O'REILLY®

Security

BUILD BETTER DEFENSES

oreillysecuritycon.com
#oreillysecurity

Drilling into Network Data
Apache Drill Workshop

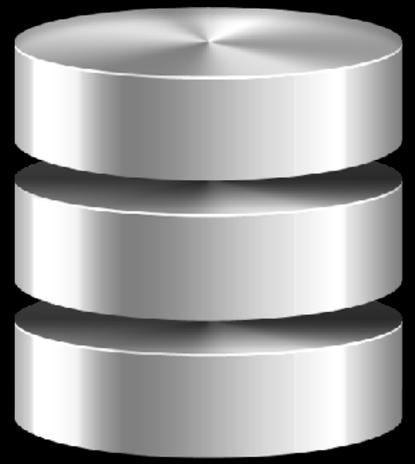
Charles S. Givre
cgivre@thedataist.com
@cgivre
thedataist.com

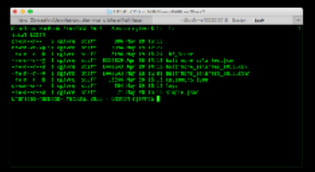
What is Drill?

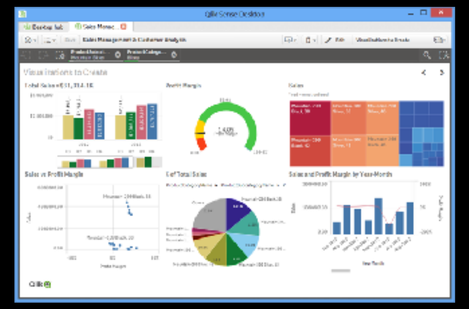
Data is not arranged in an optimal way for ad-hoc analysis

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

ETL







APACHE DRILL

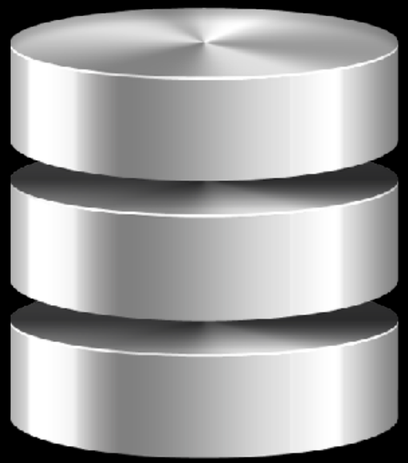
hadoop

ETL

HIVE

CSV

JSON



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

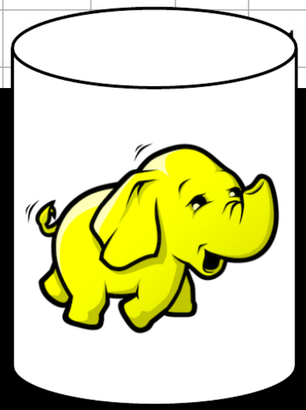
Quick Demo

Thank you Jair Aguirre!!

Quick Demo

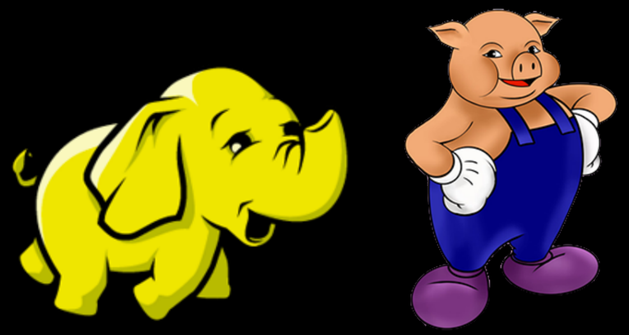
yearID	lgID	teamID	franchID	divID	Rank	G	Ghome	W	L	DivWin	WCWin	LgWin	WSWin	R	AB	H	2B	3B	HR	E
1871	NA	BS1	BNA		3	31		20	10			N		401	1372	426	70	37	3	
1871	NA	CH1	CNA		2	28		19	9			N		302	1196	323	52	21	10	
1871	NA	CL1	CFC		8	29		10	19			N		249	1186	328	35	40	7	
1871	NA	FW1	KEK		7	19		7	12			N		137	746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

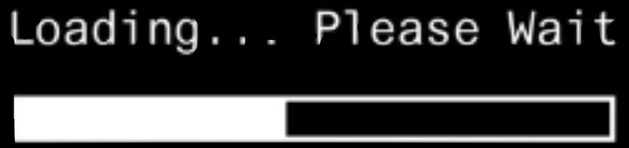
anlahman.com/baseball-archive/statistics



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:
0.89 seconds!!

NoSQL, No Problem

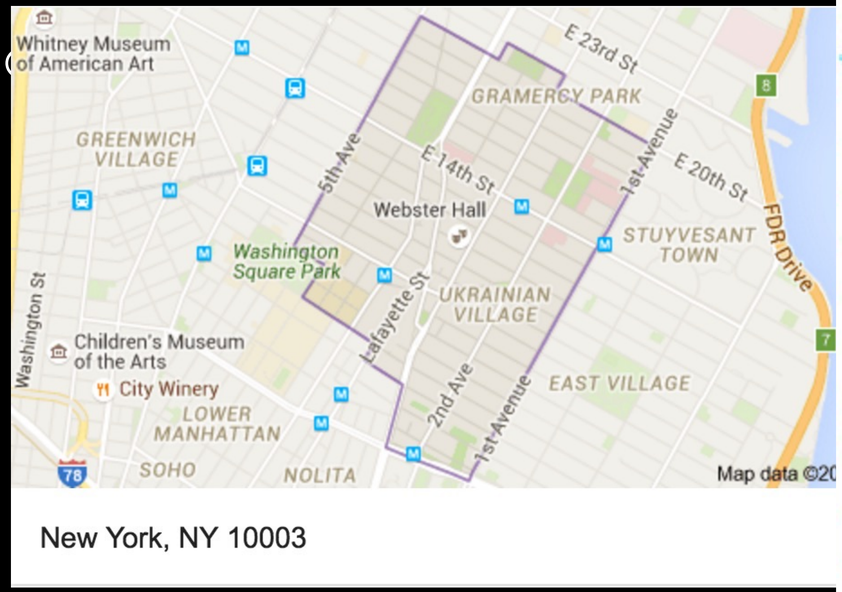
NoSQL, No Problem


```
{
  "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;
```

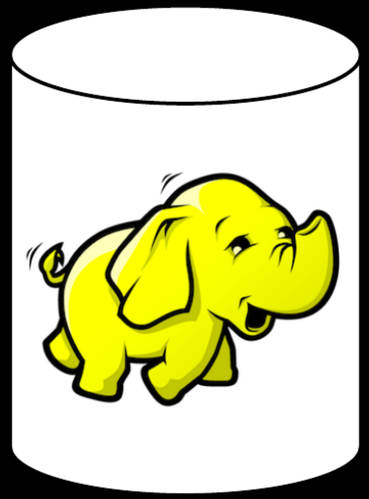


zip	rests
10003	686
10019	675
10036	611
10001	520
10022	485
10013	480
10002	471
10011	467
10016	433
10014	428

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

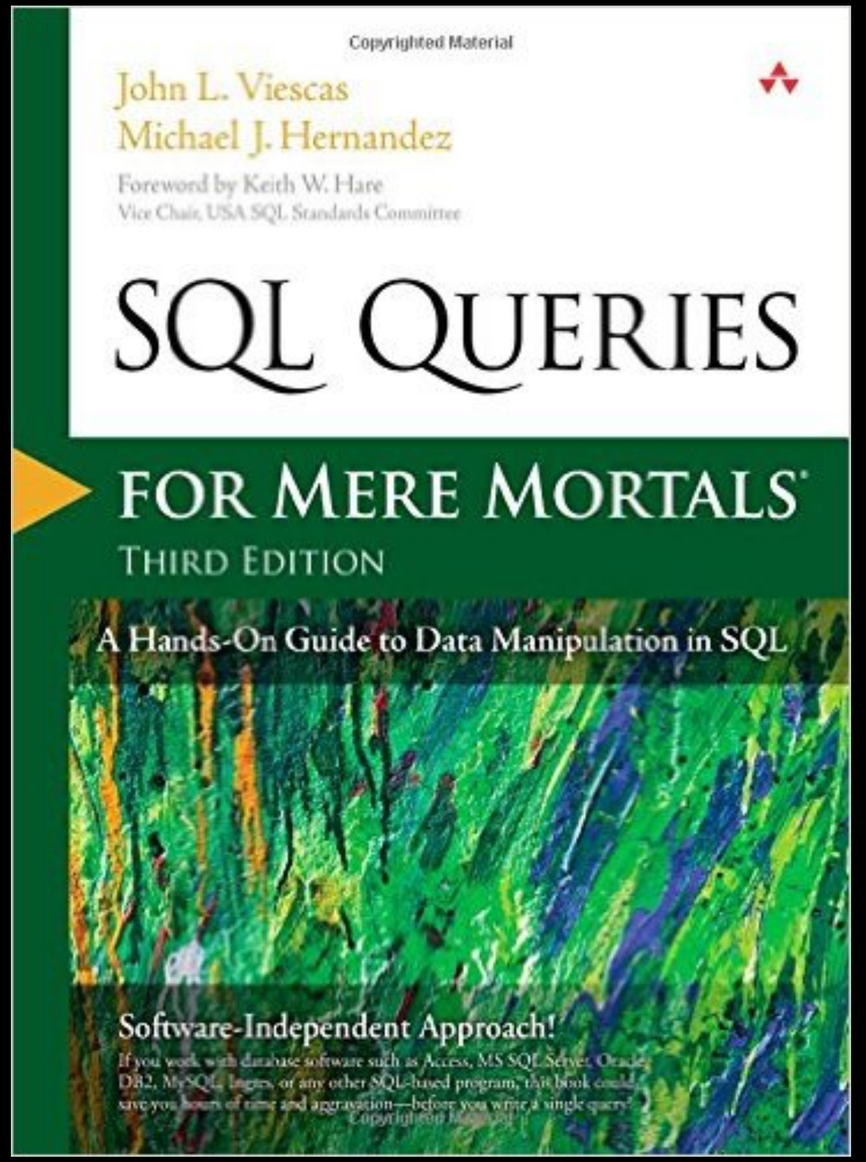
Execution Time: 0.502 Seconds


WARNING
DANGER
DO NOT PULL
HANDLE
TOMCAT
RANGER
CENTURION

To follow along, please download the files at:
https://github.com/cgivre/drillworkshop

Game Plan

- Query basic delimited data
- Query nested data
- Query log files
- Connect multiple data sources
- Write UDFs
- Query Drill using Python



http://amzn.to/2eFIVaj

Installing Drill

Installing Drill

1. Download Tarball from drill.apache.org
2. Unzip Tarball.

Starting Drill

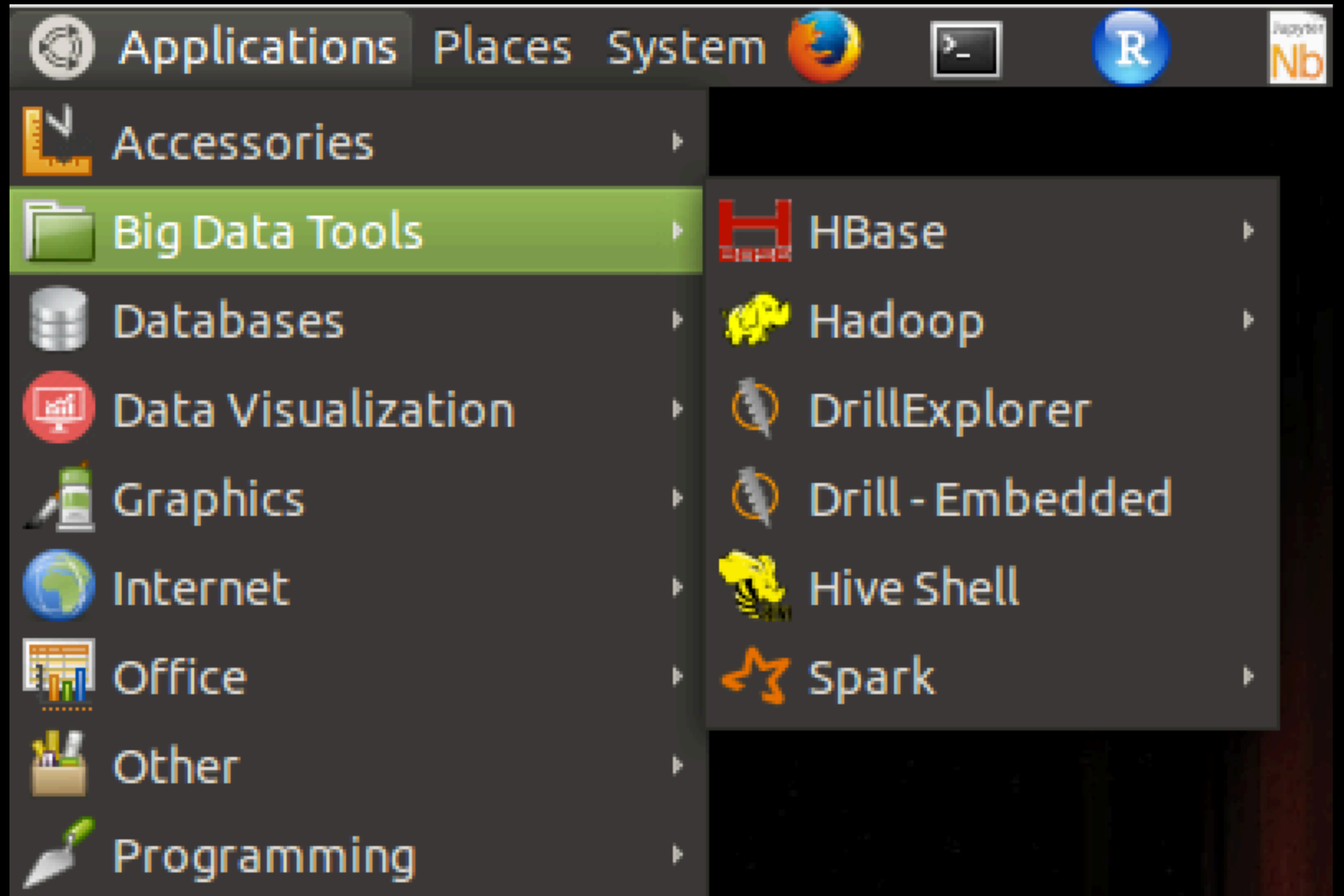
Starting Drill

Embedded Mode: For use on a standalone system

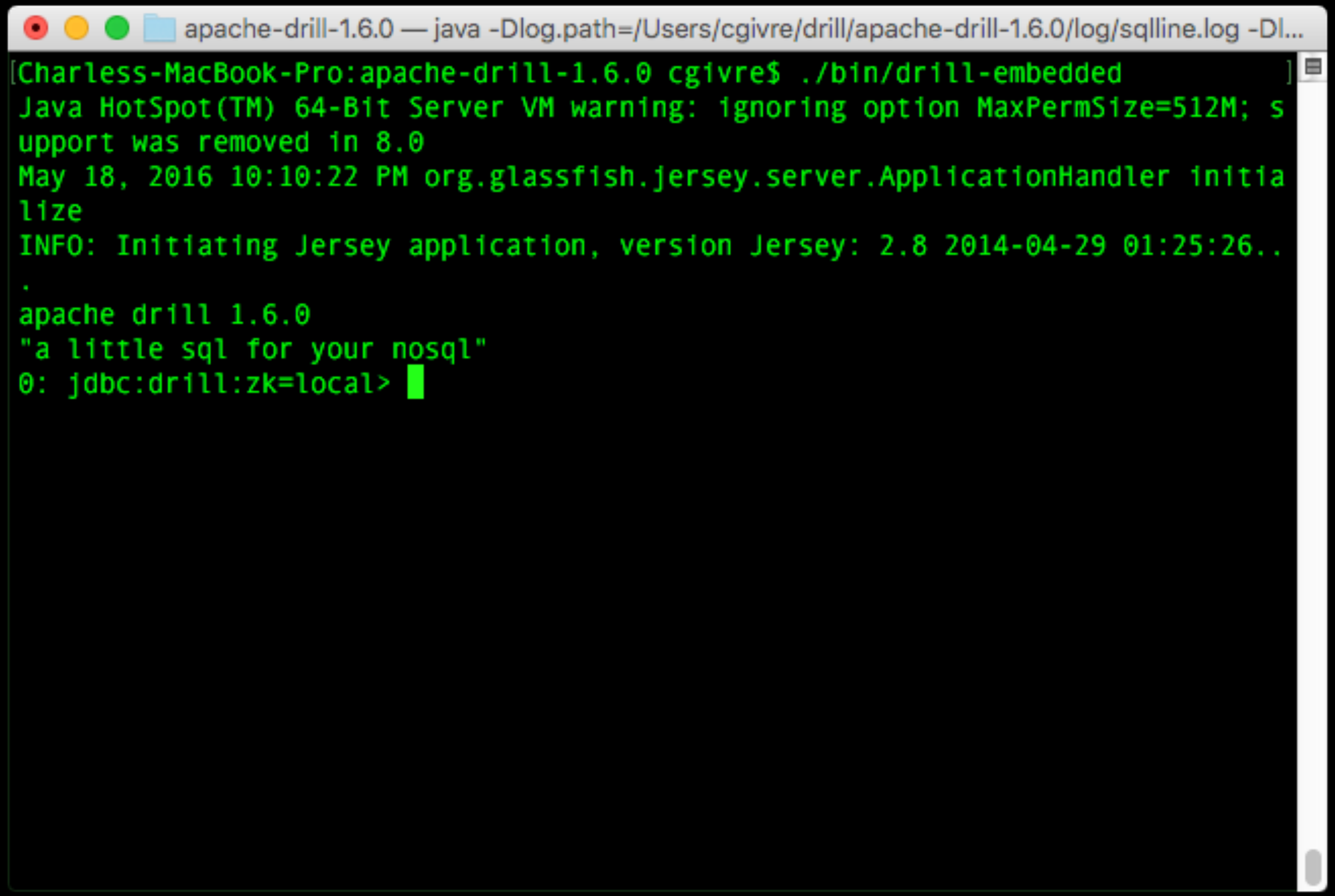
```
$./bin/drill-embedded
```


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

Starting Drill

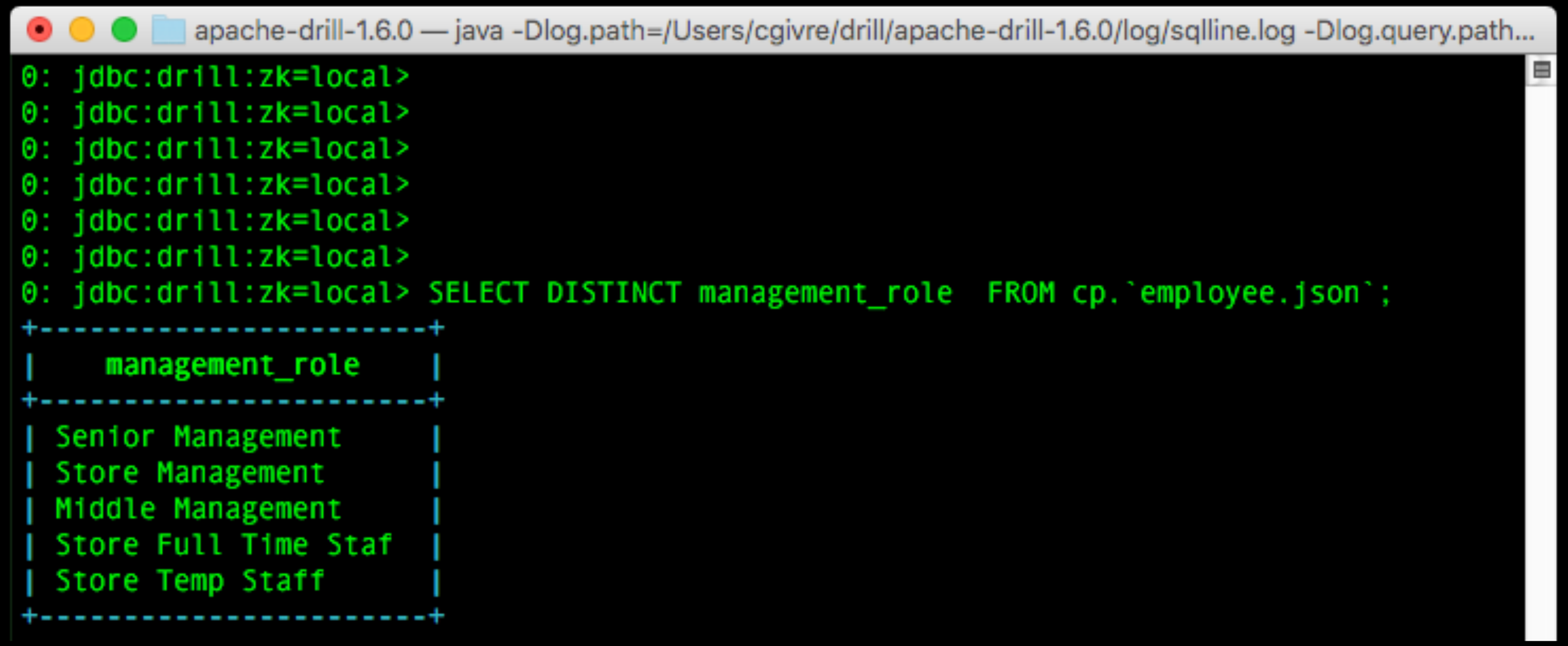


Querying Drill



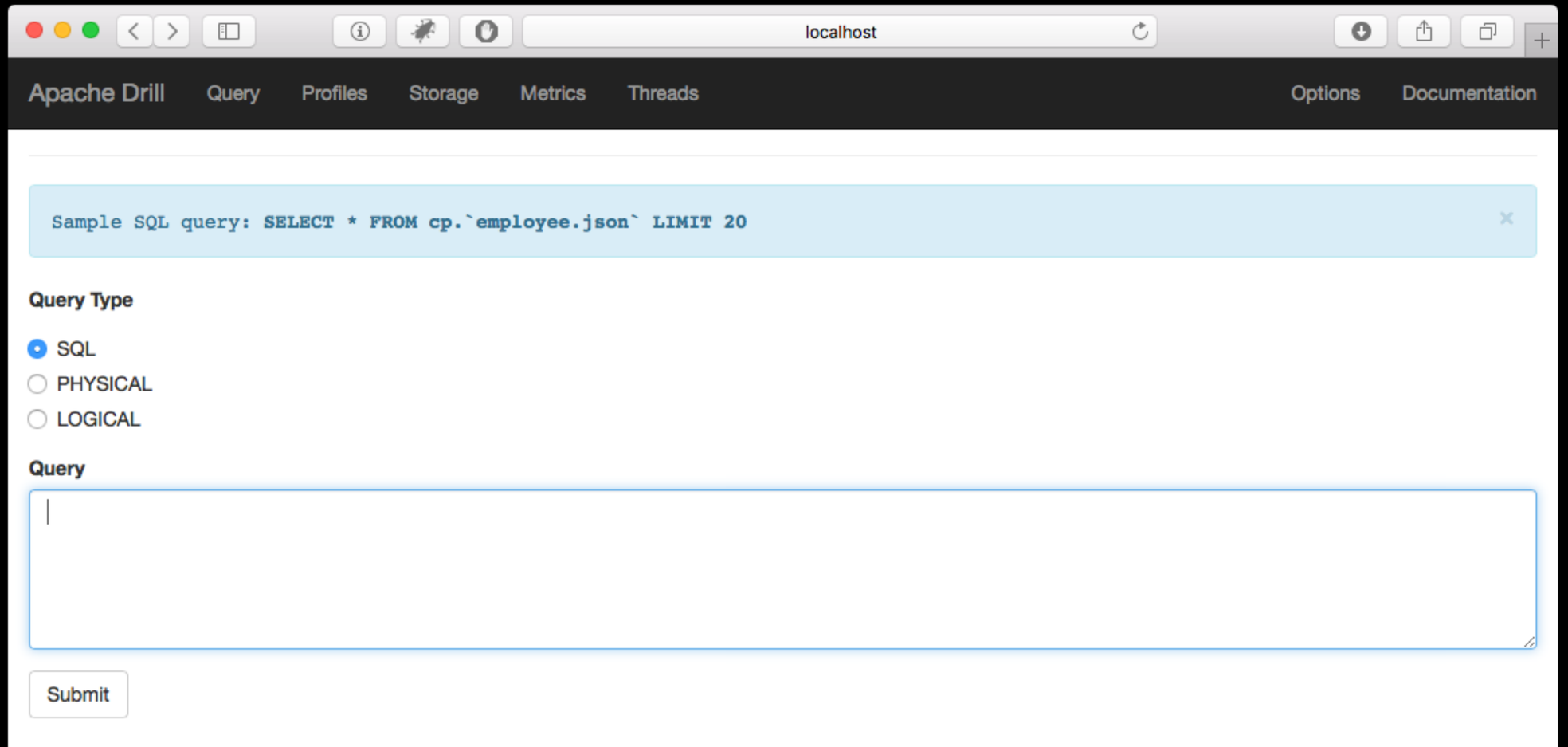
Querying Drill

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



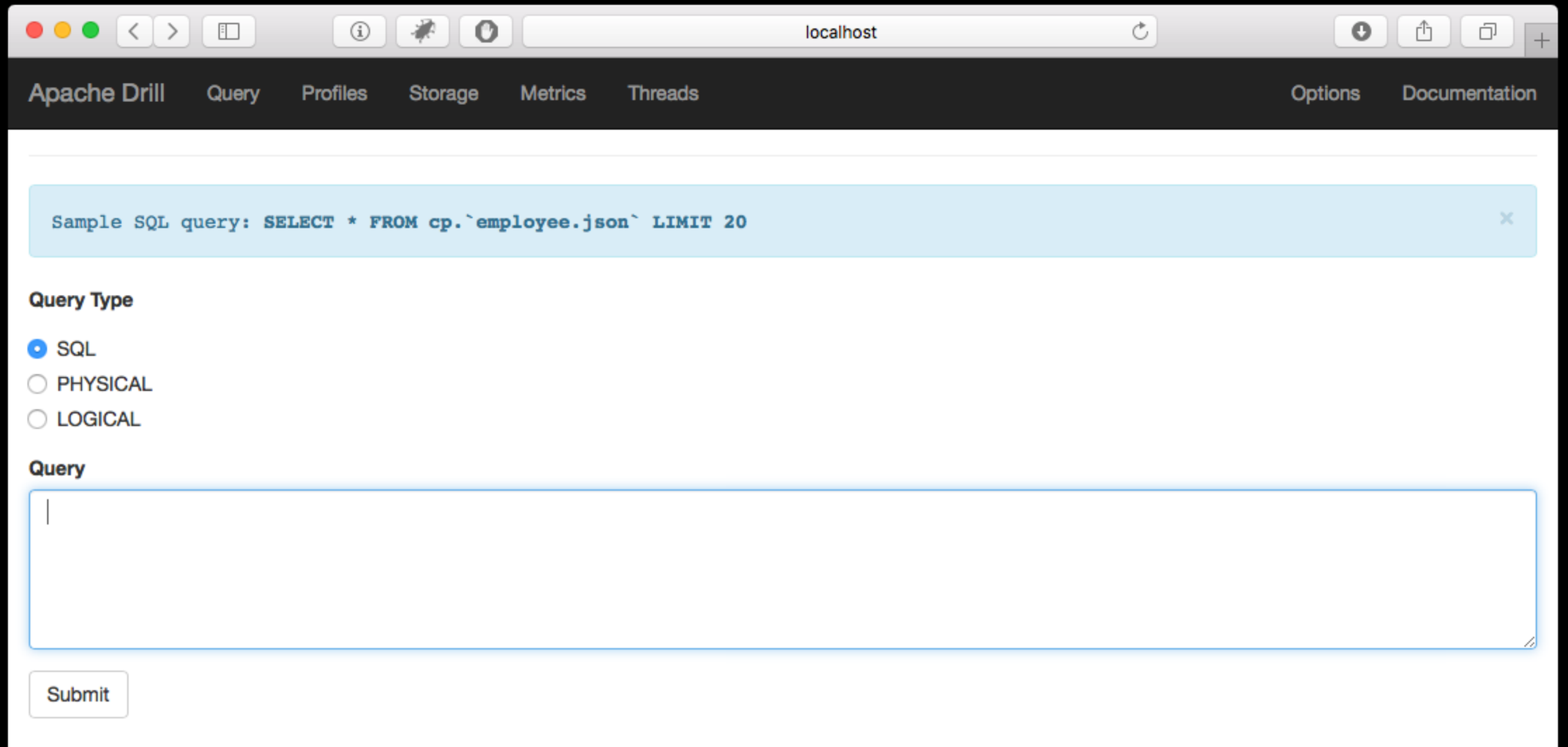
Querying Drill

http://localhost:8047



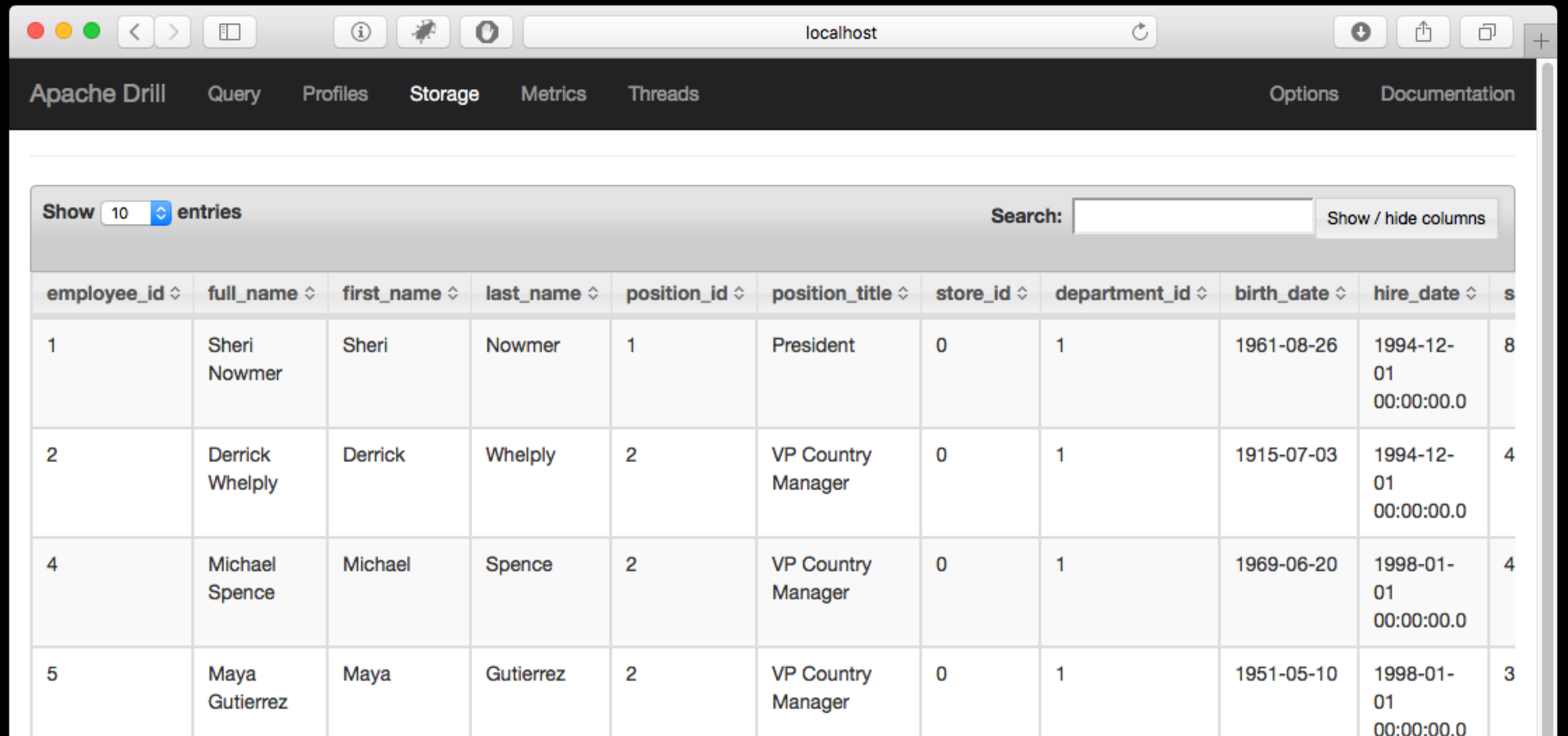
Querying Drill

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



Querying Drill

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



employee_id	full_name	first_name	last_name	position_id	position_title	store_id	department_id	birth_date	hire_date	s
1	Sheri Nowmer	Sheri	Nowmer	1	President	0	1	1961-08-26	1994-12-01 00:00:00.0	8
2	Derrick Whelply	Derrick	Whelply	2	VP Country Manager	0	1	1915-07-03	1994-12-01 00:00:00.0	4
4	Michael Spence	Michael	Spence	2	VP Country Manager	0	1	1969-06-20	1998-01-01 00:00:00.0	4
5	Maya Gutierrez	Maya	Gutierrez	2	VP Country Manager	0	1	1951-05-10	1998-01-01 00:00:00.0	3

Querying Drill

```
SELECT <fields>
FROM <table>
WHERE <optional logical condition>
```

Querying Drill

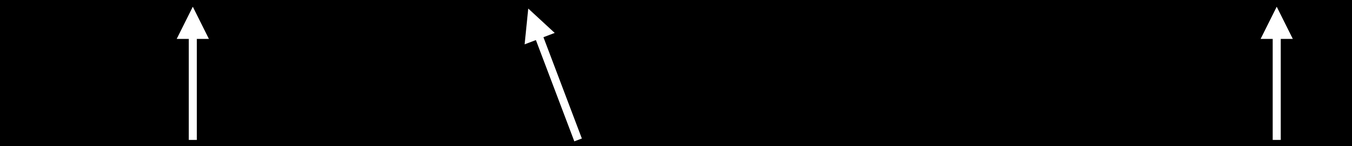
```
SELECT name, address, email
FROM customerData
WHERE age > 20
```

Querying Drill

```
SELECT name, address, email
FROM dfs.logs.`/data/customers.csv`
WHERE age > 20
```

Querying Drill

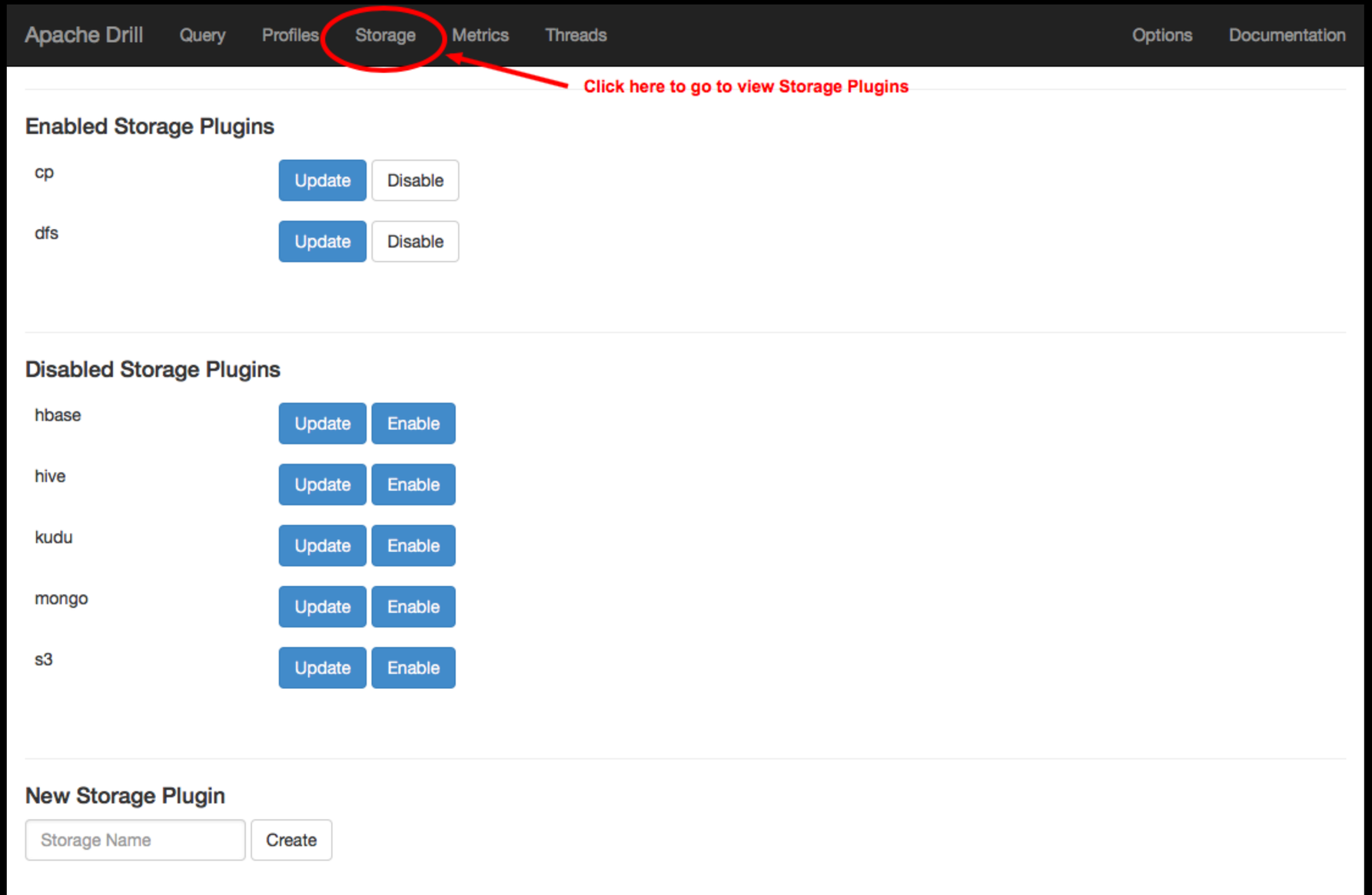
```
FROM dfs.logs.`/data/customers.csv`
```



Querying Drill

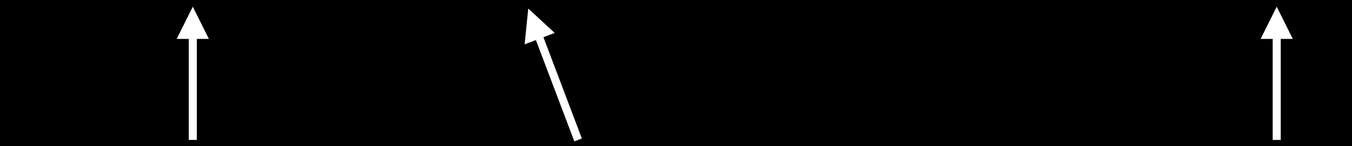
Plugins Supported	Description
cp	Queries files in the Java ClassPath
dfs	File System. Can connect to remote filesystems such as Hadoop
hbase	Connects to HBase
hive	Integrates Drill with the Apache Hive metastore
kudu	Provides a connection to Apache Kudu
mongo	Connects to mongoDB
RDBMS	Provides a connection to relational databases such as MySQL, Postgres, Oracle and others.
S3	Provides a connection to an S3 cluster

Querying Drill



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`
```

In Class Exercise:
Create a Workspace

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

1. First, download all the files from https://github.com/cgivre/drillworkshop 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

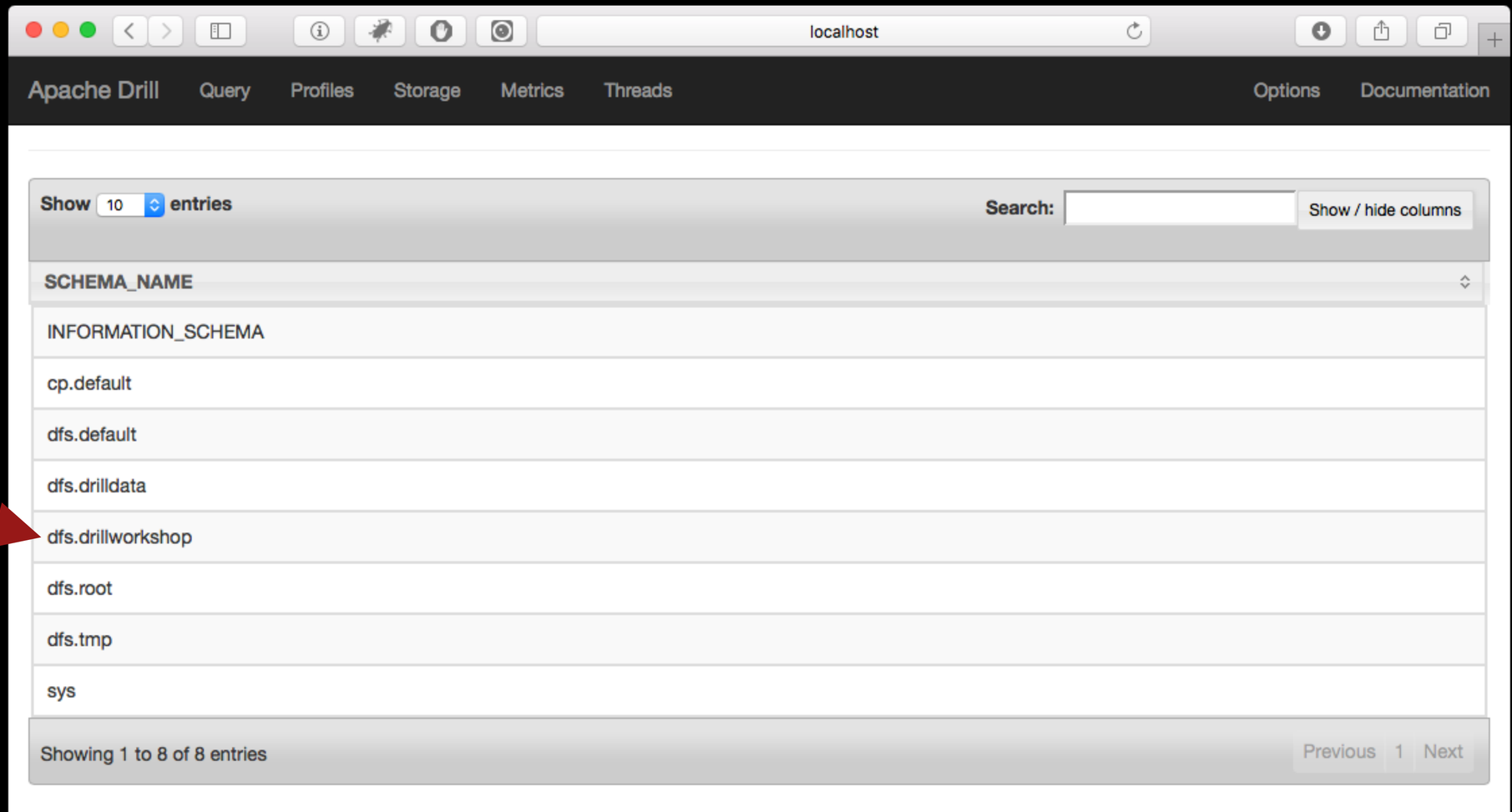
```
"drillworkshop": {
  "location": "<path to your files>",
  "writable": true,
  "defaultInputFormat": null
}
```

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

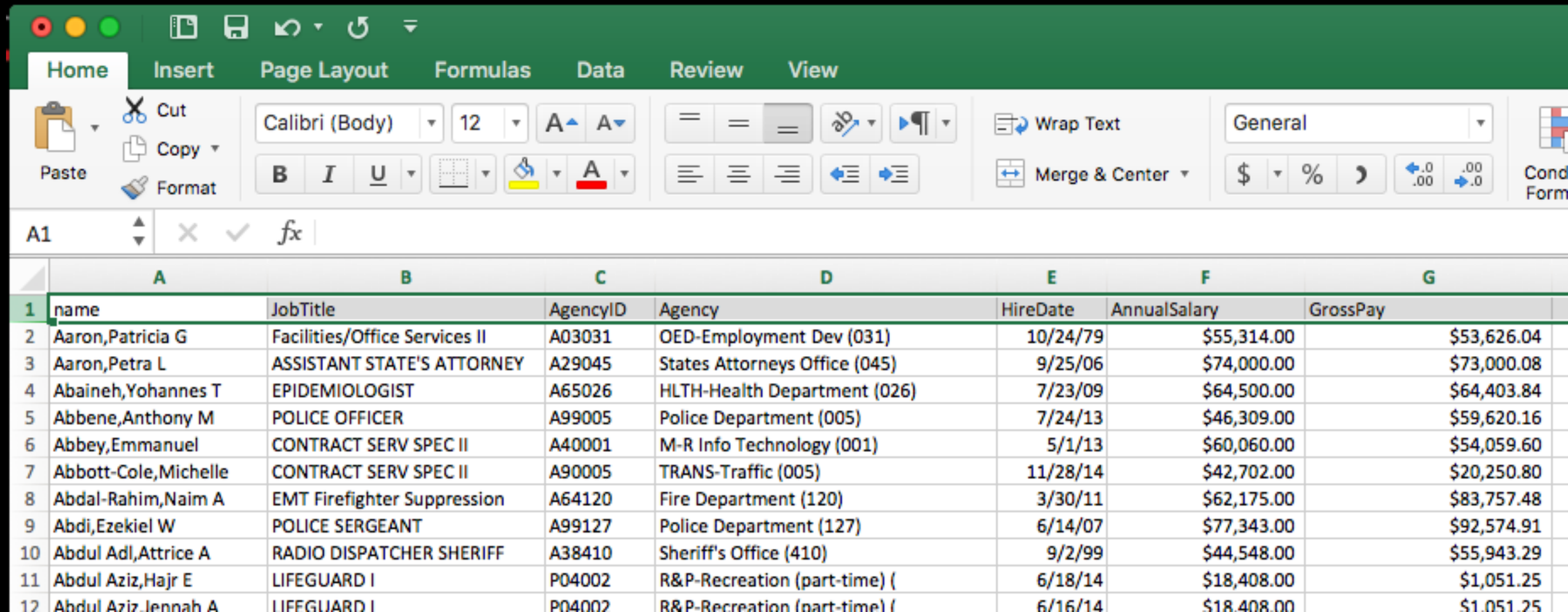
Querying Drill

```
SHOW databases;
```

Success!!



Querying Drill



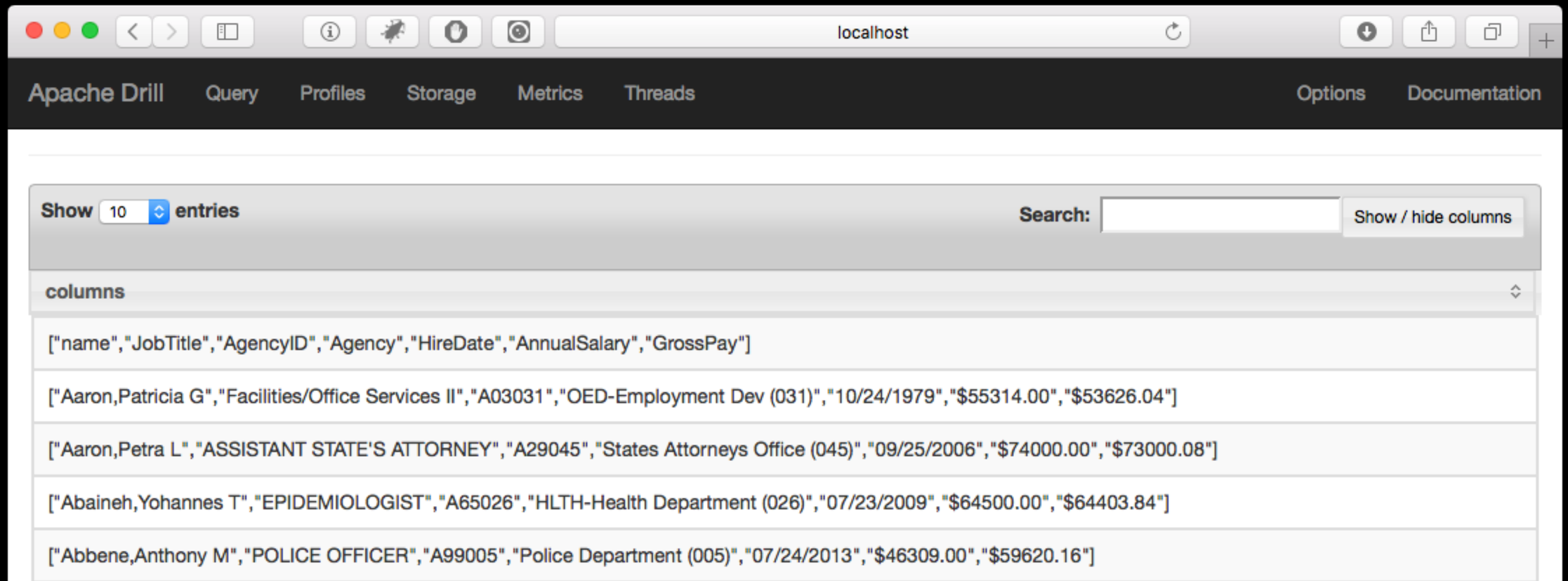
	A	B	C	D	E	F	G
1	name	JobTitle	AgencyID	Agency	HireDate	AnnualSalary	GrossPay
2	Aaron,Patricia G	Facilities/Office Services II	A03031	OED-Employment Dev (031)	10/24/79	$55,314.00	$53,626.04
3	Aaron,Petra L	ASSISTANT STATE'S ATTORNEY	A29045	States Attorneys Office (045)	9/25/06	$74,000.00	$73,000.08
4	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

Querying Drill

```
SELECT *
FROM dfs.drillworkshop.`csv/baltimore_salaries_2015.csv`
LIMIT 10
```

Drill Data Types

```
SELECT *
FROM dfs.drillworkshop.`csv/baltimore_salaries_2015.csv`
LIMIT 10
```



Drill Data Types

Simple Data Types

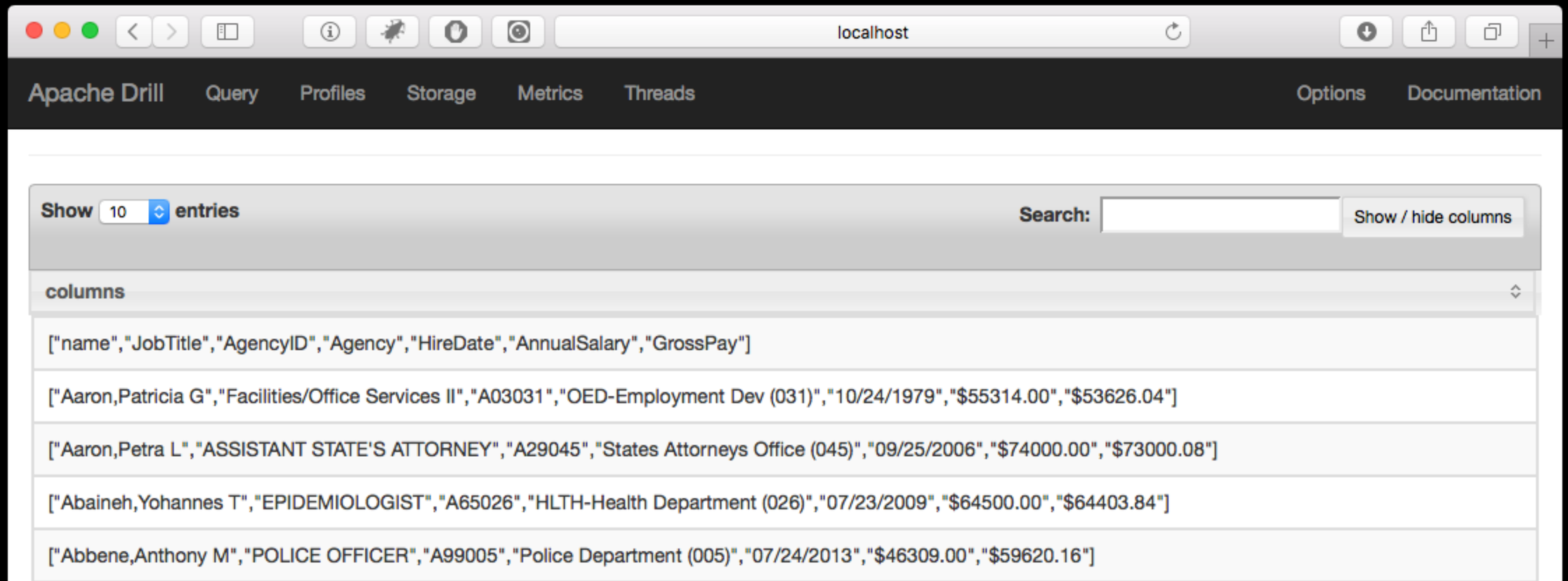
- Integer/BigInt/SmallInt
- Float/Decimal/Double
- Varchar/Binary
- Date/Time/Interval/Timestamp

Complex Data Types

- Arrays
- Maps

Querying Drill

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



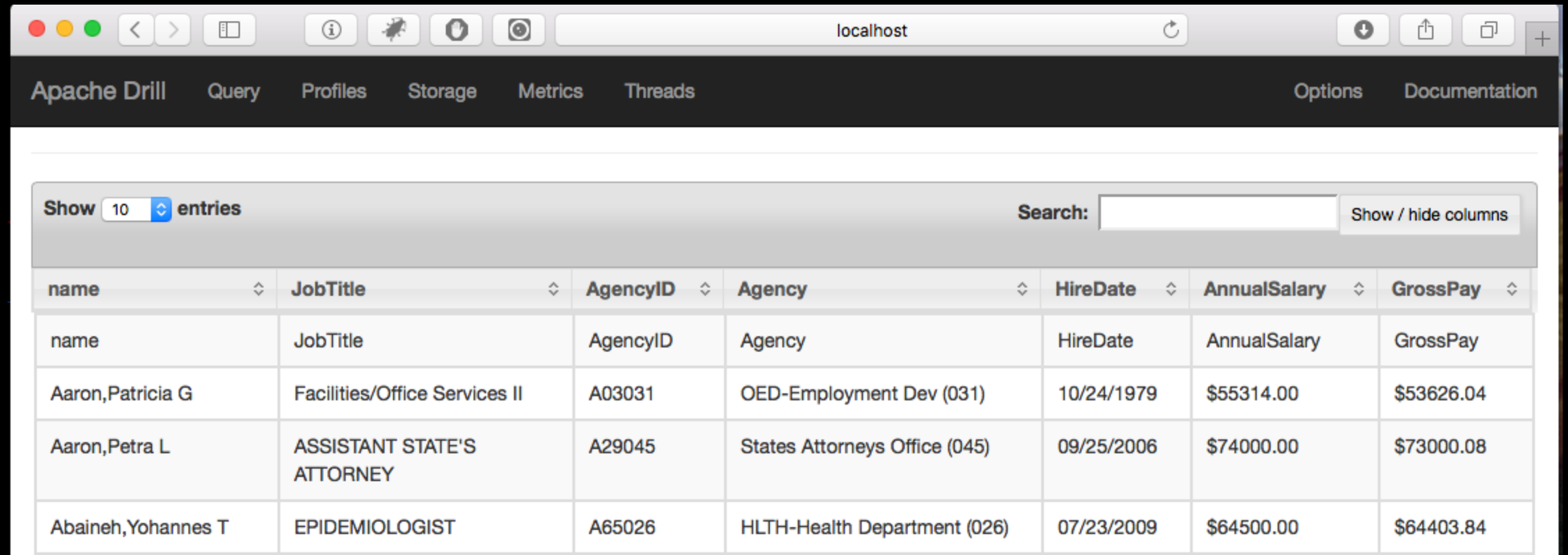
columns[*n*]

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.drillworkshop.`csv/baltimore_salaries_2015.csv`
LIMIT 10
```

Querying Drill

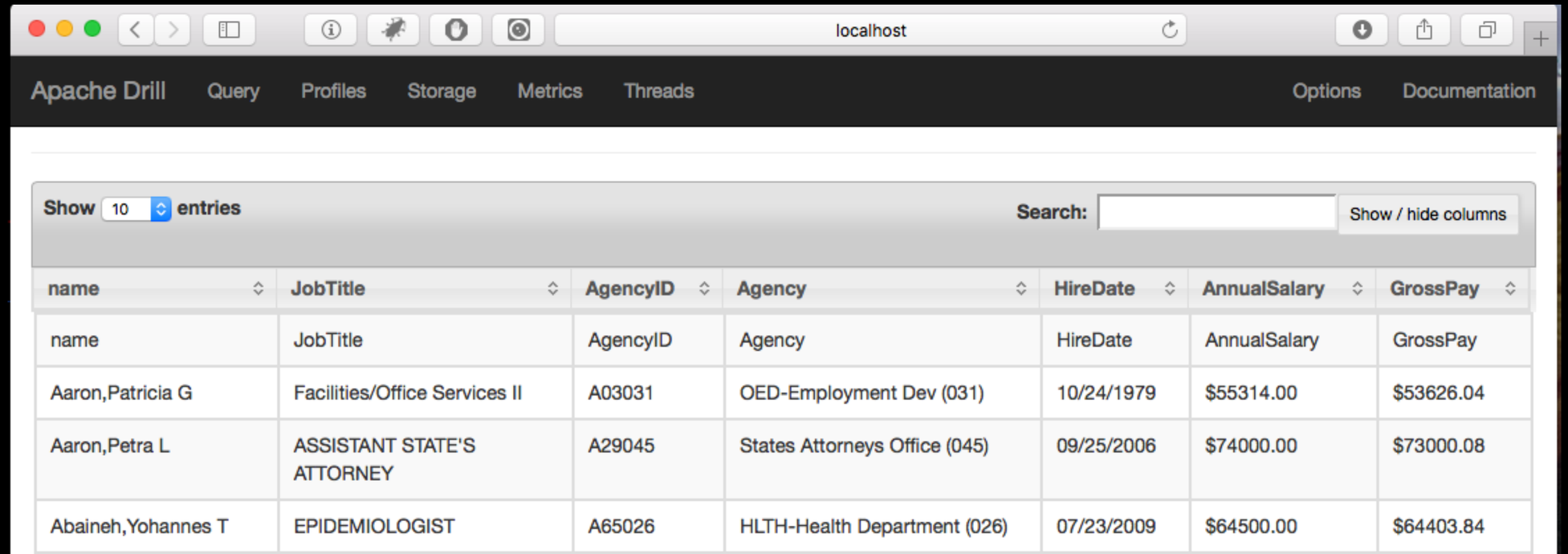
```
SELECT columns[0] AS name,
columns[1] AS JobTitle,
. . .
FROM dfs.drillworkshop.`csv/baltimore_salaries_2015.csv`
LIMIT 10
```



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.drillworkshop.`csv/baltimore_salaries_2015.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

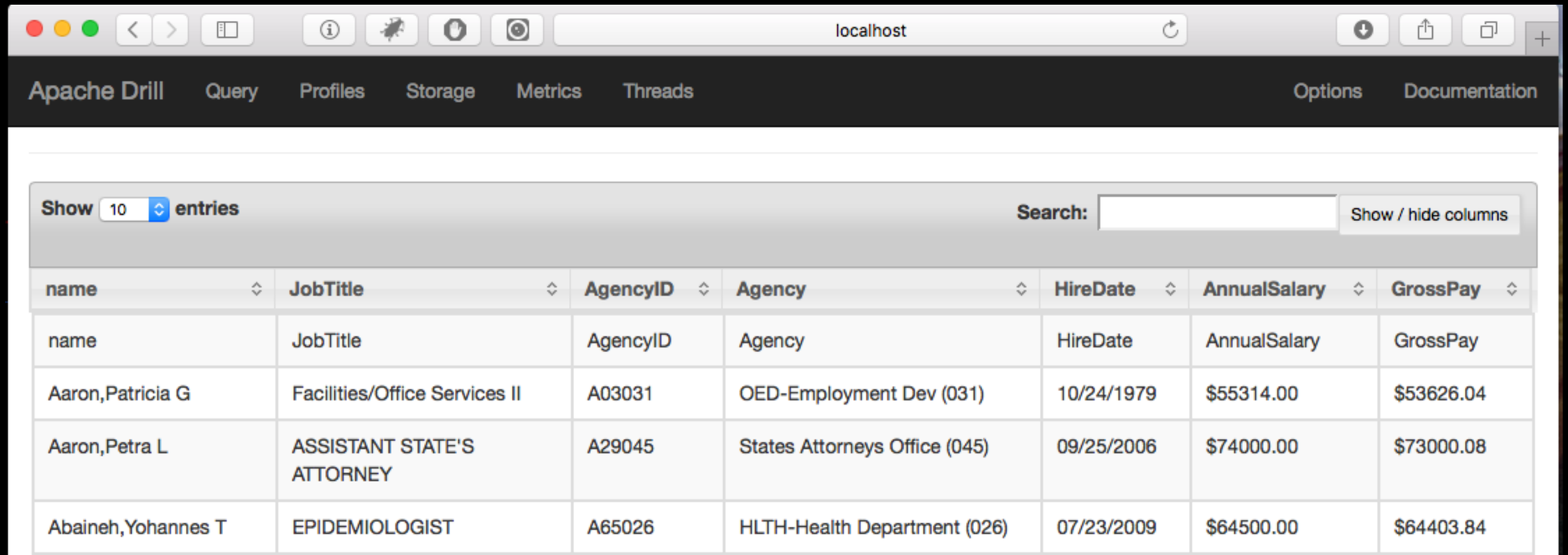
File Extension	File Type
.psv	Pipe separated values
.csv	Comma separated value files
.csvh	Comma separated value with header
.tsv	Tab separated values
.json	JavaScript Object Notation files
.avro	Avro files (experimental)
.seq	Sequence Files

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

Querying Drill

```
SELECT *
FROM dfs.drillworkshop.`csv/baltimore_salaries_2015.csvh`
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

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 dfs.drillworkshop.`baltimore_salaries_2015.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.drillworkshop.`csv/baltimore_salaries_2015.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.drillworkshop.`csv/baltimore_salaries_2015.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
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/

```
SELECT LTRIM( AnnualPay, '$' ) AS annualPay
FROM dfs.drillworkshop.`csv/baltimore_salaries_2015.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.

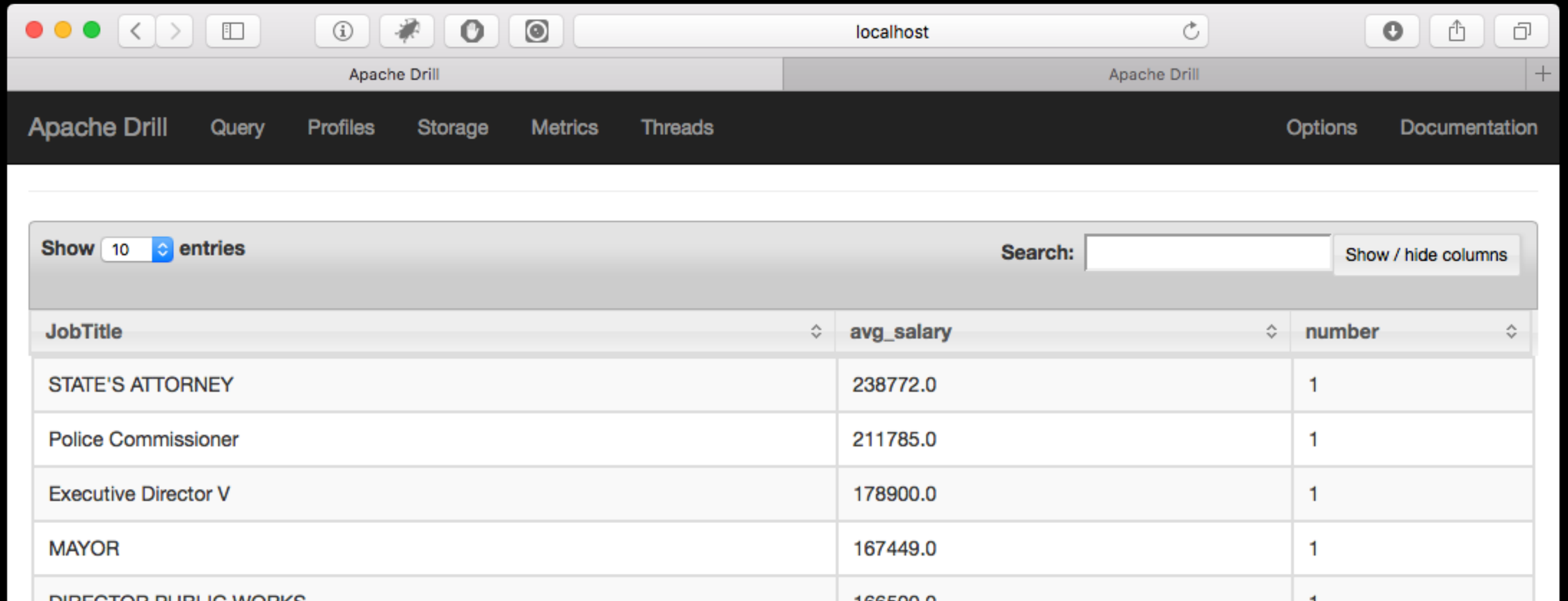
Complete documentation can be found here:

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

```
SELECT CAST( LTRIM( AnnualPay, '$' ) AS FLOAT ) AS
annualPay
FROM dfs.drillworkshop.`csv/baltimore_salaries_2015.csvh`
```

```
SELECT JobTitle,
AVG( CAST( LTRIM( AnnualSalary, '$' ) AS FLOAT) ) AS
avg_salary,
COUNT( DISTINCT name ) AS number
FROM dfs.drillworkshop.`csv/baltimore_salaries_2015.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.drillworkshop.`csv/baltimore_salaries_2015.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

```
SELECT JobTitle, AVG( TO_NUMBER( AnnualSalary, '¤' )) AS
avg_salary, COUNT( DISTINCT name ) AS number
FROM dfs.drillworkshop.`csv/baltimore_salaries_2015.csvh`
GROUP BY JobTitle
Order By avg_salary DESC
```

Working with Dates & Times

Working with Dates & Times

CAST(<field> **AS DATE**)

CAST(<field> **AS TIME**)

Working with Dates & Times

TO_DATE(<field>, ‘<format>’)

TO_TIMESTAMP(<field>, ‘<format>’)

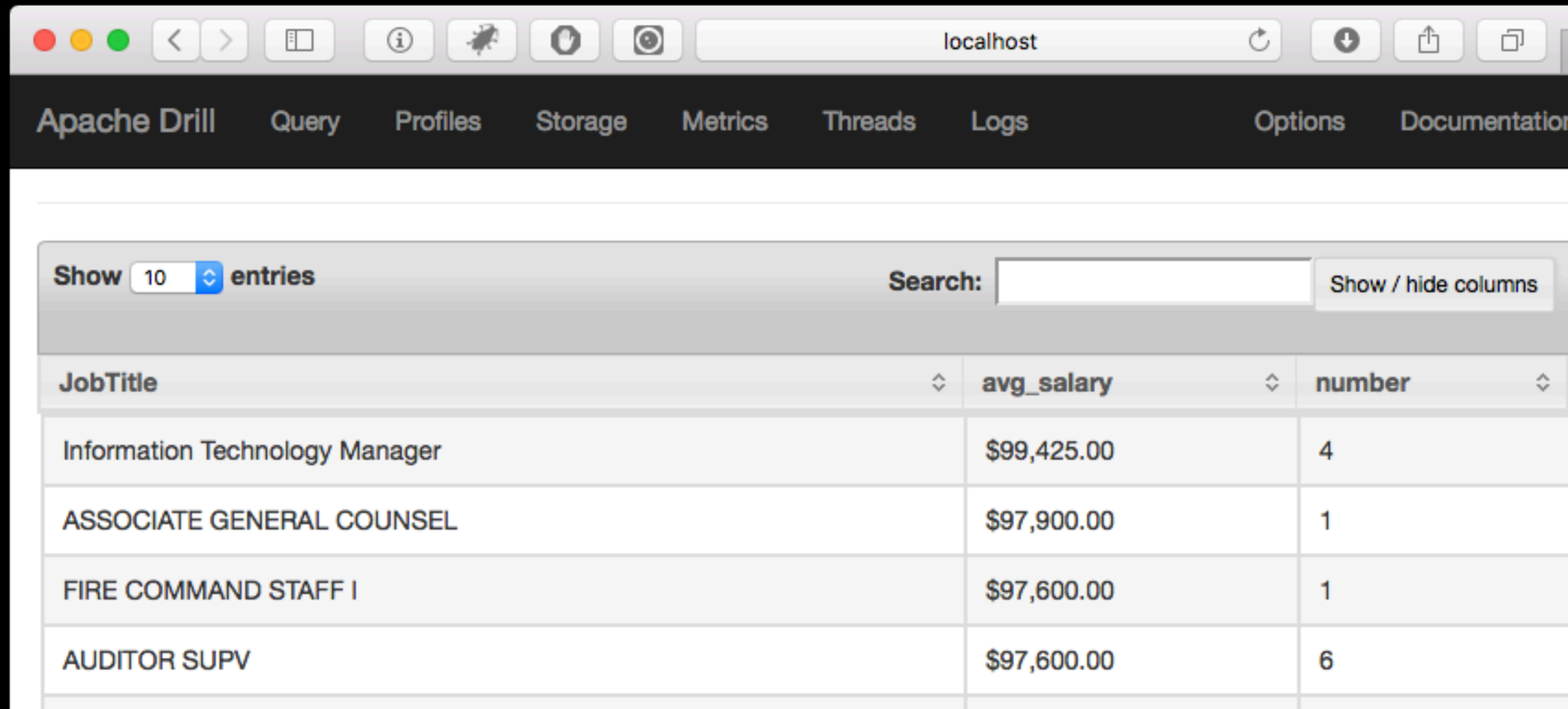
Working with Dates & Times

Symbol	Meaning	Presentation	Examples
G	era	text	AD
C	century of era (>=0)	number	20
Y	year of era (>=0)	year	1996
x	weekyear	year	1996
w	week of weekyear	number	27
e	day of week	number	2
E	day of week	text	Tuesday; Tue
y	year	year	1996
D	day of year	number	189
M	month of year	month	July; Jul; 07
d	day of month	number	10
a	halfday of day	text	PM
K	hour of halfday (0~11)	number	0
h	clockhour of halfday (1~12)	number	12
H	hour of day (0~23)	number	0
k	clockhour of day (1~24)	number	24
m	minute of hour	number	30
s	second of minute	number	55
S	fraction of second	number	978
z	time zone	text	Pacific Standard Time; PST
Z	time zone offset/id	zone	-0800; -08:00; America/Los_Angeles
	escape for text	delimiter	
'	single quote	literal	

TO_CHAR(<field>, <format>)

TO_CHAR(<field>, <format>)

```
SELECT JobTitle,
TO_CHAR( AVG( TO_NUMBER( AnnualSalary, '¤' )), '¤#,###.00' ) AS avg_salary,
COUNT( DISTINCT name ) AS number
FROM dfs.drillworkshop.`csv/baltimore_salaries_2015.csvh`
GROUP BY JobTitle
Order By avg_salary DESC
```



Intervals

```
SELECT date2,
date5,
(TO_DATE( date2, 'MM/dd/yyyy' ) - TO_DATE( date5, 'yyyy-MM-dd' )) as date_diff
FROM dfs.drillworkshop.`csv/dates.csvh
```

date_diff
P249D
P-5D
P-312D
P-315D
P-171D

Intervals

P249D

- P (Period) marks the beginning of a period of time.
- Y follows a number of years.
- M follows a number of months.
- D follows a number of days.
- H follows a number of hours 0-24.
- M follows a number of minutes.
- S follows a number of seconds and optional milliseconds

Intervals

```
SELECT date2,
date5,
 (TO_DATE( date2, 'MM/dd/yyyy' ) - TO_DATE( date5, 'yyyy-MM-dd
)) as date_diff,
EXTRACT( day FROM (TO_DATE( date2, 'MM/dd/yyyy' ) -
TO_DATE( date5, 'yyyy-MM-dd' )))
FROM dfs.drillworkshop.`csv/dates.csvh`
```

date_diff	EXPR$3
P249D	249
P-5D	-5
P-312D	-312
P-315D	-315

Other Date/Time Functions

- AGE(timestamp):
- EXTRACT(field FROM time_exp): Extract a part of a date, time or interval
- CURRENT_DATE()/CURRENT_TIME()/NOW()
- DATE_ADD()/DATE_SUB(): Adds or subtracts two dates

For complete documentation: http://drill.apache.org/docs/date-time-functions-and-arithmetic/

In Class Exercise:
Parsing Dates and Times

In this exercise you will find a data file called dates.csvh which contains 5 columns of random dates in various formats:

- **date1** is in ISO 8601 format
- **date2** is MM/DD/YYYY ie: 03/12/2016
- **date3** is: Sep 19, 2016
- **date4** is formatted: Sun, 19 Mar 2017 00:15:28 -0700
- **date5** is formatted like database dates: YYYY-mm-dd: 2016-10-03

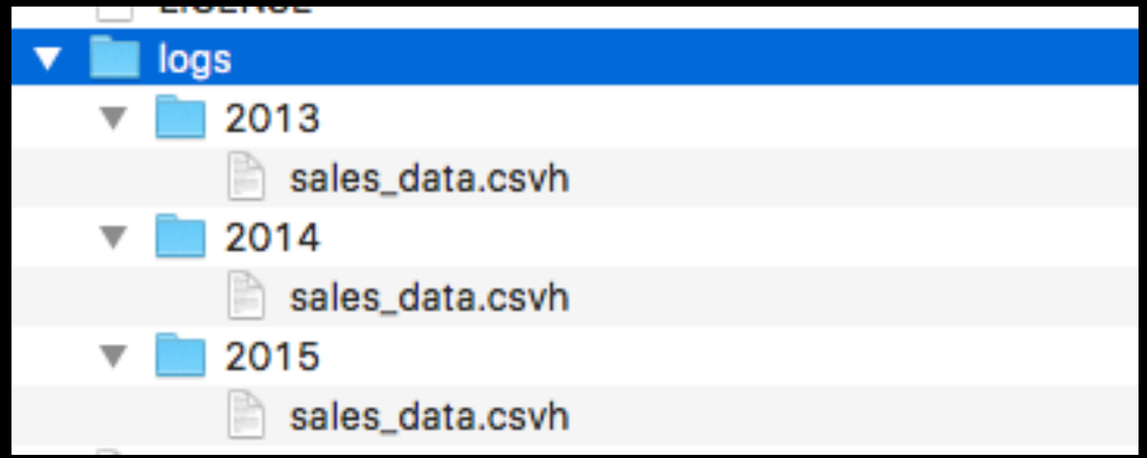
For this exercise, complete the following steps:

1. Using the various methods, (**CAST()**, **TO_DATE()**) we have discussed, convert each column into a date (or time) as appropriate.
2. Reformat **date5** so that it is in the same format as **date3**.
3. Find all the dates rows where **date3** occurs after **date5**.
4. Create a histogram table of **date2** by weekday: IE: Sunday 5, Monday 4, etc
5. Find all the entries in **date5** that are **more than 1 year old**

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

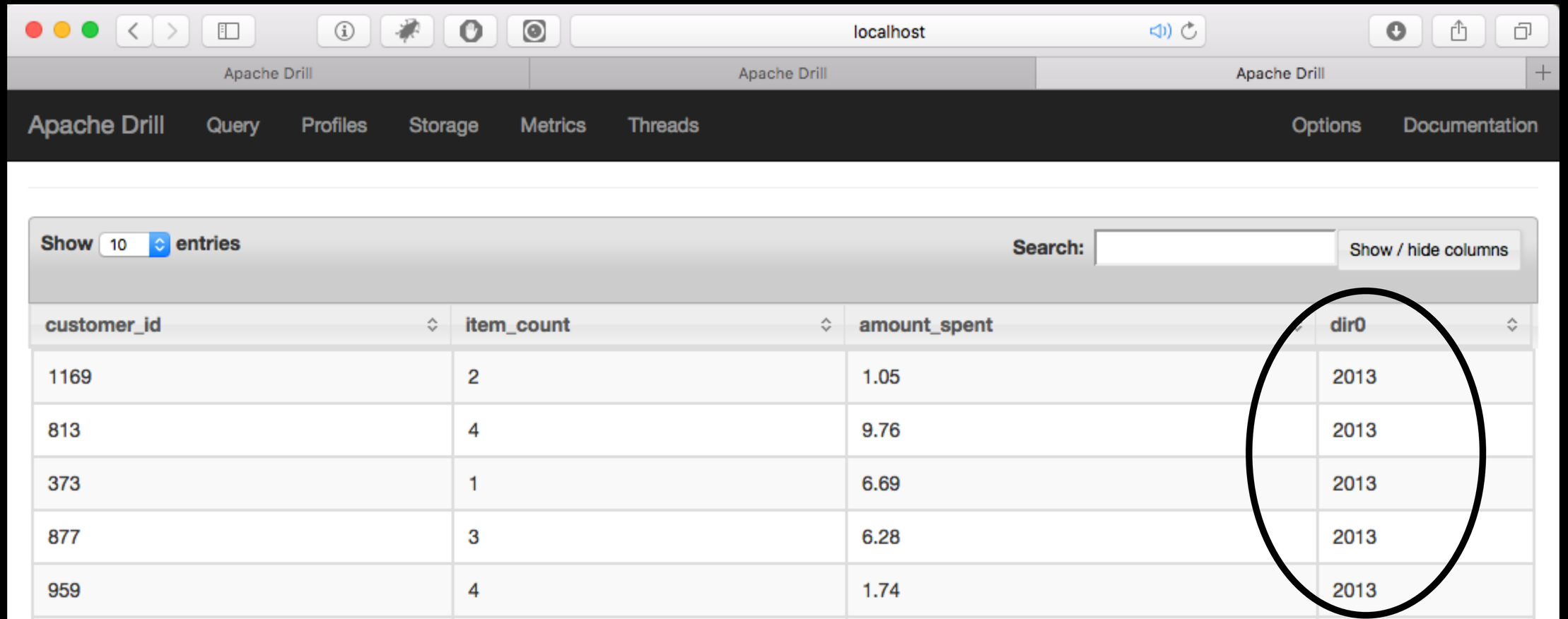
Problem: You have multiple log files which you would like to analyze

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



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

```
SELECT *
FROM dfs.drillworkshop.`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

```
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.drillworkshop.`logs/`
GROUP BY dir0
```

Let’s look at JSON data

Let’s look at JSON data

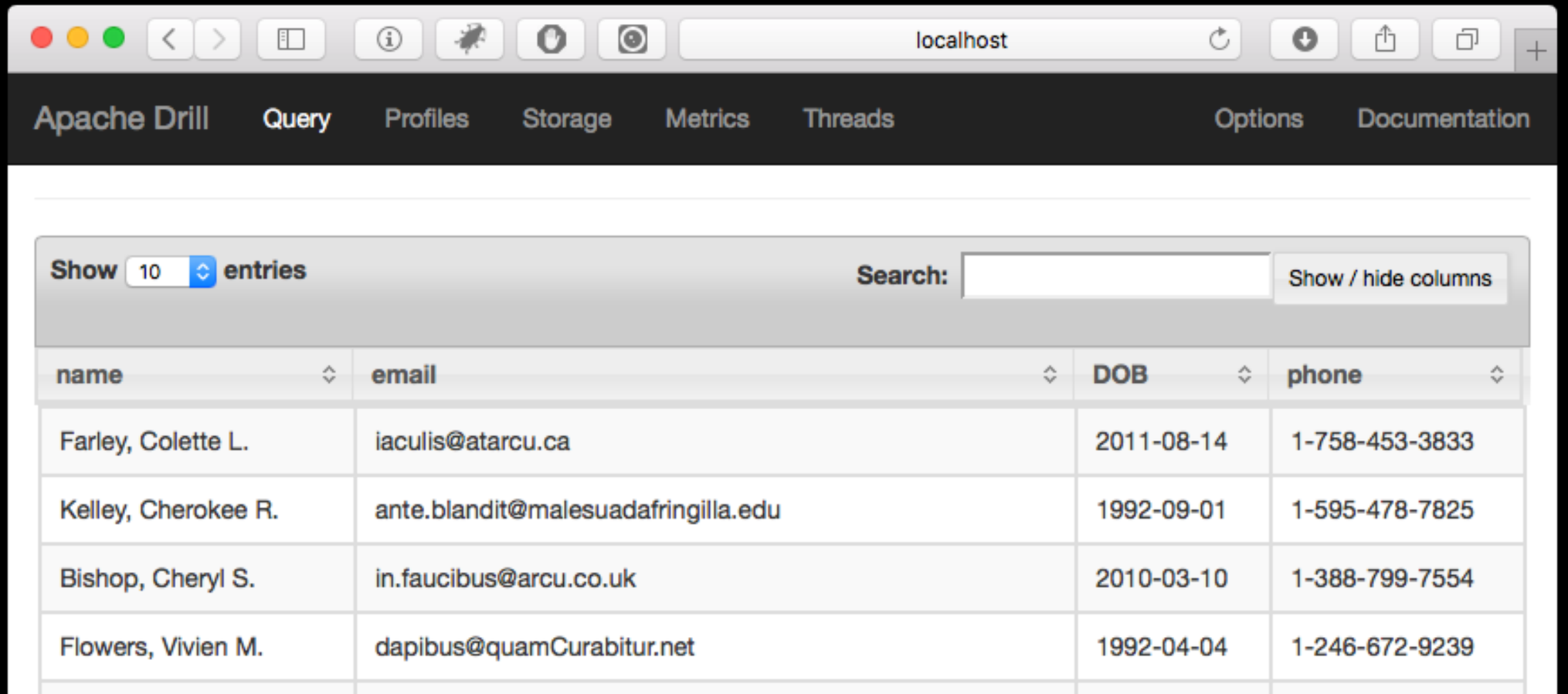
```
[
   {
      "name": "Farley, Colette L.",
      "email": "iaculis@atarcu.ca",
      "DOB": "2011-08-14",
      "phone": "1-758-453-3833"
   },
   {
      "name": "Kelley, Cherokee R.",
      "email": "ante.blandit@malesuadafringilla.edu",
      "DOB": "1992-09-01",
      "phone": "1-595-478-7825"
   }
…
]
```

Let's look at JSON data

```
SELECT *
FROM dfs.drillworkshop.`json/customers.json`
```

Let's look at JSON data

```
SELECT *
FROM dfs.drillworkshop.`json/customers.json`
```



Let's look at JSON data

```
SELECT *
FROM dfs.drillworkshop.`json/customers.json`
```

What about nested data?

Please open
`baltimore_salaries.json`
in a text editor

```
{
  "meta" : {
    "view" : {
      "id" : "nsfe-bg53",
      "name" : "Baltimore City Employee Salaries FY2015",
      "attribution" : "Mayor's Office",
      "averageRating" : 0,
      "category" : "City Government",
      …
      "       "format" : { }
      },
},
  "data" : [ [ 1, "66020CF9-8449-4464-AE61-B2292C7A0F2D", 1, 1438255843, "393202",
1438255843, "393202", null, "Aaron,Patricia G", "Facilities/Office Services II",
"A03031", "OED-Employment Dev (031)", "1979-10-24T00:00:00", "55314.00", "53626.04" ]
, [ 2, "31C7A2FE-60E6-4219-890B-AFF01C09EC65", 2, 1438255843, "393202", 1438255843,
"393202", null, "Aaron,Petra L", "ASSISTANT STATE'S ATTORNEY", "A29045", "States
Attorneys Office (045)", "2006-09-25T00:00:00", "74000.00", "73000.08" ]
```

```
{
  "meta" : {
    "view" : {
      "id" : "nsfe-bg53",
      "name" : "Baltimore City Employee Salaries FY2015",
      "attribution" : "Mayor's Office",
      "averageRating" : 0,
      "category" : "City Government",
      …
      "       "format" : { }
      },
},
  "data" : [ [ 1, "66020CF9-8449-4464-AE61-B2292C7A0F2D", 1, 1438255843, "393202",
1438255843, "393202", null, "Aaron,Patricia G", "Facilities/Office Services II",
"A03031", "OED-Employment Dev (031)", "1979-10-24T00:00:00", "55314.00", "53626.04" ]
, [ 2, "31C7A2FE-60E6-4219-890B-AFF01C09EC65", 2, 1438255843, "393202", 1438255843,
"393202", null, "Aaron,Petra L", "ASSISTANT STATE'S ATTORNEY", "A29045", "States
Attorneys Office (045)", "2006-09-25T00:00:00", "74000.00", "73000.08" ]
```

```
{
  "meta" : {
    "view" : {
      "id" : "nsfe-bg53",
      "name" : "Baltimore City Employee Salaries FY2015",
      "attribution" : "Mayor's Office",
      "averageRating" : 0,
      "category" : "City Government",
      …
      "       "format" : { }
      },
},
  "data" : [ [ 1, "66020CF9-8449-4464-AE61-B2292C7A0F2D", 1,
1438255843, "393202", 1438255843, "393202", null,
"Aaron,Patricia G", "Facilities/Office Services II", "A03031",
"OED-Employment Dev (031)", "1979-10-24T00:00:00", "55314.00",
"53626.04" ]
, [ 2, "31C7A2FE-60E6-4219-890B-AFF01C09EC65", 2, 1438255843,
"393202", 1438255843, "393202", null, "Aaron,Petra L",
"ASSISTANT STATE'S ATTORNEY", "A29045", "States Attorneys
Office (045)", "2006-09-25T00:00:00", "74000.00", "73000.08" ]
```

```
"data" : [
  [ 1,
  "66020CF9-8449-4464-AE61-B2292C7A0F2D",
  1,
  1438255843,
  "393202",
  1438255843,
  “393202",
   null,
  "Aaron,Patricia G",
  "Facilities/Office Services II",
  "A03031",
  "OED-Employment Dev (031)",
  "1979-10-24T00:00:00",
  “55314.00",
   “53626.04"
   ]
```

Drill has a series of functions for nested data

Please run

```
ALTER SYSTEM SET `store.json.all_text_mode` = true;
```

in Drill

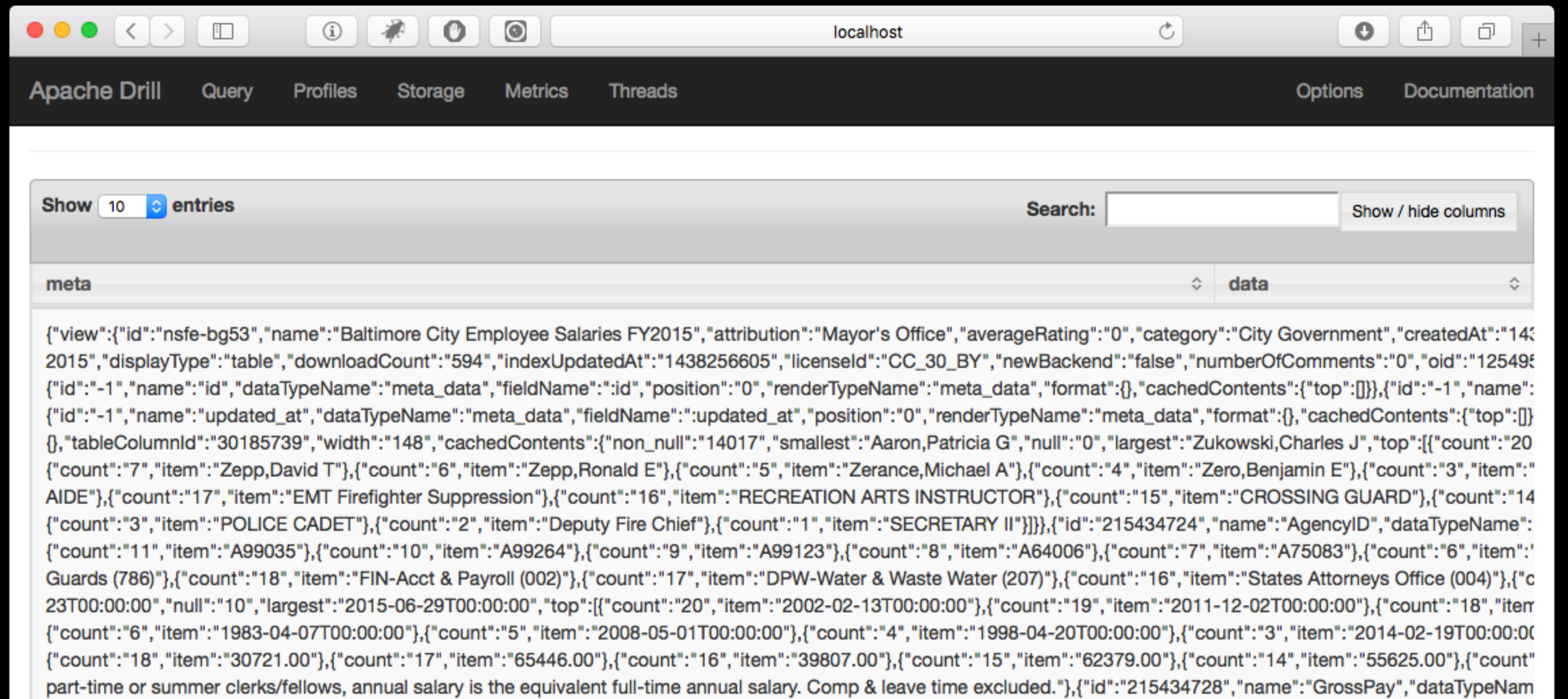
Let’s look at this data in Drill

Let's look at this data in Drill

```
SELECT *
FROM dfs.drillworkshop.`baltimore_salaries.json`
```

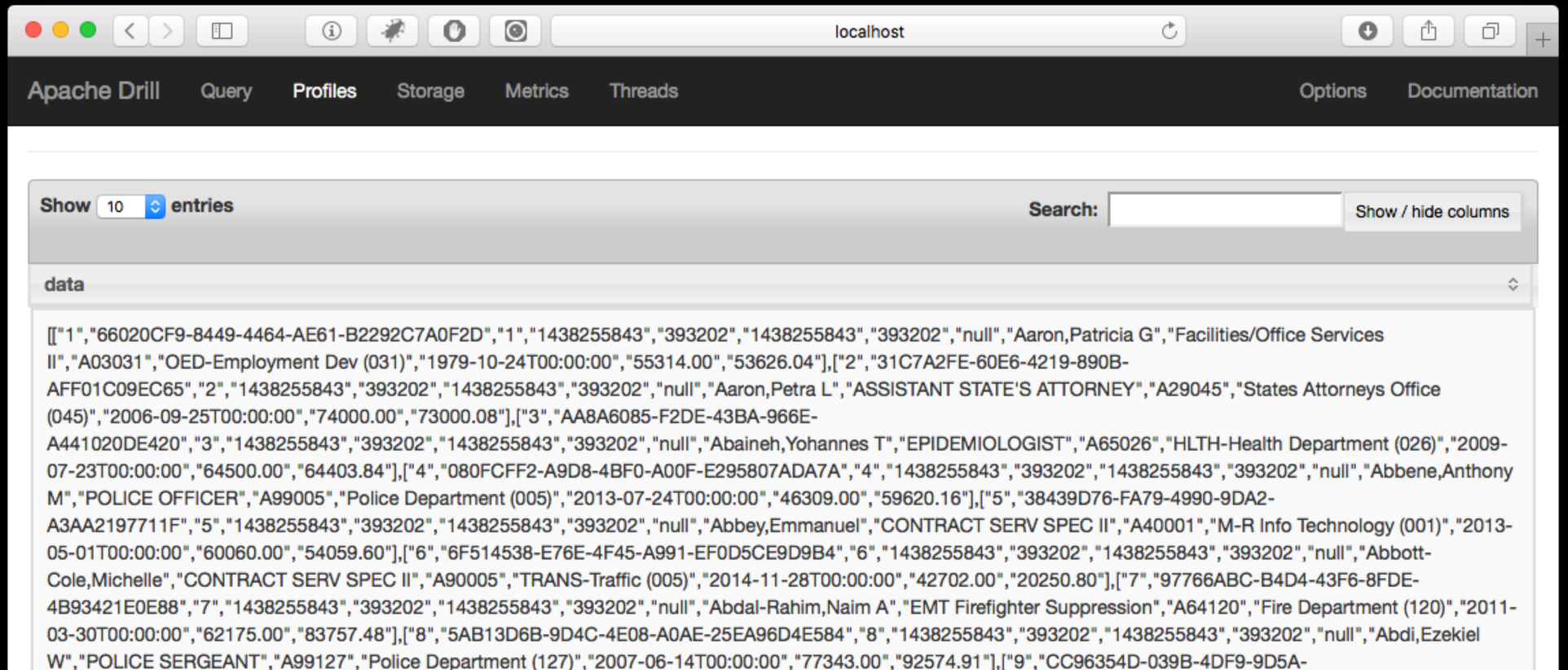
Let's look at this data in Drill

```
SELECT *
FROM dfs.drillworkshop.`baltimore_salaries.json`
```



Let's look at this data in Drill

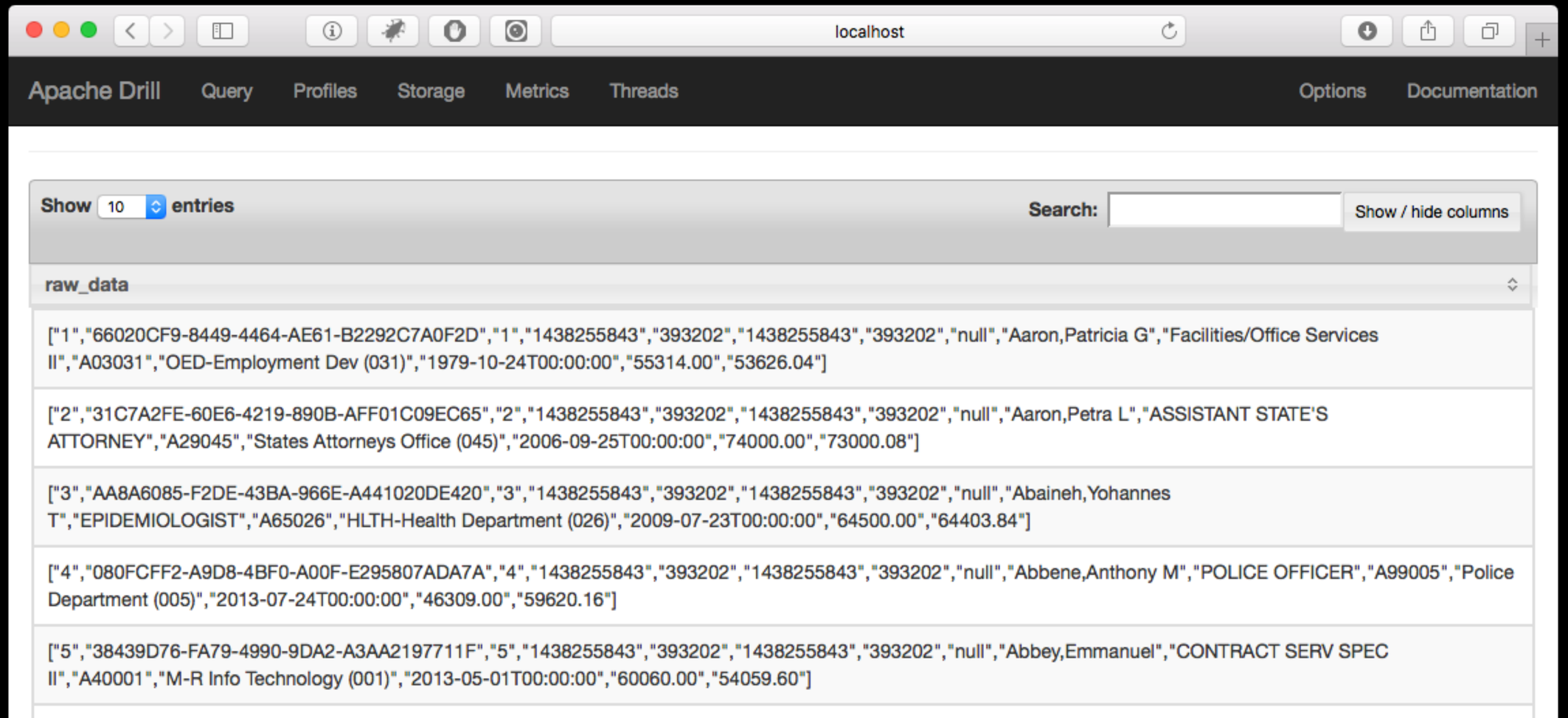
```
SELECT data
FROM dfs.drillworkshop.`baltimore_salaries.json`
```



`FLATTEN( <json array> )`
separates elements in a repeated field into individual records.

```
SELECT FLATTEN( data ) AS raw_data
FROM dfs.drillworkshop.`baltimore_salaries.json`
```

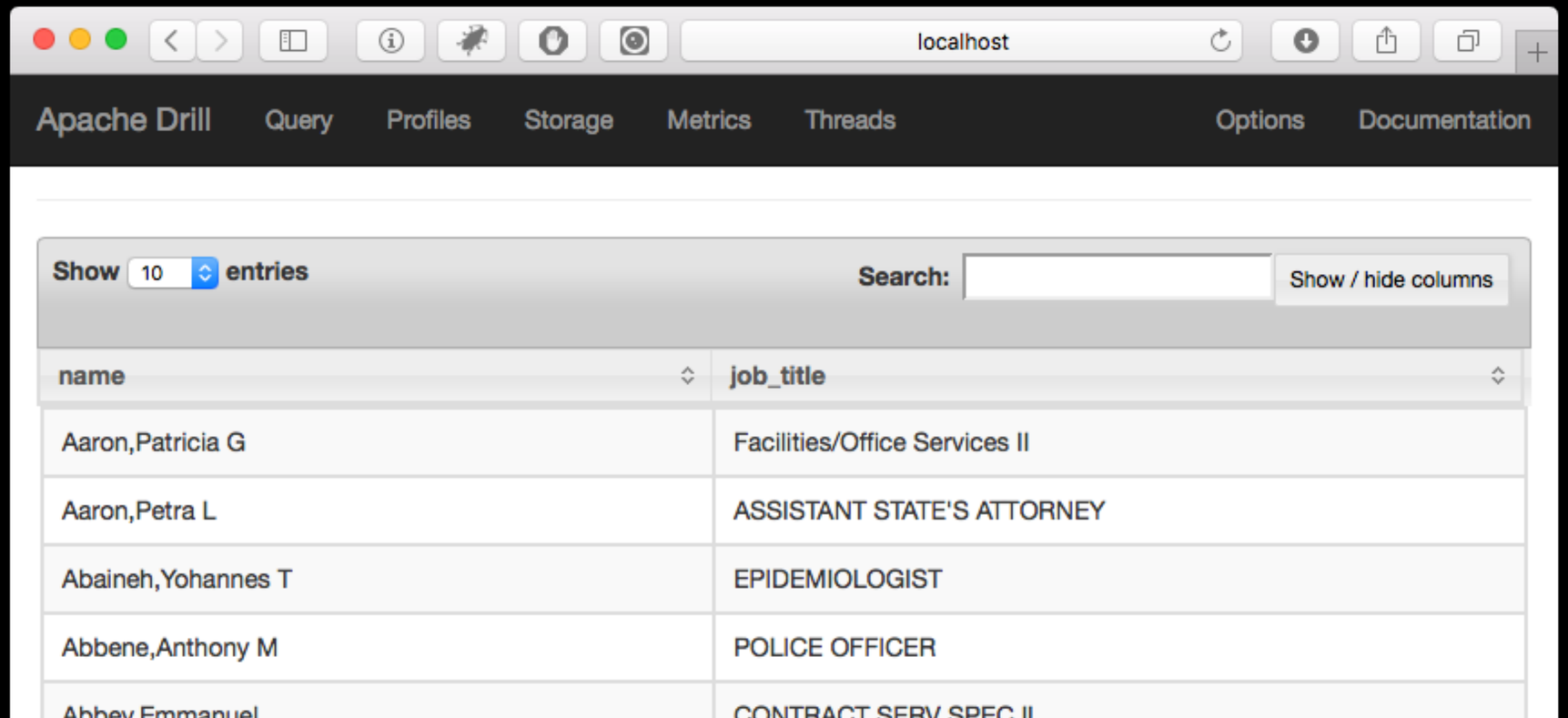
```
SELECT FLATTEN( data ) AS raw_data
FROM dfs.drillworkshop.`baltimore_salaries.json`
```



```
SELECT FLATTEN( data ) AS raw_data
FROM dfs.drillworkshop.`baltimore_salaries.json`
```

```
SELECT raw_data[8] AS name …
FROM
(
SELECT FLATTEN( data ) AS raw_data
FROM dfs.drillworkshop.`baltimore_salaries.json`
)
```

```
SELECT raw_data[8] AS name, raw_data[9] AS job_title
FROM
(
SELECT FLATTEN( data ) AS raw_data
FROM dfs.drillworkshop.`baltimore_salaries.json`
)
```



In Class Exercise

Using the JSON file, recreate the earlier query to find the average salary by job title and how many people have each job title.

HINT: Don't forget to CAST() the columns...

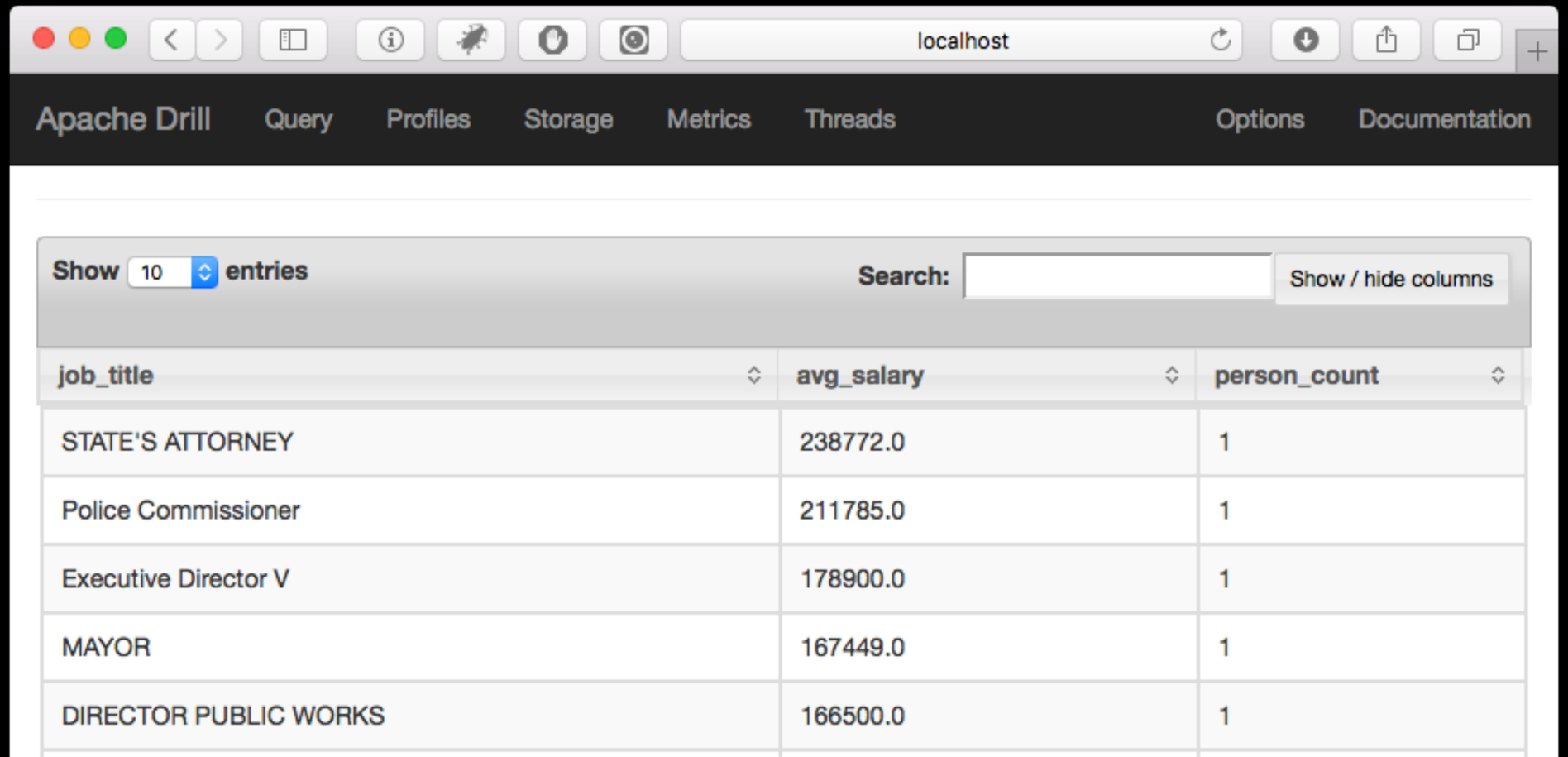
HINT 2: GROUP BY does NOT support aliases.

In Class Exercise

Using the JSON file, recreate the earlier query to find the average salary by job title and how many people have each job title.

```
SELECT raw_data[9] AS job_title,
AVG( CAST( raw_data[13] AS DOUBLE ) ) AS avg_salary,
COUNT( DISTINCT raw_data[8] ) AS person_count
FROM
(
   SELECT FLATTEN( data ) AS raw_data
   FROM dfs.drillworkshop.`json/baltimore_salaries.json`
)
GROUP BY raw_data[9]
ORDER BY avg_salary DESC
```

Using the JSON file, recreate the earlier query to find the average salary by job title and how many people have each job title.

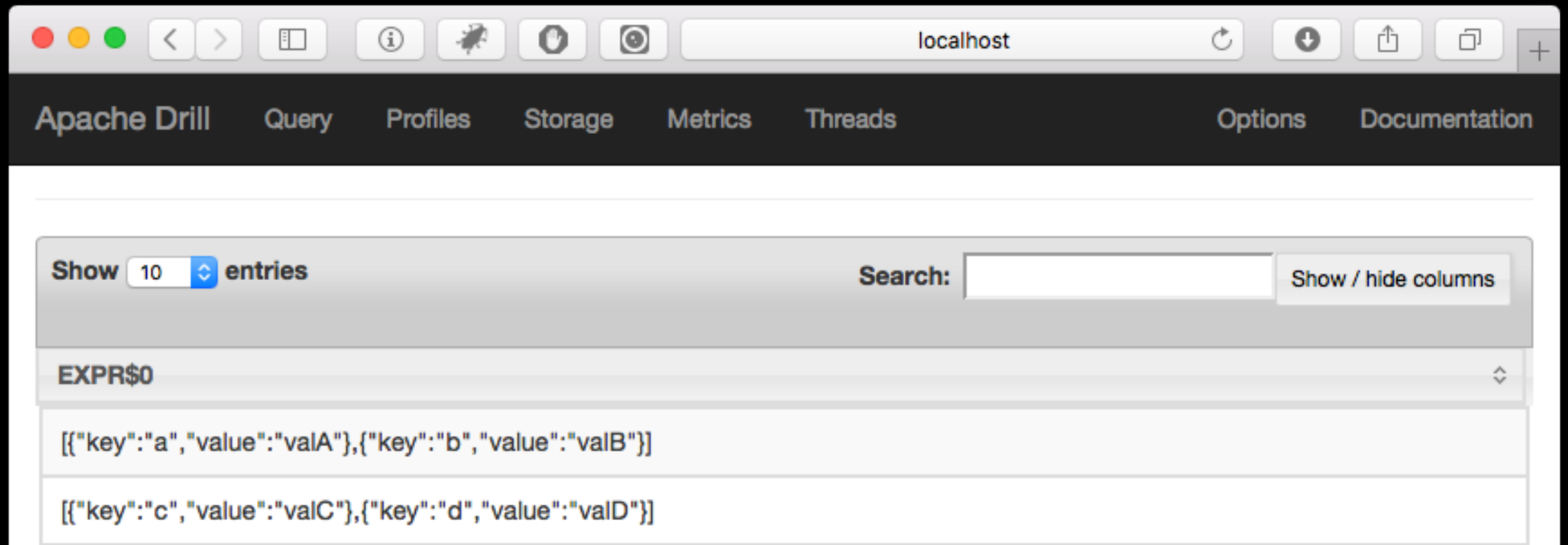


KVGEN(<map>) returns a list of keys and values in a map

```
{"rec1":{"a": "valA", "b": "valB"}}
{"rec1":{"c": "valC", "d": "valD"}}
```

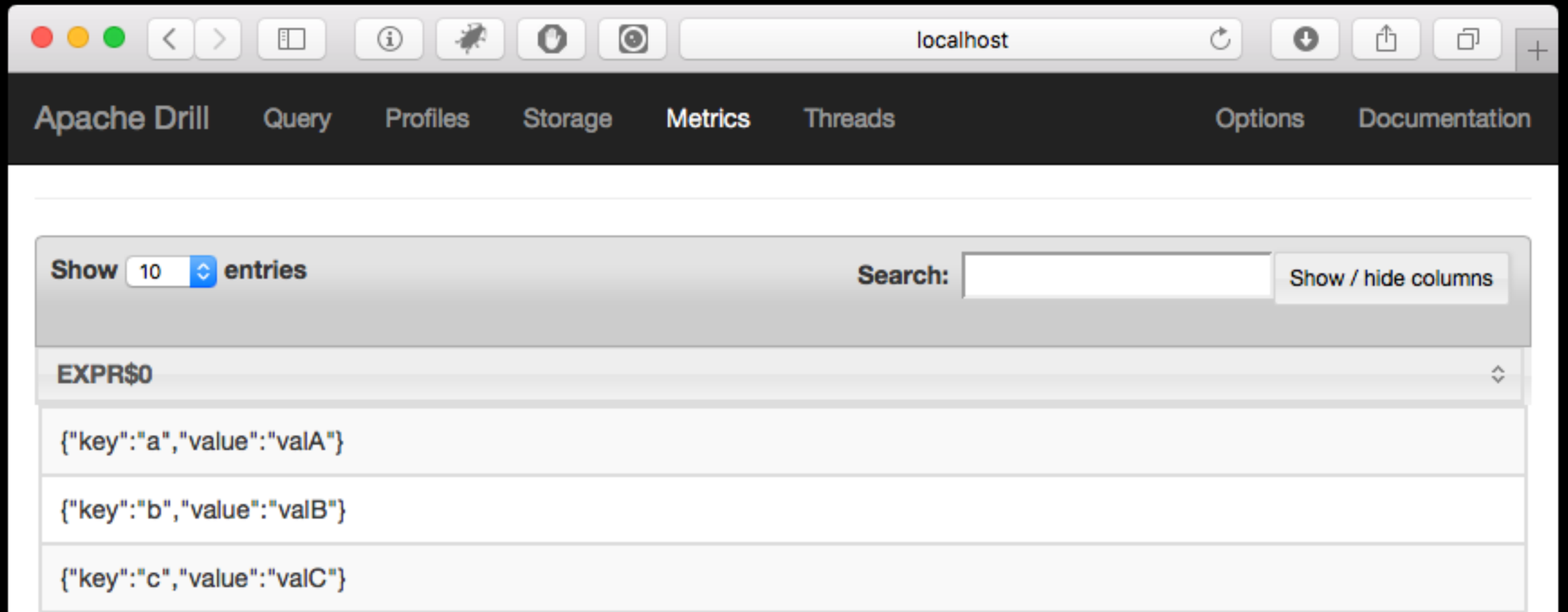
```
{"rec1":{"a": "valA", "b": "valB"}}
{"rec1":{"c": "valC", "d": "valD"}}
```

```
SELECT KVGEN( rec1 ) FROM dfs.drillworkshop.`json/simple.json`
```



```
{"rec1":{"a": "valA", "b": "valB"}}
{"rec1":{"c": "valC", "d": "valD"}}
```

```
SELECT FLATTEN( KVGEN( rec1 ) )
FROM dfs.drillworkshop.`json/simple.json`
```



Networking Functions

Networking Functions

- inet_aton(<ip>): Converts an IPv4 Address to an integer
- inet_ntoa(<int>): Converts an integer to an IPv4 address
- is_private(<ip>): Returns true if the IP is private
- in_network(<ip>,<cidr>): Returns true if the IP is in the CIDR block
- getAddressCount(<cidr>): Returns the number of IPs in a CIDR block
- getBroadcastAddress(<cidr>): Returns the broadcast address of a CIDR block
- getNetmast(<cidr>): Returns the net mask of a CIDR block
- getLowAddress(<cidr>): Returns the low IP of a CIDR block
- getHighAddress(<cidr>): Returns the high IP of a CIDR block
- parse_user_agent(<ua_string>): Returns a map of user agent information

A brief interlude… maps

A brief interlude… maps

```
SELECT parse_user_agent( columns[0] ) AS ua
FROM dfs.drillworkshop.`csv/user-agents.csv`
```

Documentation for this function is available at: https://github.com/cgivre/drill-useragent-function

A brief interlude… maps

```
SELECT parse_user_agent( columns[0] ) AS ua
FROM dfs.drillworkshop.`csv/user-agents.csv`
```

ua
{"DeviceClass":"Desktop","DeviceName":"Desktop","DeviceBrand":"Unknown","OperatingSystemClass":"Desktop","OperatingSystemName":"Windows NT","OperatingSystemVersion":"Windows XP","OperatingSystemNameVersion":"Windows XP","LayoutEngineClass":"Browser","LayoutEngineName":"Gecko","LayoutEngineVersion":"35.0","LayoutEngineVersionMajor":"35","LayoutEngineNameVersion":"Gecko 35.0","LayoutEngineNameVersionMajor":"Gecko 35","LayoutEngineBuild":"20100101","AgentClass":"Browser","AgentName":"Firefox","AgentVersion":"35.0","AgentVersionMajor":"35","AgentNameVersion":"Firefox 35.0","AgentNameVersionMajor":"Firefox 35"}
{"DeviceClass":"Desktop","DeviceName":"Desktop","DeviceBrand":"Unknown","OperatingSystemClass":"Desktop","OperatingSystemName":"Windows NT","OperatingSystemVersion":"Windows XP","OperatingSystemNameVersion":"Windows XP","LayoutEngineClass":"Browser","LayoutEngineName":"Gecko","LayoutEngineVersion":"35.0","LayoutEngineVersionMajor":"35","LayoutEngineNameVersion":"Gecko 35.0","LayoutEngineNameVersionMajor":"Gecko 35","LayoutEngineBuild":"20100101","AgentClass":"Browser","AgentName":"Firefox","AgentVersion":"35.0","AgentVersionMajor":"35","AgentNameVersion":"Firefox 35.0","AgentNameVersionMajor":"Firefox 35"}

A brief interlude… maps

```
SELECT uadata.ua.OperatingSystemName AS OS_Name
FROM (
    SELECT parse_user_agent( columns[0] ) AS ua
    FROM dfs.drillworkshop.`csv/user-agents.csv`
) AS uadata
```

table.map.key

In Class Exercise:

The file user-agents.csv is a small sample of a list of user agents gathered from a server log during an attempted attack. Using this data, answer the following questions:

1. What was the most common OS?
2. What was the most common browser?

```
SELECT uadata.ua.AgentNameVersion AS Browser,
COUNT( * ) AS BrowserCount
FROM (
   SELECT parse_user_agent( columns[0] ) AS ua
   FROM dfs.drillworkshop.`csv/user-agents.csv`
) AS uadata
GROUP BY uadata.ua.AgentNameVersion
ORDER BY BrowserCount DESC
```

Log Files

Log Files

- Drill does not natively support reading log files… yet
- If you are NOT using Merlin, included in the GitHub repo are several .jar files. Please take a second and copy them to <drill directory>/jars/3rdparty

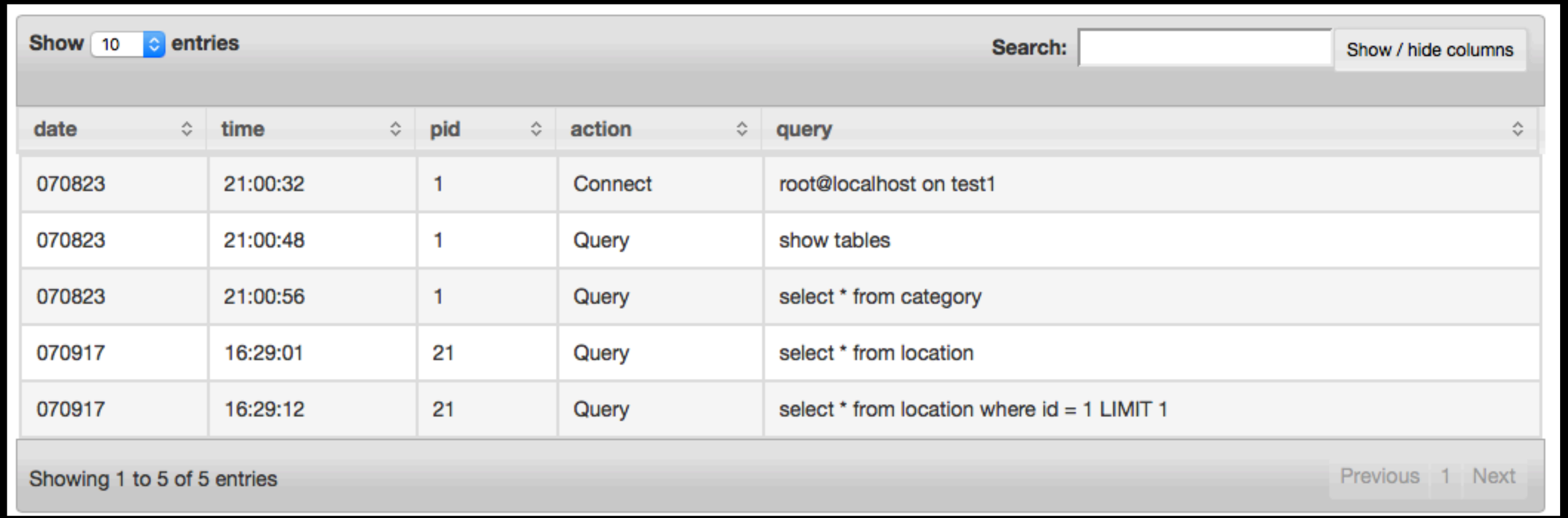
Log Files

```
070823 21:00:32       1 Connect     root@localhost on test1
070823 21:00:48       1 Query       show tables
070823 21:00:56       1 Query       select * from category
070917 16:29:01      21 Query       select * from location
070917 16:29:12      21 Query       select * from location where id = 1 LIMIT 1
```

```
log": {
      "type": "log",
      "extensions": [
        "log"
      ],
      "fieldNames": [
        "date",
        "time",
        "pid",
        "action",
        "query"
      ],
      "pattern": "(\\d{6})\\s(\\d{2}:\\d{2}:\\d{2})\\s+(\\d+)\\s(\\w+)\\s+(.+)"
    }
  }
```

```
SELECT *
FROM dfs.drillworkshop.`log_files/mysql.log`
```

```
SELECT *
FROM dfs.drillworkshop.`log_files/mysql.log`
```



Show 10 entries Search: Show / hide columns

date	time	pid	action	query
070823	21:00:32	1	Connect	root@localhost on test1
070823	21:00:48	1	Query	show tables
070823	21:00:56	1	Query	select * from category
070917	16:29:01	21	Query	select * from location
070917	16:29:12	21	Query	select * from location where id = 1 LIMIT 1

Showing 1 to 5 of 5 entries Previous 1 Next

In Class Exercise

There is a file in the repo called ‘firewall.log’ which contains entries in the following format:

```
Dec 12 03:36:23    sshd[41875]: Failed password for root from 222.189.239.10 port 1350 ssh2
Dec 12 03:36:22    sshd[41875]: Failed password for root from 222.189.239.10 port 1350 ssh2
Dec 12 03:36:22    sshlockout[15383]: Locking out 222.189.239.10 after 15 invalid attempts
Dec 12 03:36:22    sshd[41875]: Failed password for root from 222.189.239.10 port 1350 ssh2
Dec 12 03:36:22    sshlockout[15383]: Locking out 222.189.239.10 after 15 invalid attempts
Dec 12 03:36:22    sshd[42419]: Failed password for root from 222.189.239.10 port 2646 ssh2
```

In this exercise:

1. Write a regex to extract the date, process type, PID, from IP and any other information you believe may be useful from this log

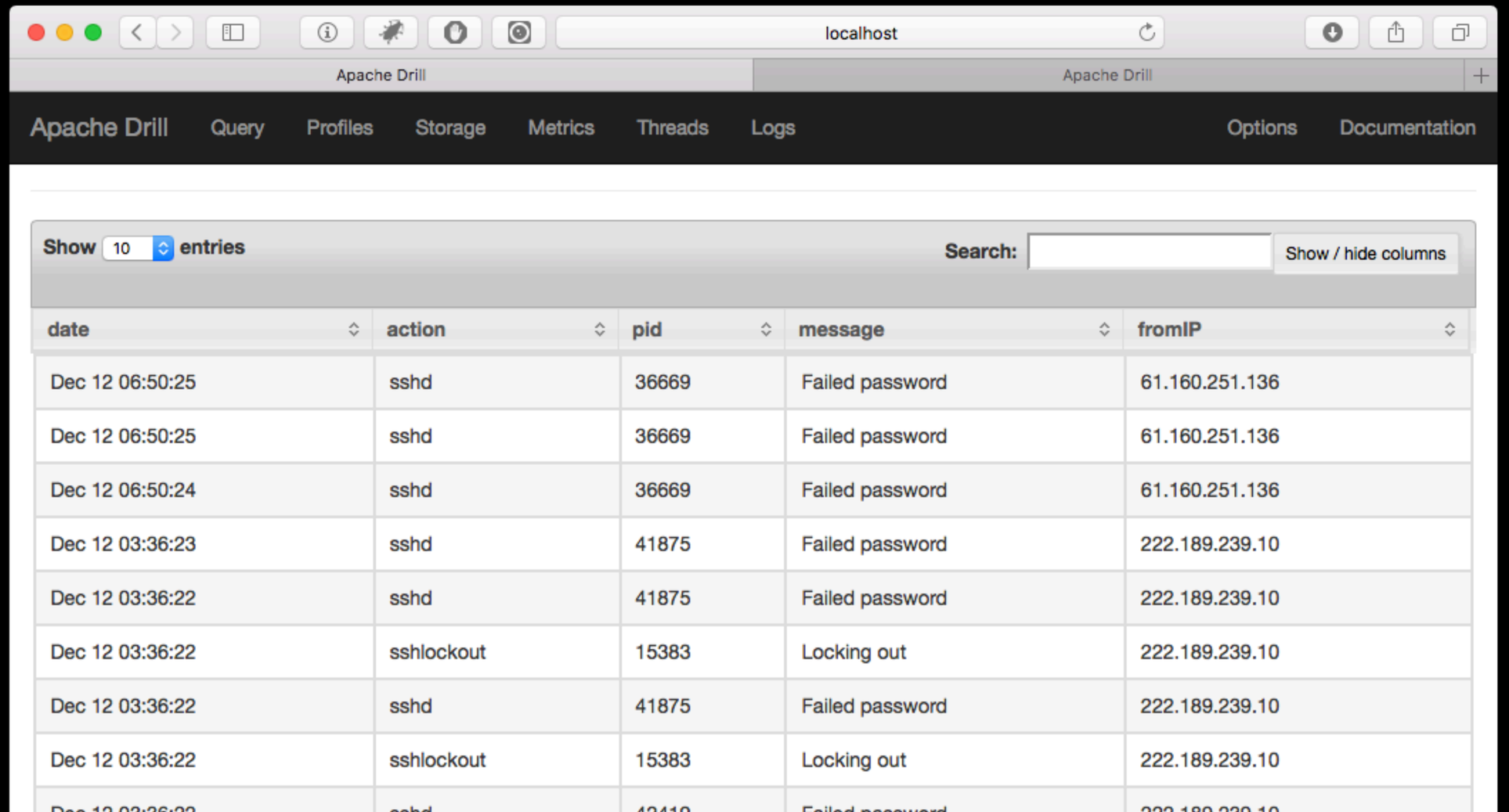
2. Use that regex to configure Drill to query this data.

3. Find all the records where the IP is in the CIDR block: 61.160.251.128/28

In Class Exercise

```
"ssdlog": {
      "type": "log",
      "extensions": [
        "ssdlog"
      ],
      "fieldNames": [
        "date",
        "action",
        "pid",
        "message",
        "fromIP"
      ],
      "pattern": "(\\w{3}\\s\\d{2}\\s\\d{2}:\\d{2}:\\d{2})\\s+(\\w+)\\[(\\d+)\\]:\\s(\\w+\\s\\w+).+?(\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3})"
    }
```

In Class Exercise



HTTPD Log Files

HTTPD Log Files

195.154.46.135 - - [25/Oct/2015:04:11:25 +0100] "GET /linux/doing-pxe-without-dhcp-control HTTP/1.1" 200 24323 "http://howto.basjes.nl/" "Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0"
23.95.237.180 - - [25/Oct/2015:04:11:26 +0100] "GET /join_form HTTP/1.0" 200 11114 "http://howto.basjes.nl/" "Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0"
23.95.237.180 - - [25/Oct/2015:04:11:27 +0100] "POST /join_form HTTP/1.1" 302 9093 "http://howto.basjes.nl/join_form" "Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0"
158.222.5.157 - - [25/Oct/2015:04:24:31 +0100] "GET /join_form HTTP/1.0" 200 11114 "http://howto.basjes.nl/" "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0 AlexaToolbar/alxf-2.21"
158.222.5.157 - - [25/Oct/2015:04:24:32 +0100] "POST /join_form HTTP/1.1" 302 9093 "http://howto.basjes.nl/join_form" "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0 AlexaToolbar/alxf-2.21"

For “documentation”: https://issues.apache.org/jira/browse/DRILL-3423

HTTPD Log Files

195.154.46.135 - - [25/Oct/2015:04:11:25 +0100] "GET /linux/doing-pxe-without-dhcp-control HTTP/1.1" 200 24323 "http://howto.basjes.nl/" "Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0"
23.95.237.180 - - [25/Oct/2015:04:11:26 +0100] "GET /join_form HTTP/1.0" 200 11114 "http://howto.basjes.nl/" "Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0"
23.95.237.180 - - [25/Oct/2015:04:11:27 +0100] "POST /join_form HTTP/1.1" 302 9093 "http://howto.basjes.nl/join_form" "Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20100101 Firefox/35.0"
158.222.5.157 - - [25/Oct/2015:04:24:31 +0100] "GET /join_form HTTP/1.0" 200 11114 "http://howto.basjes.nl/" "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0 AlexaToolbar/alxf-2.21"
158.222.5.157 - - [25/Oct/2015:04:24:32 +0100] "POST /join_form HTTP/1.1" 302 9093 "http://howto.basjes.nl/join_form" "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0 AlexaToolbar/alxf-2.21"

```
"httpd": {
      "type": "httpd",
      "logFormat": "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\"",
      "timestampFormat": null
    },
```

For “documentation”: https://issues.apache.org/jira/browse/DRILL-3423

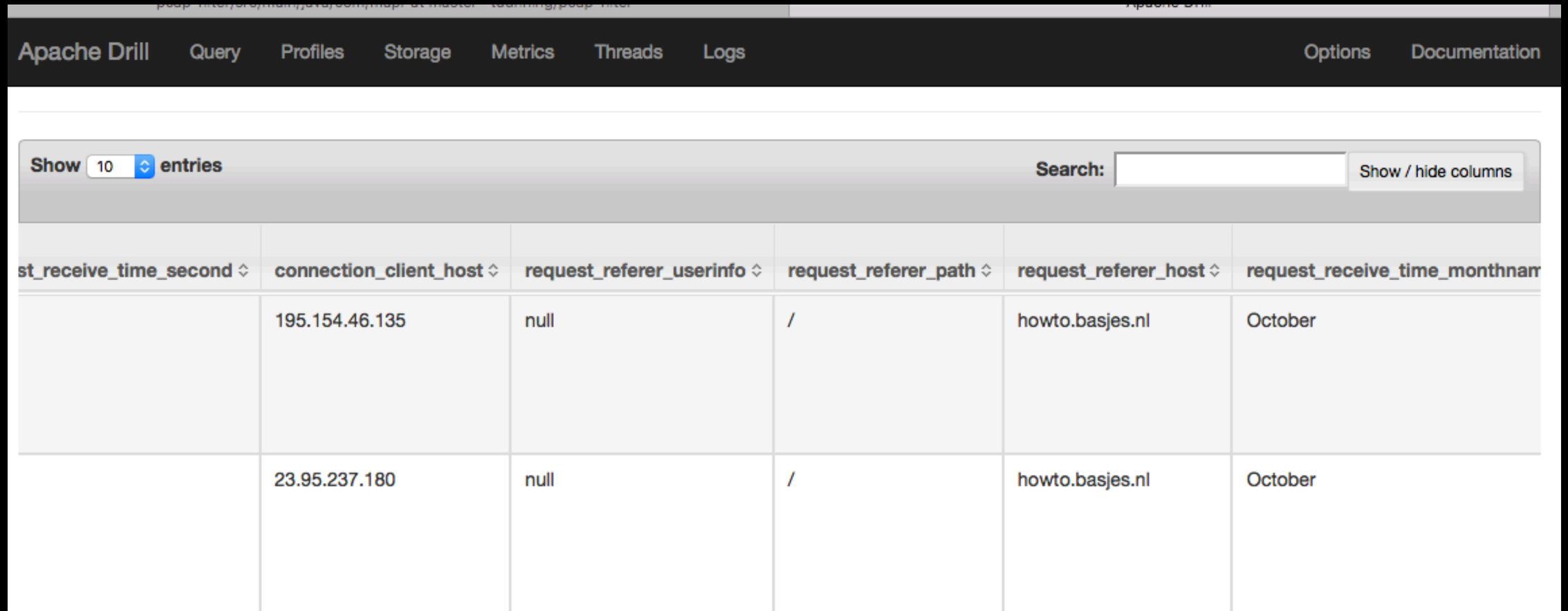
HTTPD Log Files

```
SELECT *
FROM dfs.drillworkshop.`data_files/log_files/small-server-log.httpd`
```

For “documentation”: https://issues.apache.org/jira/browse/DRILL-3423

HTTPD Log Files

```
SELECT *
FROM dfs.drillworkshop.`data_files/log_files/small-server-log.httpd`
```



HTTPD Log Files

```
SELECT request_referer, parse_url( request_referer ) AS url_data
FROM dfs.drillworkshop.`data_files/log_files/small-server-log.httpd`
```

HTTPD Log Files

```
SELECT request_referer, parse_url( request_referer ) AS url_data
FROM dfs.drillworkshop.`data_files/log_files/small-server-log.httpd`
```

Show 10 entries Search:

request_referer	url_data
http://howto.basjes.nl/	{"protocol":"http","authority":"howto.basjes.nl","host":"howto.basjes.nl","path":"/"}
http://howto.basjes.nl/	{"protocol":"http","authority":"howto.basjes.nl","host":"howto.basjes.nl","path":"/"}
http://howto.basjes.nl/join_form	{"protocol":"http","authority":"howto.basjes.nl","host":"howto.basjes.nl","path":"/join_form"}
http://howto.basjes.nl/	{"protocol":"http","authority":"howto.basjes.nl","host":"howto.basjes.nl","path":"/"}
http://howto.basjes.nl/join_form	{"protocol":"http","authority":"howto.basjes.nl","host":"howto.basjes.nl","path":"/join_form"}

In Class Exercise

There is a file in the repo called ‘hackers-access.httpd’ is a HTTPD server log. Write queries to determine:

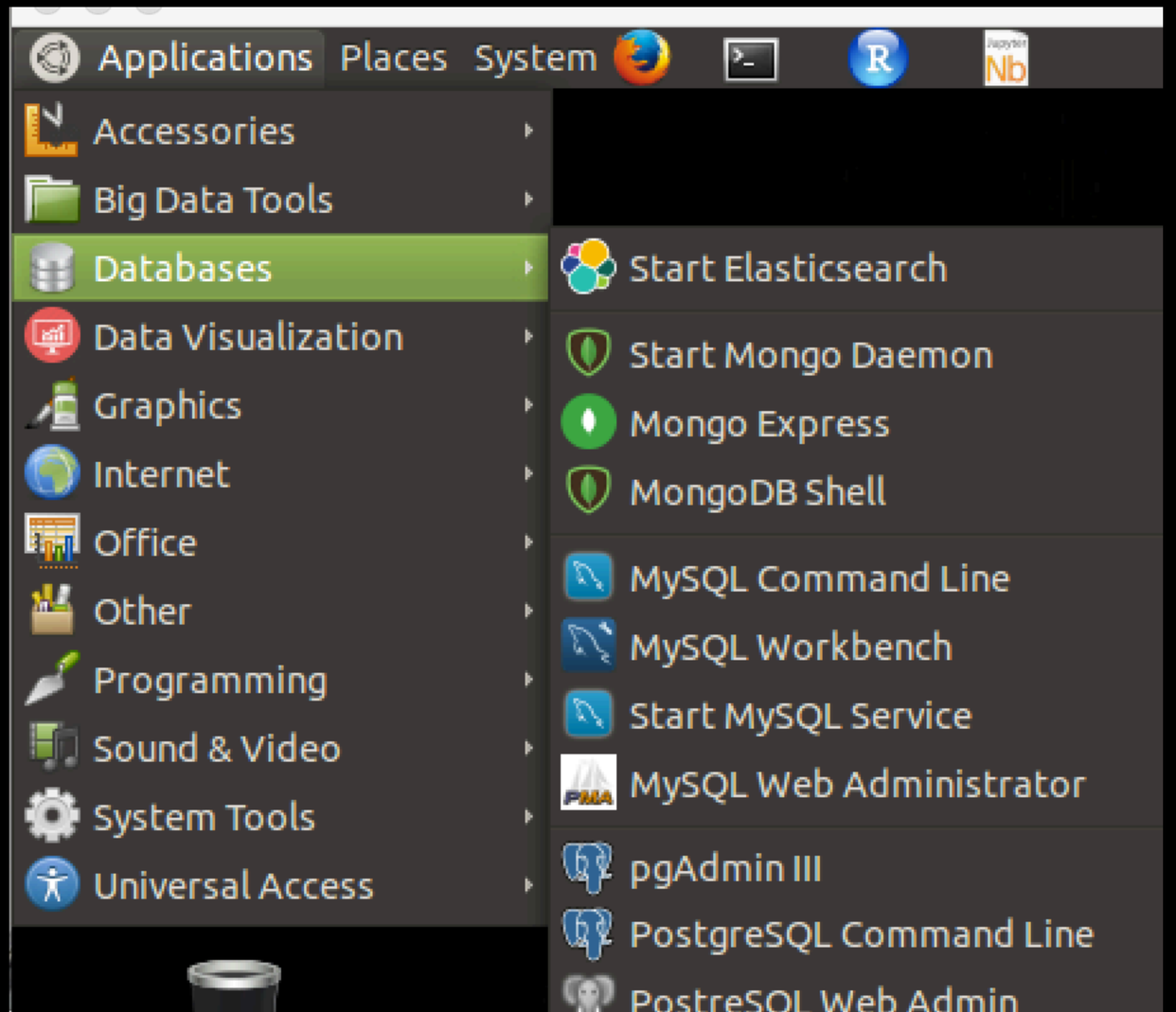
1. What is the most common browser?

2. What is the most common operating system?

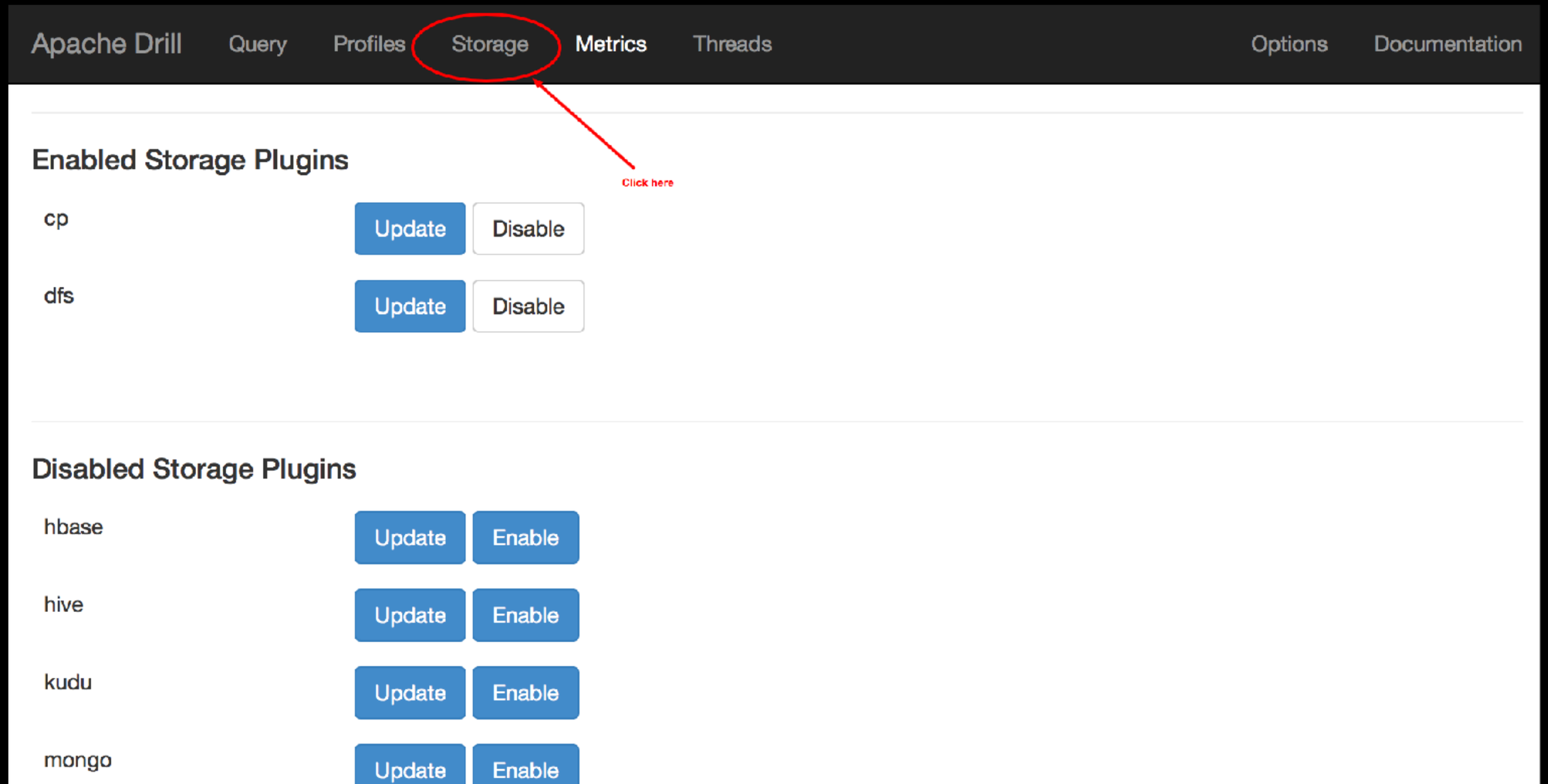
Connecting other Data Sources

Connecting other Data Sources

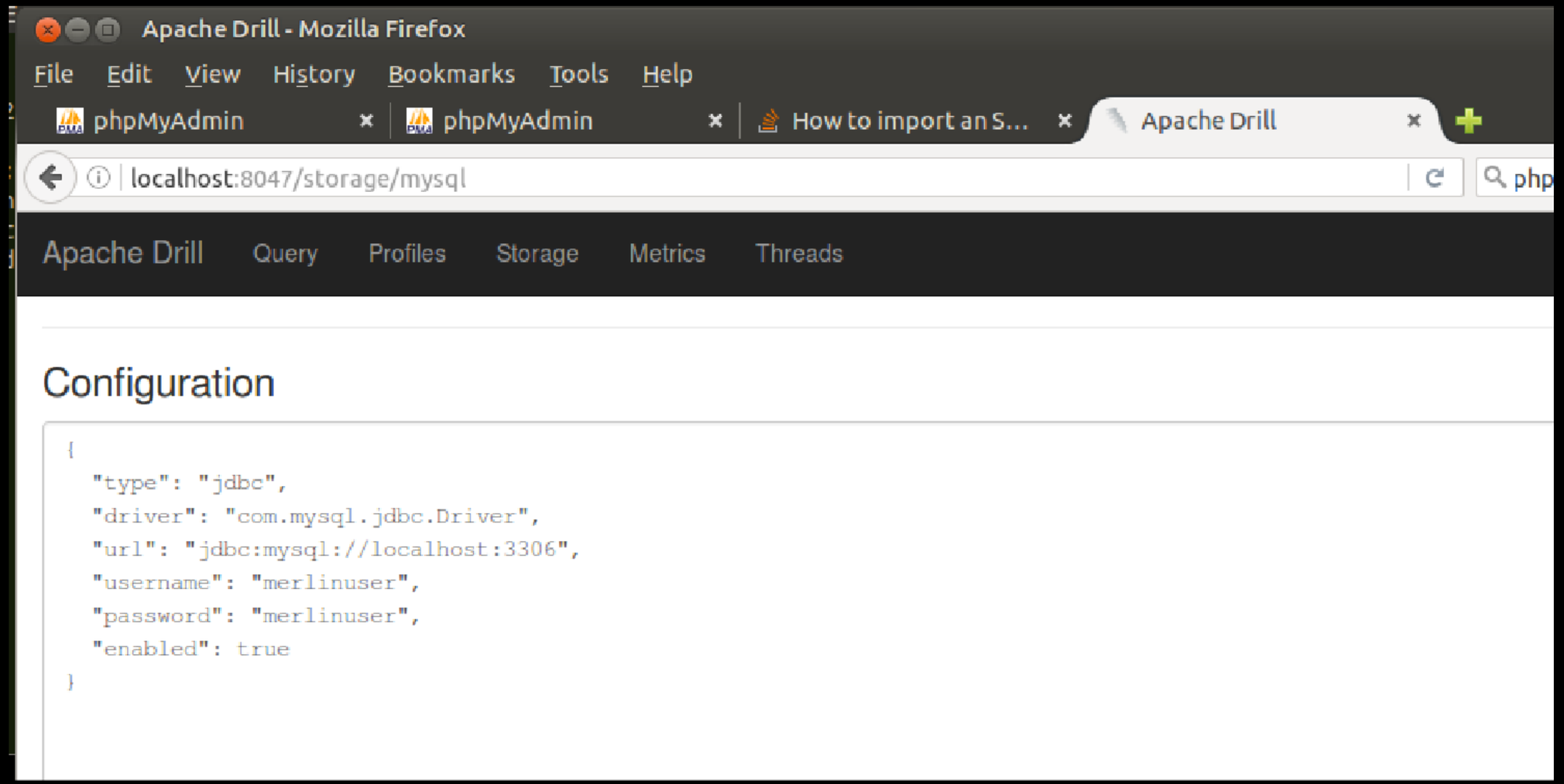
Connecting other Data Sources



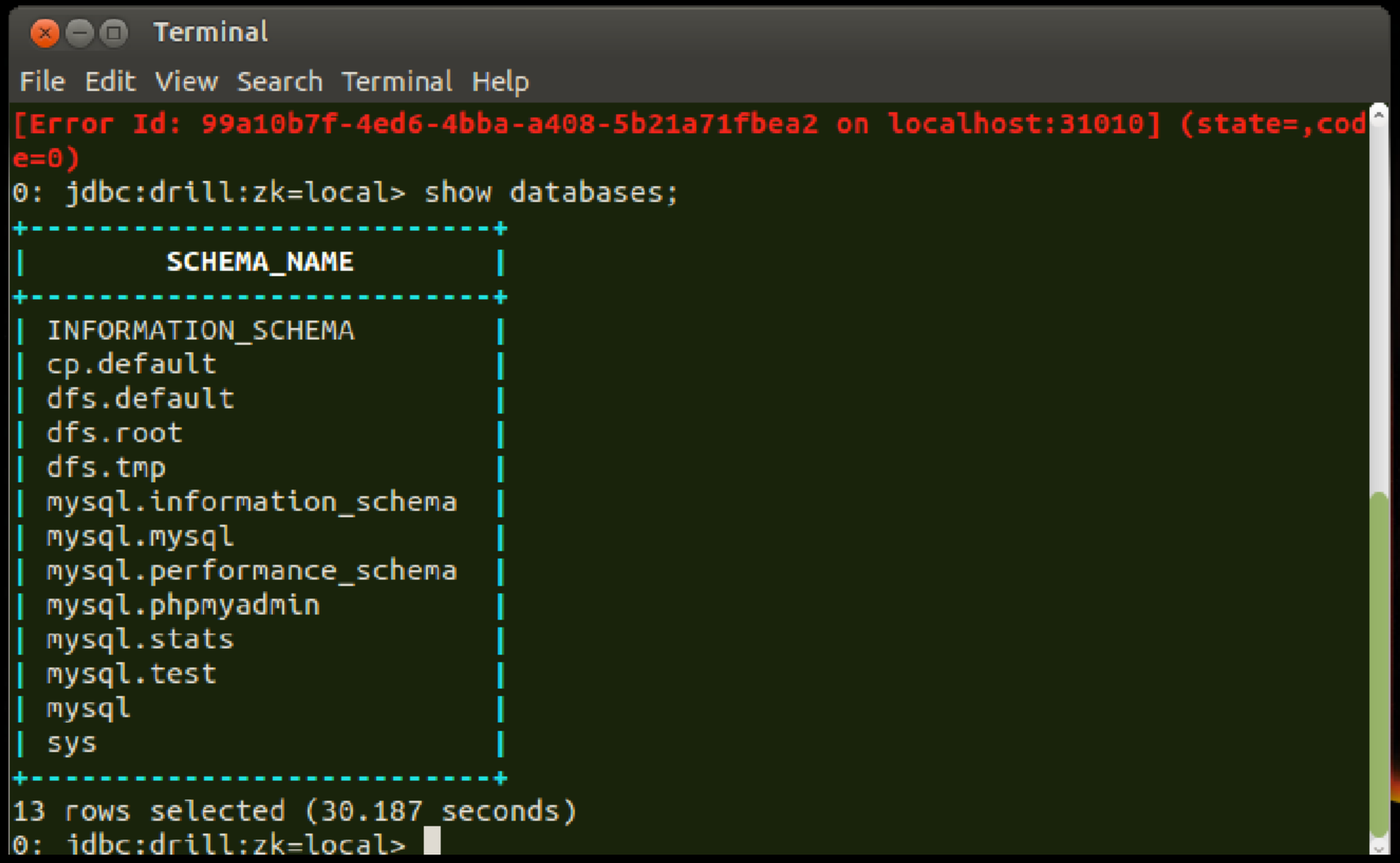
Connecting other Data Sources



Connecting other Data Sources



Connecting other Data Sources



Connecting other Data Sources

```
SELECT teams.name, SUM( batting.HR ) as hr_total
FROM batting
INNER JOIN teams ON batting.teamID=teams.teamID
WHERE batting.yearID = 1988 AND teams.yearID = 1988
GROUP BY batting.teamID
ORDER BY hr_total DESC
```

Connecting other Data Sources

```
SELECT teams.name, SUM( batting.HR ) as hr_total
FROM batting
INNER JOIN teams ON batting.teamID=teams.teamID
WHERE batting.yearID = 1988 AND teams.yearID = 1988
GROUP BY batting.teamID
ORDER BY hr_total DESC
```

Connecting other Data Sources

```
SELECT teams.name, SUM( batting.HR ) as hr_total
FROM batting
INNER JOIN teams ON batting.teamID=teams.teamID
WHERE batting.yearID = 1988 AND teams.yearID = 1988
GROUP BY batting.teamID
ORDER BY hr_total DESC
```

MySQL: 0.047 seconds

Connecting other Data Sources

```
SELECT teams.name, SUM( batting.HR ) as hr_total
FROM mysql.stats.batting
INNER JOIN mysql.stats.teams ON batting.teamID=teams.teamID
WHERE batting.yearID = 1988 AND teams.yearID = 1988
GROUP BY teams.name
ORDER BY hr_total DESC
```

MySQL: 0.047 seconds

Drill: 0.366 seconds

Writing A Drill Function

Writing A Drill Function

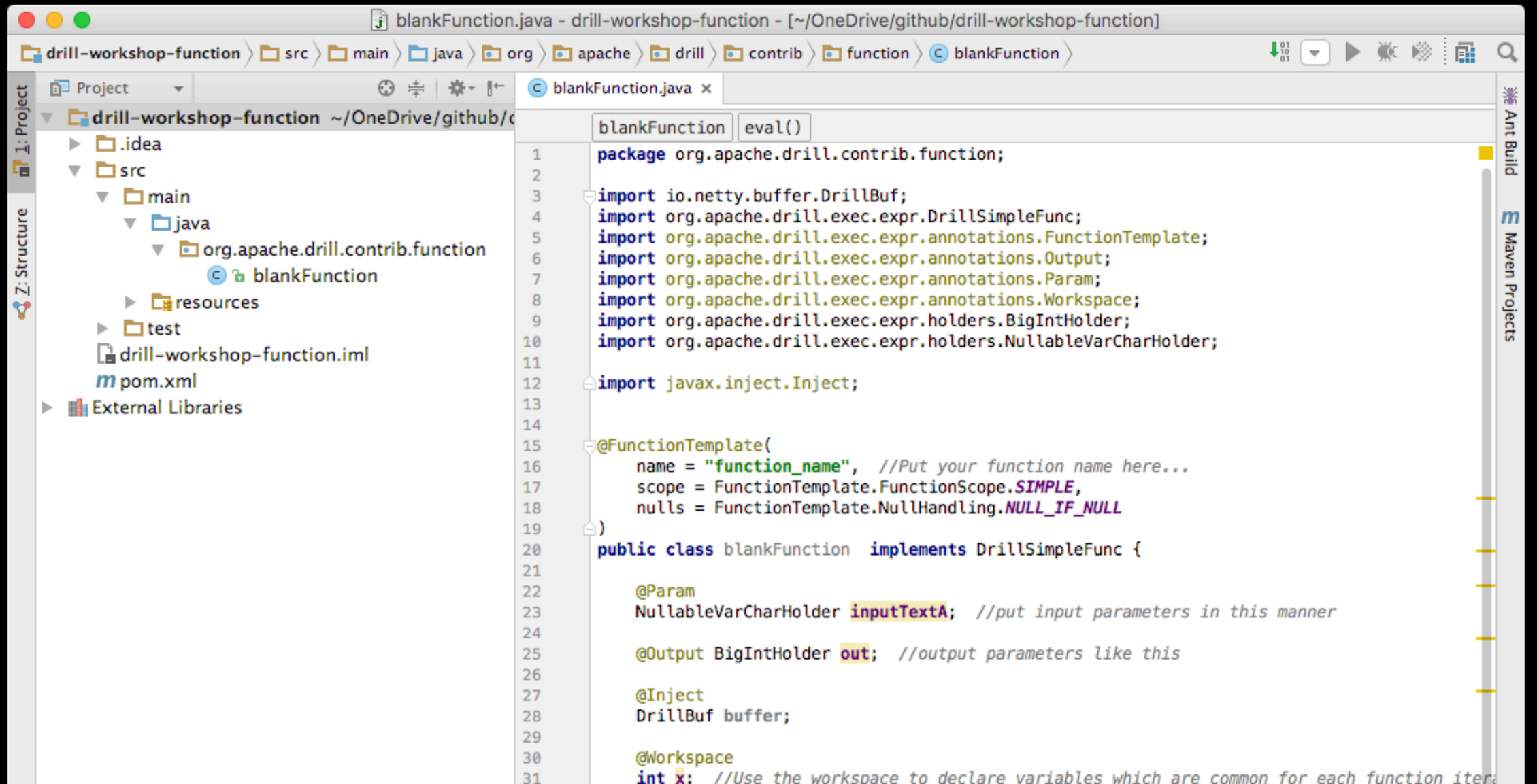
```
function doSomething( arg1, arg2 )
{
    //Something happens…
    return result value
}
```

Writing A Drill Function

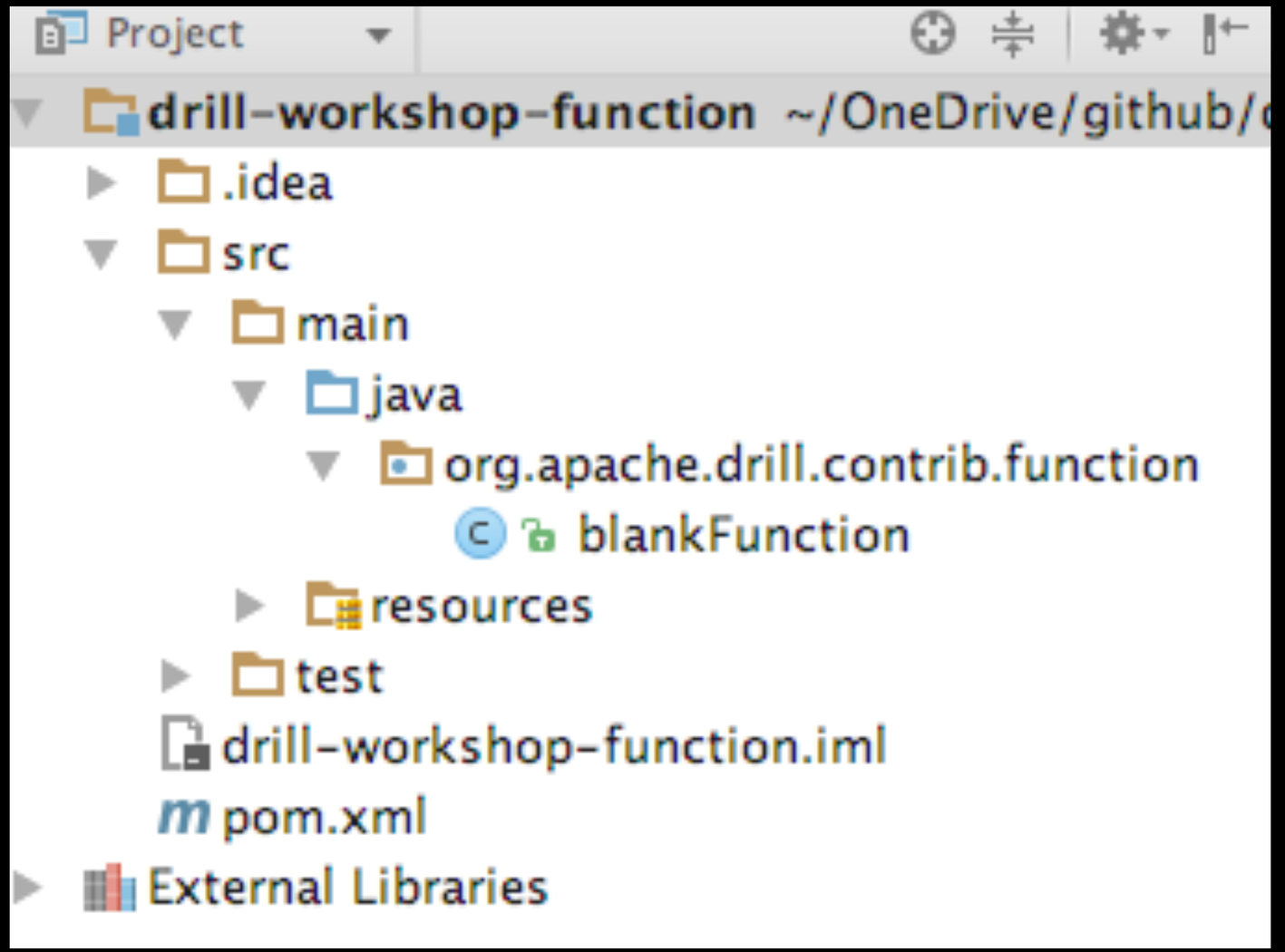
Step 1. In the workshop/udfs folder there is a .zip file called drill-workshop-function.zip, unzip that file.

Step 2. Open this folder using IntelliJ IDEA which can be found in the programming menu

Writing A Drill Function



Writing A Drill Function



Charles' Rule #1 of Drill Functions:

Use an IDE!

Writing A Drill Function

```
@FunctionTemplate(
        name = "<function name>",
        scope = FunctionTemplate.FunctionScope.SIMPLE,
        nulls = FunctionTemplate.NullHandling.NULL_IF_NULL
)
public class <function file name> implements DrillSimpleFunc {


}
```

The function name is the actual function which will be called in a query. The file name **must** match the name in the class declaration.

Writing A Drill Function: Defining Input Parameters

```
@Param
NullableVarCharHolder inputTextA;

@Param
IntHolder someInt;

@Param
Float8Holder someDecimal;

@Param
ComplexHolder someMapOrArray;

import org.apache.drill.exec.expr.holders.XXX;
```

Writing A Drill Function

```
@FunctionTemplate(
        name = "addTwo",
        scope = FunctionTemplate.FunctionScope.SIMPLE,
        nulls = FunctionTemplate.NullHandling.NULL_IF_NULL
)
public class addTwoFunction implements DrillSimpleFunc {

    @Param
    IntHolder someInt;

}
```

Writing A Drill Function: Defining Output Parameters

```
@Output
BigIntHolder out;

@Inject
DrillBuf buffer;
```

Writing A Drill Function

```
@FunctionTemplate(
        name = "addTwo",
        scope = FunctionTemplate.FunctionScope.SIMPLE,
        nulls = FunctionTemplate.NullHandling.NULL_IF_NULL
)
public class addTwoFunction implements DrillSimpleFunc {

   @Param
  IntHolder someInt;

   @Output
   BigIntHolder out;

    @Inject
    DrillBuf buffer;

}
```

Writing A Drill Function: Workspace Parameters (Optional)

```
@Workspace
BigIntHolder temp1;
```

Writing A Drill Function: The function body

```
public void setup() {}

public void eval() {}
```

Writing A Drill Function: Accessing variables

For strings:

```
String s =
org.apache.drill.exec.expr.fn.impl.StringFunctionHelpers.toStringFromUTF8(inputTextA.
start, inputTextA.end, inputTextA.buffer);
```

For Numeric Fields:

```
long x = inputParam1.value;
Double y = inputParam2.value;
```

Writing A Drill Function: The function body

```
public void setup() {}


public void eval() {
    long x = inputParam1.value;
    long result = x + 2;
}
```

Would this work?

```
return result;
```

Writing A Drill Function: The function body

```
public void setup() {}


public void eval() {
    long x = inputParam1.value;
    long result = x + 2;
    out.value = result;
}
```

Writing A Drill Function: Returning values

Any numeric:

```
out.value = <number>
```

Any String

```
String outputValue = <some string>
out.buffer = buffer;
out.start = 0;
out.end = outputValue.getBytes().length;
buffer.setBytes(0, outputValue.getBytes());
```

Writing A Drill Function: Returning values

```
org.apache.drill.exec.vector.complex.writer.BaseWriter.MapWriter queryMapWriter =
outWriter.rootAsMap();

String[] arguments = queryString.split("&");

for (int i = 0; i < arguments.length; i++) {
   String[] queryParts = <some map or array>

   org.apache.drill.exec.expr.holders.VarCharHolder rowHolder = new
     org.apache.drill.exec.expr.holders.VarCharHolder();

  byte[] rowStringBytes = <value>.getBytes();
  outBuffer.reallocIfNeeded(rowStringBytes.length);
            outBuffer.setBytes(0, rowStringBytes);

   rowHolder.start = 0;
   rowHolder.end = rowStringBytes.length;
   rowHolder.buffer = outBuffer;
   queryMapWriter.varChar(<key>).write(rowHolder);

}
```

Writing A Drill Function: Compiling & Installing

```
mvn clean package -DskipTests
```

Next, if all went well, copy the .jar files from the <project>/target folder to: <drill path>/jars/3rdparty and restart Drill.

Now you are ready to try your function in a query!

In Class Exercise

Write a Drill UDF called `digit_to_string()` which takes a single digit as an argument and returns the text version of that integer.

IE:
`digit_to_string( 3 ) = three`

What is most important is understanding how to get variables in and out of the `eval()` function.

Saving Data

Saving Data

Drill supports:

- CSV, TSV, PSV
- Parquet (default)
- JSON

Saving Data

```
ALTER SESSION SET `store.format` = '<format>';
```

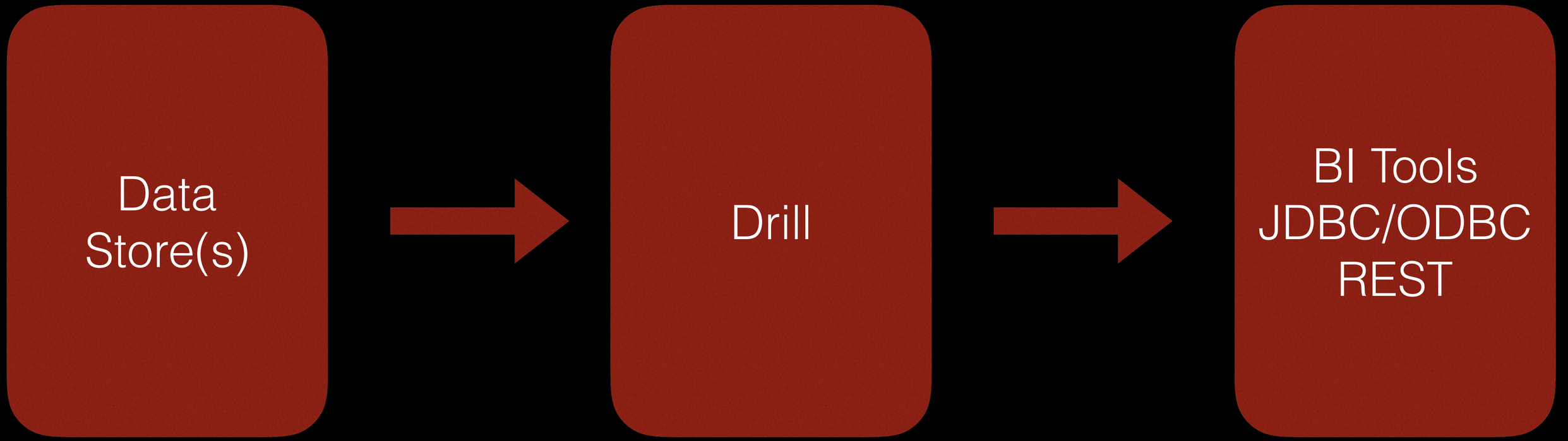
```
CREATE TABLE <file_name> AS <query>
```

```
CREATE TABLE <file_name> AS <query>
```

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

Connecting to Drill

Connecting to Drill



Connecting to Drill

```
pip install pydrill
```

Connecting to Drill

```
from pydrill.client import PyDrill
```

Connecting to Drill

```
drill = PyDrill(host='localhost', port=8047)

if not drill.is_active():
    raise ImproperlyConfigured('Please run Drill first')
```

Connecting to Drill

```
query_result = drill.query('''
 SELECT JobTitle,
  AVG( CAST( LTRIM( AnnualSalary, '$' ) AS FLOAT) ) AS
avg_salary,
COUNT( DISTINCT name ) AS number
FROM dfs.drillworkshop.`*.csvh`
GROUP BY JobTitle
Order By avg_salary DESC
LIMIT 10
''')
```

Connecting to Drill

```
df = query_result.to_dataframe()
```

In Class Exercise

Complete the PyDrill Demonstration Worksheet.

Questions?

Thank you!

Charles Givre
@cgivre
givre_charles@bah.com
thedataist.com