Data Exploration with Apache Drill - Day 2

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

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.

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.

```
SELECT Agency,
AVG( TO_NUMBER( `AnnualSalary`, '¤' ) - TO_NUMBER( `GrossPay`, '¤' )) AS avg_SalaryDelta,
COUNT( DISTINCT `EmpName` ) as emp_count,
MIN( TO_NUMBER( `AnnualSalary`, '¤' ) - TO_NUMBER( `GrossPay`, '¤' ) ) min_salary_delta,
MAX( TO_NUMBER( `AnnualSalary`, '¤' ) - TO_NUMBER( `GrossPay`, '¤' ) ) max_salary_delta
FROM dfs.drillclass.`baltimore_salaries_2016.csvh`
WHERE NOT( GrossPay ='' )
GROUP BY Agency
ORDER BY avg_SalaryDelta DESC
```

Homework

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

```
SELECT data2016.`EmpName`,
data2016.`JobTitle` AS JobTitle_2016,
data2015.`JobTitle` AS JobTitle_2015,
data2016.`AnnualSalary` AS salary_2016,
data2015.`AnnualSalary` AS salary_2015,
(TO_NUMBER( data2016.`AnnualSalary`, '¤' ) -
TO_NUMBER( data2015.`AnnualSalary`, '¤' )) AS salary_delta
FROM dfs.drillclass.`baltimore_salaries_2016.csvh` AS data2016
INNER JOIN dfs.drillclass.`baltimore_salaries_2015.csvh` AS
data2015
ON data2016.`EmpName` = data2015.`EmpName`
ORDER BY salary_delta DESC
LIMIT 10
```

Homework

(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.

```
SELECT `EmpName`,
SPLIT( `EmpName`,',' )[0] AS last_name,
SPLIT( `EmpName`,',' )[1] AS first_name
FROM dfs.drillclass.`baltimore_salaries_2016.csvh`
```

Today, we will cover:

- Dates and Times
- Nested Data (JSON)
- Other data types
- Other data sources
- Programmatically connecting to Drill

Working with Dates & Times

Working with Dates & Times

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

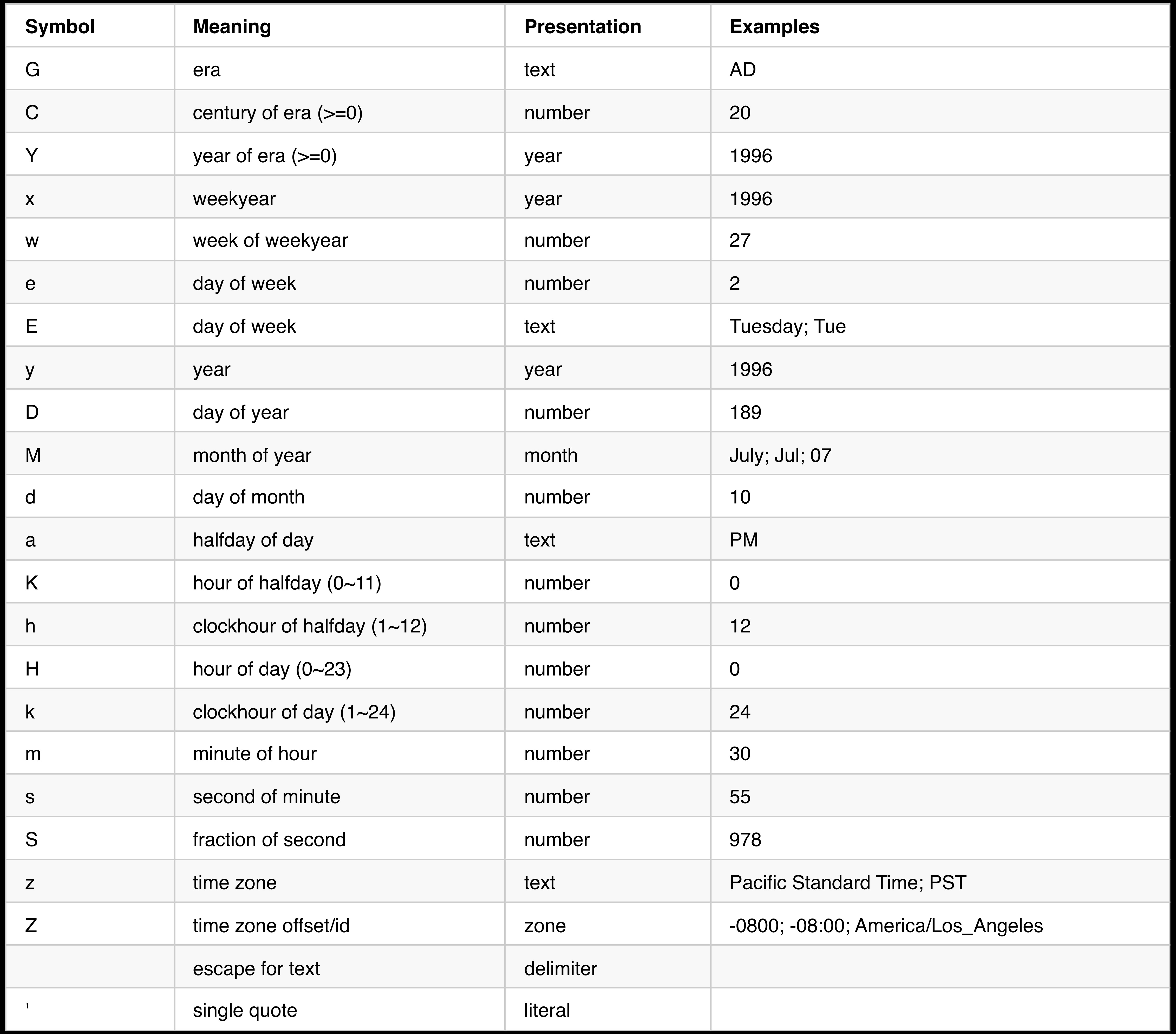
CAST(<field> **AS TIME**)

Working with Dates & Times

```
TO_DATE( <field>, '<format>')

TO_TIMESTAMP( <field>, '<format>')
```

Working with Dates & Times

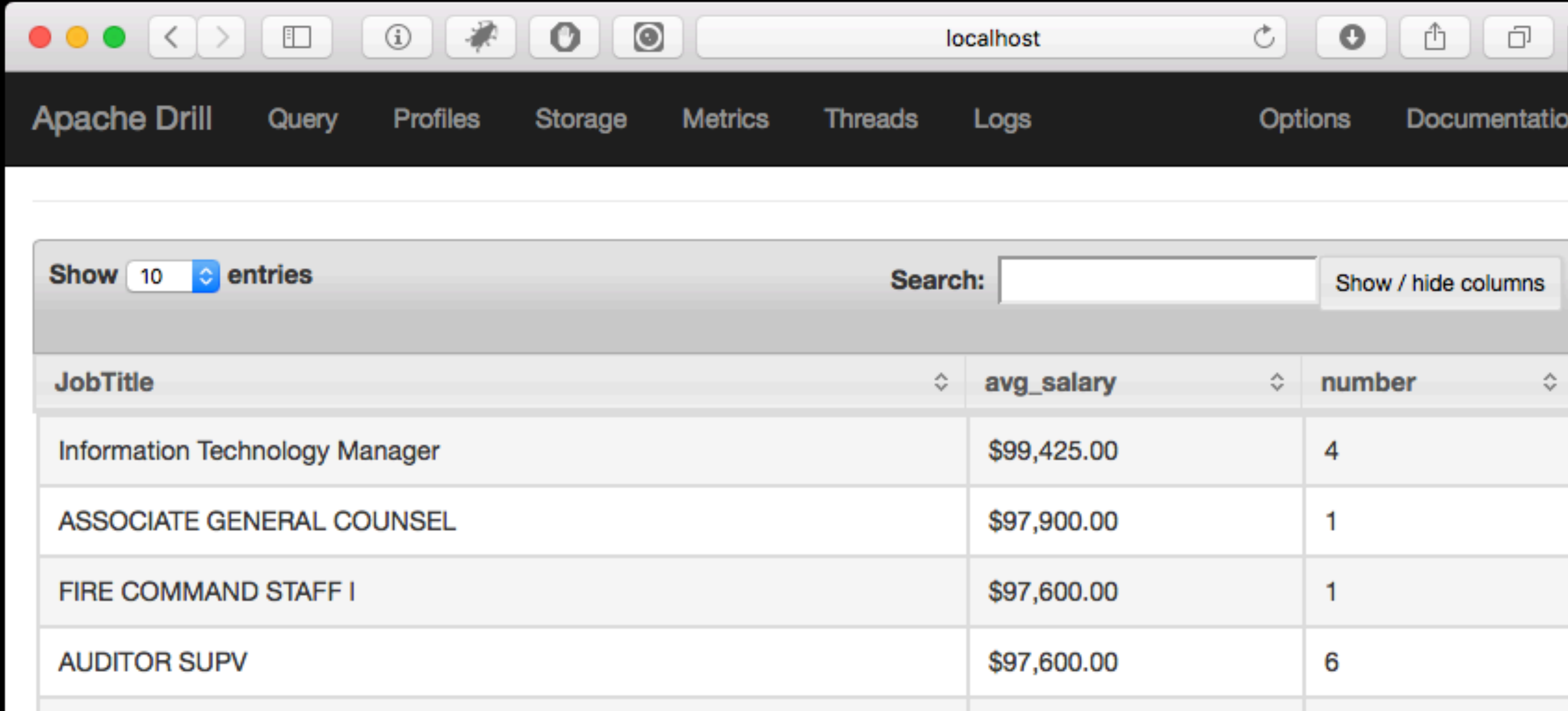


Symbol	Meaning	Presentation	Examples
G	era	text	AD
C	century of era (>=0)	number	20
Y	year of era (>=0)	year	1996
x	weekyear	year	1996
w	week of weekyear	number	27
e	day of week	number	2
E	day of week	text	Tuesday; Tue
y	year	year	1996
D	day of year	number	189
M	month of year	month	July; Jul; 07
d	day 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>)

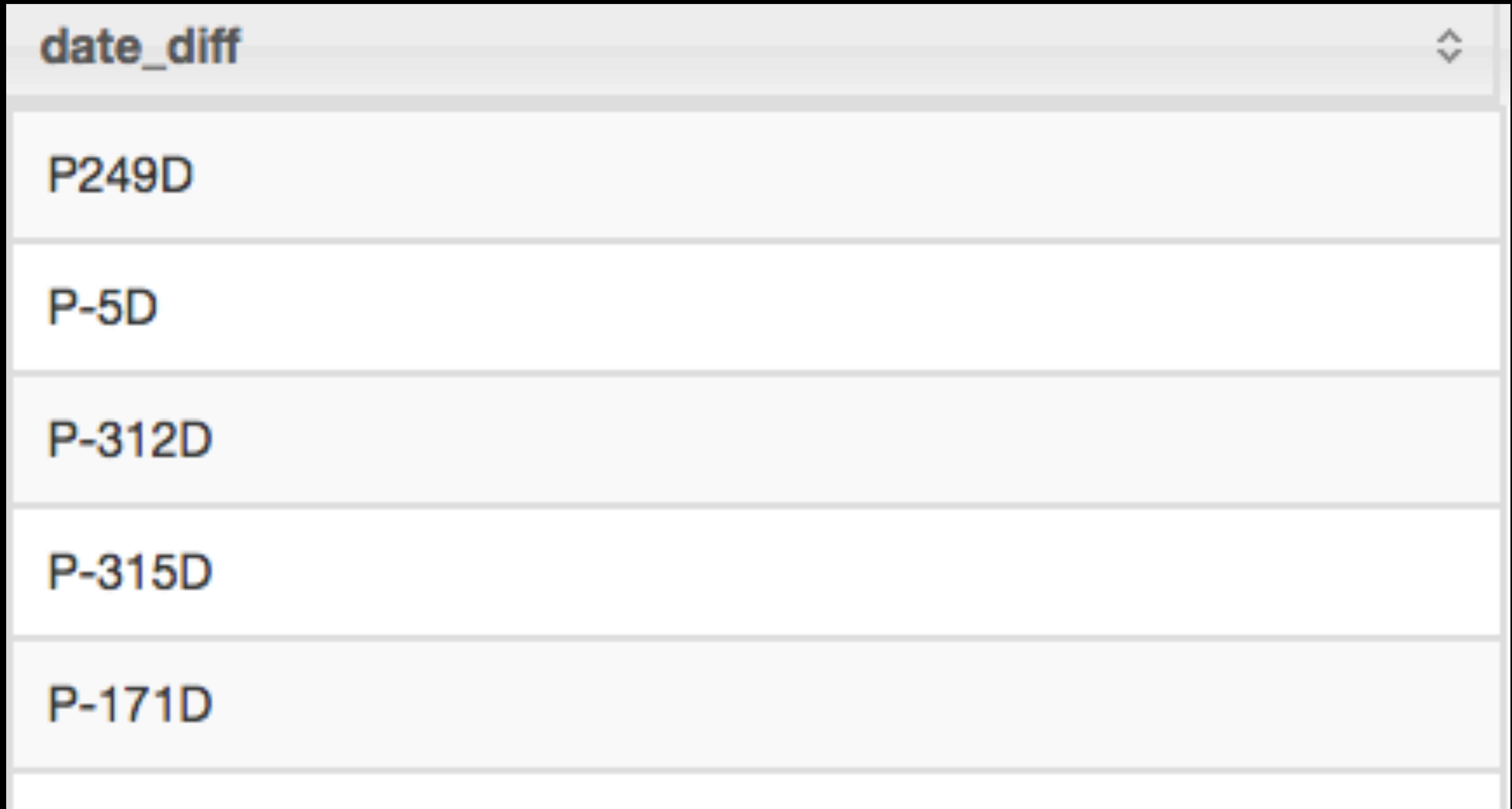
```
SELECT JobTitle,
TO_CHAR( AVG( TO_NUMBER( AnnualSalary, '¤' )), '¤#,###.00' ) AS avg_salary,
COUNT( DISTINCT name ) AS number
FROM dfs.drillclass.`baltimore_salaries_2016.csvh`
GROUP BY JobTitle
Order By avg_salary DESC
```



JobTitle	avg_salary	number
Information Technology Manager	$99,425.00	4
ASSOCIATE GENERAL COUNSEL	$97,900.00	1
FIRE COMMAND STAFF I	$97,600.00	1
AUDITOR SUPV	$97,600.00	6

Intervals

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



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

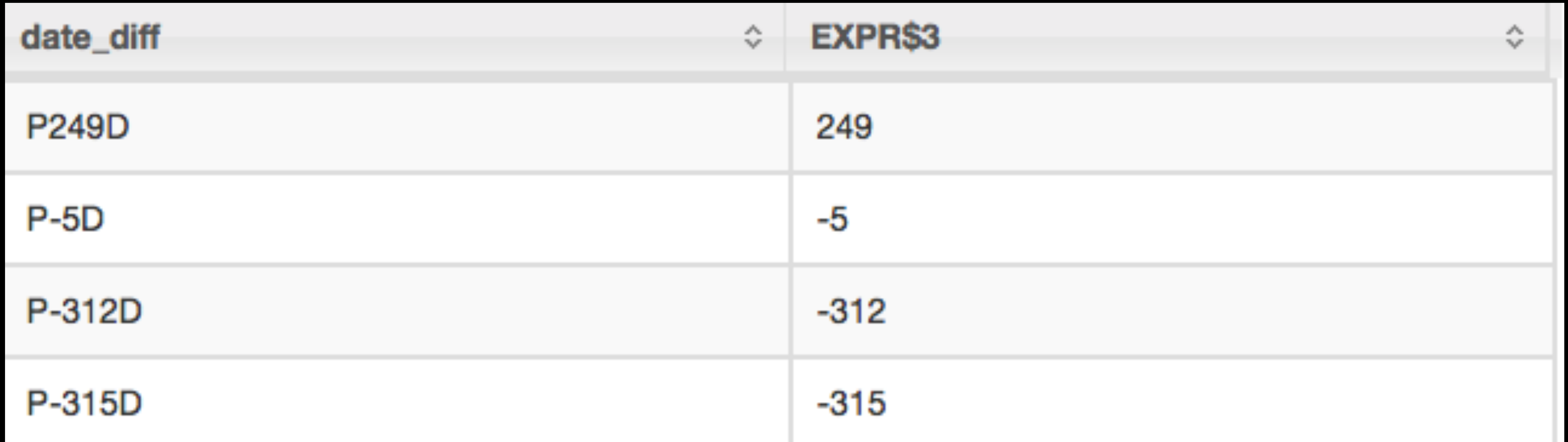
Intervals

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.drillclass.`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**

```
SELECT CAST( date1 AS DATE )
FROM dfs.drillclass.`dates.csvh`

SELECT date2, TO_DATE( date2, 'MM/dd/yyyy' )
FROM dfs.drillclass.`dates.csvh`

SELECT date3, TO_DATE( date3, 'MMM dd, yyyy' )
FROM dfs.drillclass.`dates.csvh`

SELECT date4, TO_TIMESTAMP( date4, 'EEE, dd MMM yyyy HH:mm:ss
Z' )
FROM dfs.drillclass.`dates.csvh`

SELECT date5, TO_DATE( date5, 'yyyy-MM-dd' )
FROM dfs.drillclass.`dates.csvh`
```

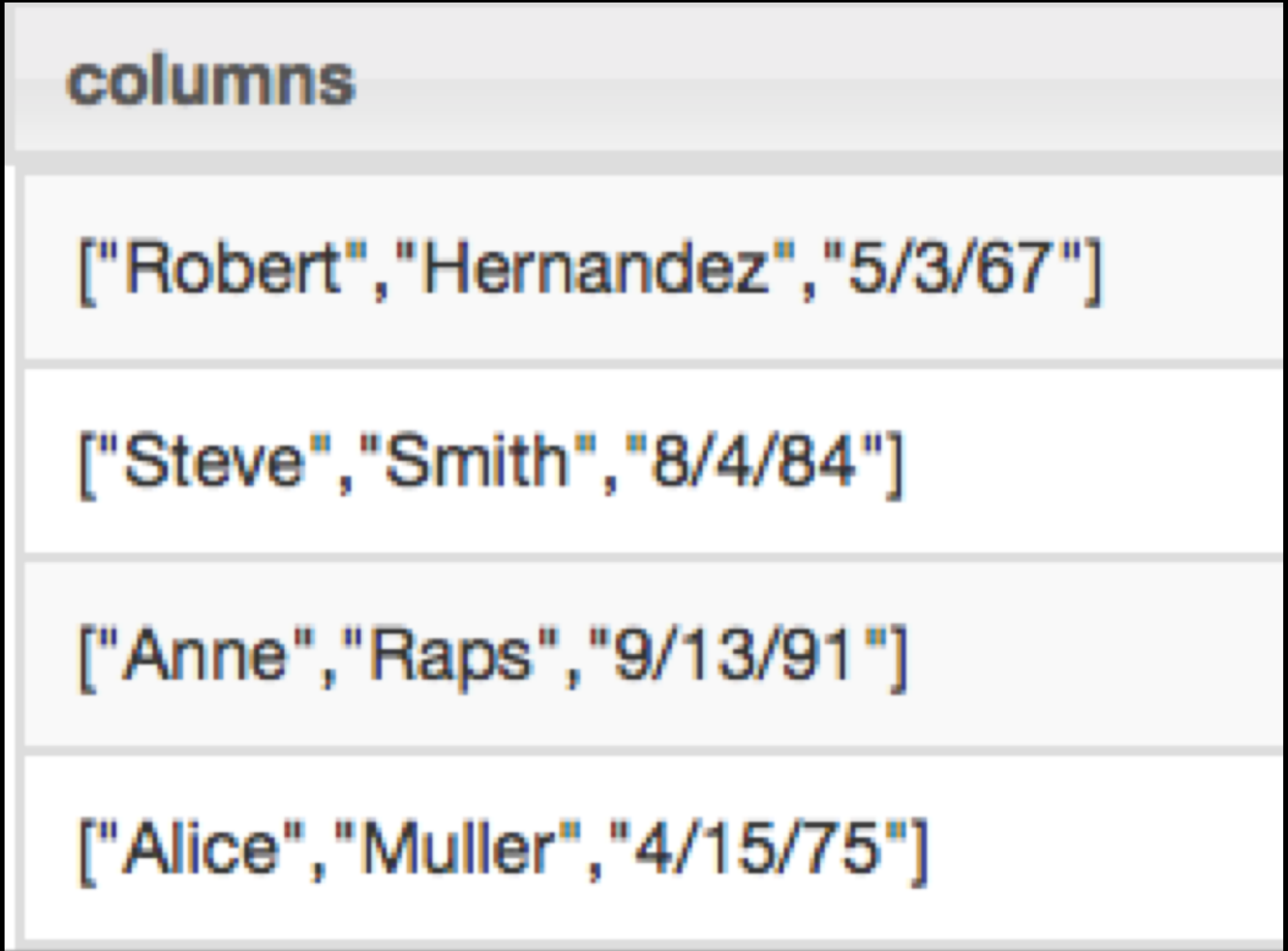
Nested Data

Complex data types: A data type which holds more than one value

Complex data types: A data type which holds more than one value

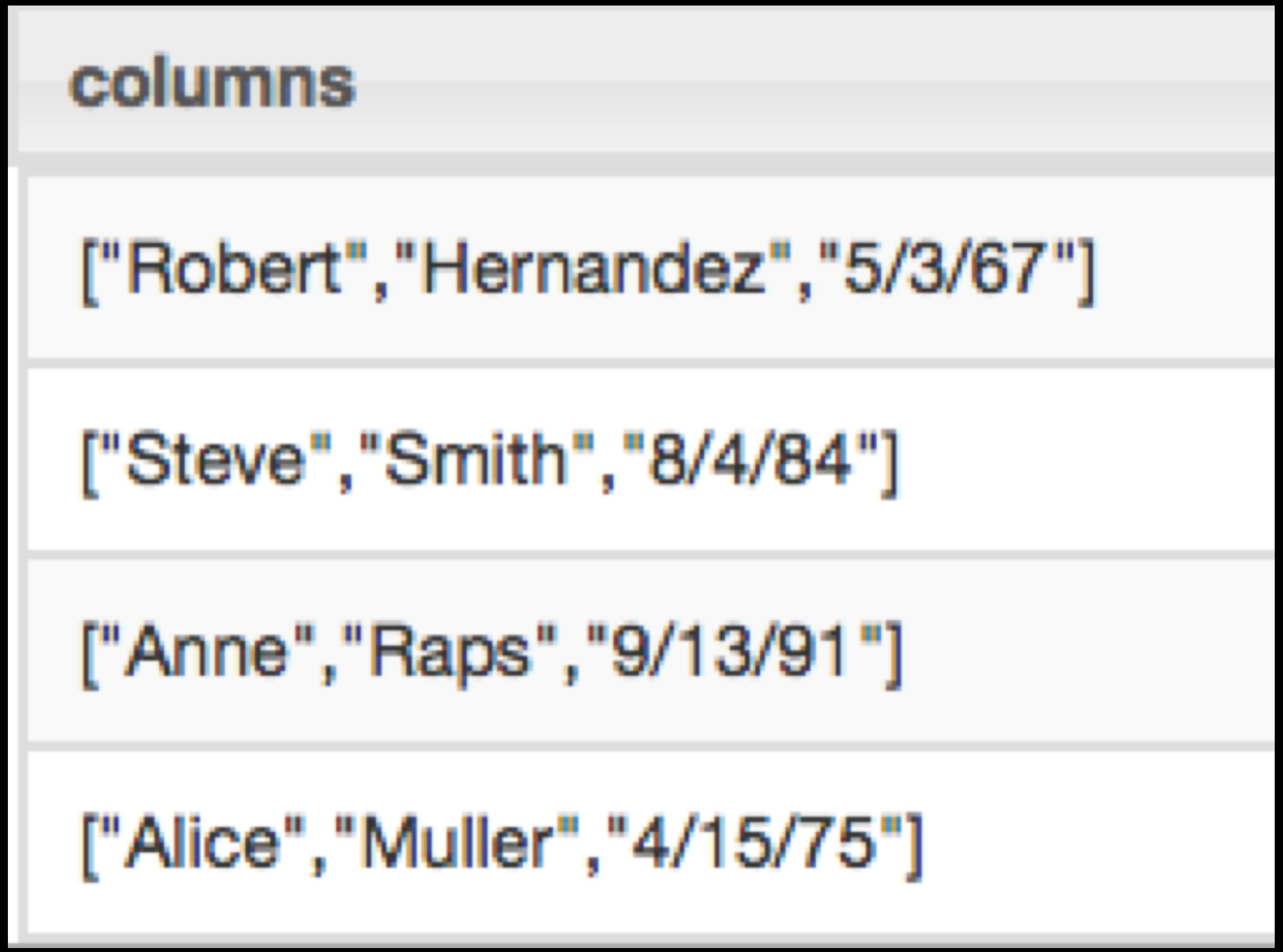
- **Array**: A complex data type **indexed by number**
- **Map**: A complex data type **indexed by a key**

Arrays in Drill



columns
["Robert","Hernandez","5/3/67"]
["Steve","Smith","8/4/84"]
["Anne","Raps","9/13/91"]
["Alice","Muller","4/15/75"]

Arrays in Drill

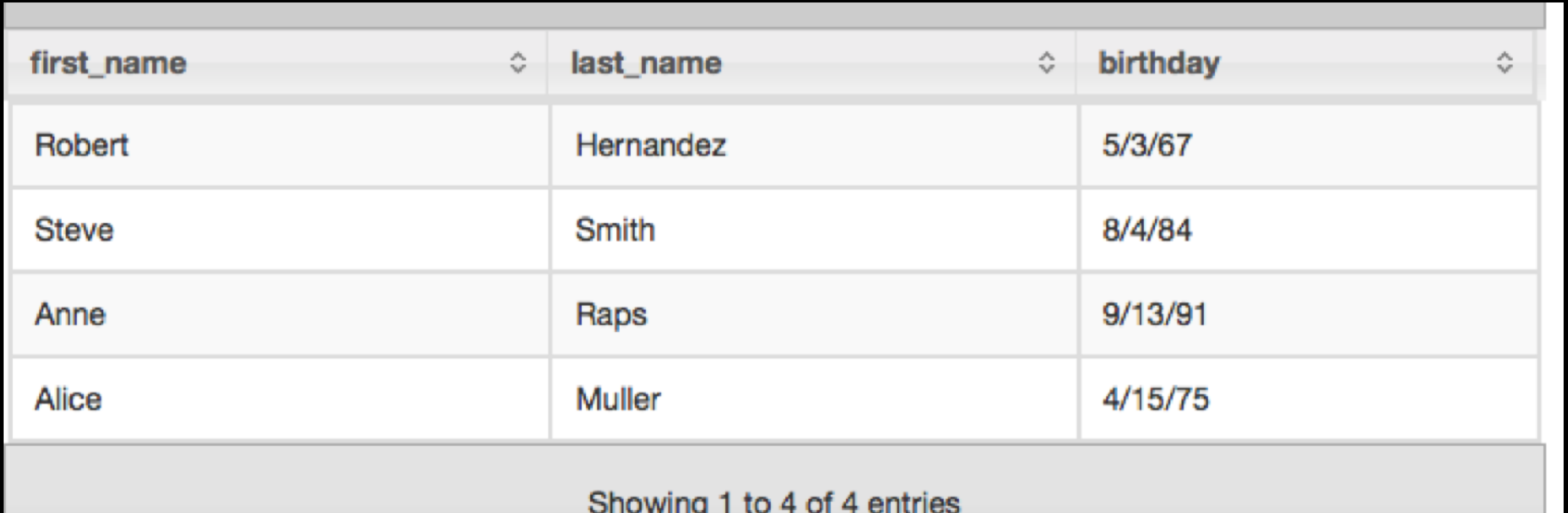


columns
["Robert","Hernandez","5/3/67"]
["Steve","Smith","8/4/84"]
["Anne","Raps","9/13/91"]
["Alice","Muller","4/15/75"]

```
SELECT columns[0] AS first_name,
columns[1] AS last_name,
columns[2] AS birthday
FROM dfs.drillclass.`customer_data.csv`
```

Arrays in Drill

```
SELECT columns[0] AS first_name,
columns[1] AS last_name,
columns[2] AS birthday
FROM dfs.drillclass.`customer_data.csv`
```



first_name	last_name	birthday
Robert	Hernandez	5/3/67
Steve	Smith	8/4/84
Anne	Raps	9/13/91
Alice	Muller	4/15/75

Showing 1 to 4 of 4 entries

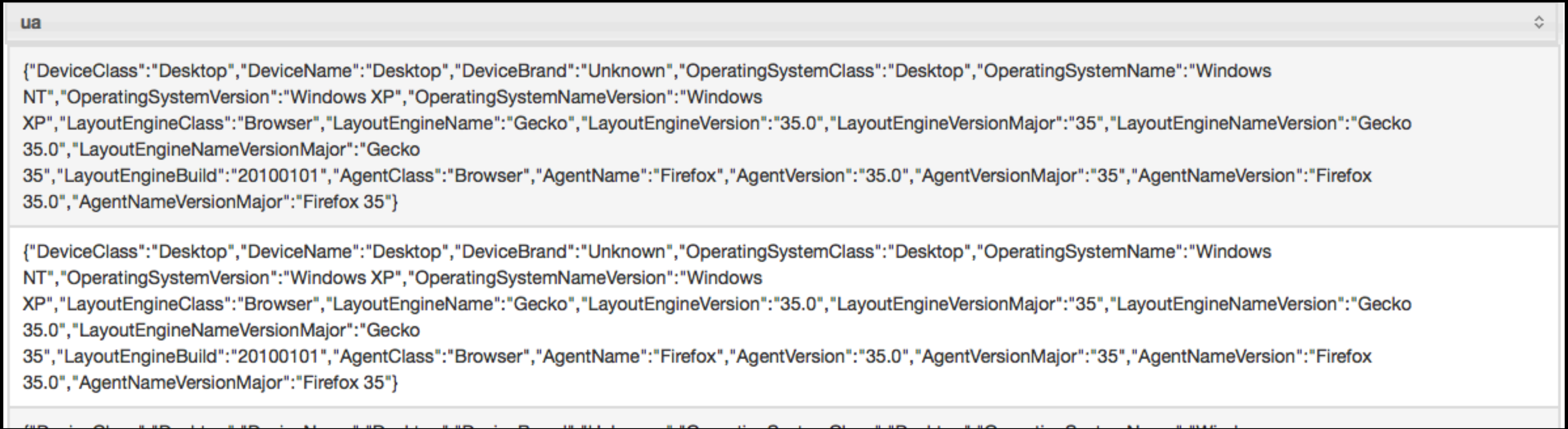
Maps (Key/Value Pairs) in Drill

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

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

Maps (Key/Value Pairs) in Drill

```
SELECT parse_user_agent( columns[0] ) AS ua
FROM dfs.drillclass.`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"}
```

Maps (Key/Value Pairs) in Drill

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

```
        {
          "DeviceClass":"Desktop",
          "DeviceName":"Macintosh",
          "DeviceBrand":"Apple",
          "OperatingSystemClass":"Desktop",
          "OperatingSystemName":"Mac OS X",
          …
           "AgentName":"Chrome",
          "AgentVersion":"39.0.2171.99",
          "AgentVersionMajor":"39",
          "AgentNameVersion":"Chrome 39.0.2171.99",
          "AgentNameVersionMajor":"Chrome 39",
          "DeviceCpu":"Intel"
        }
```

table.map.key

Maps (Key/Value Pairs) in Drill

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

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?

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?

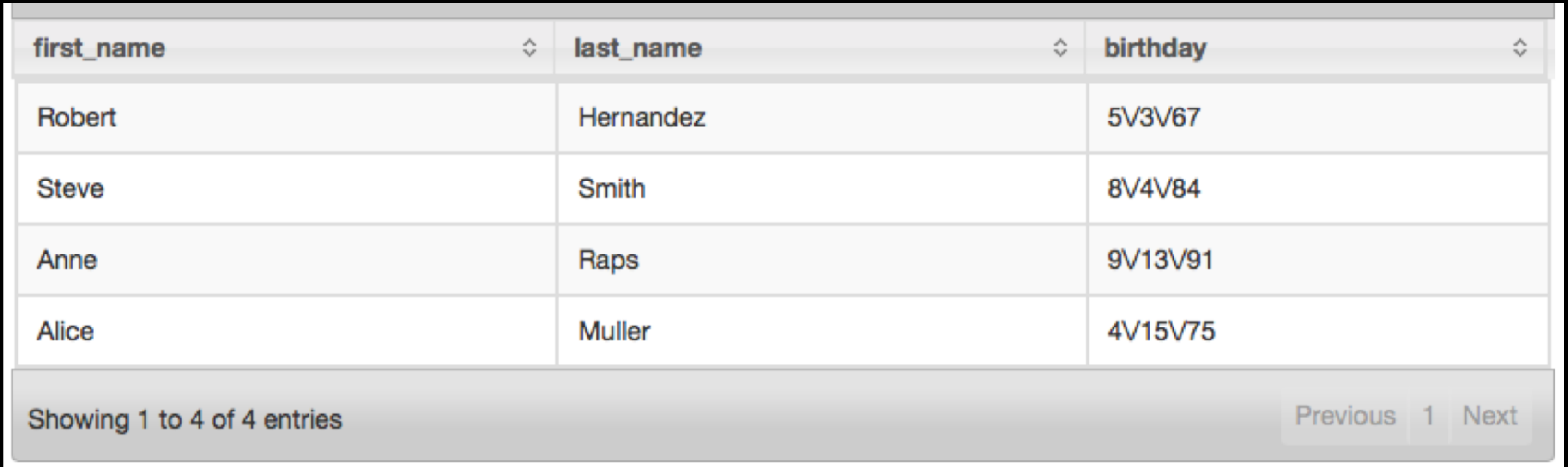
```
SELECT uadata.ua.AgentNameVersion AS Browser,
COUNT( * ) AS BrowserCount
FROM (
   SELECT parse_user_agent( columns[0] ) AS ua
   FROM dfs.drillclass.`user-agents.csv`
) AS uadata
GROUP BY uadata.ua.AgentNameVersion
ORDER BY BrowserCount DESC
```

Querying JSON Data

records.json

```
[
    {
        "first_name":"Robert",
        "last_name":"Hernandez",
        "birthday":"5\\/3\\/67"
    },{
         "first_name":"Steve",
         "last_name":"Smith",
         "birthday":"8\\/4\\/84"},
     }
]
```

```
SELECT *
FROM dfs.drillclass.`records.json`
```



first_name	last_name	birthday
Robert	Hernandez	5V3V67
Steve	Smith	8V4V84
Anne	Raps	9V13V91
Alice	Muller	4V15V75

Showing 1 to 4 of 4 entries

easy
STAPLES

Please open **split.json** in a text editor

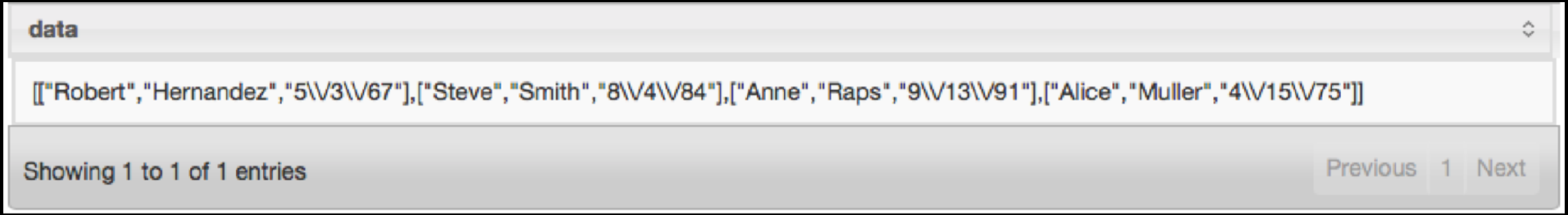
```
{
"columns":
    [
     "first_name",
      "last_name",
      "birthday"
    ],
"data":
[
 [
  "Robert",
  "Hernandez",
  "5\\/3\\/67"
 ],
 [
  "Steve",
  "Smith",
  "8\\/4\\/84"
  ],
 ]
}
```

split.json

`FLATTEN( <json array> )`

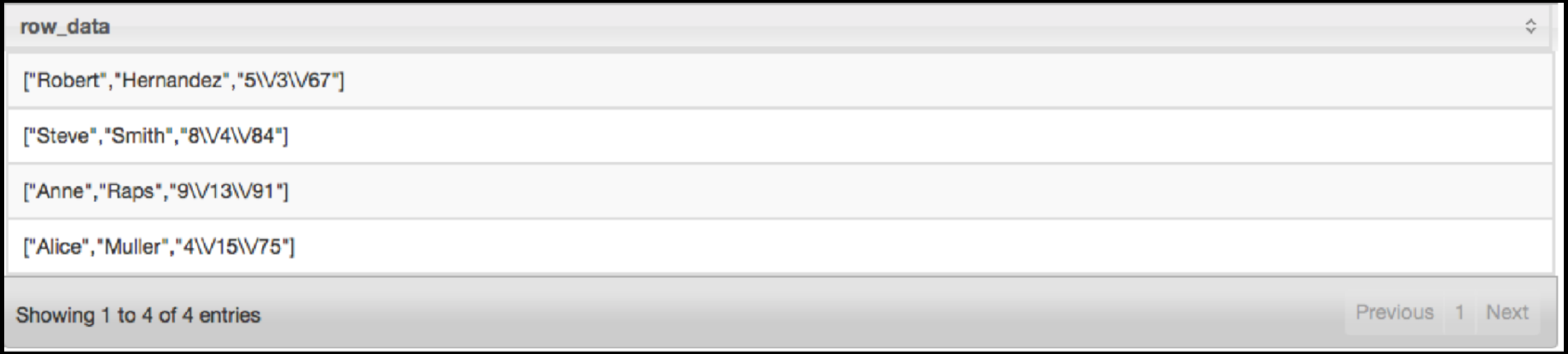
separates elements in a repeated field into individual records.

```
SELECT data
FROM dfs.drillclass.`split.json`
```



```
data
[["Robert","Hernandez","5\/3\/67"],["Steve","Smith","8\/4\/84"],["Anne","Raps","9\/13\/91"],["Alice","Muller","4\/15\/75"]]
Showing 1 to 1 of 1 entries                                              Previous  1  Next
```

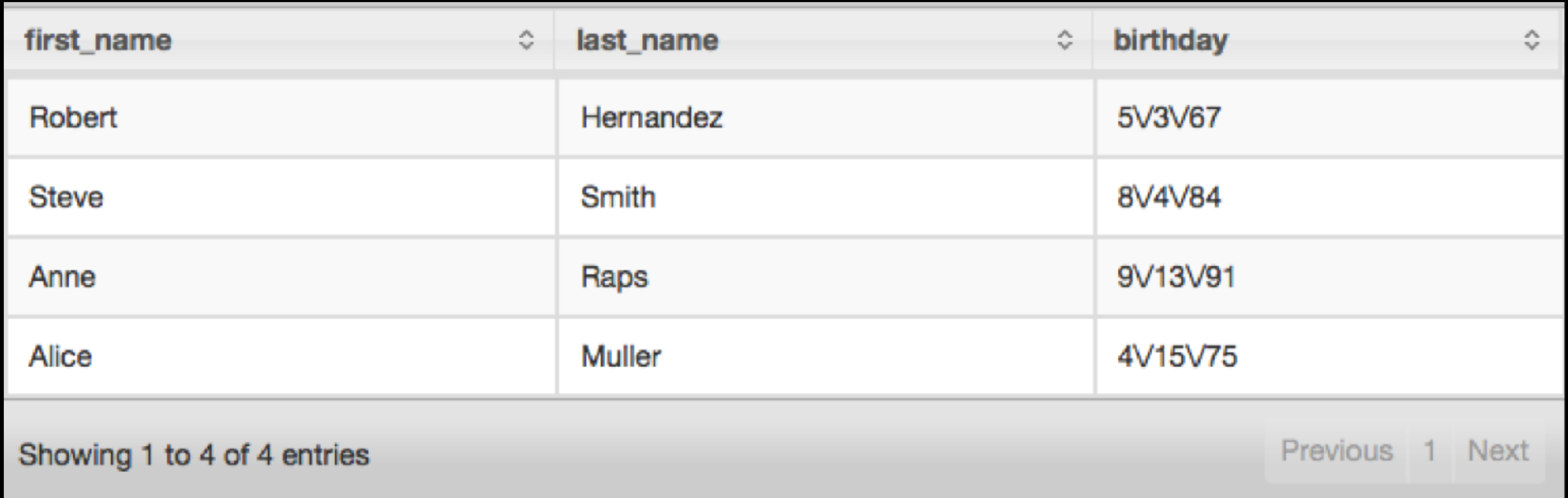
```
SELECT FLATTEN(data) AS row_data
FROM dfs.drillclass.`split.json`
```



row_data
["Robert","Hernandez","5\\/3\\/67"]
["Steve","Smith","8\\/4\\/84"]
["Anne","Raps","9\\/13\\/91"]
["Alice","Muller","4\\/15\\/75"]

Showing 1 to 4 of 4 entries

```
SELECT row_data[0] AS first_name,
row_data[1] AS last_name,
row_data[2] AS birthday
FROM
(
  SELECT FLATTEN( data ) AS row_data
  FROM dfs.drillclass.`split.json`
) AS split_data
```



first_name	last_name	birthday
Robert	Hernandez	5V3V67
Steve	Smith	8V4V84
Anne	Raps	9V13V91
Alice	Muller	4V15V75

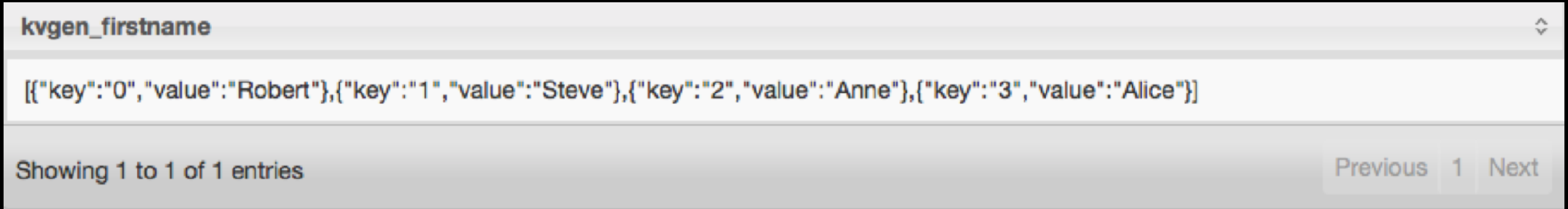
Showing 1 to 4 of 4 entries

Please open **columns.json** in a text editor

```
{
 "first_name":
  {
   "0":"Robert",
   "1":"Steve",
   "2":"Anne",
   "3":"Alice"
  },
 “last_name": {
   "0":"Hernandez",
   "1":"Smith",
   "2":"Raps",
   “3”:"Muller"
   },
"birthday":{
   "0":"5\\/3\\/67",
   "1":"8\\/4\\/84",
   "2":"9\\/13\\/91",
   "3":"4\\/15\\/75"
   }
}
```

`KVGEN(<map>)` generates key/value pairs from a column with repeated data. Often used in combination with **`FLATTEN()`**.

```
SELECT KVGEN( first_name ) AS kvgen_firstname
FROM dfs.drillclass.`columns.json`
```

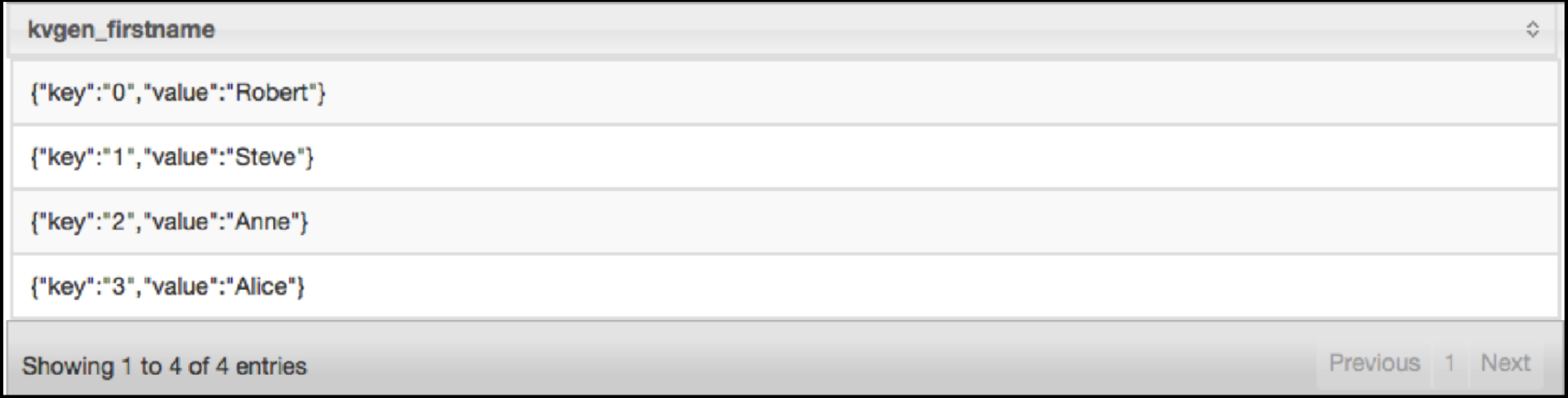


kvgen_firstname
[{"key":"0","value":"Robert"},{"key":"1","value":"Steve"},{"key":"2","value":"Anne"},{"key":"3","value":"Alice"}]

Showing 1 to 1 of 1 entries

```
SELECT FLATTEN(
  KVGEN( first_name )
) AS kvgen_firstname
FROM dfs.drillclass.`columns.json`
```

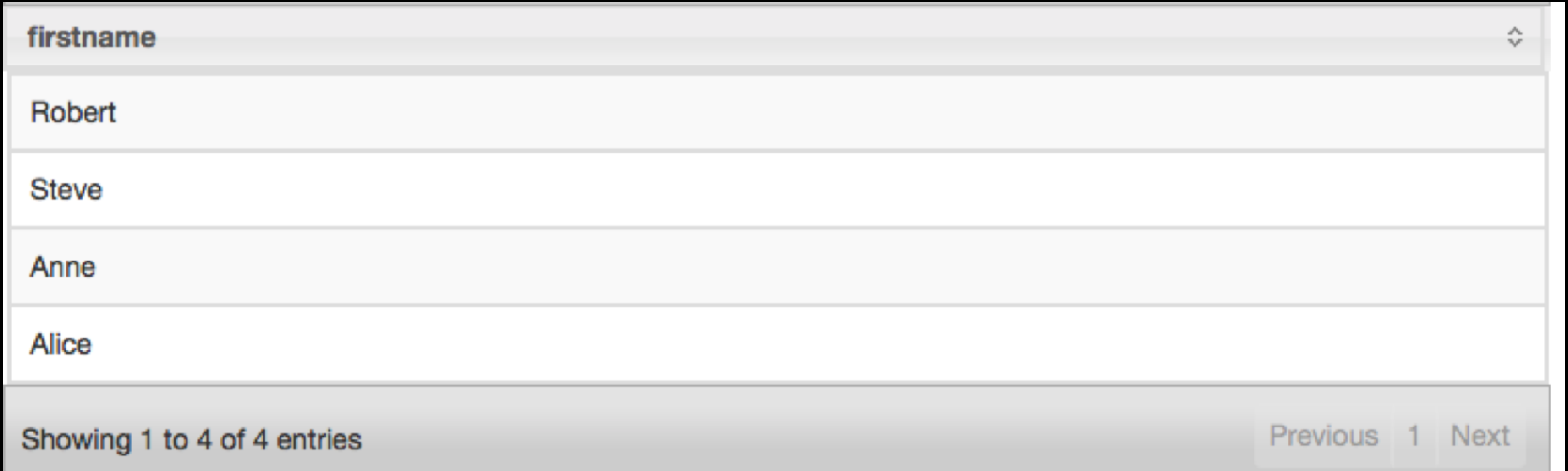
```
SELECT FLATTEN(
  KVGEN( first_name )
) AS kvgen_firstname
FROM dfs.drillclass.`columns.json`
```



kvgen_firstname
{"key":"0","value":"Robert"}
{"key":"1","value":"Steve"}
{"key":"2","value":"Anne"}
{"key":"3","value":"Alice"}

Showing 1 to 4 of 4 entries

```
SELECT FLATTEN( KVGEN( first_name ) )['value'] AS firstname
FROM dfs.drillclass.`columns.json`
```



firstname
Robert
Steve
Anne
Alice

Showing 1 to 4 of 4 entries

```
SELECT first_name, last_name, birthday
FROM
(
   SELECT row_number() OVER (ORDER BY '1') AS rownum,
   FLATTEN( KVGEN(first_name))['value'] AS first_name
   FROM dfs.drillclass.`columns.json`
) AS tbl1
JOIN
(
  SELECT row_number() OVER (ORDER BY '1') AS rownum,
  FLATTEN( KVGEN(last_name))['value'] AS last_name
  FROM dfs.drillclass.`columns.json`
) AS tbl2
ON tbl1.rownum=tbl2.rownum
JOIN
(
  SELECT row_number() OVER (ORDER BY '1') AS rownum,
  FLATTEN( KVGEN(birthday))['value'] AS birthday
  FROM dfs.drillclass.`columns.json`
) AS tbl3 ON tbl1.rownum=tbl3.rownum
```

Putting it all together…

Please run

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

in the Drill command line

Please open
`baltimore_salaries_2016.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"
     ]
```

In Class Exercise

Using the Baltimore Salaries 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.drillclass.`baltimore_salaries_2016.json`
)
GROUP BY raw_data[9]
ORDER BY avg_salary DESC
```

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 complete documentation: https://gist.github.com/cgivre/47f07a06d44df2af625fc6848407ae7c

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"
```

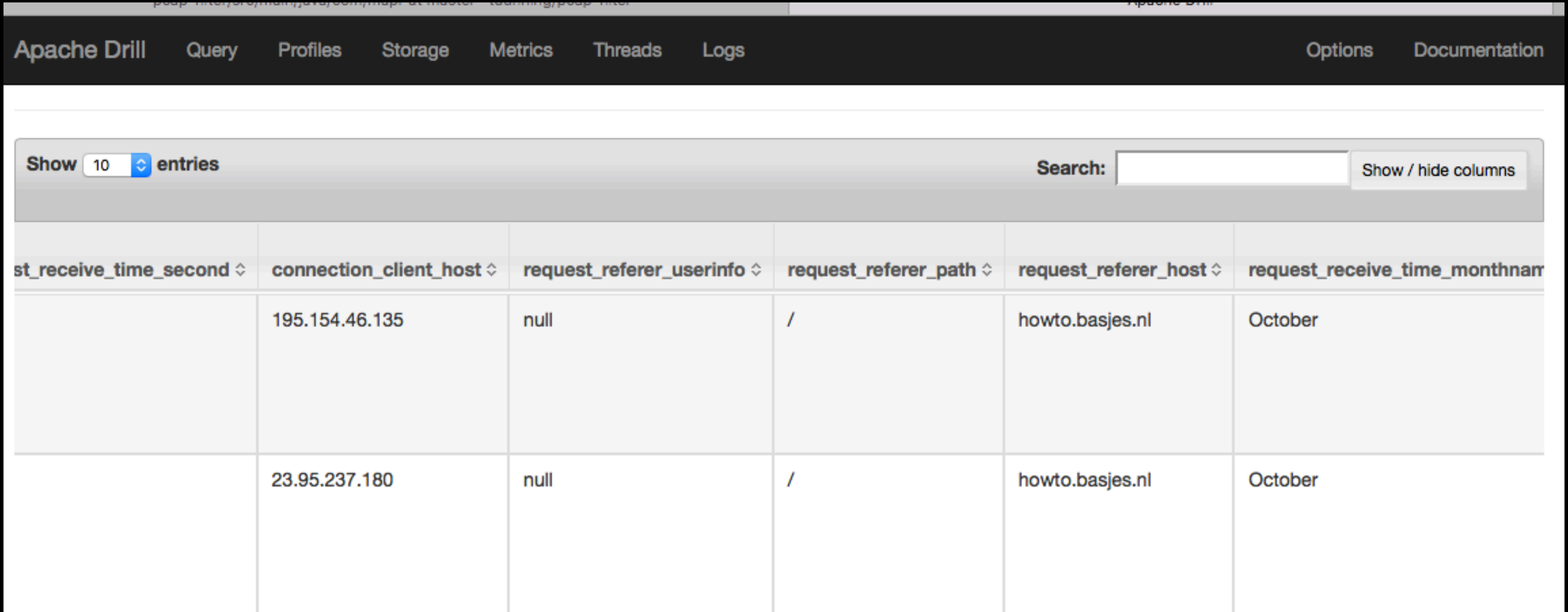
```
"httpd": {
      "type": "httpd",
      "logFormat": "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\"",
      "timestampFormat": null
    },
```

HTTPD Log Files

```
SELECT *
FROM dfs.drillclass.`small-server-log.httpd`
```

HTTPD Log Files

```
SELECT *
FROM dfs.drillclass.`small-server-log.httpd`
```



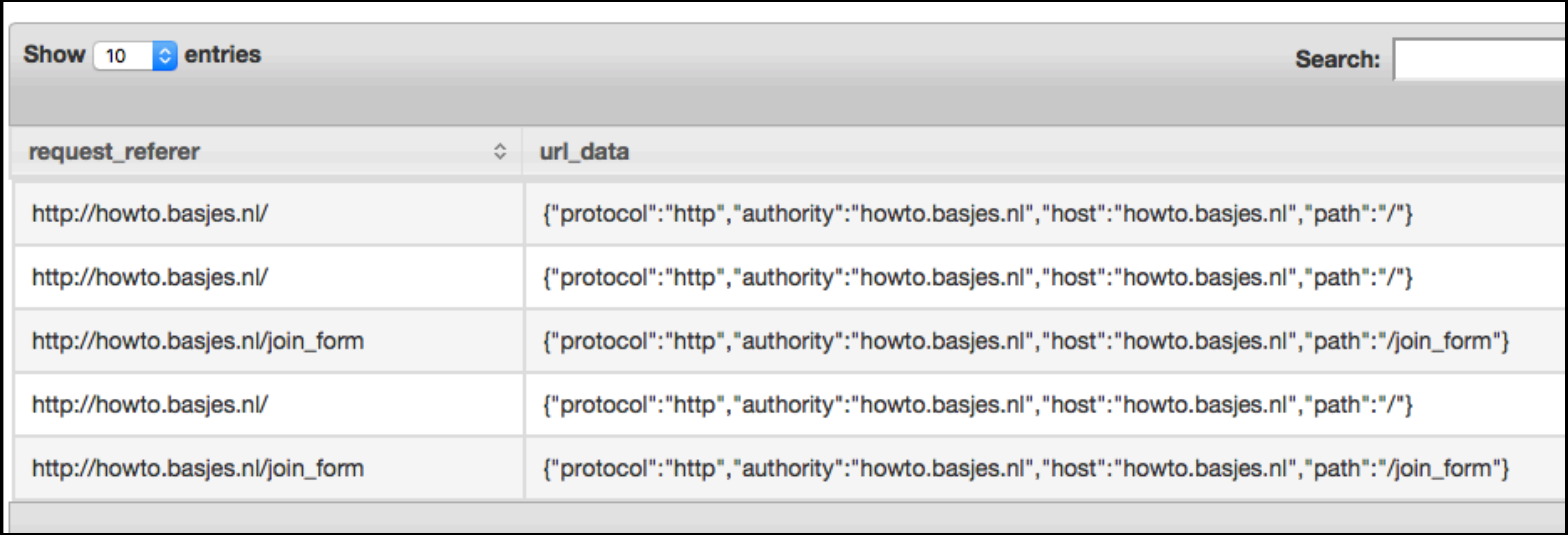
st_receive_time_second	connection_client_host	request_referer_userinfo	request_referer_path	request_referer_host	request_receive_time_monthnam
	195.154.46.135	null	/	howto.basjes.nl	October
	23.95.237.180	null	/	howto.basjes.nl	October

HTTPD Log Files

```
SELECT request_referer, parse_url( request_referer ) AS url_data
FROM dfs.drillclass.`small-server-log.httpd`
```

HTTPD Log Files

```
SELECT request_referer, parse_url( request_referer ) AS url_data
FROM dfs.drillclass.`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"}

http%3A%2F%2Fmysite.com%3Fuser%3Dcgivre
%26password%3D1234%26firstname%3DCharle
s+S

```
SELECT urldecode( <field> )
FROM…
```

http://mysite.com?
user=cgivre&password=1234&first name=Charles S

```
SELECT parse_query( urldecode( <field> ) )
FROM…
```

```
{
  “username”:”Charles”,
  “password”:”1234”,
   “name”:”Charles S”
}
```

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
- urlencode(<url>): Returns a URL encoded string
- urldecode(<url>): Decodes a URL encoded string

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?

In Class Exercise

```
SELECT ua.uadata.OperatingSystemNameVersion AS operating_system,
COUNT( * ) AS os_count
FROM
(
  SELECT parse_user_agent(`request_user-agent`) AS uadata
  FROM dfs.drillclass.`hackers-access.httpd`
) AS ua
GROUP BY ua.uadata.OperatingSystemNameVersion
ORDER BY os_count DESC
```

A Quick Demo…

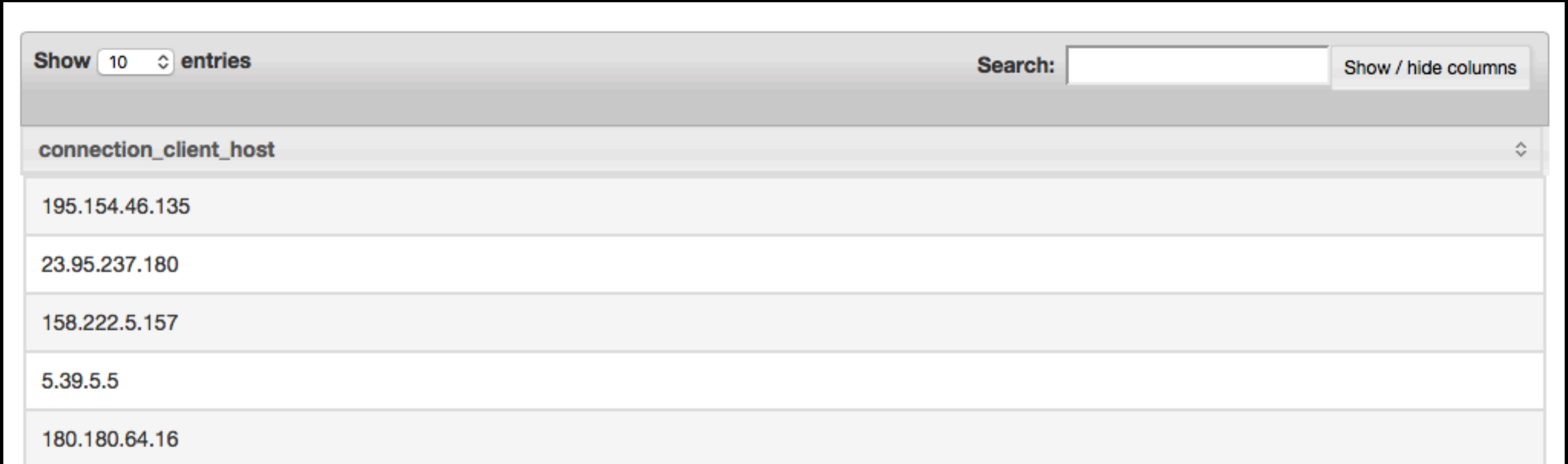
What if you wanted all the **unique** IP addresses in your server log?

A Quick Demo…

```
SELECT DISTINCT connection_client_host
FROM dfs.drillclass.`hackers-access.httpd`
```

A Quick Demo…

```
SELECT DISTINCT connection_client_host
FROM dfs.drillclass.`hackers-access.httpd`
```



Show 10 entries

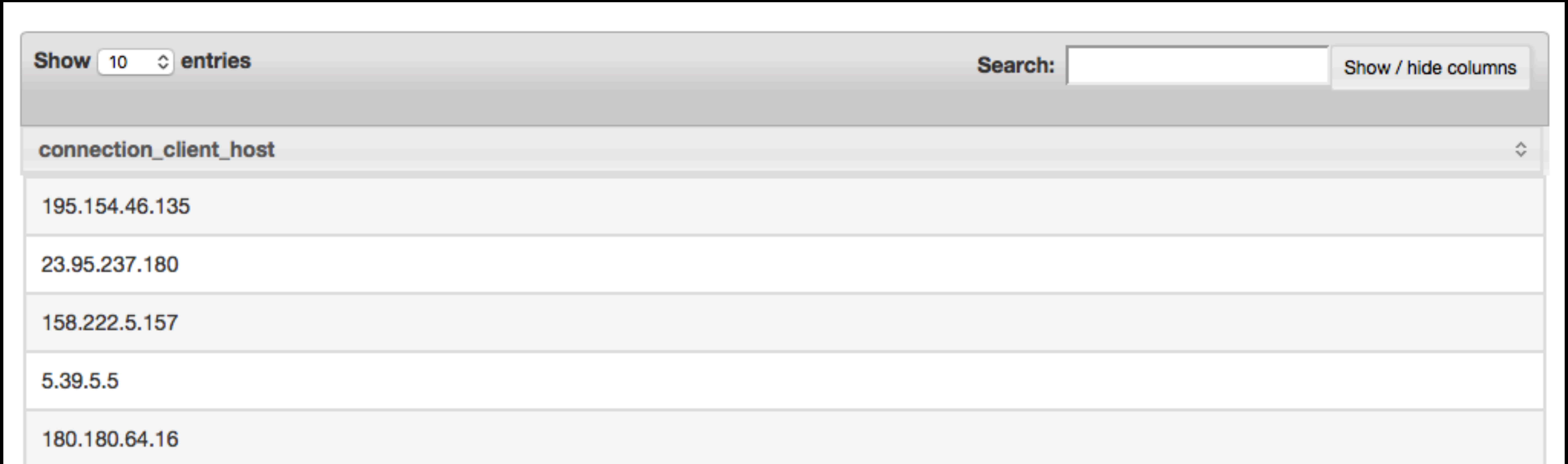
Search: Show / hide columns

connection_client_host
195.154.46.135
23.95.237.180
158.222.5.157
5.39.5.5
180.180.64.16

A Quick Demo…

```
SELECT DISTINCT connection_client_host
FROM dfs.drillclass.`hackers-access.httpd`
```

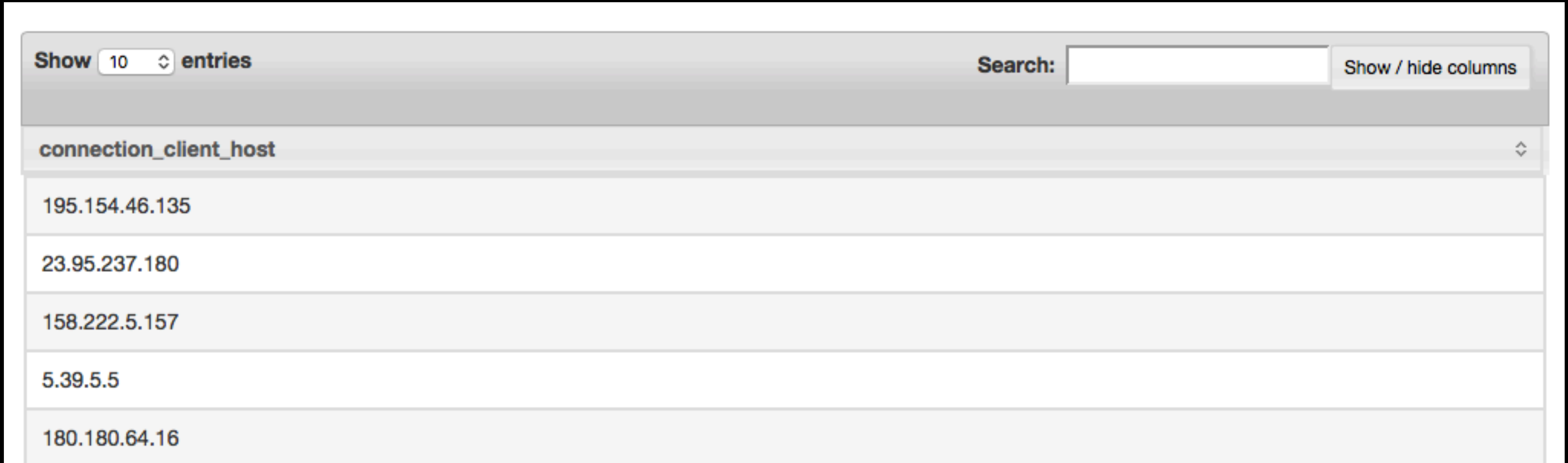
Now what if you wanted these IPs in order?



connection_client_host
195.154.46.135
23.95.237.180
158.222.5.157
5.39.5.5
180.180.64.16

A Quick Demo...

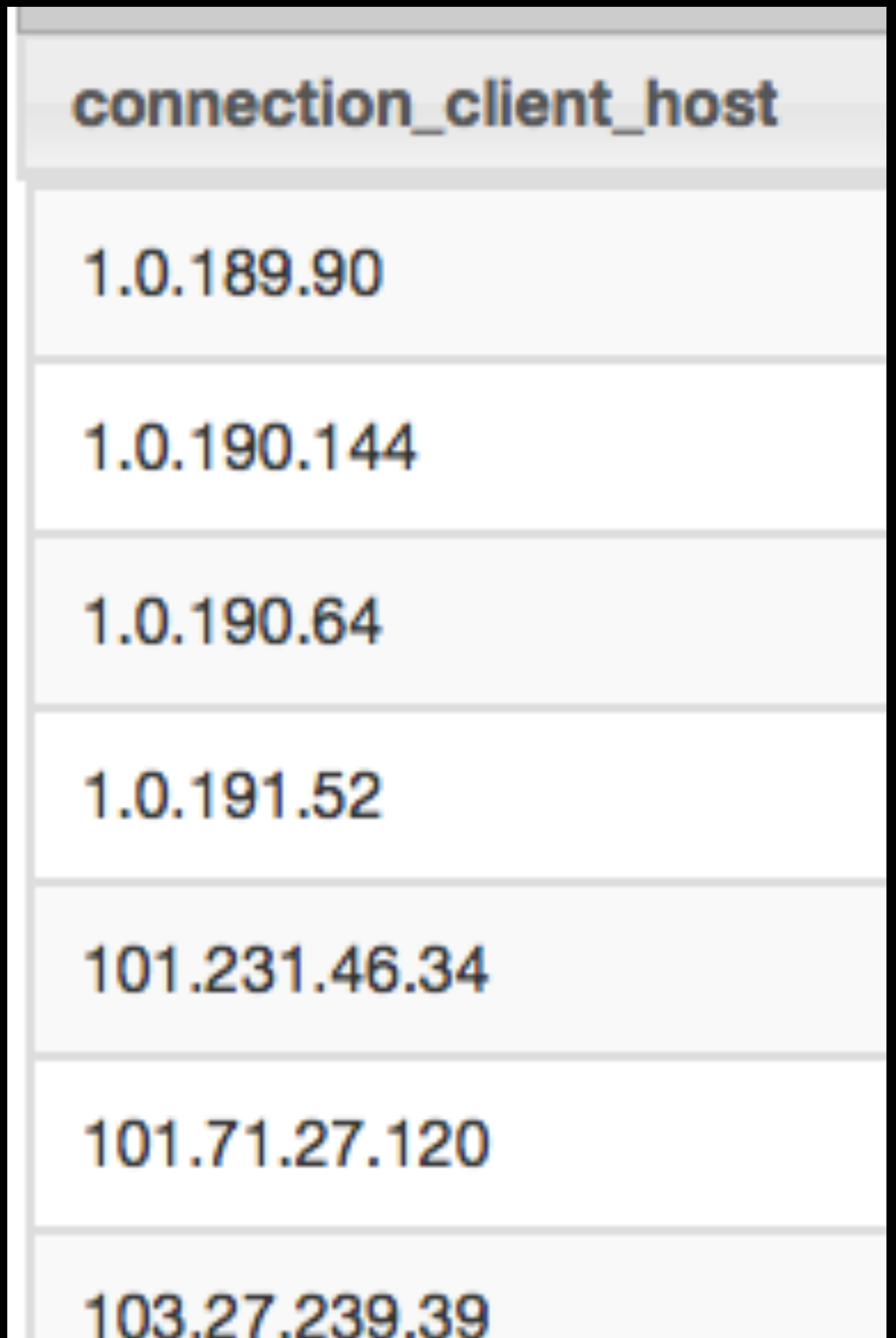
```
SELECT DISTINCT connection_client_host
FROM dfs.drillclass.`hackers-access.httpd`
WHERE regexp_matches(`connection_client_host`, '(\d{1,3}\.)
{3}\d{1,3}')
```



```
connection_client_host
195.154.46.135
23.95.237.180
158.222.5.157
5.39.5.5
180.180.64.16
```

A Quick Demo…

```
SELECT DISTINCT connection_client_host
FROM dfs.drillclass.`hackers-access.httpd`
WHERE regexp_matches(`connection_client_host`, '(\d{1,3}\.)
{3}\d{1,3}')
```



connection_client_host
1.0.189.90
1.0.190.144
1.0.190.64
1.0.191.52
101.231.46.34
101.71.27.120
103.27.239.39

What if we only wanted IPs within a certain range?

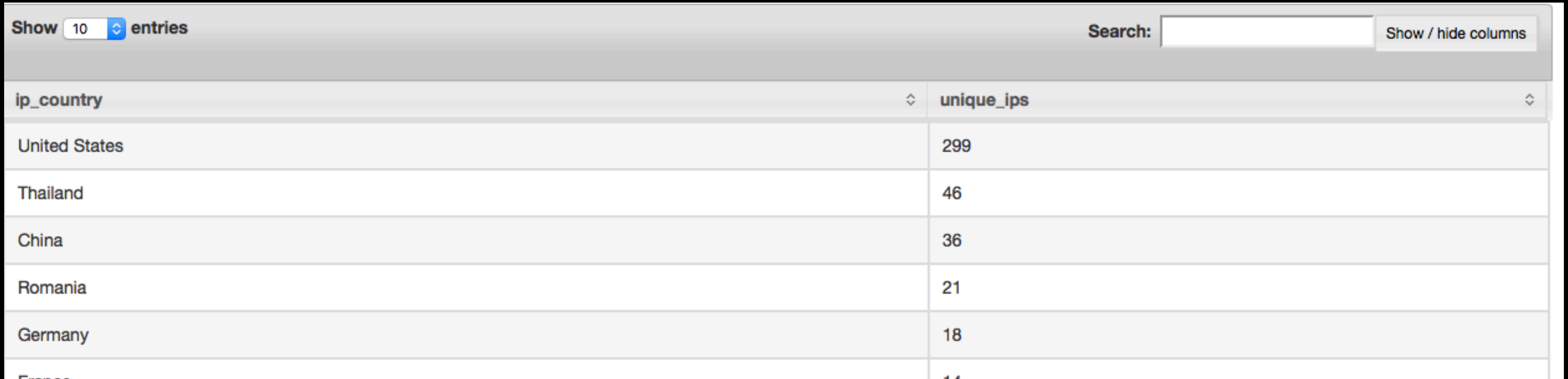
A Quick Demo...

```
SELECT DISTINCT connection_client_host
FROM dfs.drillclass.`hackers-access.httpd`
WHERE regexp_matches(`connection_client_host`, '(\d{1,3}\.)
{3}\d{1,3}') AND
inet_aton( `connection_client_host` ) >= inet_aton( '23.94.10.8' )
AND
inet_aton( `connection_client_host` ) < inet_aton( '31.187.79.31' )
ORDER BY inet_aton( `connection_client_host` ) ASC
```

What if we wanted to know what were the locations of IPs who requested certain pages?

A Quick Demo...

```
SELECT getCountryName( connection_client_host ) AS ip_country,
COUNT( DISTINCT connection_client_host ) AS unique_ips
FROM dfs.drillclass.`hackers-access.httpd`
WHERE regexp_matches(`connection_client_host`, '(\d{1,3}\.)
{3}\d{1,3}')
GROUP BY getCountryName( connection_client_host )
ORDER BY unique_ips DESC
```



ip_country	unique_ips
United States	299
Thailand	46
China	36
Romania	21
Germany	18

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

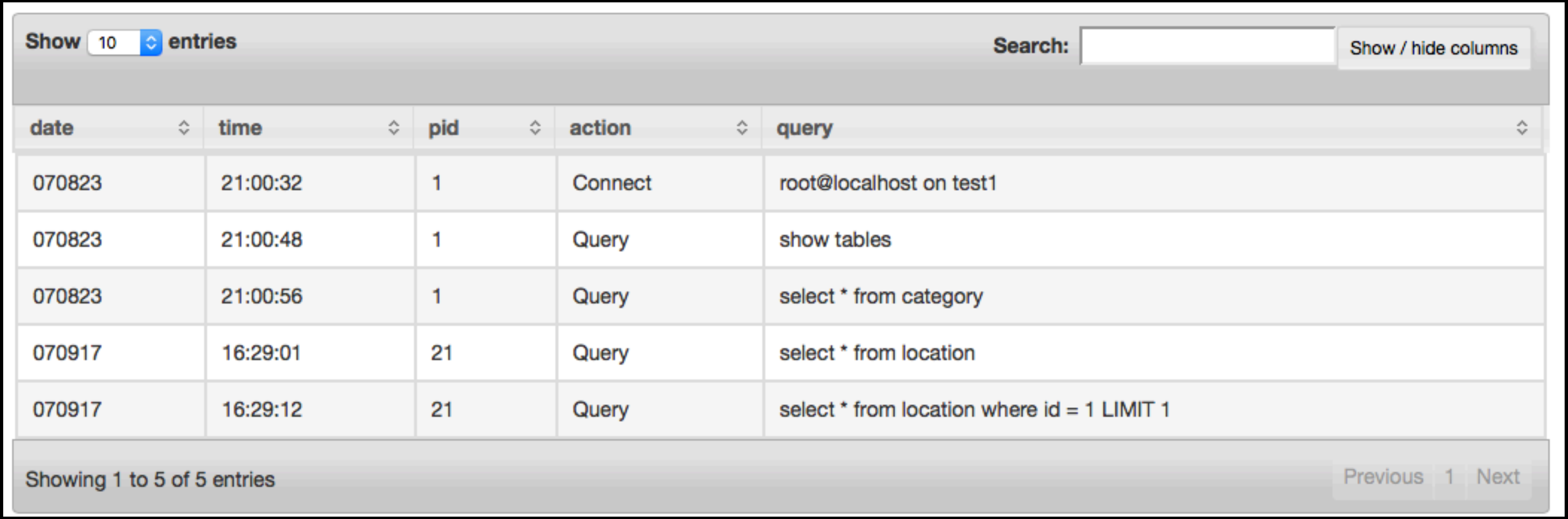
Log Files

```
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",
      "errorOnMismatch": false
      "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.drillclass.`mysql.log`
```

```
SELECT *
FROM dfs.drillclass.`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

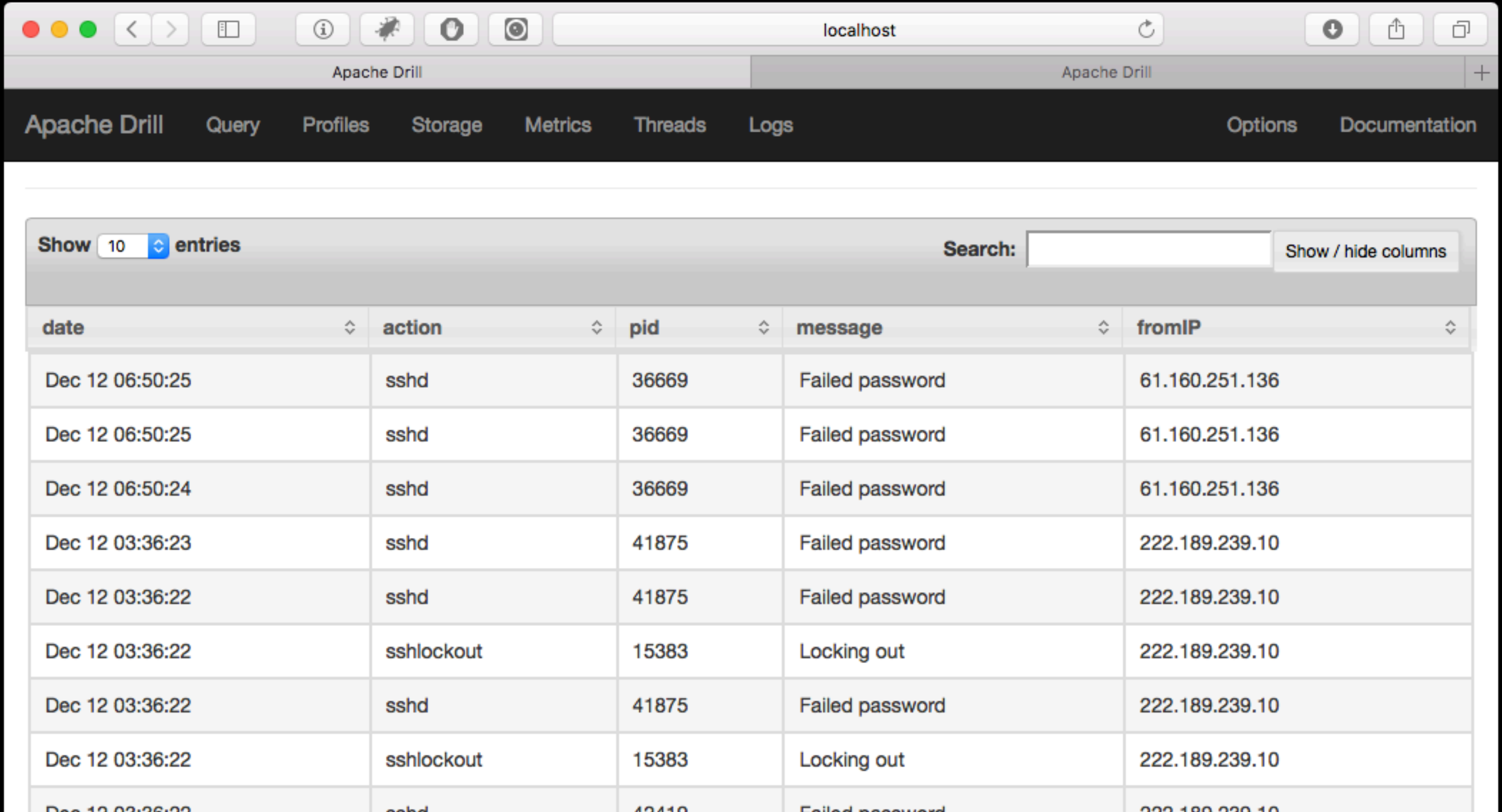
2. Use that regex to configure Drill to query this data.

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



date	action	pid	message	fromIP
Dec 12 06:50:25	sshd	36669	Failed password	61.160.251.136
Dec 12 06:50:25	sshd	36669	Failed password	61.160.251.136
Dec 12 06:50:24	sshd	36669	Failed password	61.160.251.136
Dec 12 03:36:23	sshd	41875	Failed password	222.189.239.10
Dec 12 03:36:22	sshd	41875	Failed password	222.189.239.10
Dec 12 03:36:22	sshlockout	15383	Locking out	222.189.239.10
Dec 12 03:36:22	sshd	41875	Failed password	222.189.239.10
Dec 12 03:36:22	sshlockout	15383	Locking out	222.189.239.10

Connecting other Data Sources

Connecting other Data Sources

APACHE HBASE

mongoDB®

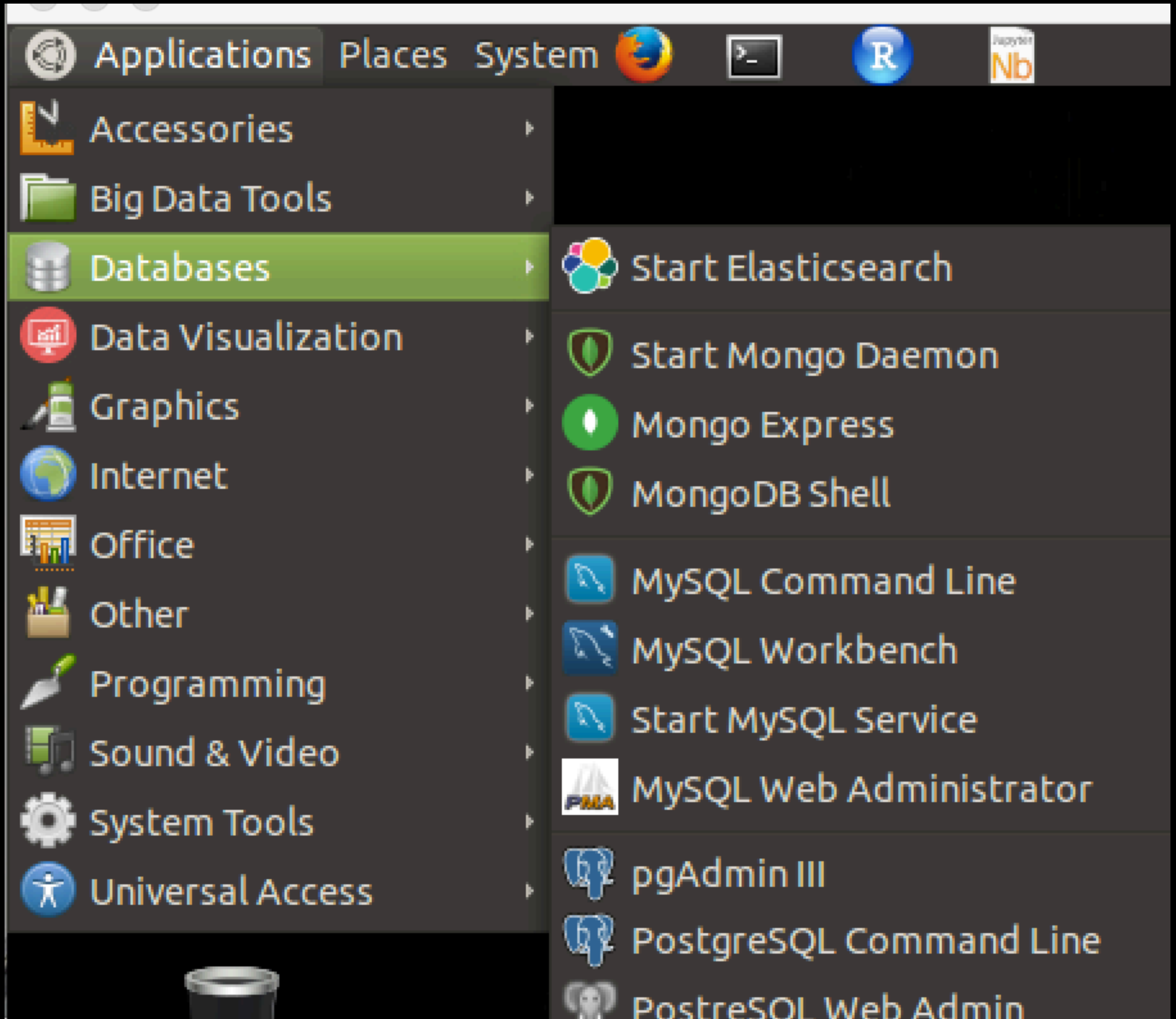
hadoop

HIVE

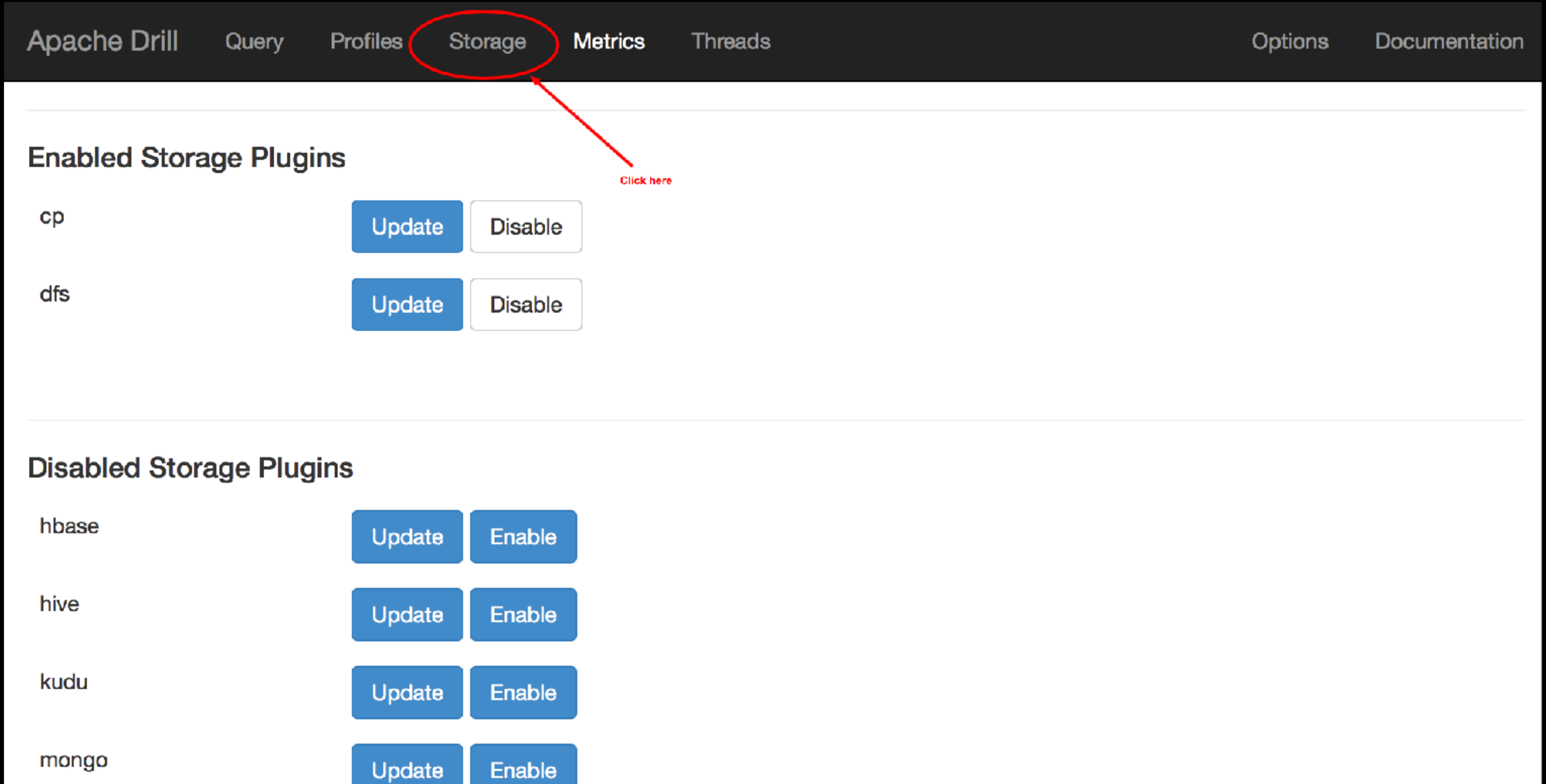
amazon web services™

MySQL®

Connecting other Data Sources



Connecting other Data Sources



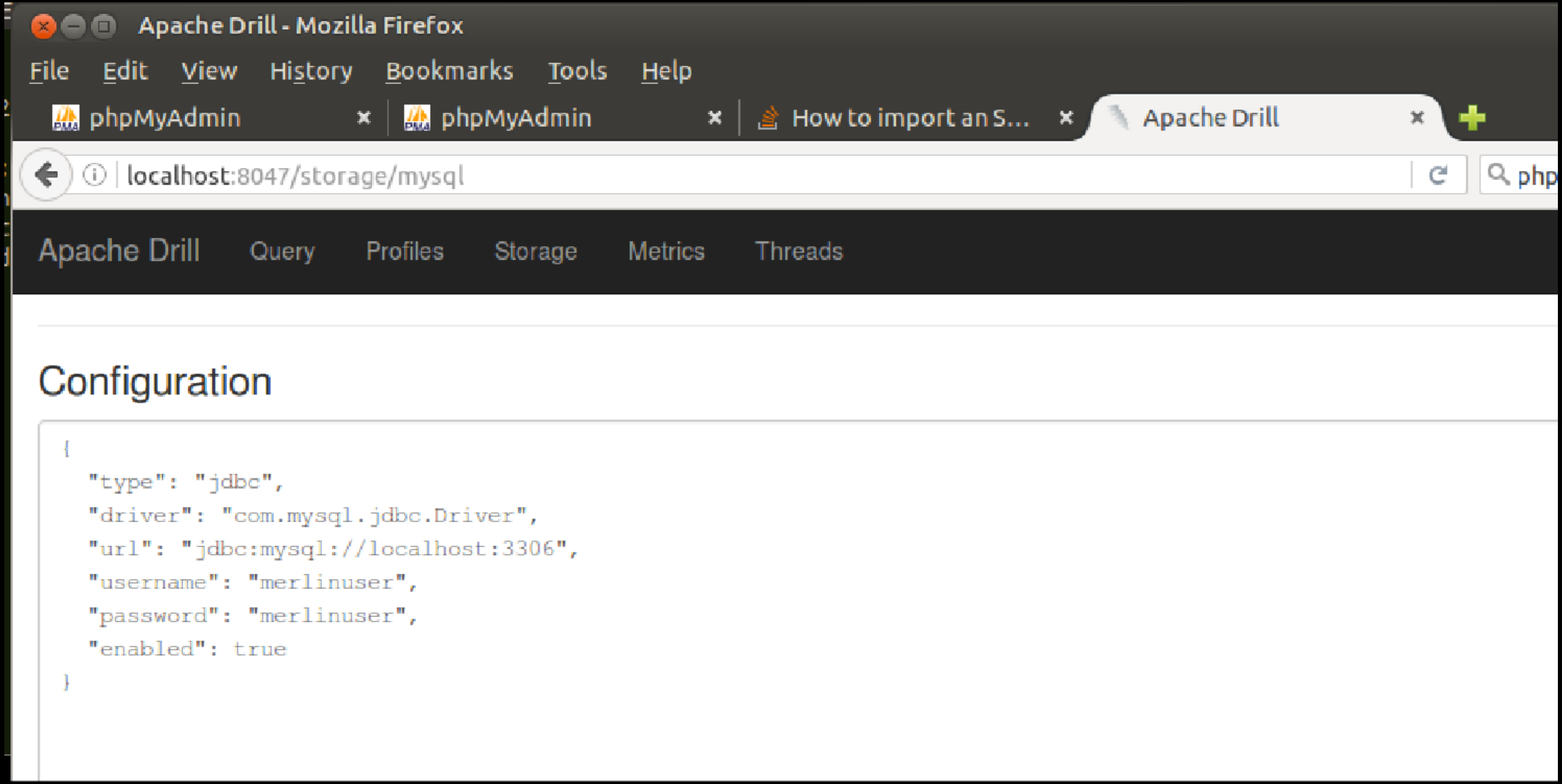
Enabled Storage Plugins

cp	Update	Disable
dfs	Update	Disable

Disabled Storage Plugins

hbase	Update	Enable
hive	Update	Enable
kudu	Update	Enable
mongo	Update	Enable

Connecting other Data Sources


Apache Drill - Mozilla Firefox
File Edit View History Bookmarks Tools Help
phpMyAdmin
phpMyAdmin
How to import an S...
Apache Drill
localhost:8047/storage/mysql
php
Apache Drill Query Profiles Storage Metrics Threads

Configuration

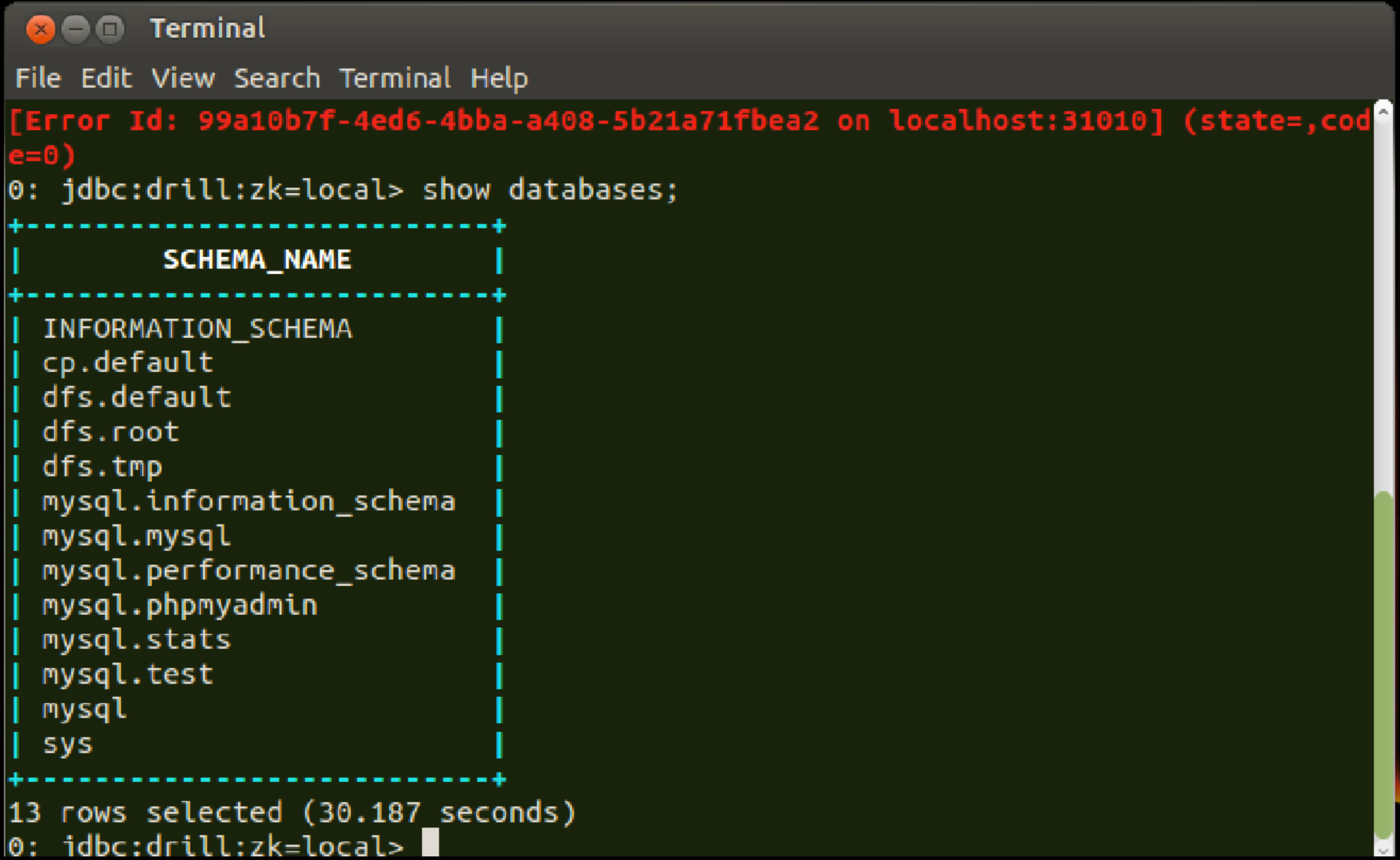
```
{
  "type": "jdbc",
  "driver": "com.mysql.jdbc.Driver",
  "url": "jdbc:mysql://localhost:3306",
  "username": "merlinuser",
  "password": "merlinuser",
  "enabled": true
}
```

MySQL™

Connecting other Data Sources

```
{
 “type”:”jdbc”,
 “driver”:”com.mysql.jdbc.Driver”,
  “url”:”jdbc:mysql://localhost:3306",
  “username”:”merlinuser”,
  “password”:”merlinuser”,
  “enabled”:true
 }
```

Connecting other Data Sources



```
[Error Id: 99a10b7f-4ed6-4bba-a408-5b21a71fbea2 on localhost:31010] (state=,cod
e=0)
0: jdbc:drill:zk=local> show databases;
+-----------------------------+
|         SCHEMA_NAME         |
+-----------------------------+
| INFORMATION_SCHEMA          |
| cp.default                  |
| dfs.default                 |
| dfs.root                    |
| dfs.tmp                     |
| mysql.information_schema    |
| mysql.mysql                 |
| mysql.performance_schema    |
| mysql.phpmyadmin            |
| mysql.stats                 |
| mysql.test                  |
| mysql                       |
| sys                         |
+-----------------------------+
13 rows selected (30.187 seconds)
0: jdbc:drill:zk=local>
```

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

hadoop

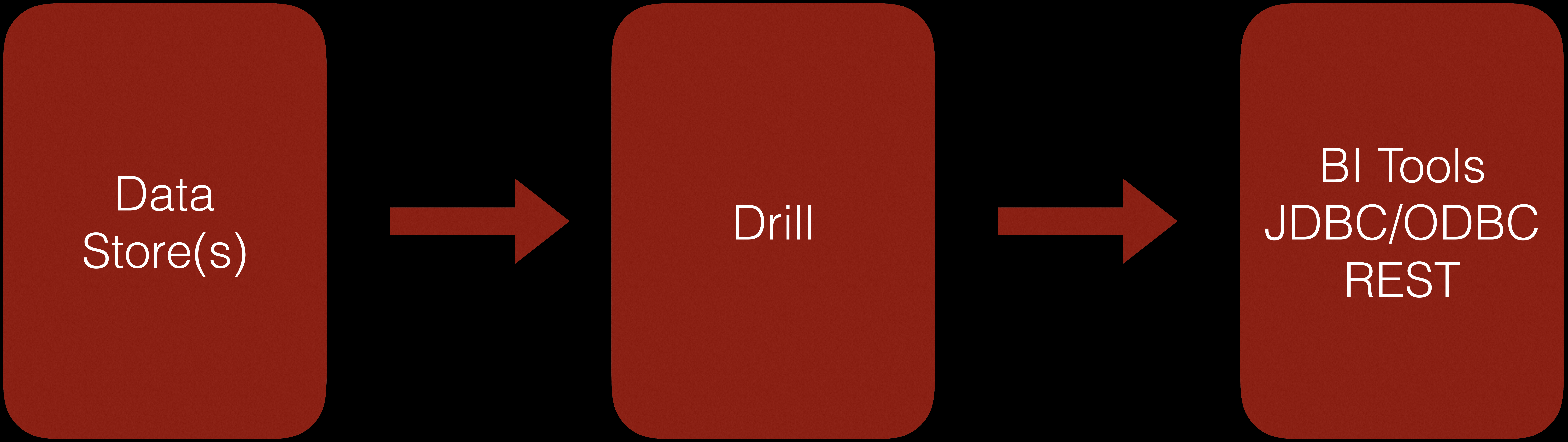
Just like DFS, except you specify a link to the Hadoop namenode.

```
{
  "type": "file",
  "enabled": true,
  "connection": "hdfs://localhost:54310",
  "config": null,
  "workspaces": {
      "demodata": {
      "location": "/user/merlinuser/demo",
      "writable": true,
      "defaultInputFormat": null
    }
  },
SELECT name, SUM( CAST( HR AS INT)) AS HR_Total
FROM hdfs.demodata.`Teams.csvh`
WHERE yearID=1988
GROUP BY name
ORDER BY HR_Total DESC
```

Connecting to Drill

Python

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.drillclass.`*.csvh`
GROUP BY JobTitle
Order By avg_salary DESC
LIMIT 10
''')
```

Connecting to Drill

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

Sergeant

- DBI
- RJDBC
- dplyr

See complete documentation: https://github.com/hrbrmstr/sergeant

```
devtools::install_github("hrbrmstr/sergeant")
```

See complete documentation: https://github.com/hrbrmstr/sergeant

```
library(sergeant)
connection <- drill_connection("localhost")
drill_active(connection)
query_result <- drill_query(connection,
"SELECT * FROM cp.`employee.json` limit 100”
)
```

See complete documentation: https://github.com/hrbrmstr/sergeant

In Class Exercise

Complete the Scripting Demonstration Worksheet.

Questions?

Thank you!!

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