Good Afternoon!

While you are waiting, please clone the class repository.

https://github.com/datadistillr/drillworkshop

Data Exploration with Apache Drill

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

Conventions for this class:

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

Expectations

- Please participate and **ask questions.**
- Please follow along and **TRY OUT** the examples yourself during the class
- All the answers are in the slide decks or GitHub repository, but please try to complete the exercises **without looking at the answers**.
- Have fun!

2 Huge Problems in Data Analytics

- Data is often stored in multiple disparate systems, and querying these systems requires understanding multiple query languages
- Joining data from these multiple disparate systems is difficult, costly and time consuming

Que feriez-vous si toutes vos données parlaient la même langue?

Què faria vostè si totes les seves dades parlaven el mateix idioma?

ሁሉም የእርስዎ ውሂብ ተመሳሳይ ቋንቋ ሲጠቀም ምን ያደርጉ ነበር?

What would you do if all your data spoke the same language?

Что бы вы сделали, если все ваши данные говорили на одном языке?

מה היית עושה אם כל הנתונים שלך מדברים באותה שפה?

اگر تمام داده های شما به همان زبان صحبت می کنند چه کاری انجام می دهید؟

Was würden Sie tun, wenn alle Ihre Daten die gleiche Sprache sprechen würden?

Apache Drill is an SQL Engine for self-describing data.

“Drill is unique among SQL query engines in that it aspires to **query data in the wild**, not just data that has been tamed and domesticated. ”

–Dr. Ted Dunning, Chief Scientist, MapR

Drill acts as a universal
translator for data.

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

Analytics teams spend up to 90% of their time preparing their data.

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

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

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

ETL is costly.

8 Wastes

The 8 Wastes are eight types of process obstacles that get in the way of providing value to the customer.

Defects
Efforts caused by rework, scrap, and incorrect information.

Overproduction
Production that is more than needed or before it is needed.

Waiting
Wasted time waiting for the next step in a process.

Non-Utilized Talent
Underutilizing people's talents, skills, & knowledge.

Transportation
Unnecessary movements of products & materials.

Inventory
Excess products and materials not being processed.

Motion
Unnecessary movements by people (e.g., walking).

Extra-Processing
More work or higher quality than is required by the customer.

© Copyright 2017 GoLeanSixSigma.com. All Rights Reserved.

Drill lets you query anything*, wherever it is*, no matter its size** using standard SQL.

* well.. almost anything
** within reason

APACHE
DRILL
ETL
Data Warehouse

APACHE
DRILL
Apache
Superset™

“Any sufficiently advanced technology is indistinguishable from magic”

—Arthur C. Clarke

You just query the data…
no schema
no centralized metadata

Drill discovers the schema dymagically

Drill is **NOT** just SQL on Hadoop

Drill scales

Drill is open source

Why should you use Drill?

Why should you use Drill?

Drill is **easy to use**

Drill is **easy to use**

Drill uses **standard ANSI SQL**

Drill is **FAST!!**

Drill is Versatile

Drill is Versatile

Natively Drill can query the following file formats:

- CSV or any delimited data
- Avro
- JSON
- Hadoop Sequence Files
- Parquet
- Apache log files
- PCAP/PCAP-NG

Plugins exist for:

- Log files
- Syslog
- Blockchain
- XML
- ACH/Swift
- Excel
- Image metadata
- ESRI Shape Files

Drill is Versatile

Natively Drill can natively query the following data sources:

- Hadoop
- HBase
- Hive
- Kafka
- MapR
- MongoDB
- Open Time Series Database
- Relational databases
- S3/Azure/Google Cloud

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

Drill is Versatile

Drill has APIs to easily build custom functions, format plugins and storage plugins.

How Drill Works

How Drill Works

How Drill Works

How Drill Works

Quick Demo

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;
```

APACHE DRILL

Execution Time:
0.232 seconds!!

NoSQL, No Problem

Drill and Non-Relational Sources

- Drill can natively query NoSQL, non-relational data stores using ANSI SQL.
- In addition the stores listed, Drill can query any data store, relational or otherwise, that has a JDBC compatible driver.
- Drill natively supports querying complex, multi-dimensional datasets.

Currently Supported

mongoDB

APACHE HBASE

OPENTSDB

APACHE KUDU

druid

In Development

Apache Solr

Couchbase

elasticsearch

cassandra

splunk>

Drill enables users to query non-relational stores **without having to learn their query language**.

Demo: Drill & MongoDB

Demo: Drill & MongoDB

```
Apache Drill 1.17.0-SNAPSHOT
"Start your SQL engine."
apache drill> show databases;
+--------------------+
|     SCHEMA_NAME    |
+--------------------+
| cp.default         |
| dfs.default        |
| dfs.drillclass     |
| dfs.gtk            |
| dfs.root           |
| dfs.test           |
| dfs.tmp            |
| information_schema |
| mongo.admin        |
| mongo.config       |
| mongo.local        |
| mongo.zips         |
| sys                |
+--------------------+
13 rows selected (1.828 seconds)
apache drill>
```

Demo: Drill & MongoDB

```
SELECT state, SUM(pop) AS population_total
FROM mongo.zips.zips
GROUP BY state
ORDER BY population_total DESC
```

state	population_total
CA	29754890
NY	17990402
TX	16984601
FL	12686644
PA	11881643
IL	11427576
OH	10846517

Demo: Drill & MongoDB

SELECT * FROM mongo.zips.zips

_id	city	loc	pop	state
1002	CUSHMAN	[-72.51565,42.377017]	36963	MA
1005	BARRE	[-72.108354,42.409698]	4546	MA
1007	BELCHERTOWN	[-72.410953,42.275103]	10579	MA
1008	BLANDFORD	[-72.936114,42.182949]	1240	MA
1001	AGAWAM	[-72.622739,42.070206]	15338	MA
1011	CHESTER	[-72.988761,42.279421]	1688	MA
1010	BRIMFIELD	[-72.188455,42.116543]	3706	MA

Demo: Drill & MongoDB

`SELECT * FROM mongo.zips.zips`

_id	city	loc	pop	state
1002	CUSHMAN	[-72.51565,42.377017]	36963	MA
1005	BARRE	[-72.108354,42.409698]	4546	MA
1007	BELCHERTOWN	[-72.410953,42.275103]	10579	MA
1008	BLANDFORD	[-72.936114,42.182949]	1240	MA
1001	AGAWAM	[-72.622739,42.070206]	15638	MA
1011	CHESTER	[-72.988761,42.279421]	1688	MA
1010	BRIMFIELD	[-72.188455,42.116543]	3706	MA

Drill has extensive support for nested (complex) data structures.

- Drill supports ***lists*** (seen at left) which are roughly equivalent to arrays. These can be accessed by index: `fieldName[0]`

- Drill also supports ***maps*** (AKA Key/Value pairs) which can be accessed using the following method:

`<table>.<map>.<key>`

All these complex data sources can be nested.

Demo: Drill & MongoDB

```
SELECT loc[0] as longitude,
loc[1] as latitude,
pop
FROM mongo.zips.zips
```

longitude	latitude	pop
-72.51565	42.377017	36963
-72.108354	42.409698	4546
-72.410953	42.275103	10579
-72.936114	42.182949	1240
-72.622739	42.070206	15338
-72.988761	42.279421	1688
-72.188455	42.116543	3706
-72.833309	42.38167	177
-72.679921	42.264319	16864

Demo: Drill & MongoDB

```
SELECT *
FROM mongo.samples.city_inspections
```

business_name	date	result	sector	address	certificate_numl
BISHWANATH BISWAS	2015-02-25T00:00:00	No Violation Issued	Mobile Food Vendor - 881	{"city":"ELMHURST","zip":11373.0,"number":4157.0,"street":"75TH ST"}	9287092
MICHAEL GOMEZ RANGHALL	2015-02-10T00:00:00	No Violation Issued	Locksmith - 062	{"city":"QUEENS VLG","zip":11427.0,"number":8823.0,"street":"214TH ST"}	9278914
NEW FINEST BUILDERS INC	2015-12-01T00:00:00	Violation Issued	Home Improvement Contractor - 100	{"city":"BROOKLYN","zip":11234.0,"number":1320.0,"street":"E 55TH ST"}	5381196
H & H FURNITURE	2015-04-29T00:00:00	Violation Issued	Home Improvement Contractor - 100	{"city":"S RICHMOND HL","zip":11419.0,"number":11420.0,"street":"101ST AVE"}	5347333
ATLIXCO DELI GROCERY INC.	2015-02-20T00:00:00	No Violation Issued	Cigarette Retail Dealer - 127	{"city":"RIDGEWOOD","zip":11385.0,"number":1712.0,"street":"MENAHAN ST"}	9278806
AZMY KIROLES	2015-12-23T00:00:00	No Violation Issued	Mobile Food Vendor - 881	{"city":"JERSEY CITY","zip":7306.0,"number":160.0,"street":"VROOM ST"}	9318079
BUILDING TO SERVE INC.	2015-07-22T00:00:00	Violation Issued	Home Improvement Contractor - 100	{"city":"JAMAICA","zip":11432.0,"number":17939.0,"street":"HILLSIDE AVE"}	5382334

Demo: Drill & MongoDB

Calculate Fails per City

```
SELECT
inspections.address.city as city, COUNT(*) as fail_count
FROM mongo.samples.city_inspections as inspections
WHERE result = 'Violation Issued' OR result = 'Fail'
GROUP BY city
ORDER BY fail_count DESC
```

city	fail_count
BROOKLYN	4333
NEW YORK	3807
BRONX	2426
STATEN ISLAND	556
JAMAICA	529
ASTORIA	385
JACKSON HEIGHTS	265
CORONA	260
FLUSHING	229
LONG ISLAND CITY	122
RIDGEWOOD	119

```
{
  "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

Drill supports standard SQL joins

NoSQL, No Problem

```
SELECT t.address.zipcode AS zip, count(name) AS rests
FROM `restaurants` t
GROUP BY t.address.zipcode
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

Drill supports standard SQL joins

Join Type	Supported
INNER	✓
LEFT	✓
LEFT OUTER	✓
RIGHT	✓
RIGHT OUTER	✓
FULL	✓
FULL OUTER	✓
LATERAL	✓
CROSS (Cartesian Product)	✓
UNION	✓

Any data Drill can query, can be joined

Simple Join Example

```
SELECT DISTINCT getCityName(pcap.src_ip) as cityName,
getCountryName(pcap.src_ip) as countryName,
getLatitude(pcap.src_ip) as latitude,
getLongitude(pcap.src_ip) as longitude,
getPostalCode(pcap.src_ip) as postal_code,
zips.pop as population
FROM dfs.test.`attack-trace.pcap` as pcap
INNER JOIN mongo.zips.zips AS zips
ON zips._id = getPostalCode(pcap.src_ip)
```

cityName	countryName	latitude	longitude	postal_code	population
Philadelphia	United States	40.0634	-74.999	19114	31199
San Jose	United States	37.3391	-121.9016	95110	17437

In this example, we are:

1. Geo-locating IP addresses and extracting lat/long, city, country and post code from raw PCAP

2. Joining this data with the ZIPCODE database in MongoDB and getting the population column

More Complex Join (Or is it?)

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 `hdfs.test.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

Questions?

Installing & Configuring Drill

Embedded

Distributed

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

Drill Requirements

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

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

Starting Drill

Embedded Mode: For use on a standalone system

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

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

```
docker run -i
--name drill-1.18.0
-p 8047:8047
-t apache/drill:1.18.0 /bin/bash
```

Drill’s Command Line Interface

apache-drill-1.6.0 — java -Dlog.path=/Users/cgivre/drill/apache-drill-1.6.0/log/sqlline.log -Dl...

```
Charless-MacBook-Pro:apache-drill-1.6.0 cgivre$ ./bin/drill-embedded
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=512M; s
upport was removed in 8.0
May 18, 2016 10:10:22 PM org.glassfish.jersey.server.ApplicationHandler initia
lize
INFO: Initiating Jersey application, version Jersey: 2.8 2014-04-29 01:25:26..
.
apache drill 1.6.0
"a little sql for your nosql"
0: jdbc:drill:zk=local>
```

Drill’s Command Line Interface

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

apache-drill-1.6.0 — java -Dlog.path=/Users/cgivre/drill/apache-drill-1.6.0/log/sqlline.log -Dlog.query.path...

```
0: jdbc:drill:zk=local>
0: jdbc:drill:zk=local>
0: jdbc:drill:zk=local>
0: jdbc:drill:zk=local>
0: jdbc:drill:zk=local>
0: jdbc:drill:zk=local>
0: jdbc:drill:zk=local> SELECT DISTINCT management_role  FROM cp.`employee.json`;
+-----------------------+
|    management_role    |
+-----------------------+
| Senior Management     |
| Store Management      |
| Middle Management     |
| Store Full Time Staf  |
| Store Temp Staff      |
+-----------------------+
```

Drill Web UI

http://localhost:8047

Drill Web UI

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

Drill Web UI

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

localhost

Apache Drill | Query | Profiles | Storage | Metrics | Threads | Options | Documentation

Show 10 entries Search: Show / hide columns

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

Drill isn’t case sensitive*

* Except when Drill is case sensitive

Drill Web UI

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

Drill Web UI

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

Drill Web UI

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

Workspaces in Drill

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

Drill Web UI

SHOW DATABASES

Drill Web UI

SHOW FILES IN <workspace>

localhost

Apache Drill | Query | Profiles | Storage | Metrics | Threads | Logs | Options | Documentation

Show 10 entries

Search: Show / hide columns

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

Workspaces in Drill

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

or

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

In Class Exercise: Create a Workspace

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

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

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

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

Querying Simple Delimited Data

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

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

http://amzn.to/2lID8yi

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

Please open people.csv

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

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

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

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

Tip: Use BACK TICKS around field names in Drill

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

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

Querying Drill

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

Storage Plugin

Workspace

Table

Querying Drill

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

Storage Plugin Workspace Table

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

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

Try it yourself!!

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

Apache Drill Query Profiles Storage Metrics Threads Logs Options Documentation

Show 10 entries Search: Show / hide columns

first_name	last_name	gender
Philip	Richardson	Male
Todd	James	Male
Jimmy	Mendoza	Male

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

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

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

Apache Drill Query Profiles Storage Metrics Threads Logs Options Documentation

Show 10 entries Search: Show / hide columns

first_name	last_name	gender
Dorothy	Fernandez	Female
Frances	Rodriguez	Female
Nancy	Nelson	Female

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

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

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

Apache Drill Query Profiles Storage Metrics Threads Logs Options Documentation

Show 10 entries Search: Show / hide columns

first_name	last_name	gender
Julia	Alvarez	Female
Nicholas	Bishop	Male
Marie	Bradley	Female

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

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

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

Apache Drill Query Profiles Storage Metrics Threads Logs Options Documentation

Show 10 entries Search: Show / hide columns

first_name	fname_length
Clarence	8
Nicholas	8
Theresa	7
Frances	7

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

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

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

Apache Drill Query Profiles Storage Metrics Threads Logs Options Documentation

Show 10 entries Search: Show / hide columns

gender	gender_count
Male	11
Female	9

Joining Datasets

Referred to as an Inner Join

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

Questions?

Querying Drill

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

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

In Class Exercise:
Create a Simple Report

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

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

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

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

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

Querying Drill

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

Drill Data Types

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

Drill Data Types

Simple Data Types

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

Complex Data Types

- Arrays
- Maps

Querying Drill

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

columns[*n*]

Querying Drill

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

Querying Drill

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

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

Querying Drill

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

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

Querying Drill

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

Querying Drill

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

Querying Drill

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

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

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

Aggregate Functions

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

Querying Drill

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

Querying Drill

Query Failed: An Error Occurred

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

Querying Drill

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

Querying Drill

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

AnnualPay has extra characters

AnnualPay is a string

Querying Drill

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

In Class Exercise:
Clean the field.

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

Complete documentation can be found here:

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

In Class Exercise:
Clean the field.

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

Complete documentation can be found here:

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

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

Drill Data Types

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

cast(<expression> AS <data type>)

In Class Exercise:
Convert to a number

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

Complete documentation can be found here:

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

In Class Exercise:
Convert to a number

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

Complete documentation can be found here:

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

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

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

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

localhost

Apache Drill | Apache Drill

Apache Drill Query Profiles Storage Metrics Threads Options Documentation

Show 10 entries Search: Show / hide columns

JobTitle	avg_salary	number
STATE'S ATTORNEY	238772.0	1
Police Commissioner	211785.0	1
Executive Director V	178900.0	1
MAYOR	167449.0	1
DIRECTOR PUBLIC WORKS	166500.0	1

TO_NUMBER(<field>, <format>)

TO_NUMBER(<field>, <format>)

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

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

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

Complete documentation can be found here:

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

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

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

Complete documentation can be found here:

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

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

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

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

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

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

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

customer_id	item_count	amount_spent	dir0
1169	2	1.05	2013
813	4	9.76	2013
373	1	6.69	2013
877	3	6.28	2013
959	4	1.74	2013

dir***n*** accesses the subdirectories

dir*n* accesses the subdirectories

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

Directory Functions

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

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

In Class Exercise:

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

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

In Class Exercise:

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

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

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

Questions?

Exercises

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.

Exercises

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
```

Exercises

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
```

Exercises

(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`
```

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	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

Previous 1 Next

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

Previous 1 Next

```
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	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

Previous 1 Next

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

Previous 1 Next

```
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

Previous 1 Next

```
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

Previous 1 Next

```
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%3DCharles+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?

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`
WHERE regexp_matches(`connection_client_host`, '(\d{1,3}\.)
{3}\d{1,3}')
```

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`
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
```

Show 10 entries Search: Show / hide columns

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

Connecting other Data Sources

Connecting other Data Sources

Connecting other Data Sources

Connecting other Data Sources

MySQL™

Connecting other Data Sources

```
{
  "type": "jdbc",
  "driver": "com.mysql.cj.jdbc.Driver",
  "url": "jdbc:mysql://localhost:3306/?serverTimezone=EST5EDT",
  "username": "<username>",
  "password": "<password>",
  "caseInsensitiveTableNames": false,
  "sourceParameters": {},
  "enabled": true
}
```

Connecting other Data Sources

Terminal

File Edit View Search Terminal Help

```
[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 REST APIs

Connecting to REST APIs

- Drill allows you to directly query REST APIs through Drill's interface
- Makes pulling reference data very easy
- Supports most APIs, as long as the API returns JSON or CSV.

```
{
  "type": "http",
  "cacheResults": false,
  "connections": {
    "sunrise": {
      "url": "https://api.sunrise-sunset.org/",
      "method": "GET",
      "headers": null,
      "authType": "none",
      "userName": null,
      "password": null,
      "postBody": null,
      "params": null,
      "dataPath": null,
      "requireTail": true,
      "inputType": "json"
    },
     "covid": {
       "url": "https://covidtracking.com/api/v1/",
       "method": "GET",
       "headers": null,
       "authType": "none",
       "userName": null,
       "password": null,
       "postBody": null,
       "params": null,
       "dataPath": null,
       "requireTail": true,
       "inputType": "json"
     }
 },
   "timeout": 100,
   "proxyHost": null,
   "proxyPort": 0,
   "proxyType": "direct",
   "proxyUsername": null,
   "proxyPassword": null,
   "enabled": true
 }
```

Querying a REST API

```
SELECT *
FROM api.covid.`states/daily.json`
```

API Name from config

Additional URL Args

Querying a REST API

```
SELECT *
FROM api.covid.`states/daily.json`
```

date	state	positive	negative	totalTestResultsSource	totalTestResults	hospitalizedCurrently	onVentilatorCurrently	recovered	dataQualityGrade	lastUpdateEt	c
20201027	AK	14798	562631	totalTestsViral	577429	81	8	7016	A	10/27/2020 03:59	2
20201027	AL	186437	1153448	totalTestsViral	1312887	1001	null	74439	A	10/27/2020 11:00	2
20201027	AR	107679	1203175	totalTestsViral	1303236	660	89	96322	A+	10/27/2020 00:00	2
20201027	AS	0	1616	totalTestsViral	1616	null	null	null	D	10/1/2020 00:00	2
20201027	AZ	240122	1490040	totalTestsPeopleViral	1724647	861	100	40169	A+	10/27/2020 00:00	2
20201027	CA	904198	17222851	totalTestsViral	18127049	3000	null	null	B	10/27/2020 02:59	2

In Class Exercise

- Using the configuration from the previous slides, create a storage plugin for the COVID tracking API
- Write a query which returns the date in date format and the hospitalization and death counts.
- EXTRA CREDIT: Produce a weekly or monthly summary of the above data

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