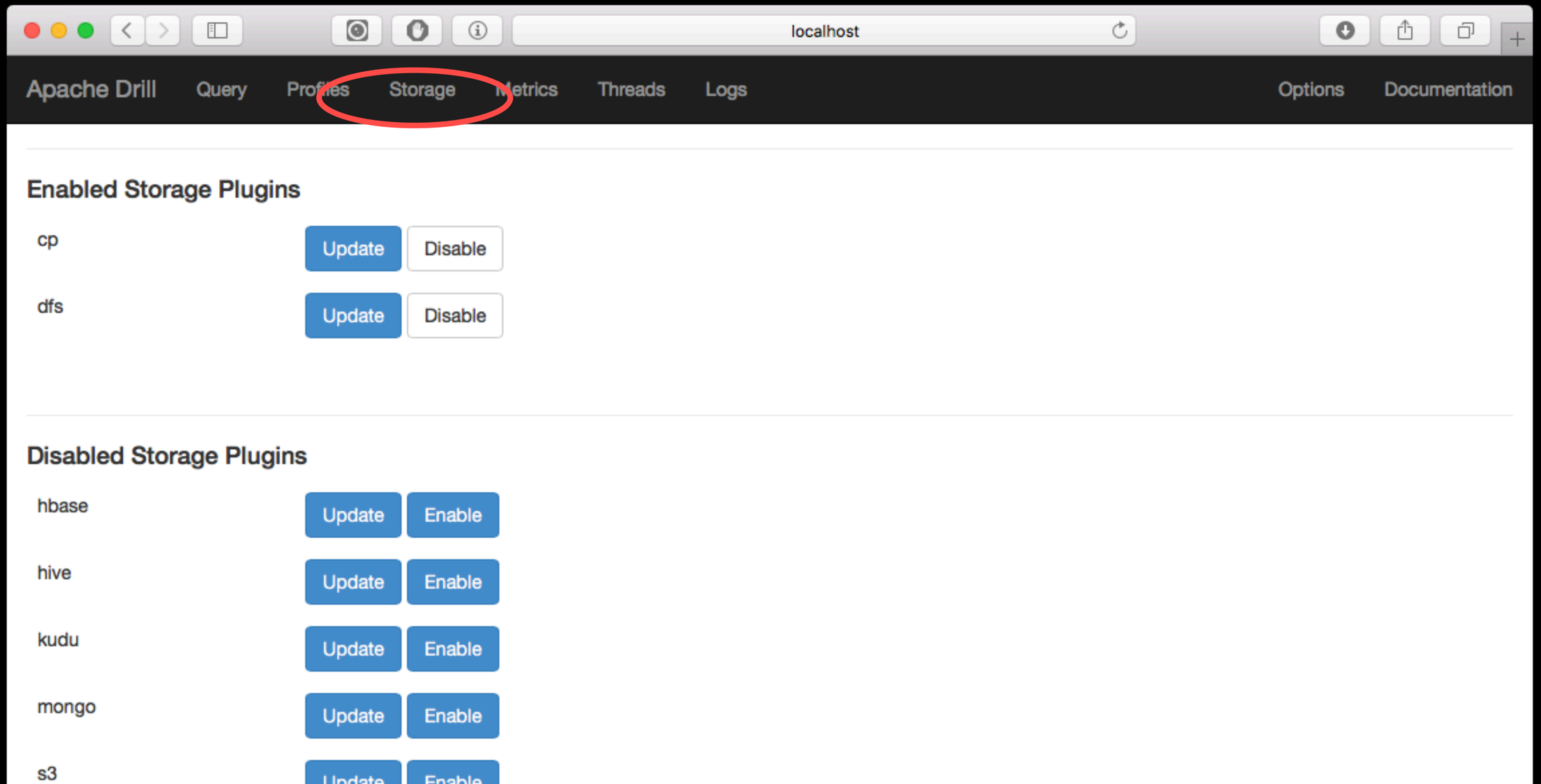
Drill isn’t case sensitive*

* Except when Drill is case sensitive

Drill Web UI


localhost
Apache Drill
Query
Profiles
Storage
Metrics
Threads
Logs
Options
Documentation
No running queries.
Completed Queries
Time
User
Query
State
Foreman
02/06/2017 12:42:26
anonymous
SELECT connection_client_host FROM dfs.drillworkshop.`log_files/small-server-log.httpd` ORDER BY INET_ATON(connection_client_host)
COMPLETED
charless-mbp-2.fios-router.home
02/06/2017 12:40:27
anonymous
SELECT connection_client_host FROM dfs.drillworkshop.`log_files/small-server-log.httpd` ORDER BY connection_client_host
COMPLETED
charless-mbp-2.fios-router.home
02/06/2017
anonymous
select connection_client_host
COMPLETED
charless-

Drill Web UI


localhost
Apache Drill
Query
Profiles
Storage
Metrics
Threads
Logs
Options
Documentation
Enabled Storage Plugins
cp
Update
Disable
dfs
Update
Disable
Disabled Storage Plugins
hbase
Update
Enable
hive
Update
Enable
kudu
Update
Enable
mongo
Update
Enable
s3
Update
Enable

Drill Web UI

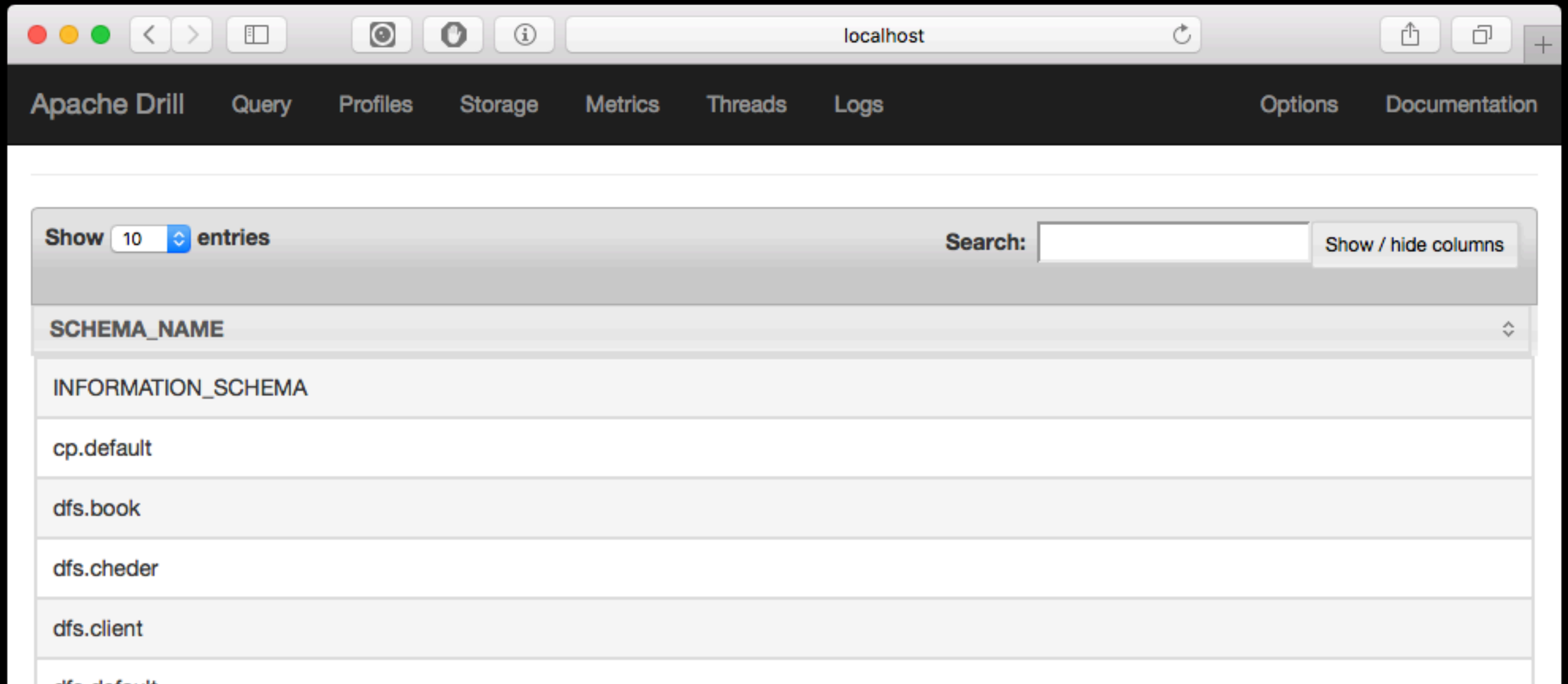
```
{
  "type": "file",
  "enabled": true,
  "connection": "file:///",
  "config": null,
  "workspaces": "workspaces": {
    "root": {
      "location": "/",
      "writable": false,
      "defaultInputFormat": null
    }…
  },
  "formats": {
    "csv": {
      "type": "text",
      "extensions": [
        "csv"
      ]      ]
    }
…
  }
```

Workspaces in Drill

- Workspaces are shortcuts to the file system. You'll want to use them when you have lengthy file paths.
- They work in any "file based" storage plugin (IE: S3, Hadoop, Local File System)

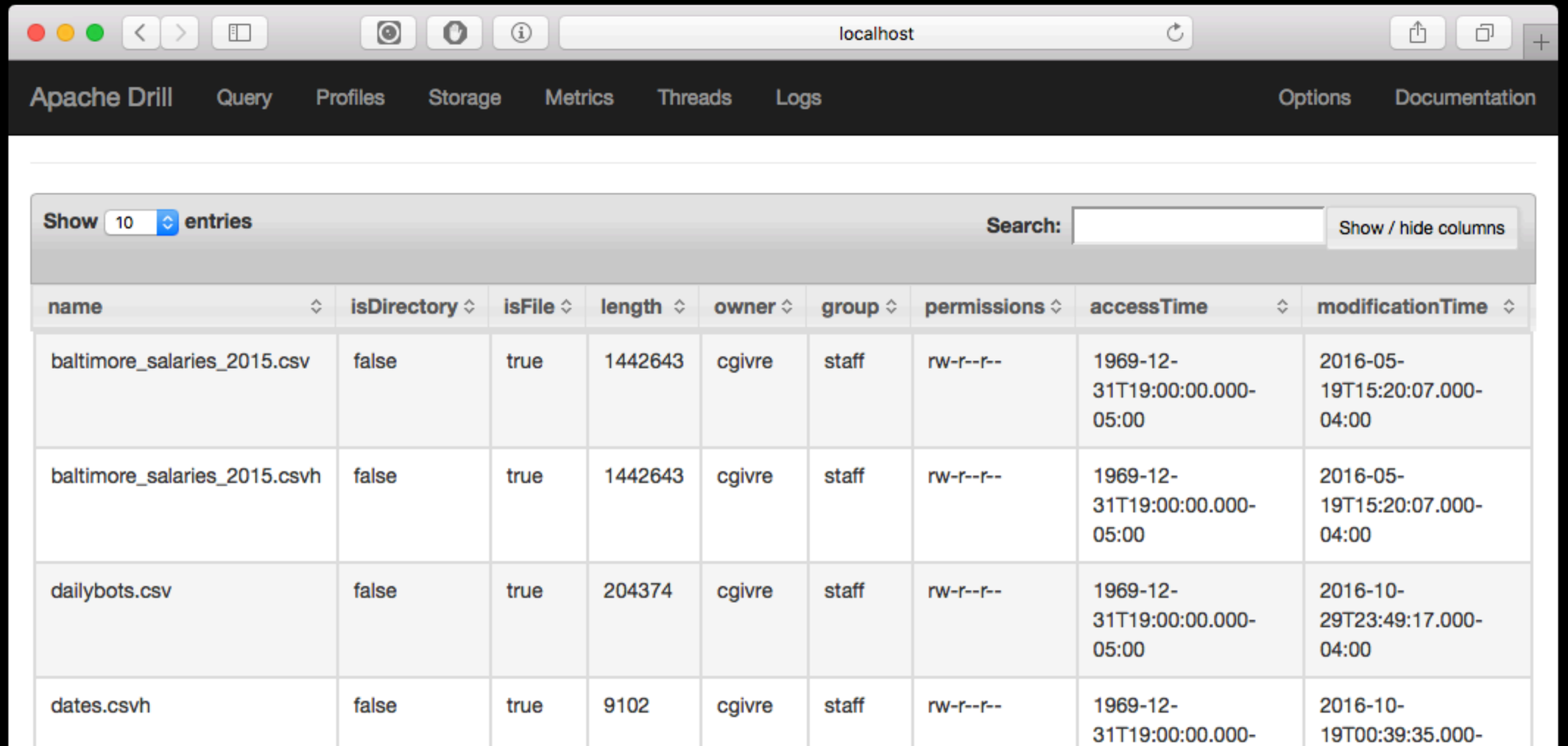
Drill Web UI

SHOW DATABASES



Drill Web UI

SHOW FILES IN <workspace>



name	isDirectory	isFile	length	owner	group	permissions	accessTime	modificationTime
baltimore_salaries_2015.csv	false	true	1442643	cgivre	staff	rw-r--r--	1969-12-31T19:00:00.000-05:00	2016-05-19T15:20:07.000-04:00
baltimore_salaries_2015.csvh	false	true	1442643	cgivre	staff	rw-r--r--	1969-12-31T19:00:00.000-05:00	2016-05-19T15:20:07.000-04:00
dailybots.csv	false	true	204374	cgivre	staff	rw-r--r--	1969-12-31T19:00:00.000-05:00	2016-10-29T23:49:17.000-04:00
dates.csvh	false	true	9102	cgivre	staff	rw-r--r--	1969-12-31T19:00:00.000-	2016-10-19T00:39:35.000-

Workspaces in Drill

```
SELECT field1, field2
FROM dfs.`/Users/cgivre/github/projects/drillclass/file1.csv`
```

or

```
SELECT field1, field2
FROM dfs.drilldata.`file1.csv`
```

In Class Exercise:
Create a Workspace

In this exercise we are going to create a workspace called '`drillclass`', which we will use for future exercises.

1. First, download all the files from https://github.com/cgivre/data-exploration-with-apache-drill and put them in a folder of your choice on your computer. **Remember the complete file path.**

2. Open the Drill Web UI and go to Storage->dfs->update

3. Paste the following into the 'workspaces' section and click update

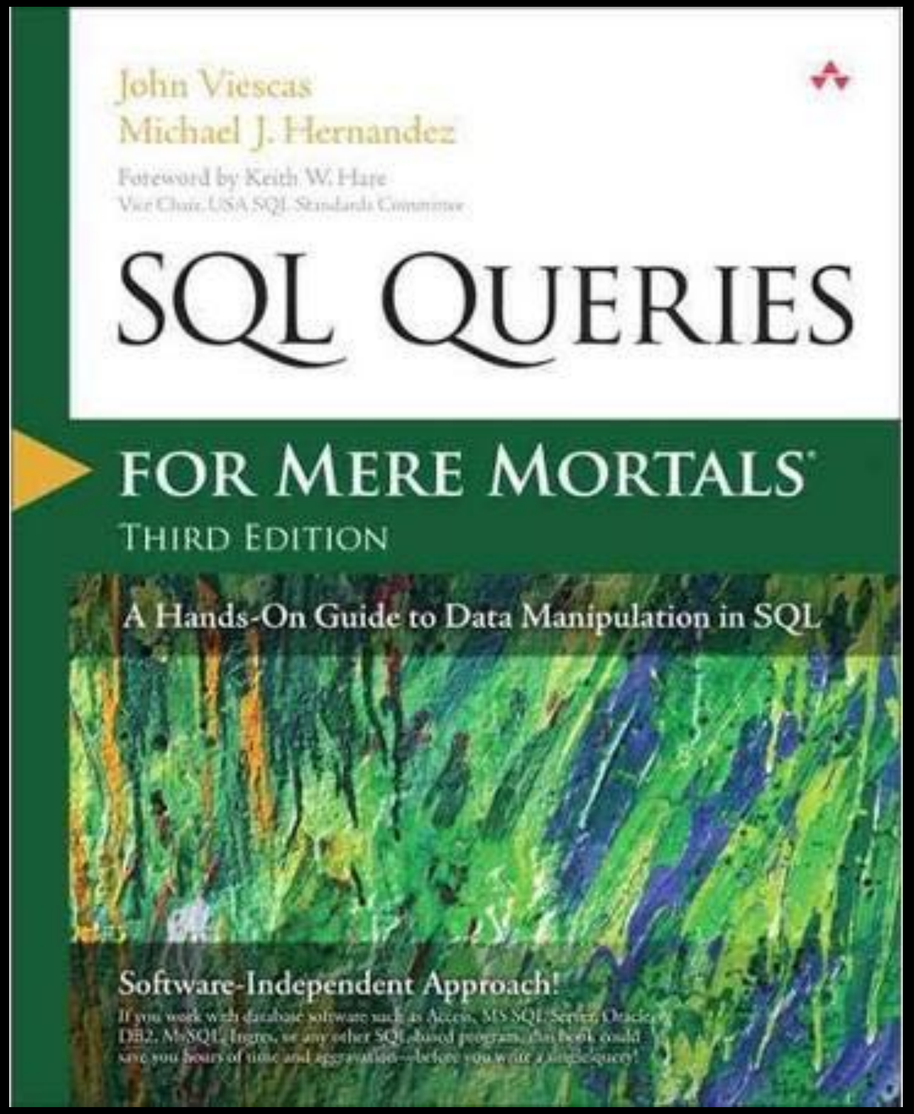
```
"drillclass": {
  "location": "<path to your files>",
  "writable": true,
  "defaultInputFormat": null
}
```

4. Execute a `show databases` query to verify that your workspace was added.

Querying Simple Delimited Data

Everything you need to know about SQL*… in 10 minutes

* well…not quite everything, but enough to get you through this session



http://amzn.to/2IlD8yi

```
SELECT <fields>
FROM <data source>
```

Please open people.csv

	A	B	C	D	E	F
1	id	first_name	last_name	email	gender	ip_address
2	1	Philip	Richardson	prichardson0@samsung.com	Male	76.20.6.56
3	2	Todd	James	tjames1@hostgator.com	Male	0.118.22.98
4	3	Jimmy	Mendoza	jmendoza2@reuters.com	Male	118.191.95.200
5	4	Jose	Morris	jmorris3@example.com	Male	106.180.20.171
6	5	Dorothy	Fernandez	dfernandez4@ask.com	Female	171.75.36.3
7	6	Patrick	Bradley	pbradley5@elpais.com	Male	20.244.138.119
8	7	Nicholas	Bishop	nbishop6@fotki.com	Male	80.54.11.247
9	8	Michael	Kelly	mkelly7@imageshack.us	Male	191.190.150.242
10	9	Russell	Coleman	rcoleman8@pbs.org	Male	50.176.191.152
11	10	Frances	Rodriguez	frodriguez9@github.io	Female	23.118.13.92
12	11	Nancy	Nelson	nnelsona@biglobe.ne.jp	Female	225.182.18.156
13	12	Theresa	Russell	trussellb@hexun.com	Female	16.14.226.63
14	13	Frances	Greene	fgreenec@sbwire.com	Female	192.88.64.253
15	14	Julia	Alvarez	jalvarezd@livejournal.com	Female	197.237.43.239

Giving a shout out to
https://www.mockaroo.com
which I used to generate most of my data for this class.

```
SELECT *
FROM <data source>
```

```
SELECT first_name,
last_name,
gender
FROM <data source>
```

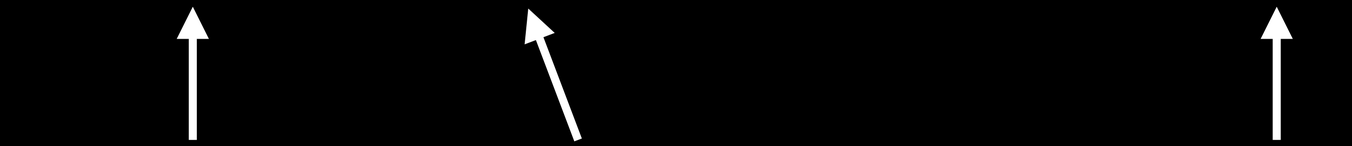
Tip: Use BACK TICKS around field names in Drill

```
SELECT `first_name`,
`last_name`,
`gender`
FROM <data source>
```

```
SELECT `first_name`,
`last_name`,
`gender`
FROM <data source>
```

Querying Drill

```
FROM dfs.logs.`/data/customers.csv`
```



Querying Drill

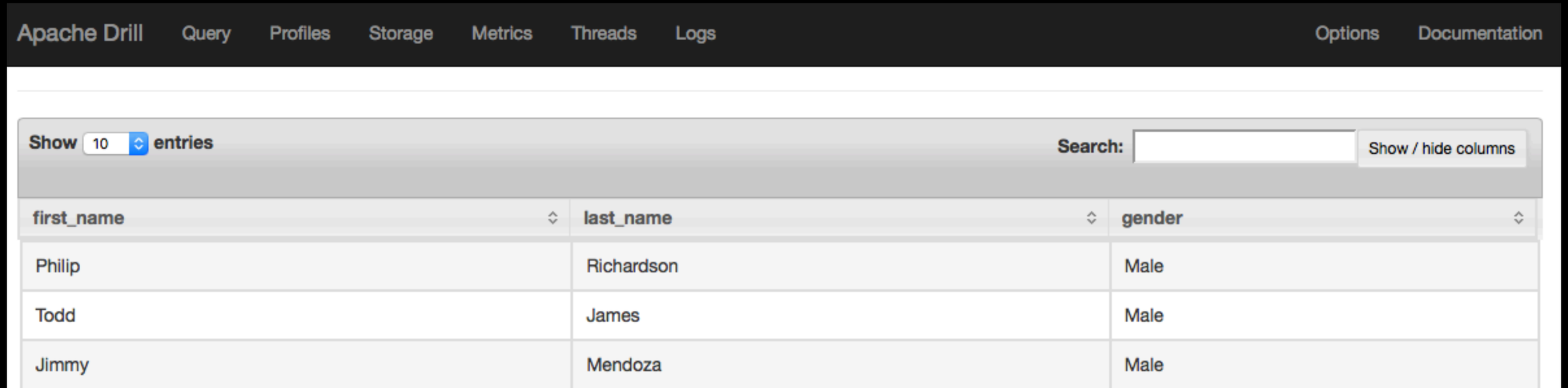
```
FROM dfs.logs.`/data/customers.csv`
```


```
FROM dfs.`/var/www/mystore/sales/data/customers.csv`
```

```
SELECT `first_name`,
`last_name`,
`gender`
FROM dfs.drillclass.`people.csvh`
```

Try it yourself!!

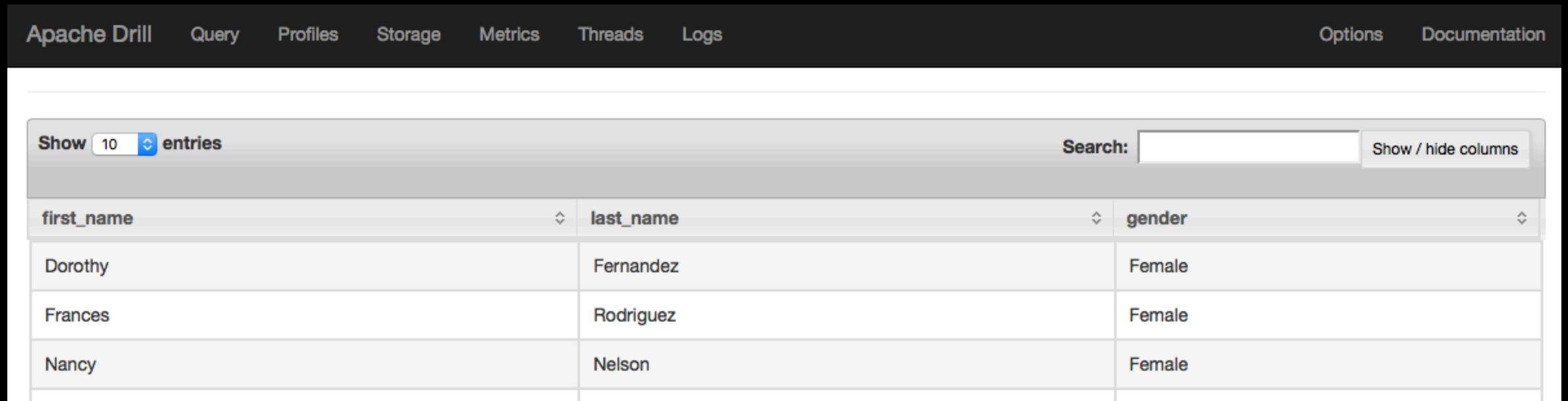
```
SELECT `first_name`,
`last_name`,
`gender`
FROM dfs.drillclass.`people.csvh`
```



```
SELECT <fields>
FROM <data source>
WHERE <logical condition>
```

```
SELECT `first_name`,
`last_name`,
`gender`
FROM dfs.drillclass.`people.csvh`
WHERE `gender` = 'Female'
```

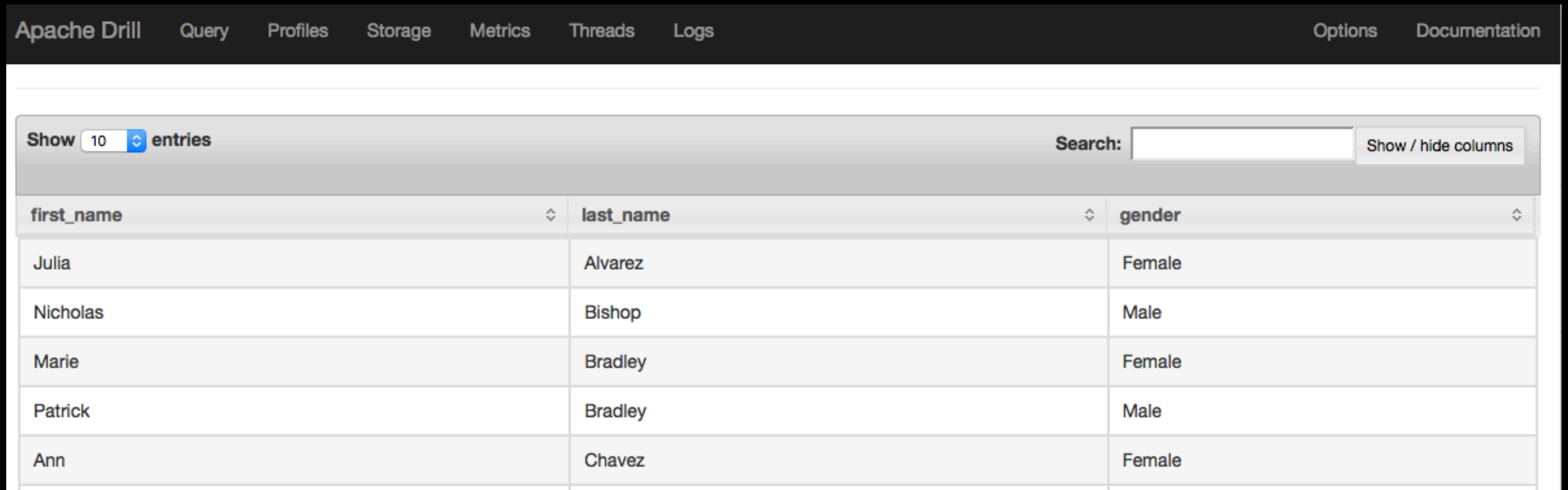
```
SELECT `first_name`,
`last_name`,
`gender`
FROM dfs.drillclass.`people.csvh`
WHERE `gender` = 'Female'
```



```
SELECT <fields>
FROM <data source>
WHERE <logical condition>
ORDER BY <field> (ASC|DESC)
```

```
SELECT `first_name`,
`last_name`,
`gender`
FROM dfs.drillclass.`people.csvh`
ORDER BY `last_name`, `first_name` ASC
```

```
SELECT `first_name`,
`last_name`,
`gender`
FROM dfs.drillclass.`people.csvh`
ORDER BY `last_name`, `first_name` ASC
```

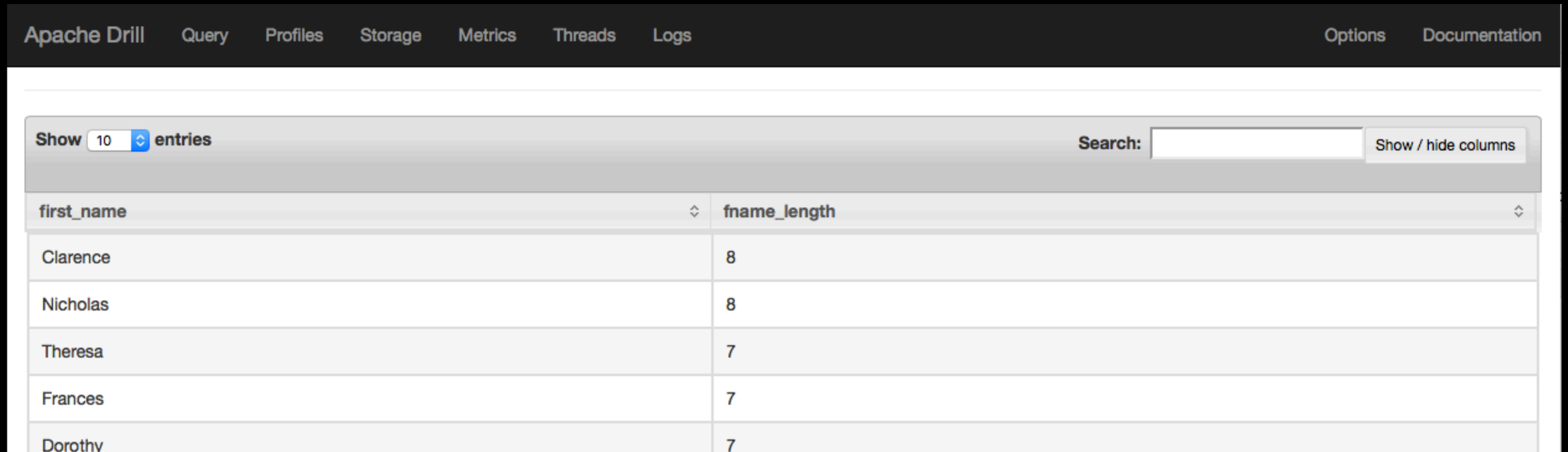


first_name	last_name	gender
Julia	Alvarez	Female
Nicholas	Bishop	Male
Marie	Bradley	Female
Patrick	Bradley	Male
Ann	Chavez	Female

```
SELECT
FUNCTION( <field> ) AS new_field
FROM <data source>
```

```
SELECT first_name,
LENGTH( `first_name` ) AS
fname_length
FROM dfs.drillclass.`people.csvh`
ORDER BY fname_length DESC
```

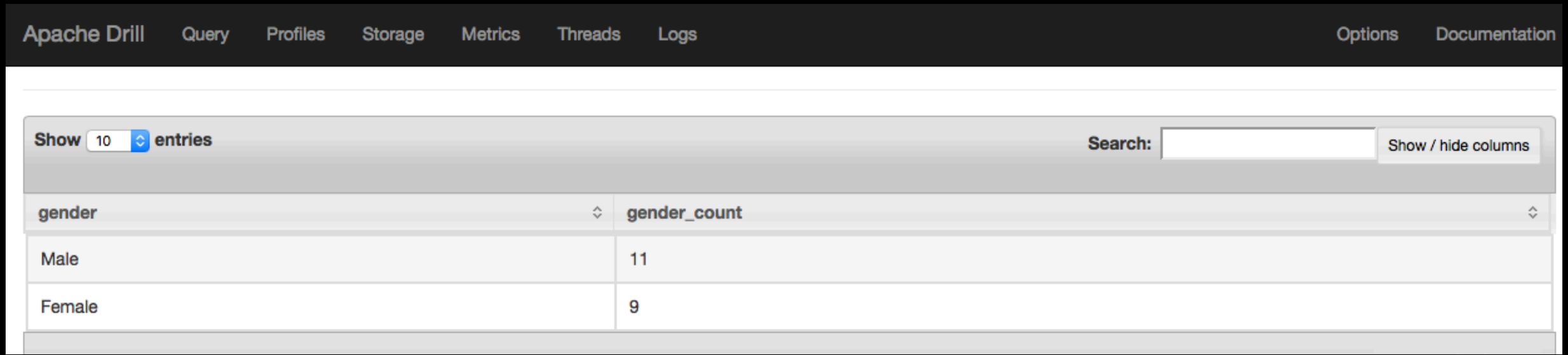
```
SELECT first_name,
LENGTH( `first_name` ) AS
fname_length
FROM dfs.drillclass.`people.csvh`
ORDER BY fname_length DESC
```



```
SELECT <fields>
FROM <data source>
GROUP BY <field>
```

```
SELECT `gender`,
COUNT( * ) AS gender_count
FROM dfs.drillclass.`people.csvh`
GROUP BY `gender`
```

```
SELECT `gender`,
COUNT( * ) AS gender_count
FROM dfs.drillclass.`people.csvh`
GROUP BY `gender`
```



Joining Datasets

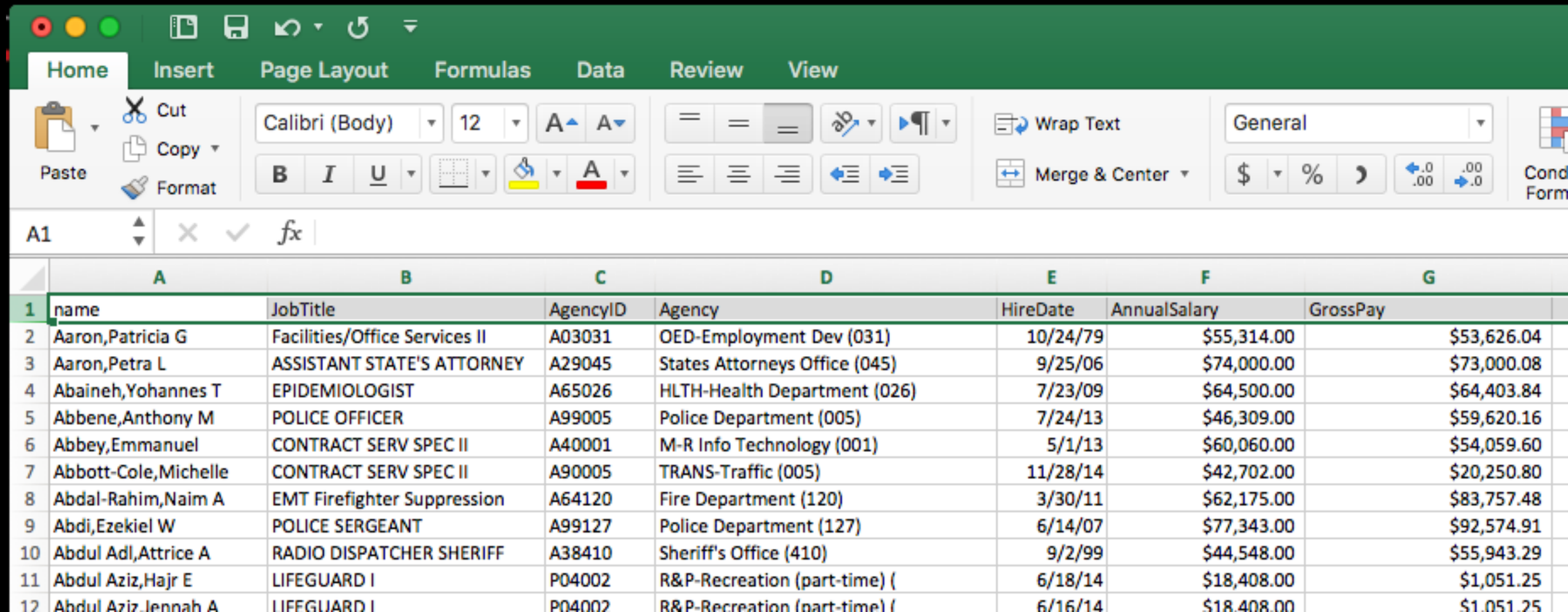

Referred to as an Inner Join
Data Set A
Data Set B

```
SELECT <fields>
FROM <data source 1> AS table1
INNER JOIN <data source 2> AS table2
ON table1.`id` = table2.`id`
```

Questions?

Querying Drill

Please take a look at baltimore_salaries_2016.csv.
This data is available at: http://bit.ly/balt-sal



	A	B	C	D	E	F	G
1	name	JobTitle	AgencyID	Agency	HireDate	AnnualSalary	GrossPay
2	Aaron,Patricia G	Facilities/Office Services II	A03031	OED-Employment Dev (031)	10/24/79	$55,314.00	$53,626.04
3	Aaron,Petra L	ASSISTANT STATE'S ATTORNEY	A29045	States Attorneys Office (045)	9/25/06	$74,000.00	$73,000.08
4	Abaineh,Yohannes T	EPIDEMIOLOGIST	A65026	HLTH-Health Department (026)	7/23/09	$64,500.00	$64,403.84
5	Abbene,Anthony M	POLICE OFFICER	A99005	Police Department (005)	7/24/13	$46,309.00	$59,620.16
6	Abbey,Emmanuel	CONTRACT SERV SPEC II	A40001	M-R Info Technology (001)	5/1/13	$60,060.00	$54,059.60
7	Abbott-Cole,Michelle	CONTRACT SERV SPEC II	A90005	TRANS-Traffic (005)	11/28/14	$42,702.00	$20,250.80
8	Abdal-Rahim,Naim A	EMT Firefighter Suppression	A64120	Fire Department (120)	3/30/11	$62,175.00	$83,757.48
9	Abdi,Ezekiel W	POLICE SERGEANT	A99127	Police Department (127)	6/14/07	$77,343.00	$92,574.91
10	Abdul Adl,Attrice A	RADIO DISPATCHER SHERIFF	A38410	Sheriff's Office (410)	9/2/99	$44,548.00	$55,943.29
11	Abdul Aziz,Hajr E	LIFEGUARD I	P04002	R&P-Recreation (part-time) (	6/18/14	$18,408.00	$1,051.25
12	Abdul Aziz,Jennah A	LIFEGUARD I	P04002	R&P-Recreation (part-time) (	6/16/14	$18,408.00	$1,051.25

In Class Exercise:
Create a Simple Report

For this exercise we will use the baltimore_salaries_2016.csvh file.

1. Create a query which returns each person's: name, jobtitle, and gross pay.

2. Create a report which contains each employee's name, job title, 2015 salary and 2016 salary. NOTE: This query requires the use of a JOIN.

```
SELECT EmpName, JobTitle, GrossPay
FROM dfs.drillclass.`baltimore_salaries_2016.csvh`
LIMIT 10
```

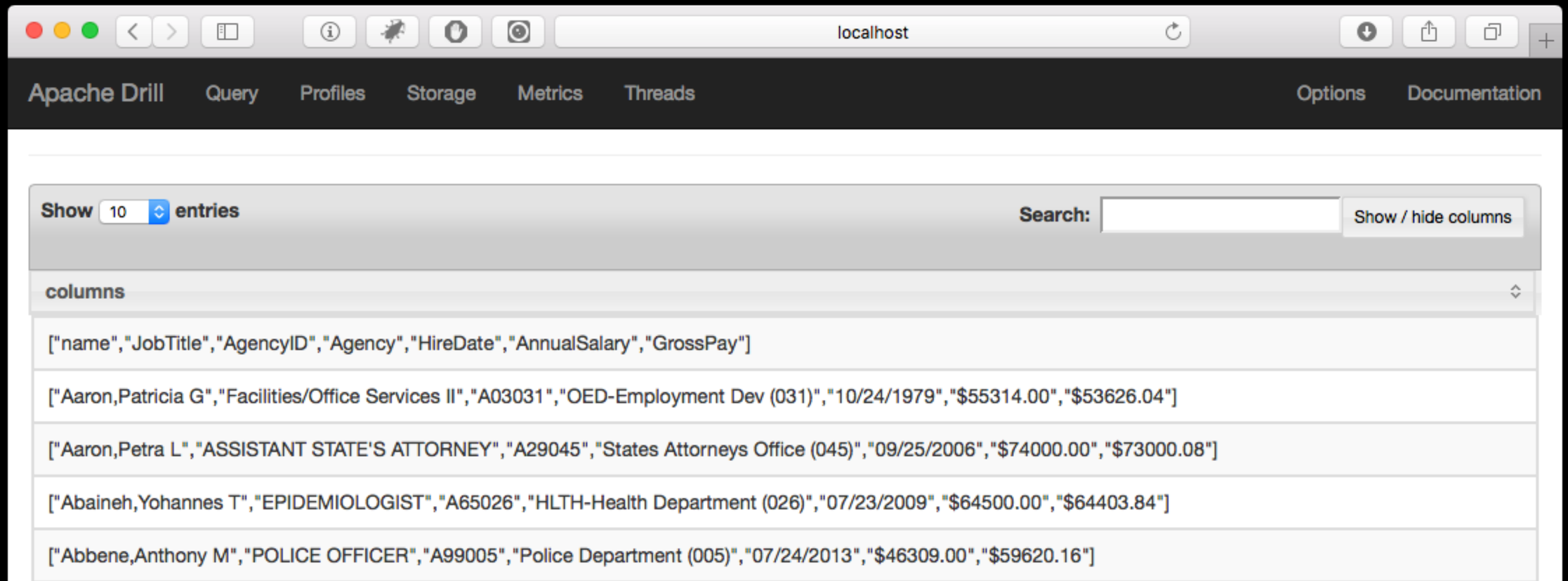
```
SELECT data2016.`EmpName`,
data2016.`JobTitle`,
data2016.`AnnualSalary` AS salary_2016,
data2015.`AnnualSalary` AS salary_2015
FROM dfs.drillclass.`baltimore_salaries_2016.csvh` AS data2016
INNER JOIN dfs.drillclass.`baltimore_salaries_2015.csvh` AS data2015
ON data2016.`EmpName` = data2015.`EmpName`
```

Querying Drill

```
SELECT *
FROM dfs.drillclass.`csv/baltimore_salaries_2016.csv`
LIMIT 10
```

Drill Data Types

```
SELECT *
FROM dfs.drillclass.`baltimore_salaries_2016.csv`
LIMIT 10
```



Drill Data Types

Simple Data Types

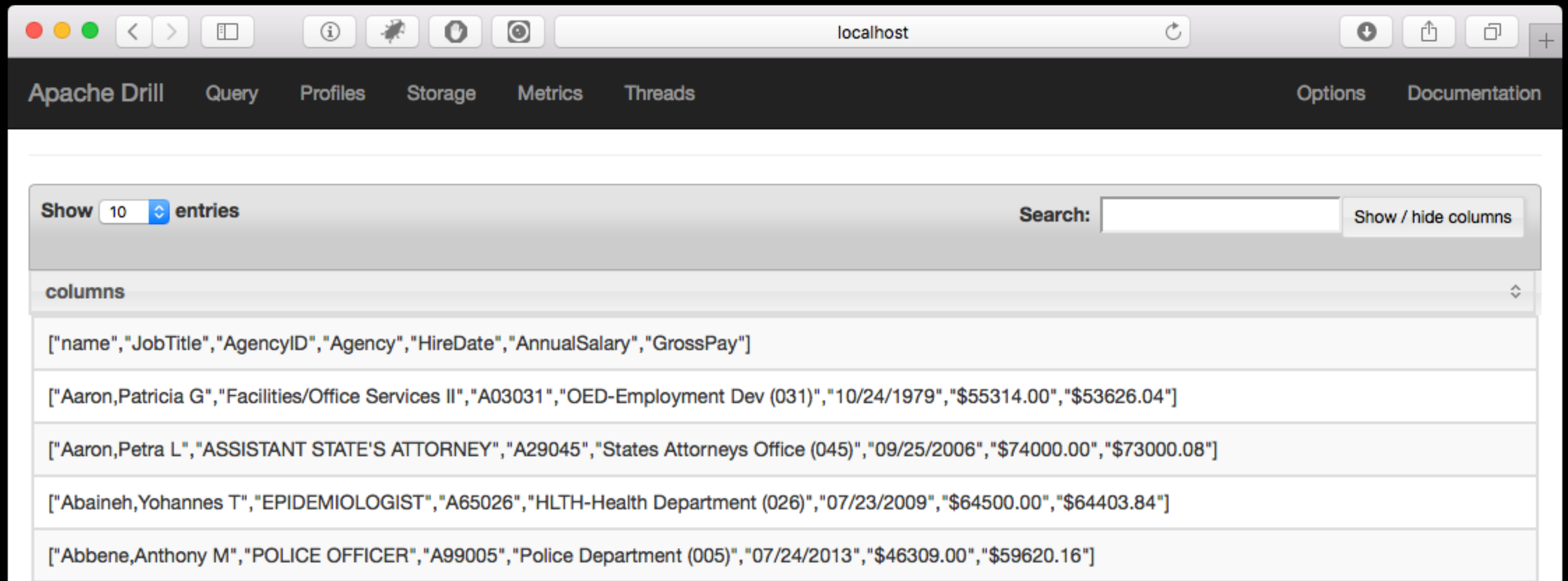
- Integer/BigInt/SmallInt
- Float/Decimal/Double
- Varchar/Binary
- Date/Time/Interval/Timestamp

Complex Data Types

- Arrays
- Maps

Querying Drill

["Aaron, Patricia G" "Facilities/Office Services"…]



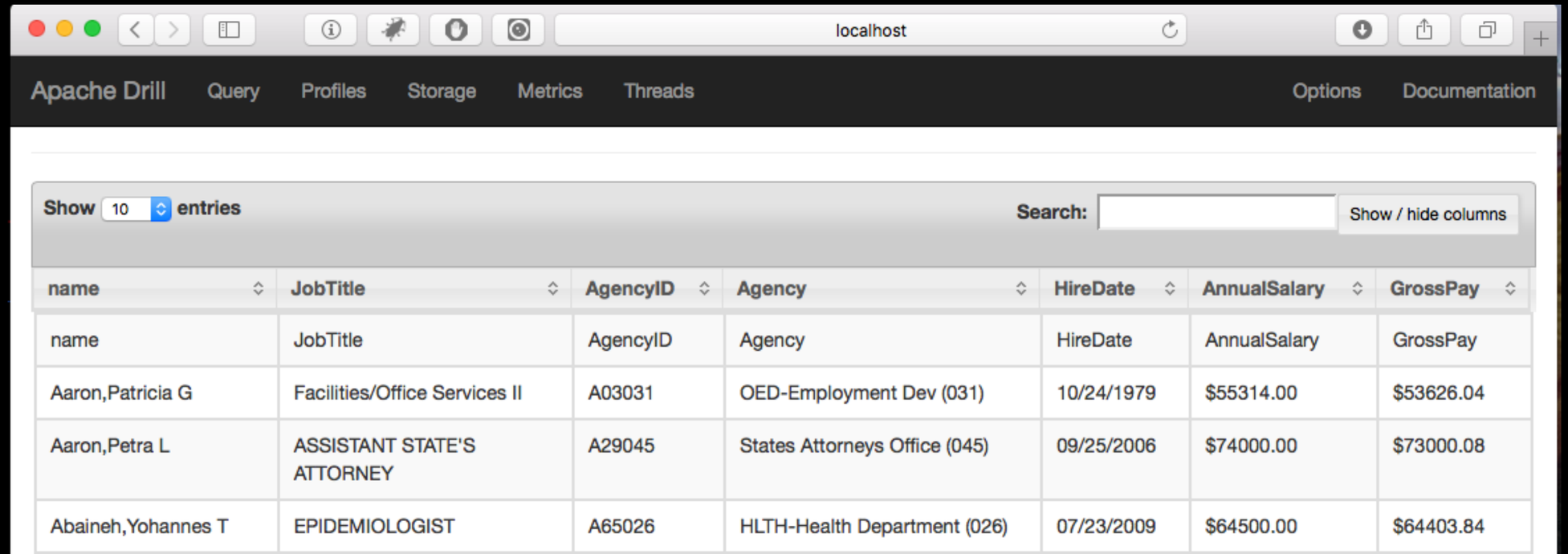
columns[*n*]

Querying Drill

```
SELECT columns[0] AS name,
columns[1] AS JobTitle,
columns[2] AS AgencyID,
columns[3] AS Agency,
columns[4] AS HireDate,
columns[5] AS AnnualSalary,
columns[6] AS GrossPay
FROM dfs.drillclass.`csv/baltimore_salaries_2016.csv`
LIMIT 10
```

Querying Drill

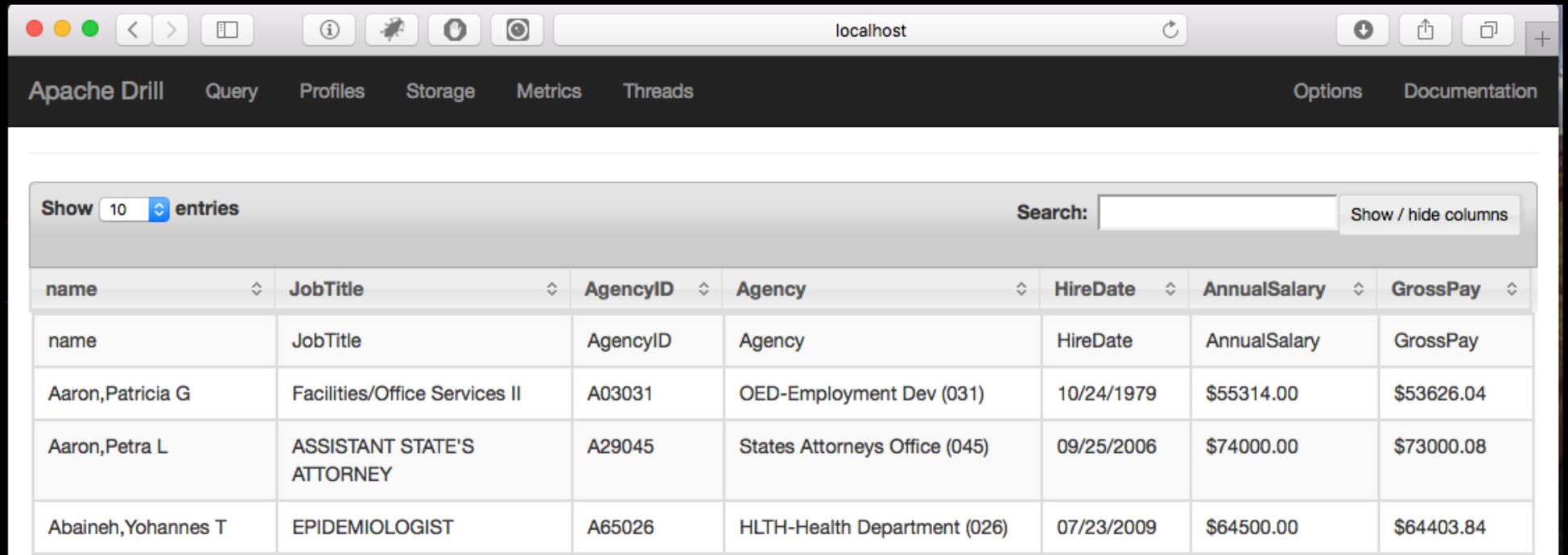
```
SELECT columns[0] AS name,
columns[1] AS JobTitle,
. . .
FROM dfs.drillclass.`csv/baltimore_salaries_2016.csv`
LIMIT 10
```



name	JobTitle	AgencyID	Agency	HireDate	AnnualSalary	GrossPay
name	JobTitle	AgencyID	Agency	HireDate	AnnualSalary	GrossPay
Aaron,Patricia G	Facilities/Office Services II	A03031	OED-Employment Dev (031)	10/24/1979	$55314.00	$53626.04
Aaron,Petra L	ASSISTANT STATE'S ATTORNEY	A29045	States Attorneys Office (045)	09/25/2006	$74000.00	$73000.08
Abaineh,Yohannes T	EPIDEMIOLOGIST	A65026	HLTH-Health Department (026)	07/23/2009	$64500.00	$64403.84

Querying Drill

```
SELECT columns[0] AS name,
columns[1] AS JobTitle,
. . .
FROM dfs.drillclass.`baltimore_salaries_2016.csv`
LIMIT 10
```



name	JobTitle	AgencyID	Agency	HireDate	AnnualSalary	GrossPay
name	JobTitle	AgencyID	Agency	HireDate	AnnualSalary	GrossPay
Aaron,Patricia G	Facilities/Office Services II	A03031	OED-Employment Dev (031)	10/24/1979	$55314.00	$53626.04
Aaron,Petra L	ASSISTANT STATE'S ATTORNEY	A29045	States Attorneys Office (045)	09/25/2006	$74000.00	$73000.08
Abaineh,Yohannes T	EPIDEMIOLOGIST	A65026	HLTH-Health Department (026)	07/23/2009	$64500.00	$64403.84

Querying Drill

```
"csvh": {
  "type": "text",
  "extensions": [
    "csvh"
  ],
  "extractHeader": true,
  "delimiter": ","
}
```

Querying Drill

File Extension	File Type
.psv	Pipe separated values
.csv	Comma separated value files
.csvh	Comma separated value with header
.tsv	Tab separated values
.json	JavaScript Object Notation files
.avro	Avro files (experimental)
.seq	Sequence Files

Querying Drill

Options	Description
comment	What character is a comment character
escape	Escape character
delimiter	The character used to delimit fields
quote	Character used to enclose fields
skipFirstLine	true/false
extractHeader	Reads the header from the CSV file

```
SELECT *
FROM table(
  dfs.drillclass.`baltimore_salaries_2016.csv`
  (
    type => 'text',
    extractHeader => true,
    fieldDelimiter => ‘,’
  )
)
```

Problem: Find the average salary of each Baltimore City job title

Aggregate Functions

Function	Argument Type	Return Type
AVG(expression)	Integer or Floating point	Floating point
COUNT(*)		BIGINT
COUNT([DISTINCT] <expression>)	any	BIGINT
MIN/MAX(<expression>)	Any numeric or date	same as argument
SUM(<expression>)	Any numeric or interval	same as argument

Querying Drill

```
SELECT JobTitle, AVG( AnnualSalary) AS avg_salary,
COUNT( DISTINCT name ) AS number
FROM drillclass.`baltimore_salaries_2016.csvh`
GROUP BY JobTitle
Order By avg_salary DESC
```

Querying Drill

Query Failed: An Error Occurred

org.apache.drill.common.exceptions.UserRemoteException: SYSTEM ERROR: SchemaChangeException: Failure while trying to materialize incoming schema. Errors: Error in expression at index -1. Error: Missing function implementation: [castINT(BIT-OPTIONAL)]. Full expression: --UNKNOWN EXPRESSION--.. Fragment 0:0 [Error Id: af88883b-f10a-4ea5-821d-5ff065628375 on 10.251.255.146:31010]

Querying Drill

```
SELECT JobTitle, AVG( AnnualSalary) AS avg_salary,
COUNT( DISTINCT name ) AS number
FROM dfs.drillclass.`baltimore_salaries_2016.csvh`
GROUP BY JobTitle
Order By avg_salary DESC
```

Querying Drill

```
SELECT JobTitle,
AVG( AnnualSalary ) AS avg_salary,
COUNT( DISTINCT name ) AS number
FROM dfs.drillclass.`baltimore_salaries_2016.csvh`
GROUP BY JobTitle
Order By avg_salary DESC
```

AnnualPay has extra characters

AnnualPay is a string

Querying Drill

Function	Return Type
BYTE_SUBSTR	BINARY or VARCHAR
CHAR_LENGTH	INTEGER
CONCAT	VARCHAR
ILIKE	BOOLEAN
INITCAP	VARCHAR
LENGTH	INTEGER
LOWER	VARCHAR
LPAD	VARCHAR
LTRIM	VARCHAR
POSITION	INTEGER
REGEXP_REPLACE	VARCHAR
RPAD	VARCHAR
RTRIM	VARCHAR
SPLIT	ARRAY
STRPOS	INTEGER
SUBSTR	VARCHAR
TRIM	VARCHAR
UPPER	VARCHAR

In Class Exercise:
Clean the field.

In this exercise you will use one of the string functions to remove the dollar sign from the ‘AnnualPay’ column.

Complete documentation can be found here:

https://drill.apache.org/docs/string-manipulation/

In Class Exercise:
Clean the field.

In this exercise you will use one of the string functions to remove the dollar sign from the ‘AnnualPay’ column.

Complete documentation can be found here:

https://drill.apache.org/docs/string-manipulation/

```
SELECT LTRIM( AnnualPay, '$' ) AS annualPay
FROM dfs.drillclass.`baltimore_salaries_2016.csvh`
```

Drill Data Types

Data type	Description
Bigint	8 byte signed integer
Binary	Variable length byte string
Boolean	True/false
Date	yyyy-mm-dd
Double / Float	8 or 4 byte floating point number
Integer	4 byte signed integer
Interval	A day-time or year-month interval
Time	HH:mm:ss
Timestamp	JDBC Timestamp
Varchar	UTF-8 encoded variable length string

cast(<expression> AS <data type>)

In Class Exercise:
Convert to a number

In this exercise you will use the cast() function to convert AnnualPay into a number.

Complete documentation can be found here:

https://drill.apache.org/docs/data-type-conversion/#cast

In Class Exercise:
Convert to a number

In this exercise you will use the cast() function to convert AnnualPay into a number.

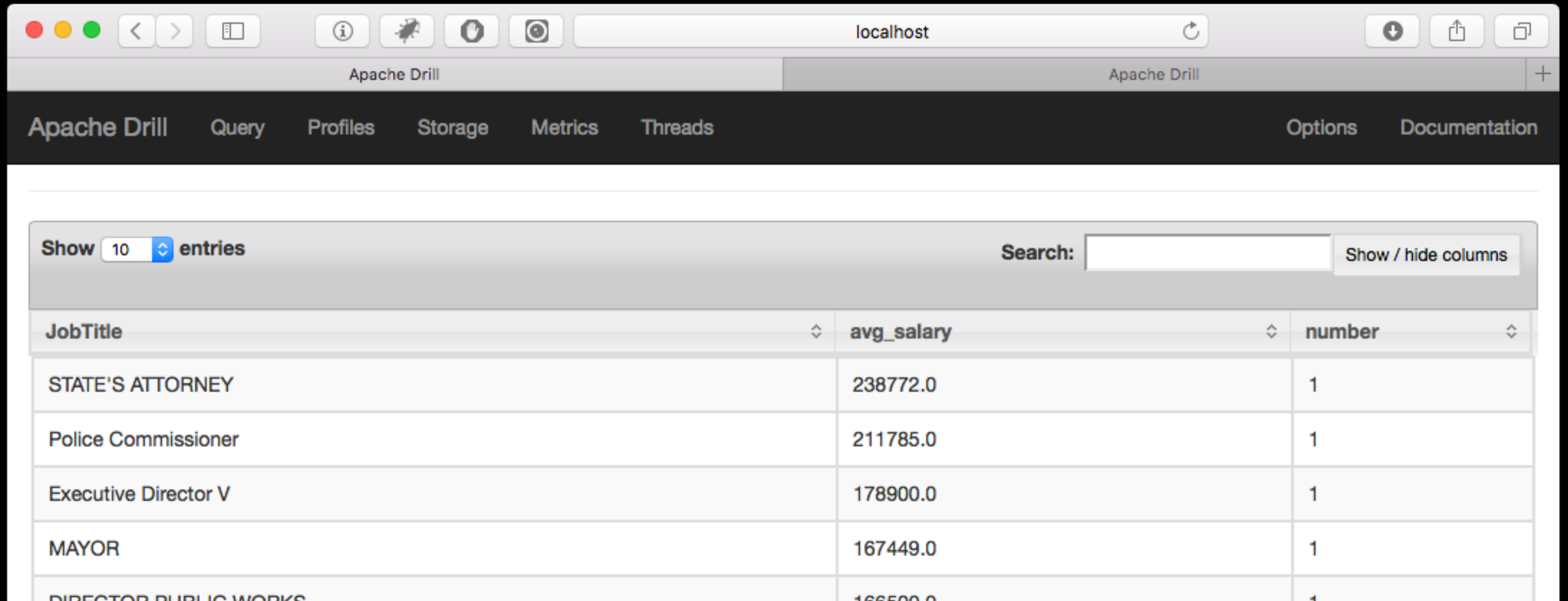
Complete documentation can be found here:

https://drill.apache.org/docs/data-type-conversion/#cast

```
SELECT CAST( LTRIM( AnnualPay, '$' ) AS FLOAT ) AS
annualPay
FROM dfs.drillclass.`csv/baltimore_salaries_2016.csvh`
```

```
SELECT JobTitle,
AVG(
   CAST(
      LTRIM( AnnualSalary, '$' ) AS FLOAT ) ) AS avg_salary,
COUNT( DISTINCT name ) AS number
FROM dfs.drillclass.`baltimore_salaries_2016.csvh`
GROUP BY JobTitle
Order By avg_salary DESC
```

```
SELECT JobTitle,
AVG( CAST( LTRIM( AnnualSalary, '$' ) AS FLOAT) ) AS avg_salary,
COUNT( DISTINCT name ) AS number
FROM dfs.drillclass.`baltimore_salaries_2016.csvh`
GROUP BY JobTitle
Order By avg_salary DESC
```



TO_NUMBER(<field>, <format>)

TO_NUMBER(<field>, <format>)

Symbol	Meaning
0	Digit
#	Digit, zero shows as absent
.	Decimal separator or monetary separator
-	Minus Sign
,	Grouping Separator
%	Multiply by 100 and show as percentage
‰ \u2030	Multiply by 1000 and show as per mille value
¤ \u00A4	Currency symbol

In Class Exercise:
Convert to a number using **TO_NUMBER()**

In this exercise you will use the **TO_NUMBER()** function to convert AnnualPay into a numeric field.

Complete documentation can be found here:

https://drill.apache.org/docs/data-type-conversion/#to_number

In Class Exercise:
Convert to a number using **TO_NUMBER()**

In this exercise you will use the **TO_NUMBER()** function to convert AnnualPay into a numeric field.

Complete documentation can be found here:

https://drill.apache.org/docs/data-type-conversion/#to_number

```
SELECT JobTitle,
AVG( TO_NUMBER( AnnualSalary, '¤' )) AS avg_salary,
COUNT( DISTINCT `EmpName` ) AS number
FROM dfs.drillclass.`baltimore_salaries_2016.csvh`
GROUP BY JobTitle
Order BY avg_salary DESC
```

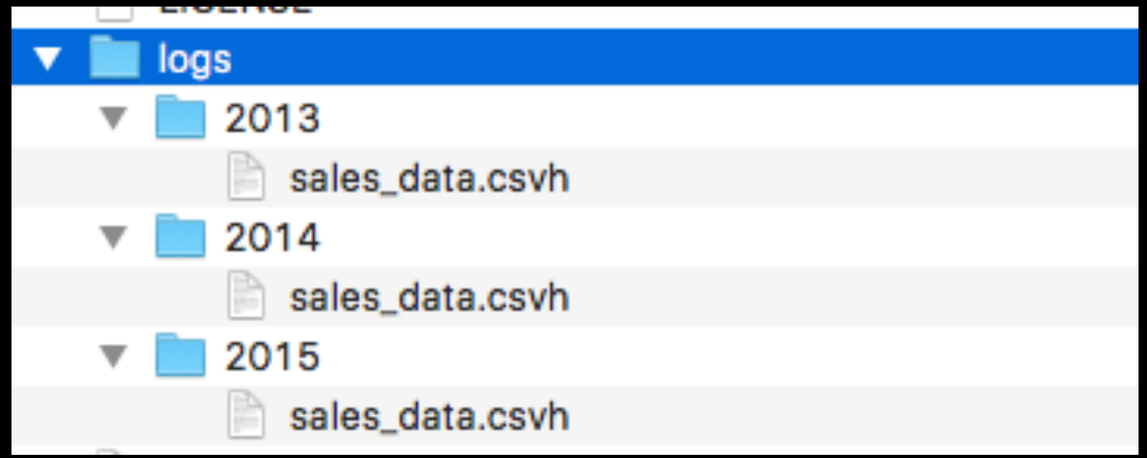
Topics for Tomorrow

- Dealing with dates and times
- Nested Data
- Reading other data types
- Programmatically connecting to Drill
- Connecting other data sources

Problem: You have files spread across many directories which you would like to analyze

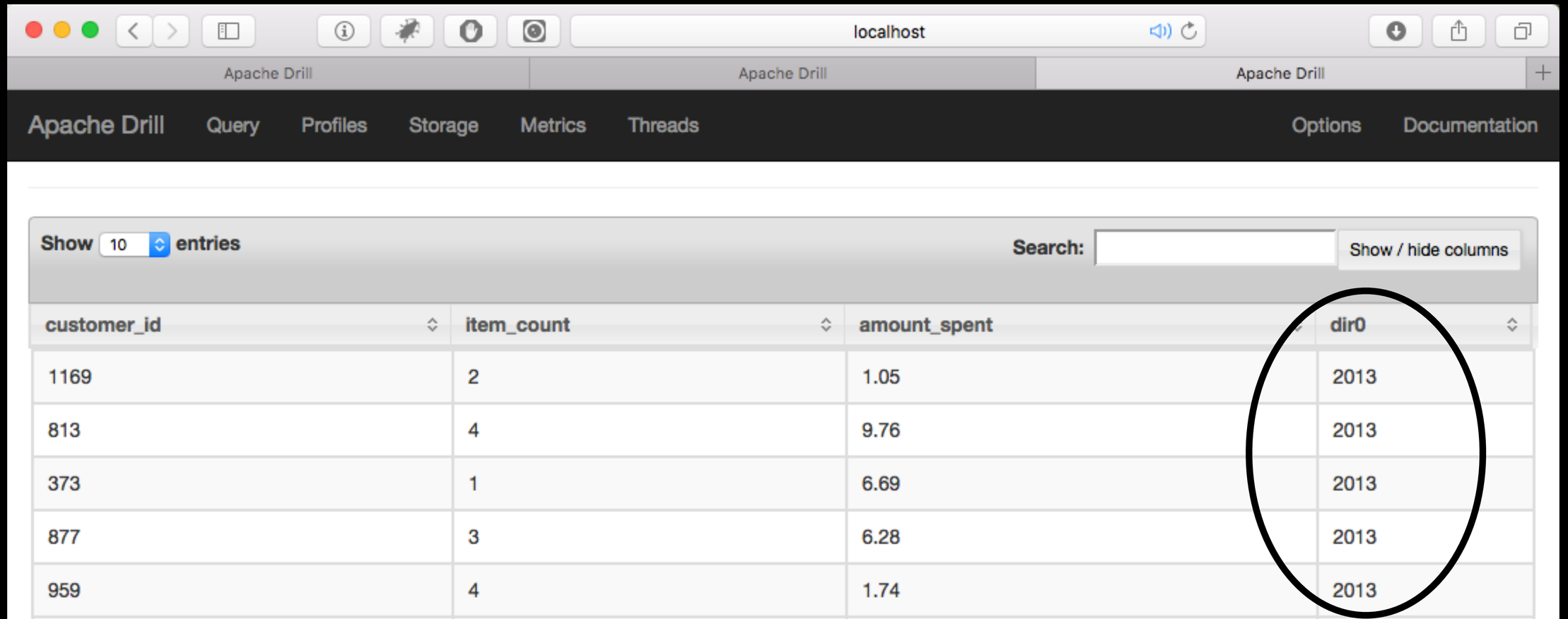
Problem: You have multiple log files which you would like to analyze

- In the sample data files, there is a folder called ‘logs’ which contains the following structure:



```
SELECT *
FROM dfs.drillclass.`logs/`
LIMIT 10
```

```
SELECT *
FROM dfs.drillclass.`logs/`
LIMIT 10
```



dir*n* accesses the subdirectories

dir*n* accesses the subdirectories

```
SELECT *
FROM dfs.drilldata.`logs/`
WHERE dir0 = '2013'
```

Directory Functions

Function	Description
MAXDIR(), MINDIR()	Limit query to the first or last directory
IMAXDIR(), IMINDIR()	Limit query to the first or last directory in case insensitive order.

```
WHERE dir<n> = MAXDIR('<plugin>.<workspace>', '<filename>')
```

In Class Exercise:

Find the total number of items sold by year and the total dollar sales in each year.

HINT: Don’t forget to CAST() the fields to appropriate data types

In Class Exercise:

Find the total number of items sold by year and the total dollar sales in each year.

HINT: Don't forget to CAST() the fields to appropriate data types

```
SELECT dir0 AS data_year,
SUM( CAST( item_count AS INTEGER ) ) as total_items,
SUM( CAST( amount_spent AS FLOAT ) ) as total_sales
FROM dfs.drillclass.`logs/`
GROUP BY dir0
```

Questions?

Homework

Using the Baltimore Salaries dataset write queries that answer the following questions:

1. In 2016, calculate the average difference in GrossPay and Annual Salary by Agency. HINT: Include `WHERE NOT( GrossPay ='' )` in your query. For extra credit, calculate the number of people in each Agency, and the min/max for the salary delta as well.

2. Find the top 10 individuals whose salaries changed the most between 2015 and 2016, both gain and loss.

3. (Optional Extra Credit) Using the various string manipulation functions, **split** the name function into two columns for the last name and first name. HINT: Don't overthink this, and review the sides about the columns array if you get stuck.

Thank you!!