

**MAPR** Academy

**Data Analysis with Apache Drill**

Lesson 1: SQL Queries with Drill

© 2015 MapR Technologies **MAPR** 1





What is Drill?



© 2015 MapR Technologies **MAPR** 2

- Apache Drill is a query engine for Big Data exploration.





What is Drill?



© 2015 MapR Technologies  3

- Apache Drill uses standard ANSI SQL to query a variety of structured, semi-structured and unstructured data types
- Drill is built with a modular architecture that can expand its access to any additional data types through custom plugins
- Drill is designed from the ground up to support high-performance analysis on the semi-structured and rapidly evolving data coming from modern Big Data applications, while still providing the familiarity and ecosystem of industry standard ANSI SQL.





## What is Drill?



© 2015 MapR Technologies  4

- Can access drill through JDBC/ODBC, REST, C++ and JAVA API
- out of box ODBC client tool called Drill explorer (ODBC) and out of box command line tool called SQLLine (JDBC)





## Learning Goals

- ▶ Perform Familiar SQL Queries: Structured Data
- ▶ Perform Familiar SQL Queries: Range of Data Types
- ▶ Demonstrate Drill & Semi-structured Data

© 2015 MapR Technologies  5

After you have finished this lesson, you will be able to use Drill to:

- Perform familiar SQL queries on structured data
- Perform familiar SQL queries on a range of data types
- Demonstrate how Drill can be used to explore semi-structured data



 Use Case

© 2015 MapR Technologies  6

- Big data exploration with Drill is used in a variety of industries.
- Some common uses include improving information security, finding new opportunities for financial investment, analyzing medical information and examining retail data to increase sales and improve the customer experience.





Use Case

© 2015 MapR Technologies **MAPR** 7

- Large retail businesses, such as the The Big Office Supply Company, often employ data analysts to study their customer activity and purchasing habits.
- From this analysis, they hope to identify such advantages as new sales opportunities and product enhancements based on customer feedback.
- At the Big Office Supply Company, the first thing we want our data analysts to do is to help us expand our sales opportunities.
- We are a business that sells office supplies, after all, and we need to sell more of these supplies to expand our business.





## Familiar SQL Queries: Structured Data



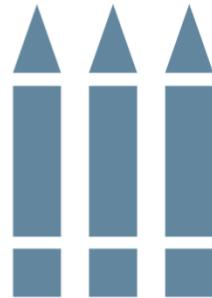
© 2015 MapR Technologies **MAPR** 8

- Our marketing department wants to begin a new sales promotion
- We need to help them decide when and where to focus promotional materials .





## Familiar SQL Queries: Structured Data



© 2015 MapR Technologies **MAPR** 9

We've asked our data analysts to tell us:

- the top grossing month of sales
- the rank of countries based on their gross sales in that month
- and the top 10 products based on total sales volume

Once we have this information, we can then determine what is the best time to start our promotion, and what areas we should focus on. We can then build a promotion using our top selling products, and target the locations and time to get our best expected ROI





## Familiar SQL Queries: Structured Data

```
orders(order_id BIGINT, month STRING, cust_id BIGINT,  
state STRING, prod_id BIGINT, order_total INT)
```

order_id	month	cust_id	state	prod_id	order_total
67212	June	10001	ca	909	13
70302	May	10004	ga	420	11

© 2015 MapR Technologies 10

## 1. Build table TABLE orders(order\_id BIGINT, month STRING, cust\_id BIGINT, country STRING, prod\_id BIGINT, order\_total INT)

**(Add column header and a couple of rows of data as spoken)**

- Our company has offloaded its customer data from a transactional Oracle system into a Hive table on a Hadoop cluster.
- This table, called ‘orders,’ includes columns for the order ID number, the month the order was placed, the customer ID of the person who placed the order, the country where the customer lives, the product ID of the order and the total amount of the order.
- Our data analysts will use Drill to query this table and find the information that we asked them for





## Familiar SQL Queries: Structured Data

Query 1

```
SELECT `month`, SUM(order_total) as sales  
FROM hive.orders  
GROUP BY `month`  
ORDER BY sales desc;
```

Query 2

```
SELECT `month`, `state`, SUM(order_total) as sales  
FROM hive.orders  
WHERE `month`='June'  
GROUP BY `month`, `state`  
ORDER BY sales desc;
```

Query 3

```
SELECT `prod_id`, SUM(order_total) as sales  
FROM hive.orders  
GROUP BY `prod_id`  
ORDER BY 2 desc limit 10;
```

© 2015 MapR Technologies  11

### 2. Image of CLI query, and then results

- First, they will make the following query to determine the top month of gross sales ...

### 3. Image of CLI query, and then results

- Then, we will query for the sales total in that month for each state...

### 4. Image of CLI query, and then results

- And finally they will determine the top 10 products based on volume of sales .





Refer to Lab Guide Exercise 1.1



**1. Build table TABLE orders(order\_id BIGINT, month STRING, cust\_id BIGINT, country STRING, prod\_id BIGINT, order\_total INT)**

**(Add column header and a couple of rows of data as spoken)**

- Our company has offloaded its customer data from a transactional Oracle system into a Hive table on a Hadoop cluster.
- This table, called ‘orders,’ includes columns for the order ID number, the month the order was placed, the customer ID of the person who placed the order, the country where the customer lives, the product ID of the order and the total amount of the order.
- Our data analysts will use Drill to query this table and find the information that we asked them for

**2. Image of CLI query, and then results**

- First, they will make the following query to





## Knowledge Check

**Please indicate if the following statement is true or false:**

I can use Drill to easily perform standard ANSI SQL queries on structured data

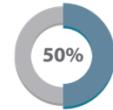
**Please indicate if the following statement is true or false:**

I can use Drill to easily perform standard ANSI SQL queries on structured data. **(TRUE)**





## Familiar SQL Queries: Range of Data Types



© 2015 MapR Technologies  14

- Our marketing department is pleased with the results that they received about sales times and locations, but our data analysts are not satisfied.
- They don't feel that just looking at customer purchases gives us the full picture of who is most interested in our products.





## Familiar SQL Queries: Range of Data Types

```
jallen — root@maprdemo:/mapr/demo.mapr.com/data/nested clicks — 225x56
[{"trans_id":31928,"date":"2014-04-26","time":"12:17:12","user_info":{"cust_id":22526,"device":"IOSS","state":"il"}, "trans_info":{"prod_id":1174,2}, "purch_flag":false}, {"trans_id":31026,"date":"2014-04-28","time":"13:50:29","user_info":{"cust_id":16368,"device":"AOS4.2","state":"nc"}, "trans_info":{"prod_id":1}, "purch_flag":false}, {"trans_id":33848,"date":"2014-04-18","time":"08:44:42","user_info":{"cust_id":23449,"device":"IOSS","state":"oh"}, "trans_info":{"prod_id":1582}, "purch_flag":false}, {"trans_id":32283,"date":"2014-04-18","time":"08:27:47","user_info":{"cust_id":28323,"device":"IOSS","state":"ca"}, "trans_info":{"prod_id":178,47}, "purch_flag":false}, {"trans_id":32284,"date":"2014-04-18","time":"08:27:47","user_info":{"cust_id":28323,"device":"IOSS","state":"ca"}, "trans_info":{"prod_id":178,47}, "purch_flag":false}, {"trans_id":38422,"date":"2014-04-23","time":"01:46:05","user_info":{"cust_id":15957,"device":"IOS7","state":"sc"}, "trans_info":{"prod_id":1528}, "purch_flag":true}, {"trans_id":35898,"date":"2014-04-18","time":"13:28:56","user_info":{"cust_id":28677,"device":"IOS7","state":"ny"}, "trans_info":{"prod_id":1}, "purch_flag":false}, {"trans_id":32421,"date":"2014-04-15","time":"11:00:53","user_info":{"cust_id":23599,"device":"IOS7","state":"il"}, "trans_info":{}}, {"trans_id":32422,"date":"2014-04-15","time":"11:00:53","user_info":{"cust_id":23599,"device":"IOS7","state":"il"}, "trans_info":{}}, {"trans_id":35273,"date":"2014-04-02","time":"01:53:28","user_info":{"cust_id":15340,"device":"IOS7","state":"ca"}, "trans_info":{"prod_id":155,31}, "purch_flag":false}, {"trans_id":30778,"date":"2014-04-13","time":"01:20:05","user_info":{"cust_id":16996,"device":"AOS4.2","state":"oh"}, "trans_info":{"prod_id":149,499,151}, "purch_flag":true}, {"trans_id":32120,"date":"2014-04-28","time":"08:58:19","user_info":{"cust_id":21482,"device":"IOSS","state":"m"}, "trans_info":{"prod_id":1}, "purch_flag":false}, {"trans_id":32086,"date":"2014-04-15","time":"21:00:15","user_info":{"cust_id":15344,"device":"IOS7","state":"ny"}, "trans_info":{"prod_id":127}, "purch_flag":false}, {"trans_id":32087,"date":"2014-04-15","time":"21:00:15","user_info":{"cust_id":15344,"device":"IOS7","state":"ny"}, "trans_info":{"prod_id":127}, "purch_flag":false}, {"trans_id":32955,"date":"2014-04-15","time":"28:41:59","user_info":{"cust_id":15492,"device":"AOS4.4","state":"il"}, "trans_info":{"prod_id":242,8,23,166,26,8,81}, "purch_flag":false}, {"trans_id":32643,"date":"2014-04-26","time":"15:07:00","user_info":{"cust_id":15137,"device":"AOS4.2","state":"tx"}, "trans_info":{"prod_id":1452,19,3,15,954,41}, "purch_flag":false}

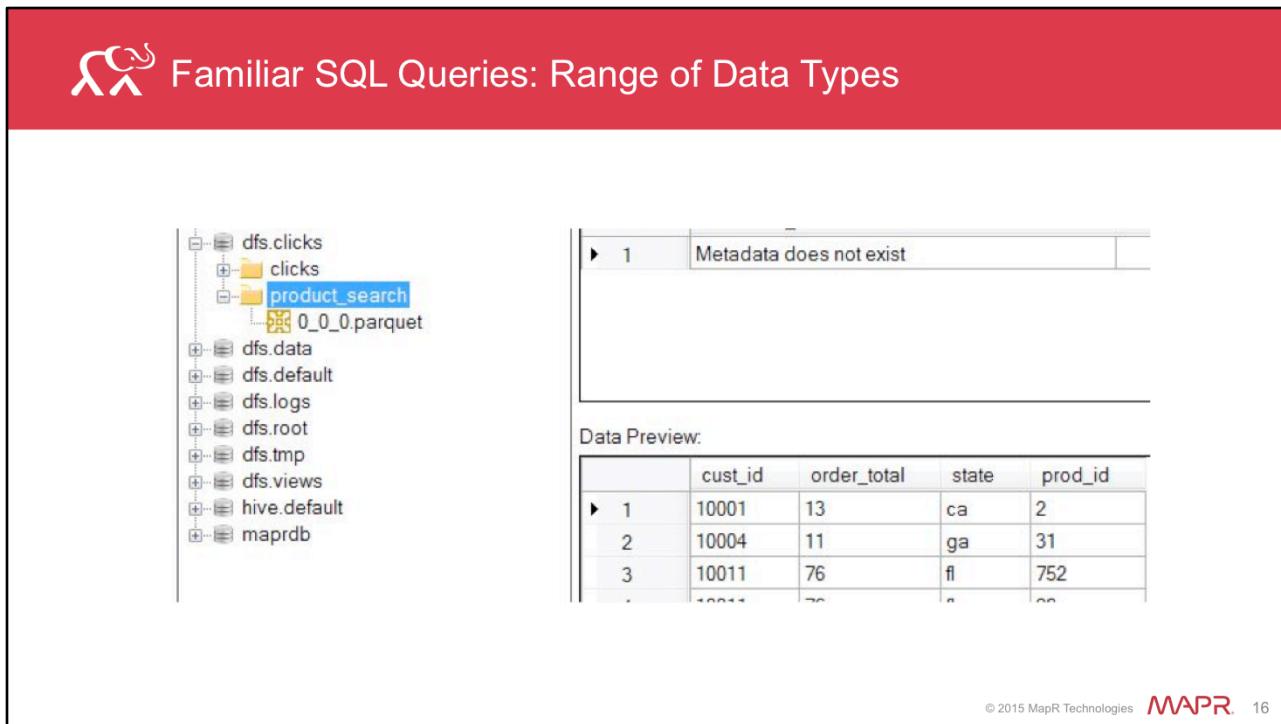
[ @ ○ ○ jallen — root@maprdemo:/mapr/demo.mapr.com/data/flat/logs/2014/6 — ssh — 181x57
{"trans_id":24102,"date":"06/29/2014","time":"05:16:08","cust_id":4,"device":"IOSS","state":"il","camp_id":0,"Keywords":null,"prod_id":133,"purch_flag":false}, {"trans_id":24105,"date":"06/25/2014","time":"01:23:44","cust_id":3414,"device":"IOSS","state":"ut","camp_id":15,"Keywords":null,"prod_id":95,"purch_flag":false}, {"trans_id":24109,"date":"06/17/2014","time":"19:32:51","cust_id":43,"device":"IOS7","state":fl,"camp_id":9,"Keywords":null,"prod_id":247,"purch_flag":false}, {"trans_id":24124,"date":"06/22/2014","time":05:39:40,"cust_id":4,"device":"IOSS","state":ny,"camp_id":7,"Keywords":null,"prod_id":14,"purch_flag":false}, {"trans_id":24125,"date":"06/11/2014","time":06:45:09,"cust_id":814,"device":IOSS,"state":wa,"camp_id":13,"Keywords":null,"prod_id":287,"purch_flag":false}, {"trans_id":24128,"date":"06/18/2014","time":16:44:37,"cust_id":1919,"device":IOSS,"state":tn,"camp_id":0,"Keywords":null,"prod_id":91,"purch_flag":false}, {"trans_id":24129,"date":"06/30/2014","time":03:39:47,"cust_id":166537,"device":IOSS,"state":ca,"camp_id":13,"Keywords":null,"prod_id":375,"purch_flag":false}, {"trans_id":24162,"date":"06/21/2014","time":13:18:18,"cust_id":10,"device":AOS4.3,"state":oh,"camp_id":12,"Keywords":null,"prod_id":380,"purch_flag":false}, {"trans_id":24164,"date":"06/12/2014","time":08:41:21,"cust_id":8,"device":IOSS,"state":il,"camp_id":10,"Keywords":null,"prod_id":33,"purch_flag":false}, {"trans_id":24166,"date":"06/15/2014","time":22:30:54,"cust_id":0,"device":IOSS,"state":oh,"camp_id":0,"Keywords":null,"prod_id":21,"purch_flag":false}, {"trans_id":24177,"date":"06/27/2014","time":16:47:21,"cust_id":278,"device":AOS4.4,"state":oh,"camp_id":9,"Keywords":null,"prod_id":430,"purch_flag":false}, {"trans_id":24178,"date":"06/14/2014","time":22:10:50,"cust_id":12,"device":IOSS,"state":az,"camp_id":14,"Keywords":null,"prod_id":227,"purch_flag":false}, {"trans_id":24187,"date":"06/27/2014","time":13:37:15,"cust_id":2580,"device":IOSS,"state":md,"camp_id":1,"Keywords":null,"prod_id":0,"purch_flag":false}]


```

© 2015 MapR Technologies 15

- As users visit our site and view our product pages, the server collects both click data in a nested JSON format and log files as flat text.





The screenshot shows the MapR interface. On the left, there is a file tree view with the following structure:

```

    /dfs.clicks
      - clicks
      - product_search
        - 0_0_0.parquet
    /dfs.data
    /dfs.default
    /dfs.logs
    /dfs.root
    /dfs.tmp
    /dfs.views
    /hive.default
    /maprdb
  
```

The 'product\_search' folder and its '0\_0\_0.parquet' file are highlighted in blue. To the right of the file tree is a data preview window. At the top, it says '1 Metadata does not exist'. Below that is a table titled 'Data Preview:' with the following data:

	cust_id	order_total	state	prod_id
1	10001	13	ca	2
2	10004	11	ga	31
3	10011	76	fl	752
...	...	...	...	...

At the bottom right of the interface, it says '© 2015 MapR Technologies MAPR 16'.

- We also collect all user product search requests, which are regularly converted into parquet format.
- Parquet is a format designed to improve query speed, and will be described in more detail in later lessons.
- Our data analysts want to query all of these files for additional insights

### Nested JSON

[/mapr/demo.mapr.com/data/nested/clicks/clicks.json](http://mapr/demo.mapr.com/data/nested/clicks/clicks.json)

### Flat Log

[/mapr/demo.mapr.com/data/flat/logs/2014/6/log.json](http://mapr/demo.mapr.com/data/flat/logs/2014/6/log.json)

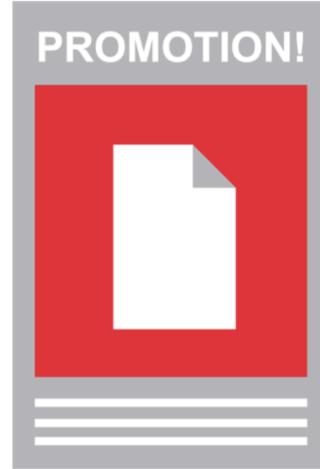
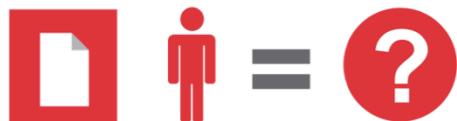
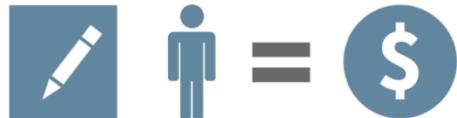
### Parquet

[/mapr/demo.mapr.com/data/nested/cust\\_orders/0\\_0\\_0.parquet](http://mapr/demo.mapr.com/data/nested/cust_orders/0_0_0.parquet)





## Familiar SQL Queries: Range of Data Types



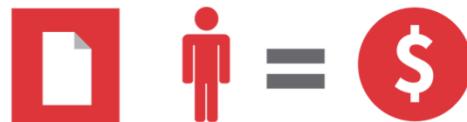
© 2015 MapR Technologies **MAPR** 17

- First, they will look at the search data to see what are the most popular items searched over the last year, and target those items in our promotion
- Next, they use the click and log data to look at who is viewing our site, but did not buy the product at which they were looking.
- Since we know they have interest in that specific product, we can target our promotion to that item and expect a higher rate of return





## Familiar SQL Queries: Range of Data Types



© 2015 MapR Technologies **MAPR** 18

- Also, they want to look at exactly what different people are purchasing
- When we couple this with information from our products database, we can then provide specific, targeted coupons for upgrades or consumables to match those products





## Familiar SQL Queries: Range of Data Types



### Without Drill:

- Enlist help from Engineer
- Lengthy ETL process
- Increased effort, cost & time

© 2015 MapR Technologies **MAPR** 19

- Normally, our data analysts would be out of luck trying to use SQL on these types of files.
- To query data of this type, they would either need to enlist a data engineer to help them write a MapReduce program, or perform a lengthy ETL process to flatten the data and create a centralize schema for it.





## Familiar SQL Queries: Range of Data Types



### With Drill:

- Easy query
- Discover data on the fly
- No upfront schema definitions
- Manipulate nested data without flattening at design or run time
- Standard ANSI SQL
- Better expected results

© 2015 MapR Technologies **MAPR** 20

- Drill makes querying semi-structured data extremely easy.
- Drill can discover the schema of the data on the fly, allowing our data analysts to query the data without creating any upfront schema definitions.
- Drill's flexible data model makes it easy to query and manipulate nested data without any flattening at design time nor at run time.
- Our data analysts can use Drill to query our JSON and parquet files using the standard ANSI SQL that they are familiar with, just like they did with our structured database.
- Because of this, Drill allows our data analysts to quickly and easily gain new insights into our data, and get much better expected results from our new promotion

We will discuss more about how Drill discovers the schema of data on the fly in the next lesson





## Familiar SQL Queries: Range of Data Types

```
products(HBASE_ROW_KEY, details:name,
details:category, pricing:price)
```

HBASE_ROW_KEY	name	category	price
5	12-1/2 Diameter Round Wall Clock	Office Furnishings	20
6	12 Colored Short Pencils	Pens & Art Supplies	3

© 2015 MapR Technologies 21

### **1. show JSON file and flat file directory tree, zoomed out from server icon**

- The web click and log are stored on our servers. The JSON click files are large files of nested data, while the log files are flat text and are stored in folders saved by year, month and day. Drill can access these files directly, without the need to move them to into a data warehouse or even onto our Hadoop cluster\

### **2. The parquet files are a nested columnar format, and are offloaded into our Hadoop file system daily.**

- The parquet files are a nested columnar format, and are offloaded into our Hadoop file system daily.





## Familiar SQL Queries: Range of Data Types

Query 1

```
SELECT *
FROM (SELECT name, count(*)
      FROM (SELECT
              FLATTEN(clicks.trans_info.prod_id) name
             FROM dfs.`data/nested/clicks/clicks.parquet` clicks) a
        GROUP BY name
       ORDER BY 2 desc) b limit 4;
```

Query 2

```
SELECT clicks.user_info.cust_id
FROM dfs.`data/nested/clicks/clicks.json` clicks
WHERE clicks.trans_info.purch_flag = 'false';
```

© 2015 MapR Technologies 22

Our Data Analysts will use Drill to perform similar SQL queries as before, this time using the nested JSON data, the flat text files, the parquet files, the HBase table, and also combining these different data types into a single query, on the fly.

### **Query 4. Image of CLI query, and then results**

First, they will use the following query to look at the parquet files, and determine the top 4 searched for items over the last year.

### **Query 5. Image of CLI query, and then results**

- They will then make the following query on nested JSON and flat text files to see who visited our site, but did not buy the product or products that they were viewing

### **Query 6. Image of CLI query, and then results**





## Familiar SQL Queries: Range of Data Types

### Query 3

```
SELECT cust_clk.cust_name, prod.name as prod_name, cust_clk.`date`  
FROM hive.products as prod,  
(SELECT cast(cust.personal.name as varchar(40)) as cust_name, clk.prod_id, clk.`date`  
FROM maprdb.customers as cust,  
(SELECT cast(clicks.user_info.cust_id as bigint) as cust_id, cast  
(FLATTEN(clicks.trans_info.prod_id) as bigint) as prod_id, clicks.trans_info.purch_flag as  
purch_flag, to_date(clicks.`date`, 'yyyy-mm-dd') as `date`  
FROM dfs.clicks.`/clicks@clicks.json` as clicks) as clk  
WHERE cast(cust.row_key as bigint) = clk.cust_id and clk.purch_flag = 'false') as cust_clk  
WHERE cust_clk.prod_id = prod.prod_id  
and cust_clk.`date` between '2014-01-01' and '2014-03-31'
```

© 2015 MapR Technologies 23

Our Data Analysts will use Drill to perform similar SQL queries as before, this time using the nested JSON data, the flat text files, the parquet files, the HBase table, and also combining these different data types into a single query, on the fly.

### Query 4. Image of CLI query, and then results

First, they will use the following query to look at the parquet files, and determine the top 4 searched for items over the last year.

### Query 5. Image of CLI query, and then results

- They will then make the following query on nested JSON and flat text files to see who visited our site, but did not buy the product or products that they were viewing

### Query 6. Image of CLI query, and then results





Refer to Lab Guide Exercise 1.2



**1. Show JSON file and flat file directory tree, zoomed out from server icon**

- The web click and log are stored on our servers.
- The JSON click files are large files of nested data, while the log files are flat text and are stored in folders saved by year, month and day.
- Drill can access these files directly, without the need to move them to into a data warehouse or even onto our Hadoop cluster .

**2. Show parquet format zoomed out from a cluster icon**

- The parquet files are a nested columnar format, and are offloaded into our Hadoop file system daily.

**3. Build table products**

**(HBASE\_ROW\_KEY, details:name, details:category, pricing:price)**





## Knowledge Check

**Please select all of the answers that apply**

With Drill, we can use ANSI SQL to query:

- Structured data in Hive or HBase
- Unstructured JSON or parquet data
- Flat text files
- Nested data
- Any combination of these data types

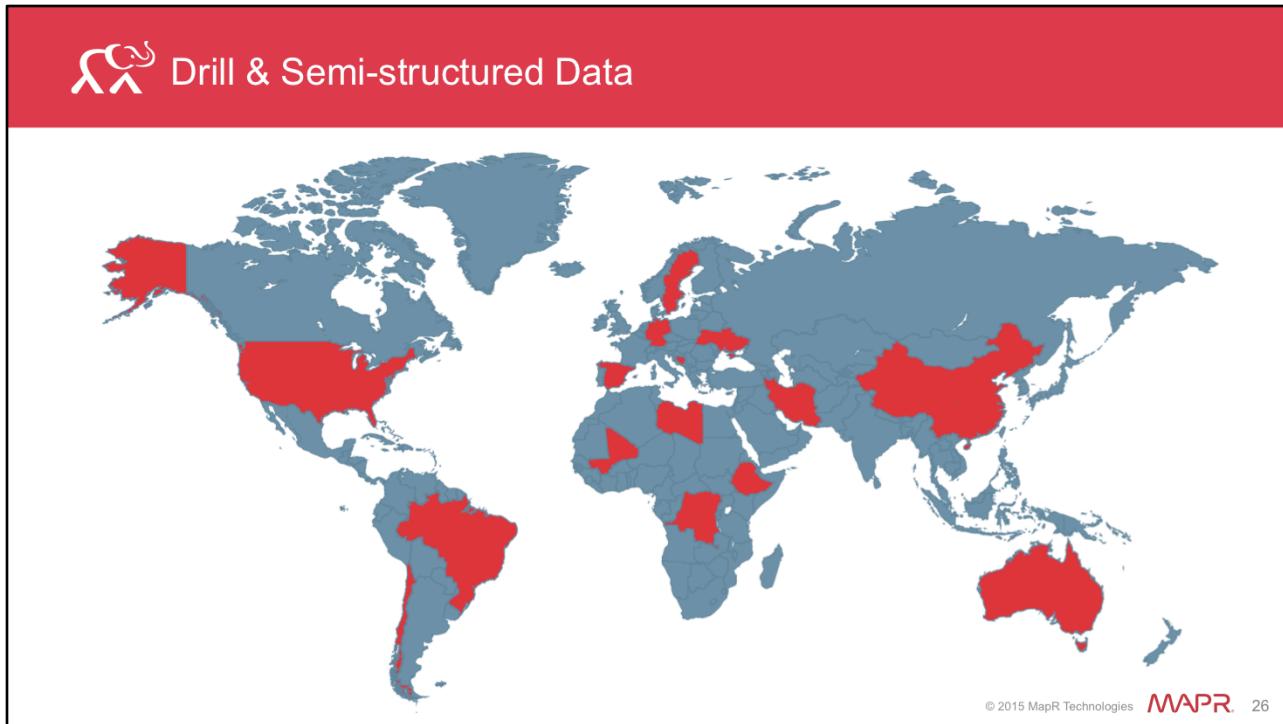
© 2015 MapR Technologies  25

**Please select all of the answers that apply**

With Drill, we can use ANSI SQL to query:

- Structured data in Hive or HBase
  - Unstructured JSON or parquet data
  - Flat text files
  - Nested data
  - Any combination of these data types
- (Check all)**





- Our data analysts have done a great job using drill to easily query a variety of data sources to help us target our promotions to get a much higher rate of return for our marketing dollars.
- Yet we have just begun to learn how drill can be used to access our company's data.

Drill & Semi-structured Data

Tableau

MicroStrategy

SAS  
THE POWER TO KNOW

	dir0	trans_id	date	time	cust_id	device	state	camp_id
1	2	12115	02/23/2013	19:48:24	3	IOSS	az	5
2	2	12127	02/26/2013	19:42:03	11459	IOSS	wa	10
3	2	12138	02/09/2013	05:49:01	1	IOSS	ca	7
4	2	12139	02/23/2013	06:58:20	1	AOS4.4	ms	7
5	2	12145	02/10/2013	10:14:56	10	IOSS	mi	6
6	2	12157	02/15/2013	02:49:22	102	IOSS	ny	5
7	2	12176	02/19/2013	08:39:02	28	IOSS	or	0
8	2	12194	02/24/2013	08:26:17	125445	IOSS	ar	0
9	2	12236	02/05/2013	01:40:05	10	IOSS	nj	2
10	2	12249	02/03/2013	04:45:47	21725	IOSS	nj	5

© 2015 MapR Technologies 27

- In addition to writing SQL queries with the command line, drill provides JDBC and ODBC interfaces that can be used to connect to standard BI tools like Tableau, Microstrategy or SAS, and the ODBC driver can be used to access a GUI interface called Drill Explorer

The screenshot shows the Drill Explorer application window. At the top, there's a red header bar with the text "Drill & Semi-structured Data" and the Drill logo. Below the header, the main interface has three main sections: "Schemas" on the left, "Metadata" in the center, and "Data Preview" on the right.

**Schemas:** A tree view showing various schemas and their contents. Visible nodes include cp.default, dfs.clicks, dfs.data, dfs.default, dfs.logs, logs (with sub-nodes 2012, 2013, 2014), dfs.root, dfs.tmp, dfs.views, hive.default, and maprdb.

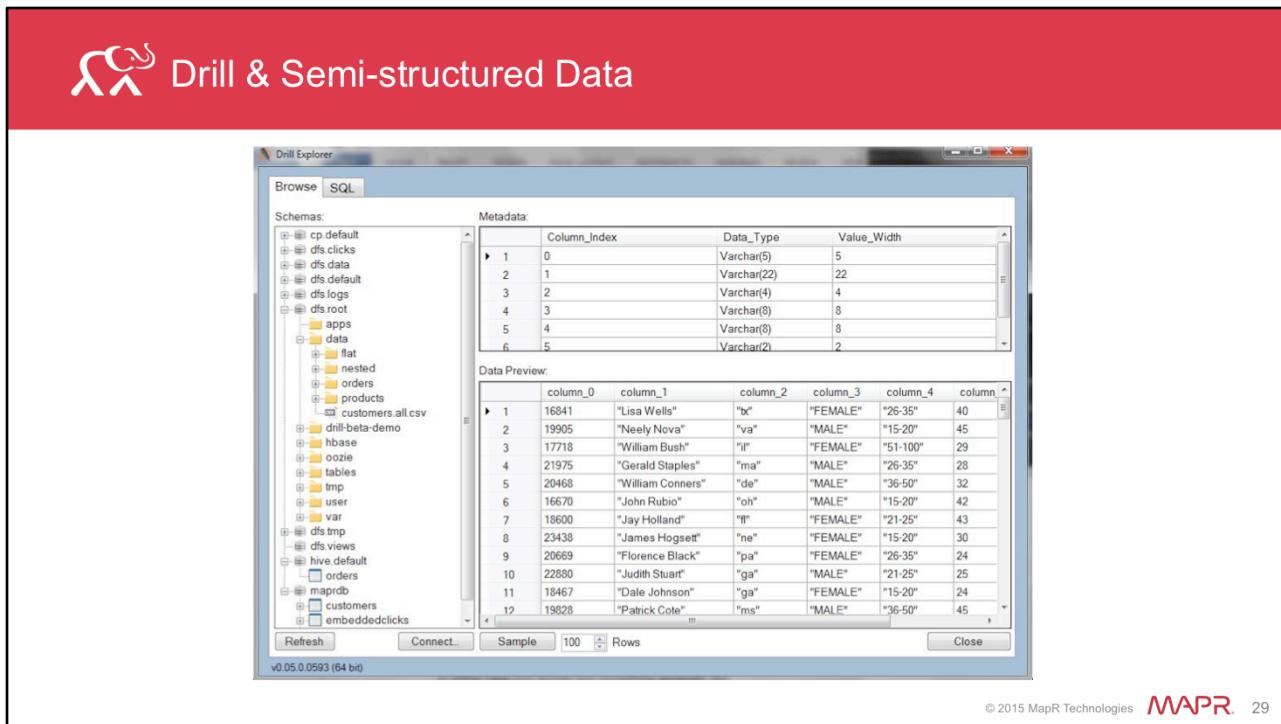
**Metadata:** An empty panel intended for displaying metadata information.

**Data Preview:** A table showing a sample of data from a schema. The columns are dir0, trans\_id, date, time, cust\_id, device, state, and camp\_id. The data consists of 10 rows of transactional records.

	dir0	trans_id	date	time	cust_id	device	state	camp_id
1	2	12115	02/23/2013	19:48:24	3	iOS5	az	5
2	2	12127	02/26/2013	19:42:03	11459	iOS5	wa	10
3	2	12138	02/09/2013	05:49:01	1	iOS6	ca	7
4	2	12139	02/23/2013	06:58:20	1	AOS4.4	ms	7
5	2	12145	02/10/2013	10:14:56	10	iOS5	mi	6
6	2	12157	02/15/2013	02:49:22	102	iOS5	ny	5
7	2	12176	02/19/2013	08:39:02	28	iOS5	or	0
8	2	12194	02/24/2013	08:26:17	125445	iOS5	ar	0
9	2	12236	02/05/2013	01:40:05	10	iOS5	nj	2
10	2	12249	02/03/2013	04:45:47	21725	iOS5	nj	5

At the bottom of the window, there are buttons for Refresh, Connect..., Sample, 100 Rows, and Close. A status bar at the very bottom indicates "v0.05.0.0593 (64 bit)".

- Drill explorer can be used by our data analysts to examine and understand the metadata available in the schema-less and semi-structured data before they query it with visualization tools



The screenshot shows the Drill & Semi-structured Data interface. At the top, there's a red header bar with the text "Drill & Semi-structured Data" and the MAPR logo. Below the header is a "Drill Explorer" window. The window has three main sections: "Schemas" on the left, "Metadata" in the center, and "Data Preview" on the right.

**Schemas:**

- cp default
- dfs clicks
- dfs data
- dfs default
- dfs logs
- dfs root
- data
  - flat
  - nested
  - orders
  - products
  - customers all csv
- drill-beta-demo
- hbase
- oozie
- tables
- tmp
- user
- var
- dfs tmp
- dfs views
- hive default
- maprdb
  - orders
  - customers
  - embeddedclicks

**Metadata:**

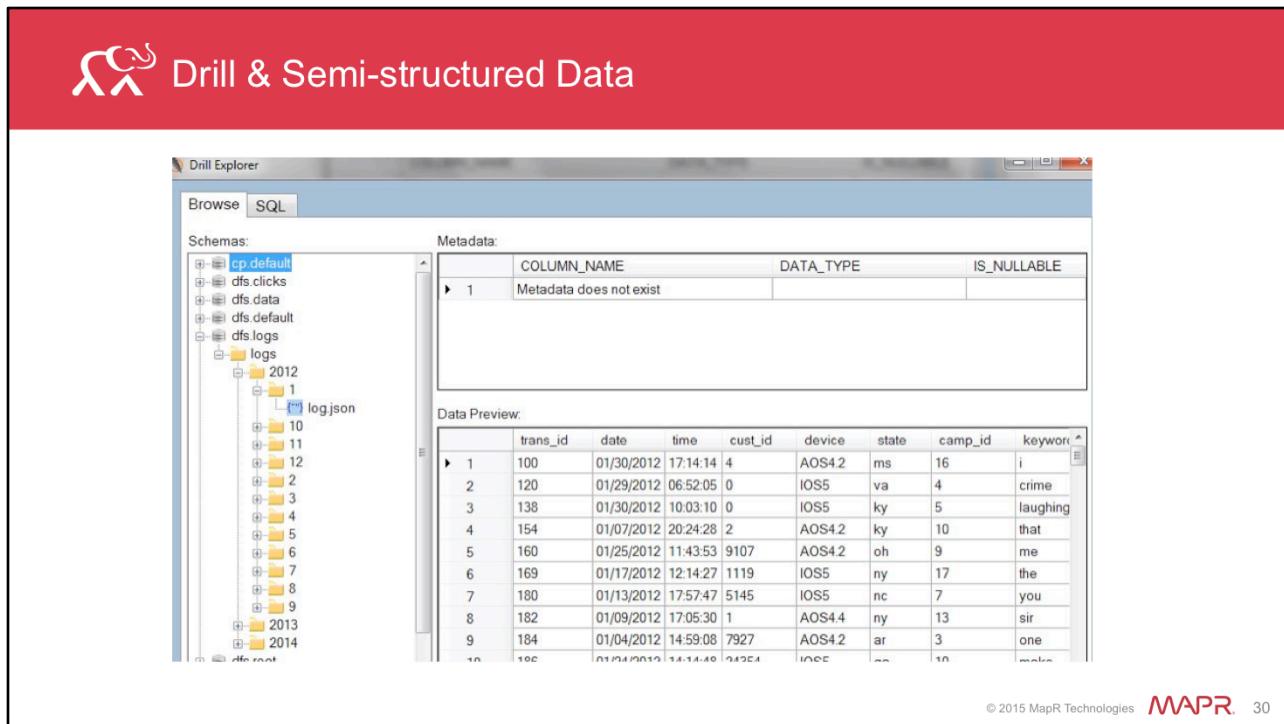
Column_Index	Data_Type	Value_Width
0	Varchar(5)	5
1	Varchar(22)	22
2	Varchar(4)	4
3	Varchar(8)	8
4	Varchar(8)	8
5	Varchar(2)	2

**Data Preview:**

column_0	column_1	column_2	column_3	column_4	column_5
16841	"Lisa Wells"	"b"	"FEMALE"	"26-35"	40
19905	"Neely Nova"	"va"	"MALE"	"15-20"	45
17718	"William Bush"	"lf"	"FEMALE"	"51-100"	29
21975	"Gerald Staples"	"ma"	"MALE"	"26-35"	28
20468	"William Conners"	"de"	"MALE"	"36-50"	32
16670	"John Rubio"	"oh"	"MALE"	"15-20"	42
18600	"Jay Holland"	"f"	"FEMALE"	"21-25"	43
23438	"James Hogsett"	"ne"	"FEMALE"	"15-20"	30
20669	"Florence Black"	"pa"	"FEMALE"	"26-35"	24
22880	"Judith Stuart"	"ga"	"MALE"	"21-25"	25
18467	"Dale Johnson"	"ga"	"FEMALE"	"15-20"	24
19828	"Patrick Cote"	"ms"	"MALE"	"36-50"	45

At the bottom of the Drill Explorer window, there are buttons for Refresh, Connect..., Sample, 100 Rows, and Close. The footer of the interface also includes the MAPR logo and the number 29.

- We can give drill access to multiple data sources, structured, unstructured, known or even unknown



The screenshot shows the Drill Explorer interface with a red header bar containing the text "Drill & Semi-structured Data" and the Drill logo.

The main window displays the following sections:

- Schemas:** A tree view showing available schemas: cp.default, dfs.clicks, dfs.data, dfs.default, and dts.logs. Under dts.logs, there is a folder named logs containing sub-folders for the years 2012, 2013, and 2014. Within the 2012 folder, there are sub-folders labeled 1 through 14, and one file named log.json.
- Metadata:** A table titled "Metadata" with one row showing "COLUMN\_NAME" as "Metadata does not exist", "DATA\_TYPE" as "", and "IS\_NULLABLE" as "".
- Data Preview:** A table titled "Data Preview" showing a sample of data from the log.json file. The columns are trans\_id, date, time, cust\_id, device, state, camp\_id, and keyword. The data rows are as follows:

	trans_id	date	time	cust_id	device	state	camp_id	keyword
1	100	01/30/2012	17:14:14	4	AOS4.2	ms	16	i
2	120	01/29/2012	06:52:05	0	IOS5	va	4	crime
3	138	01/30/2012	10:03:10	0	IOS5	ky	5	laughing
4	154	01/07/2012	20:24:28	2	AOS4.2	ky	10	that
5	160	01/25/2012	11:43:53	9107	AOS4.2	oh	9	me
6	169	01/17/2012	12:14:27	1119	IOS5	ny	17	the
7	180	01/13/2012	17:57:47	5145	IOS5	nc	7	you
8	182	01/09/2012	17:05:30	1	AOS4.4	ny	13	sir
9	184	01/04/2012	14:59:08	7927	AOS4.2	ar	3	one
10	196	01/14/2012	14:14:40	2354	IOS5	ca	10	makes

At the bottom right of the interface, it says "© 2015 MapR Technologies MAPR 30".

- Without the need for knowing the schema for the data, the type of data, without a data engineer writing any code and even without typing a single line of SQL, our data analysts can explore all of these different data sources.

The screenshot shows the Drill Explorer interface. The title bar says "Drill & Semi-structured Data". The left pane is titled "Schemas" and shows a tree structure of data sources: cp.default, dfs.clicks, dfs.data, dfs.default, dfs.logs (with subfolders 2012, 2013, 2014), dfs.root, dfs.tmp, dfs.views, hive.default (with subfolders orders, maprdb, customers, address, loyalty, personal, embeddedclicks, products), and maprdb. The "orders" table under "hive.default" is selected. The right pane has two sections: "Metadata: hive.default.orders" which lists columns: order\_id (BIGINT), month (VARCHAR), cust\_id (BIGINT), state (VARCHAR), prod\_id (BIGINT), and order\_total (BIGINT); and "Data Preview" which shows a table with 11 rows of sample data:

	order_id	month	cust_id	state	prod_id	order_total
1	67212	June	10001	ca	909	13
2	70302	June	10004	ga	420	11
3	69090	June	10011	fl	44	76
4	68834	June	10012	ar	0	81
5	71220	June	10018	az	411	24
6	61287	June	1001	nj	104	134
7	68553	June	10021	ca	117	67
8	68109	June	10022	tx	337	10
9	68526	June	10025	mi	11	63
10	69362	June	10028	tx	430	65
11	68624	June	10030	fl	808	51

At the bottom, there are buttons for Refresh, Connect, Sample (set to 100), Rows, Close, and a copyright notice: © 2015 MapR Technologies.

- They can quickly and easily view the structure, size and contents of each of these data sources.

The image shows a screenshot of the Drill & Semi-structured Data interface. At the top, there's a red header bar with the text "Drill & Semi-structured Data" and a logo. Below the header is a "Drill Explorer" window. The window has tabs for "Browse" and "SQL", with "SQL" selected. In the SQL tab, there's a text area containing the following SQL query:

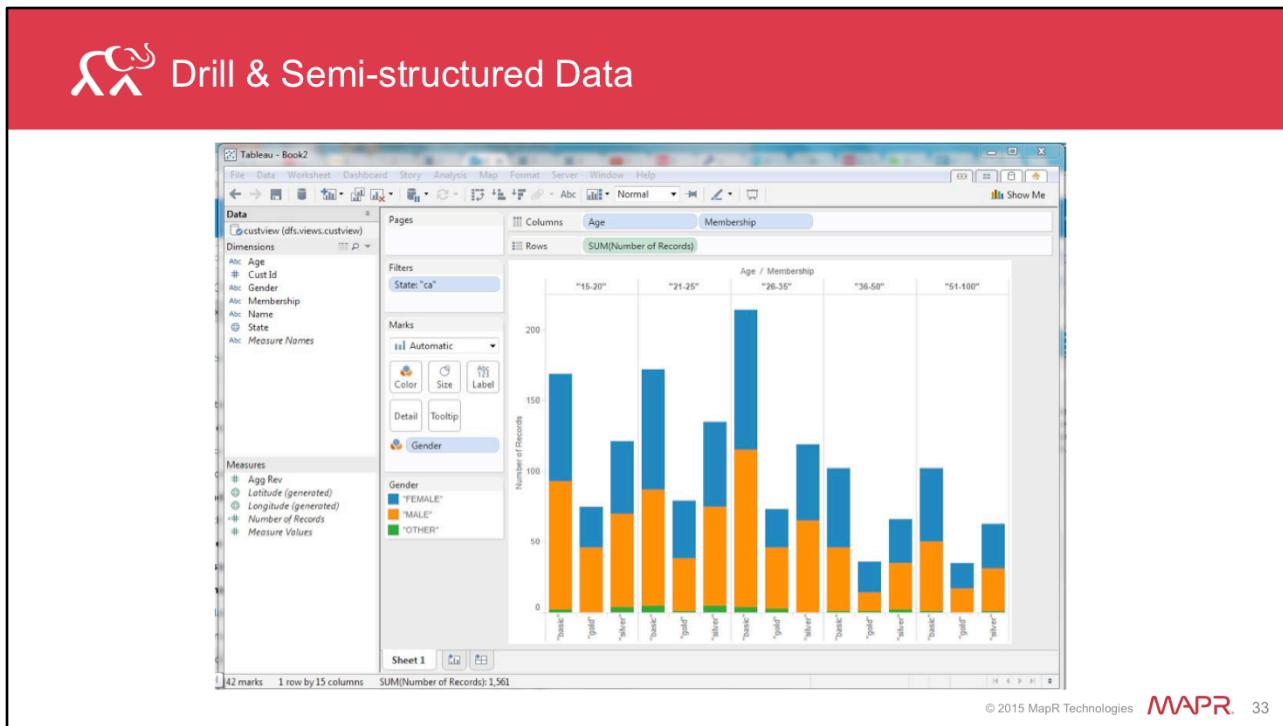
```
select * from `clicks/clicks.json` limit 2;
```

Below the SQL area, it says "Total Number of Records: 2". A table is displayed with the following data:

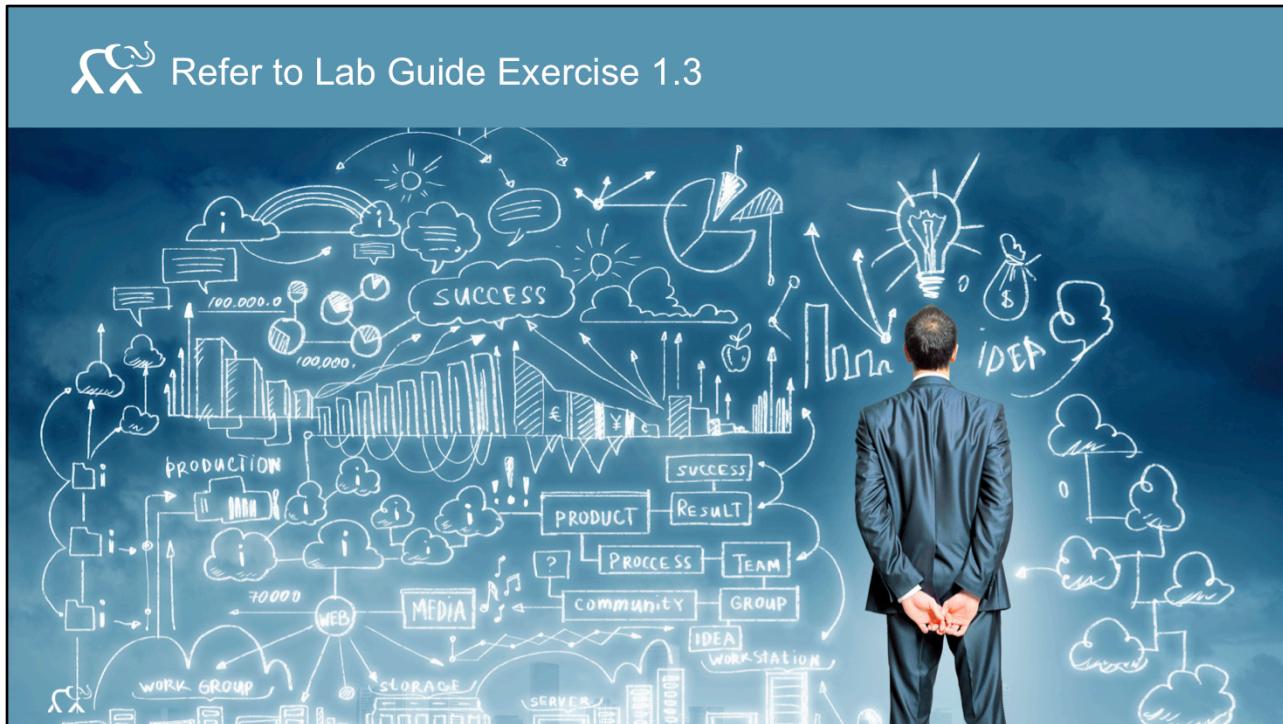
	trans_id	date	time	user_info	trans_info
▶	1	2014-04-26	12:17:12	{ "cust_id": 22526, "device": "IOS5", "state": "il"}	{ "prod_id": [174, 2], "purch_flag": "false" }
	2	2014-04-20	13:50:29	{ "cust_id": 16368, "device": "AOS4.2", "state": "nc"}	{ "prod_id": [], "purch_flag": "false" }

At the bottom right of the interface, there's a copyright notice: "© 2015 MapR Technologies MAPR 32".

- If they find something of interest, Drill will provide the SQL code for that section of data.
- As we change what we look at in the browser, the SQL will automatically update for us, and we can always edit the SQL as needed



- With Drill Explorer our data analysts can get a better understanding of our data through quick exploration, and then easily visualize areas of interest with the BI tools that they already use, such as Tableau, Microstrategy or SAS



## 1. Data Analyst Image

- The Drill explorer gives our Data Analysts a unique tool to learn about data from multiple sources, and even unknown data..

### 1. Drill Explorer Launch Page

- When we first open the Drill Explorer, we see all of the data sources available to drill. This includes our structured HBase table, as well as our unstructured JSON and parquet data, and any other data that drill can access.

### 2. Select data source and view the data

- We can select any one of these data sources, and the data will appear in the browser window.
- We don't have to know anything about the data or



## Knowledge Check

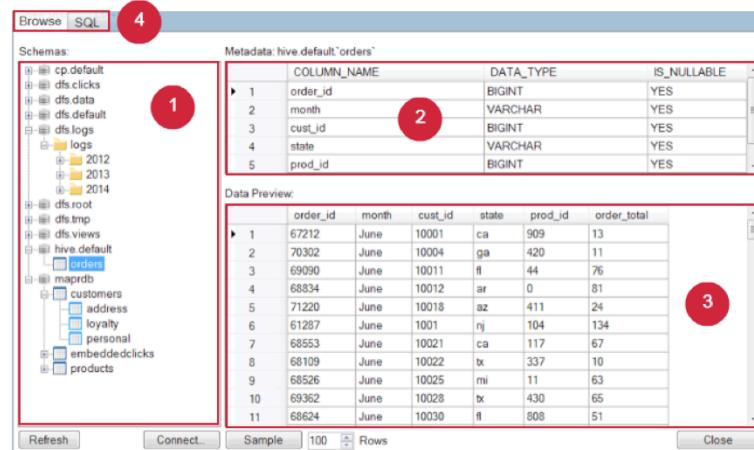
- View all data sources available to Drill
- Explore data saved in selected sources
- See meta data for selected source
- Choose between exploring data sources or SQL use to query this data

4

1

2

3



**Schemas:**

- cp default
- dfs clicks
- dfs data
- dfs default
- dfs logs
  - logs
  - 2012
  - 2013
  - 2014
- dfs root
- dfs tmp
- dfs views
- hive default
  - orders
- maprdb
  - customers
  - address
  - loyalty
  - personal
  - embeddedclicks
  - products

**Metadata: hive.default.orders**

COLUMN_NAME	DATA_TYPE	IS_NULLABLE
order_id	BIGINT	YES
month	VARCHAR	YES
cust_id	BIGINT	YES
state	VARCHAR	YES
prod_id	BIGINT	YES

**Data Preview:**

order_id	month	cust_id	state	prod_id	order_total
67212	June	10001	ca	909	13
70302	June	10004	ga	420	11
69990	June	10011	fl	44	76
68834	June	10012	ar	0	81
71220	June	10018	az	411	24
61287	June	1001	nj	104	134
68553	June	10021	ca	117	67
68109	June	10022	tx	337	10
68526	June	10025	mi	11	63
69362	June	10028	tx	430	65
68824	June	10030	fl	808	51

© 2015 MapR Technologies  35

1  
3  
2  
4





Next Steps

**Lesson 2:**  
Querying  
Self Describing Data© 2015 MapR Technologies **MAPR** 36

Congratulations, you have completed the first lesson of this course. Continue to lesson 2 to learn about what types of data we can access when using drill, how drill can discover the schema of unknown or complex data, and dig deeper into how drill interacts with data.

**Lesson 2:**  
Querying Self Describing Data with Drill

**MAPR**<sup>®</sup> Academy

## Data Analysis with Apache Drill

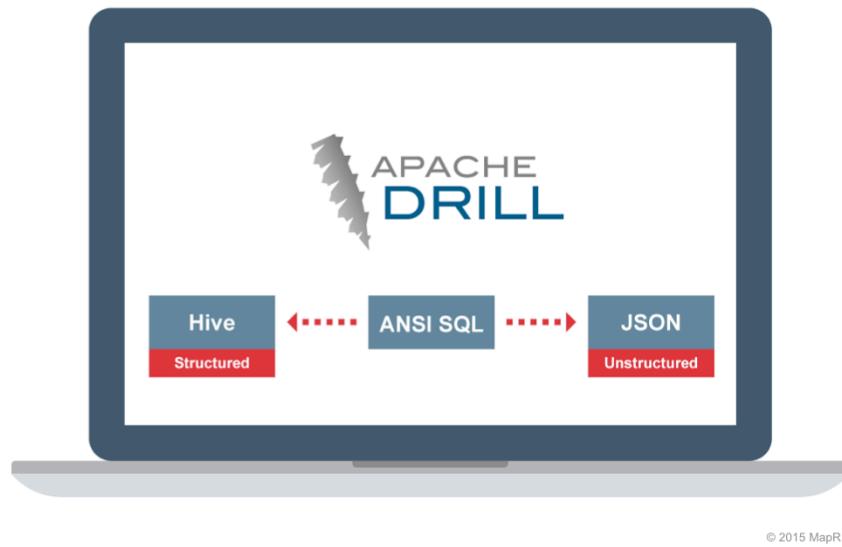
Lesson 2: Query Self Describing Data

© 2015 MapR Technologies **MAPR** 37





## What is Drill?



© 2015 MapR Technologies **MAPR** 38

In the previous lesson, we learned about how easily Apache Drill can be used to perform standard SQL queries on structured data, and the power of using Drill to simultaneously query structured and semi-structured data.



The Drill Explorer interface is shown, featuring a red header bar with the text "What is Drill?". The main window displays the "Drill Explorer" title bar with tabs for "Browse" and "SQL". On the left, a tree view titled "Schemas:" shows various database structures. In the center, the "Metadata: hive.default.orders" section displays a table with columns: order\_id, month, cust\_id, state, prod\_id, and order\_total. Below it, the "Data Preview" section shows a sample of 11 rows from the orders table.

	order_id	month	cust_id	state	prod_id	order_total
1	67212	June	10001	ca	909	13
2	70302	June	10004	ga	420	11
3	69090	June	10011	fl	44	76
4	68834	June	10012	ar	0	81
5	71220	June	10018	az	411	24
6	61287	June	1001	nj	104	134
7	68553	June	10021	ca	117	67
8	68109	June	10022	tx	337	10
9	68526	June	10025	mi	11	63
10	69362	June	10028	tx	430	65
11	68624	June	10030	fl	808	51

© 2015 MapR Technologies **MAPR** 39

We also introduced the concept of exploring unknown data with Drill Explorer, opening up new understanding of our data without typing a single line of code.



The screenshot shows the Drill Explorer application window. On the left, there's a tree view of schemas and tables. A magnifying glass icon is overlaid on the 'Data Preview' table, which contains sample data from the 'orders' table. The table has columns: order\_id, month, cust\_id, state, prod\_id, order\_qty, and total. The data preview shows 11 rows of sample data.

	order_id	month	cust_id	state	prod_id	order_qty	total
1	67212	May	10001	ca	909	1	100
2	70302	May	10004	ga	420	1	100
3	69090	June	10011	fl	44	1	100
4	68834	June	10012	ar	0	1	100
5	71220	June	10013	nj	104	134	100
6	61287	June	1001	nj	104	134	100
7	68553	June	10021	ca	117	67	100
8	68109	June	10022	tx	337	10	100
9	68526	June	10025	mi	11	63	100
10	69362	June	10028	tx	430	65	100
11	68624	June	10030	fl	808	51	100

- In this lesson we will look more deeply into the queries we performed in lesson one. We will discuss how Drill is able to determine the schema of data on the fly, and look at how drill interacts with different types of data.

What is Drill?

Tableau Book2

Data: dsvviews.customers

Dimensions: Age, City, Gender, Membership, Name, State

Measures: Agg Rev, Longitude (generated), Number of Records, Measure Values

Filters: State: "ca"

Marks: SUM(Number of Records)

Pages: Age / Membership

Rows: SUM(Number of Records)

Columns: Age

Legend: FEMALE, MALE, OTHER

© 2015 MapR Technologies MAPR 41

We will also continue to examine the Drill Explorer, and see how we can create new SQL views that we can use to easily visualize our data with standard BI tools via ODBC connectivity.





## Learning Goals

- ▶ Define Data Types
- ▶ Define Data Interaction & Schema Detection
- ▶ Create Complex Query Examples
- ▶ Create Views for use with BI Tools

© 2015 MapR Technologies  42

When you have finished this lesson, you will be able to

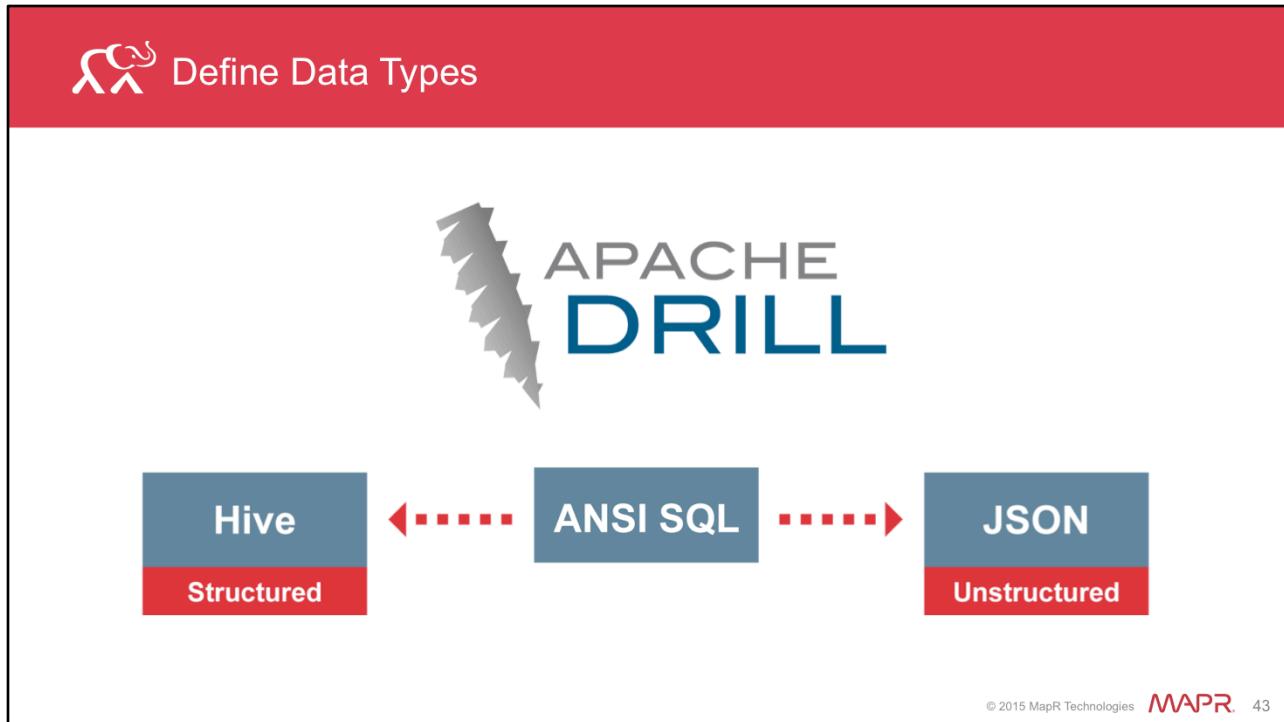
Define the different types of data we can explore and query with Drill

Describe how Drill interacts with these data types, and discovers their schema

Explore data to create queries using multiple data sources, including complex data like arrays and maps

Create views to visualize data in our existing BI tools without writing any code

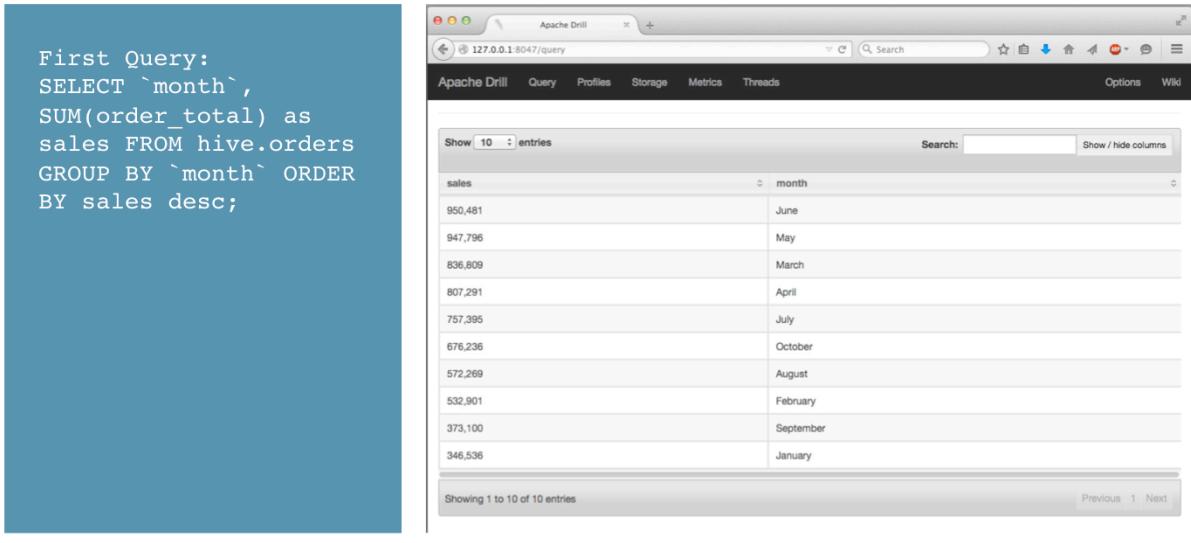




In the previous lesson, we performed SQL queries on several different data types; structured and semi-structured tables, and streaming click data.

These different data types are not just an evolution in our company's data, though. Rather, they each perform a different purpose, and each type of data is put in the format it is, because it best suits that purpose.





The screenshot shows the Apache Drill interface running in a web browser at 127.0.0.1:8047/query. The title bar says "Apache Drill". The main area displays a table with two columns: "sales" and "month". The data is as follows:

sales	month
950,481	June
947,796	May
836,809	March
807,291	April
757,395	July
676,236	October
572,269	August
532,901	February
373,100	September
346,536	January

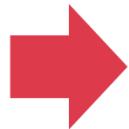
At the bottom, it says "Showing 1 to 10 of 10 entries".

© 2015 MapR Technologies MAPR 44

Looking back at the queries we made, first we pulled some data from our orders database, stored as a Hive table. Hive is a structured tabular data format, stored on a distributed file system like MapR-FS.



## Define Data Types



order_id	month	cust_id	state	prod_id	order_total
67212	June	10001	ca	909	13
70302	May	10004	ga	420	11
15451	Feburary	10009	in	107	26
95413	August	10019	oh	594	40

© 2015 MapR Technologies  45

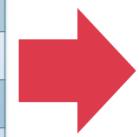
With structured data, the schema of the table is fixed. A standard relational database requires that we know the schema of our database before we load any data.





## Define Data Types

order_id	month	cust_id	state	prod_id	order_total
67212	June	10001	ca	909	13
70302	May	10004	ga	420	11
15451	Feburary	10009	in	107	26
95413	August	10019	oh	594	40



© 2015 MapR Technologies MAPR 46

With most SQL tools, we also have to know the schema of the database when we pull information back out again. Later in this lesson, we will discuss how Drill is able to discover the schema of a Hive table for you.





## Define Data Types

	Fixed Schema	Dynamic Schema
Simple	<b>Structured Tables:</b> <ul style="list-style-type: none"><li>• CSV</li><li>• TSV</li></ul>	
Complex		

© 2015 MapR Technologies  47

Structured data is flat, and very fast and easy to query, which is why relational databases have been so successful for so long. Structured data is also rigid, however, and does not allow for the schema of the data to change without overhauling the entire database, and all of the records it holds. Structured data is well suited for data sets that we know will be consistent, like that from our order form. Every order will contain the same set of fields, yielding consistent data.





## Define Data Types

cust_id	name	state	gender	age	agg_rev	membership
16841	Lisa Wells	tx	FEMALE	26-35	40	basic
19905	Neely Nova	va	MALE	15-20	45	basic
17718	William Bush	il	FEMALE	51-100	29	basic
21975	Gerald Staples	ma	MALE	26-35	28	basic

© 2015 MapR Technologies 48

The second set of queries included our customer and product data stored in HBase tables. Like our structured data, HBase also stores flat data in a table. HBase is an evolution of Google's Big Table, and was originally designed to be used with a distributed file system like MapR-FS.





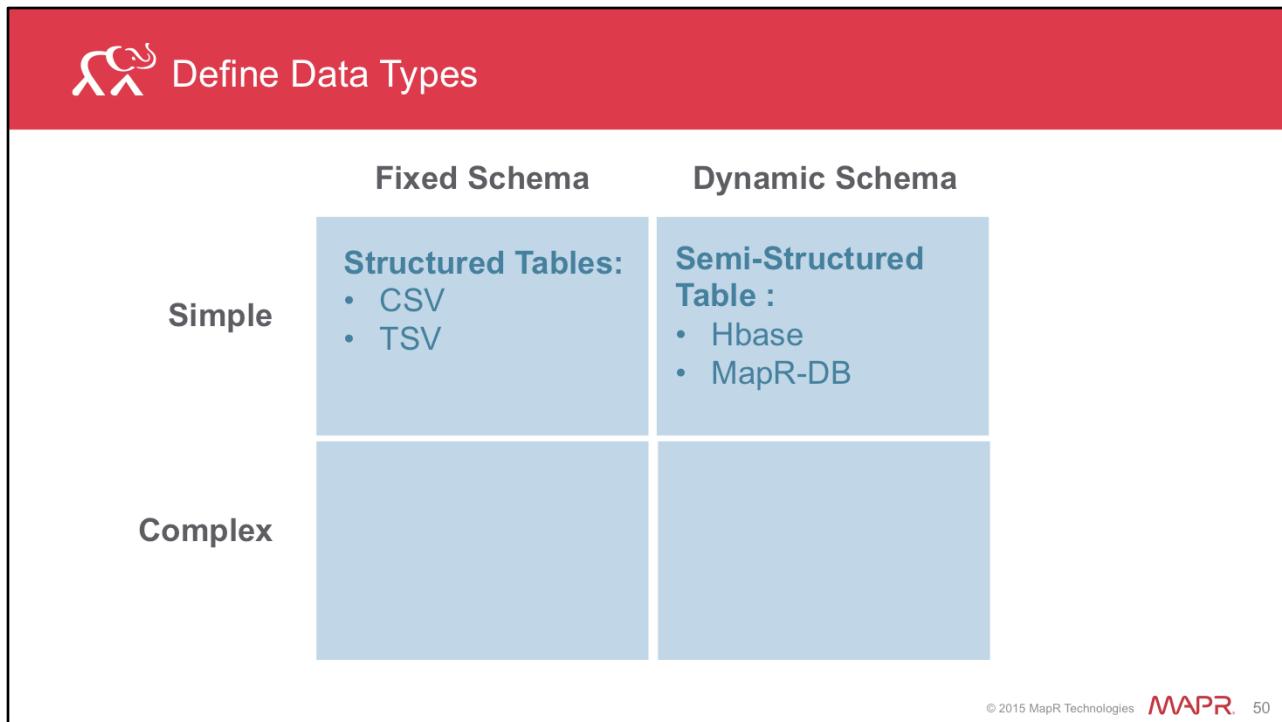
## Define Data Types

cust_id	name	state	gender	age	agg_rev	membership	reviews
16841	Lisa Wells	tx	FEMALE	26-35	40	basic	NULL
19905	Neely Nova	va	MALE	15-20	45	basic	NULL
17718	William Bush	il	FEMALE	51-100	29	basic	NULL
21975	Gerald Staples	ma	MALE	26-35	28	basic	TiqA9ybgewk

© 2015 MapR Technologies MAPR 49

Different from the structured data, HBase uses key-value pairs to locate the requested data, and this allows HBase the advantage of having a dynamic schema. We can change the structure of the table on the fly without affecting any previously stored data. With HBase, we can add new columns and rows to our table as we come across new data needs, and fill only the fields for a row that have data. HBase tables will often have a large amount of NULL value cells, as most rows will not share all of the columns that are filled other rows.





HBase is well suited for a data set that needs to be flexible. Since HBase stores data in binary form, we can store whatever data we need to in our table, access it through a key value pair, and not have to worry about the structure of the data.



 Define Data Types



↓

<b>cust_id</b>	<b>name</b>	<b>state</b>	<b>gender</b>	<b>age</b>	<b>agg_rev</b>	<b>membership</b>	<b>reviews</b>
16841	Lisa Wells	tx	FEMALE	26-35	40	basic	NULL
19905	Neely Nova	va	MALE	15-20	45	basic	NULL
17718	William Bush	il	FEMALE	51-100	29	basic	NULL
21975	Gerald Staples	ma	MALE	26-35	28	basic	TiqA9ybgewk

© 2015 MapR Technologies  51

We are using HBase for our customer and product databases. We can store text data, metadata, customer or product images and anything else about our customer and our products all in the same database. We also have the flexibility to add new types of data, such as advanced customer security or product videos, as they come available.



 Define Data Types

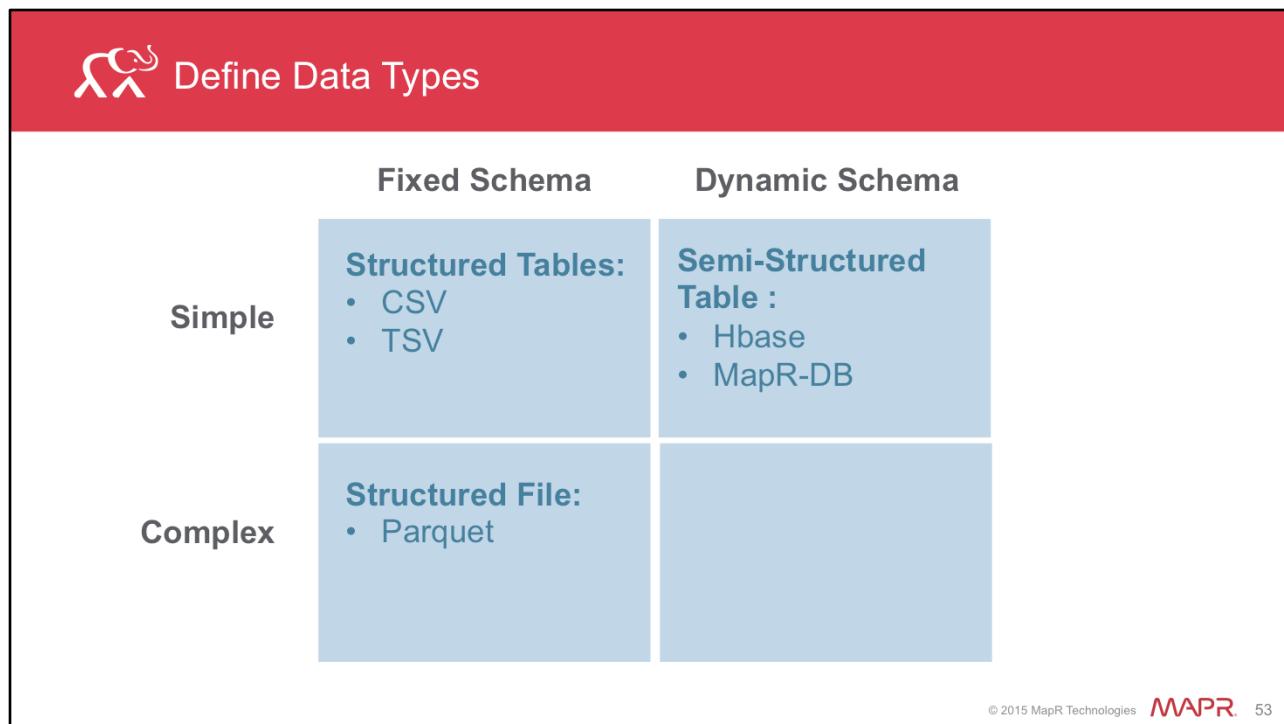
Data Preview:

	trans_id	date	time	user_info	trans_info
▶ 1	31920	2014-04-26	12:17:12	{ "cust_id": 22526, "device": "IOS5", "state": "il"}	{ "prod_id": [174, 2], "purch_flag": "false"}
2	31026	2014-04-20	13:50:29	{ "cust_id": 16368, "device": "AOS4.2", "state": "nc"}	{ "prod_id": [], "purch_flag": "false"}
3	33848	2014-04-10	04:44:42	{ "cust_id": 21449, "device": "IOS6", "state": "oh"}	{ "prod_id": [582], "purch_flag": "false"}
4	32383	2014-04-18	06:27:47	{ "cust_id": 20323, "device": "IOS5", "state": "oh"}	{ "prod_id": [710, 47], "purch_flag": "false"}
5	32359	2014-04-19	23:13:25	{ "cust_id": 15360, "device": "IOS5", "state": "ca"}	{ "prod_id": [0, 8, 170, 173, 1, 124, 46, 764, 30, 711, 0, 3, 25], "purch_flag": "false"}
6	30422	2014-04-23	01:46:05	{ "cust_id": 15957, "device": "IOS7", "state": "sc"}	{ "prod_id": [628], "purch_flag": "false"}
7	35898	2014-04-18	13:28:56	{ "cust_id": 20677, "device": "IOS7", "state": "ny"}	{ "prod_id": [], "purch_flag": "false"}
8	32421	2014-04-15	11:00:53	{ "cust_id": 23599, "device": "IOS5", "state": "ri"}	{ "prod_id": [1, 0, 87, 170, 445, 6, 0, 2, 7, 7, 110], "purch_flag": "false"}
9	39447	2014-04-21	01:34:06	{ "cust_id": 16122, "device": "IOS6", "state": "il"}	{ "prod_id": [26, 504, 1, 47, 31, 116, 167, 71, 20, 305, 554, 0, 189, 384], "purch_flag": "false"}
10	35373	2014-04-02	06:53:28	{ "cust_id": 15342, "device": "IOS5", "state": "ms"}	{ "prod_id": [15, 0], "purch_flag": "false"}
11	30778	2014-04-13	01:20:05	{ "cust_id": 16996, "device": "AOS4.2", "state": "oh"}	{ "prod_id": [40, 499, 15], "purch_flag": "true"}
12	32120	2014-04-28	06:58:19	{ "cust_id": 21402, "device": "IOS5", "state": "nm"}	{ "prod_id": [], "purch_flag": "false"}
13	30706	2014-04-15	21:08:15	{ "cust_id": 15344, "device": "IOS7", "state": "ny"}	{ "prod_id": [27], "purch_flag": "false"}
14	38207	2014-04-28	18:57:55	{ "cust_id": 16841, "device": "AOS4.2", "state": "mo"}	{ "prod_id": [181, 0, 33, 708, 92], "purch_flag": "false"}
15	32955	2014-04-15	20:41:58	{ "cust_id": 15492, "device": "AOS4.4", "state": "wi"}	{ "prod_id": [242, 0, 2, 3, 166, 26, 0, 0], "purch_flag": "false"}
...	...	...	...	...	...

© 2015 MapR Technologies  52

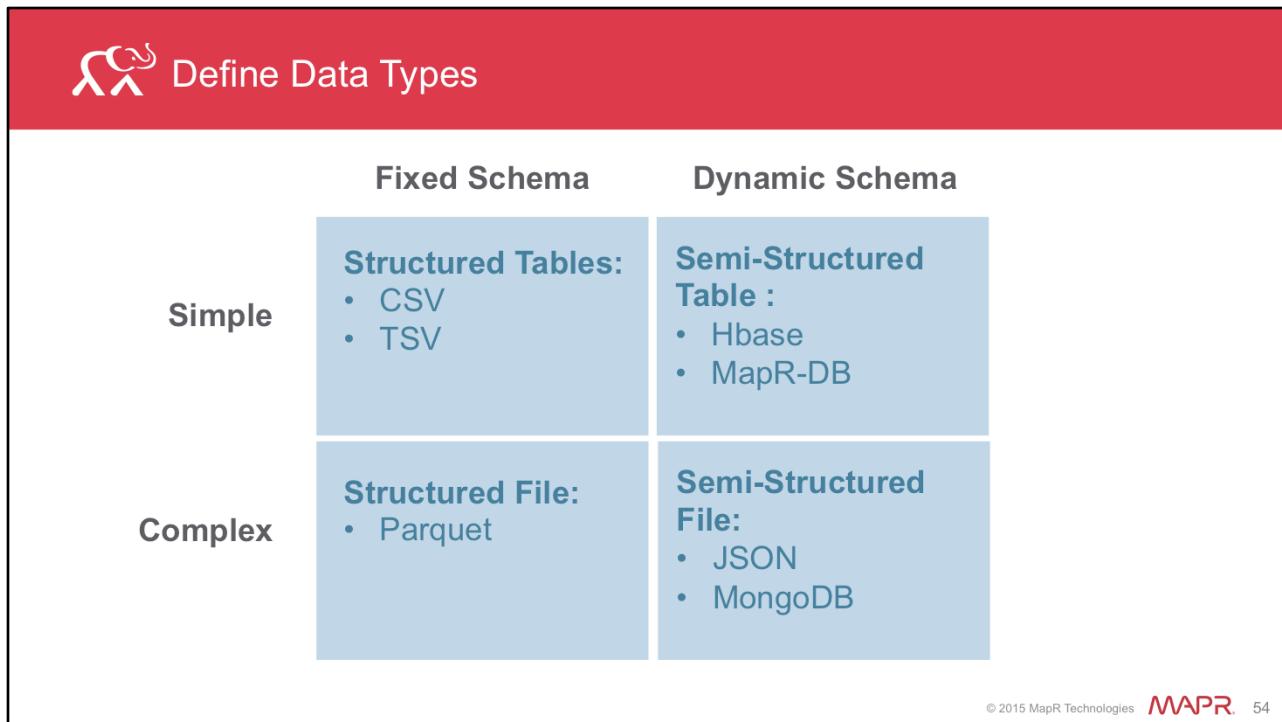
In our second set of exercises, we also queried web click data that is stored as parquet and JSON files. Parquet stores data in a columnar format, and was designed to handle complex, nested data in a Hadoop framework.





Parquet uses a pre-defined, fixed schema, similar to our structured tables. Parquet files are very fast to query. Since we frequently query our user product search logs, we periodically convert these files into parquet format for faster query times.





We also keep our click data in JSON format. JSON is the most flexible of our file formats. JSON files can be complex and nested, like our parquet files, and also being schema-less, like our HBase tables.

With Drill we can, and have used familiar ANSI SQL to query data that fits into any of these categories. Drill gives us unparalleled ease and flexibility to query our companies data.

*(highlight the entire table)*

*(rollover quadrants of the table to show:*

*upper left: Flat, structured data is fast to query, but very inflexible to changes. Best suited for known data sets that will not change.*

*upper right: Semi-structured or NoSQL tables are very flexible, and still provide structure to our data. Best suited for very large*



		Fixed Schema	Dynamic Schema
Simple	<b>Structured Tables:</b> <ul style="list-style-type: none"><li>• CSV</li><li>• TSV</li></ul>	<b>Semi-Structured Table :</b> <ul style="list-style-type: none"><li>• Hbase</li><li>• MapR-DB</li></ul>	
Complex	<b>Structured File:</b> <ul style="list-style-type: none"><li>• Parquet</li></ul>	<b>Semi-Structured File:</b> <ul style="list-style-type: none"><li>• JSON</li><li>• MongoDB</li></ul>	

© 2015 MapR Technologies  55

Which quadrants contain data types that Drill can use to run SQL queries?

Which quadrants contain data types that other SQL tools can use SQL to run queries?





Define Data Interaction & Schema Detection

```
SELECT `month`, SUM(order_total) as sales  
FROM hive.orders GROUP BY `month` ORDER BY sales desc;
```

© 2015 MapR Technologies  56

When we look more closely at our queries, we can see how drill interacts with each of these different types of data.

For example, going back to our first query...





Define Data Interaction & Schema Detection

```
SELECT `month`, SUM(order_total) as sales
FROM hive.orders GROUP BY `month` ORDER BY sales desc;
```

© 2015 MapR Technologies  57

the SELECT statement defines what data we are pulling out of the table, and what action we are going to perform on it, just like we are used to from any standard SQL query.





Define Data Interaction & Schema Detection

```
SELECT `month`, SUM(order_total) as sales  
FROM hive.orders GROUP BY `month` ORDER BY sales desc;
```

© 2015 MapR Technologies **MAPR** 58

In the FROM statement, however, we start to see how Drill interacts with our data sources. `hive.orders` tells Drill where to find the data we are requesting in the select statement.



The screenshot shows the Drill Explorer interface. On the left, a tree view titled 'Schemas' displays various database structures. A red callout box labeled 'Storage system type' points to the 'hive default' schema, which is highlighted in blue. The tree includes nodes like cp.default, dfs.clicks, dfs.data, dfs.default, dfs.logs (with sub-folders for 2012, 2013, and 2014), dfs.root, dfs.tmp, dfs.views, maprdb (with sub-tables customers, address, loyalty, personal, embeddedclicks, and products), and the highlighted hive.default. Below the tree are 'Refresh' and 'Connect...' buttons. On the right, a SQL query window contains the following code:

```
SELECT `month`, SUM(order_total)  
as sales FROM hive.orders GROUP  
BY `month` ORDER BY sales desc;
```

At the bottom right of the interface, there is a copyright notice: © 2015 MapR Technologies MAPR. 59

In this case, hive points Drill to the Hive storage on our cluster.



The screenshot shows the Drill Explorer interface. On the left, a tree view of schemas is displayed under 'Schemas': cp.default, dfs.clicks, dfs.data, dfs.default, dfs.logs (with subfolders 2012, 2013, 2014), dfs.root, dfs.tmp, dfs.views, hive.default (with table 'orders' selected), maprdb (with subfolders customers, address, loyalty, personal, embeddedclicks, products). Below the tree are 'Refresh' and 'Connect...' buttons. On the right, a SQL query is shown:

```
SELECT `month`, SUM(order_total)
as sales FROM hive.orders GROUP
BY `month` ORDER BY sales desc;
```

Annotations with red arrows point to specific parts of the query:

- A red box labeled 'Storage system type' points to the word 'hive' in the query.
- A red box labeled 'Table Name' points to the word 'orders' in the query.

At the bottom right of the interface, it says '© 2015 MapR Technologies MAPR 60'.

and then 'orders' is the name of the table from which we are requesting data.



## Define Data Interaction &amp; Schema Detection

```
SELECT clicks.user_info.cust_id  
FROM dfs.`data/nested/clicks/clicks.json` clicks  
WHERE clicks.trans_info.purch_flag = 'false';
```

© 2015 MapR Technologies  61

Then, examining the fifth query we made,





## Define Data Interaction &amp; Schema Detection

Storage system type

```
SELECT clicks.user_info.cust_id  
FROM dfs.`data/nested/clicks/clicks.json` clicks  
WHERE clicks.trans_info.purch_flag = 'false';
```

© 2015 MapR Technologies  62

the FROM statement points Drill to the DFS filesystem





## Define Data Interaction &amp; Schema Detection

Storage system type

```
SELECT clicks.user_info.cust_id  
FROM dfs.`data/nested/clicks/clicks.json` clicks  
WHERE clicks.trans_info.purch_flag = 'false';
```

Path Name

© 2015 MapR Technologies 63

and then `data/nested/clicks/clicks.json` provides the pathname to the file where Drill can find the data we are requesting.

With Drill, we don't have to connect to the database prior to submitting a query. All we have to do is include the data source location in the FROM statement of our query, and Drill finds the data we are requesting.



The screenshot shows the Drill Explorer application window. At the top, there is a red header bar with the title "Define Data Interaction & Schema Detection" and the MAPR logo. Below the header, the main interface is divided into two main sections: "Schemas" on the left and "Metadata" and "Data Preview" on the right.

**Schemas:** A tree view showing various data sources. The "hive default" schema is expanded, revealing the "orders" table. Other schemas listed include "cp default", "dfs clicks", "dfs data", "dfs default", "dfs logs", "dfs root", "dfs tmp", "dfs views", "maprdb", and "customers".

**Metadata:** A table titled "Metadata: hive.default.orders" showing the schema of the "orders" table. The columns are: COLUMN\_NAME, DATA\_TYPE, and IS\_NULLABLE. The data rows are:

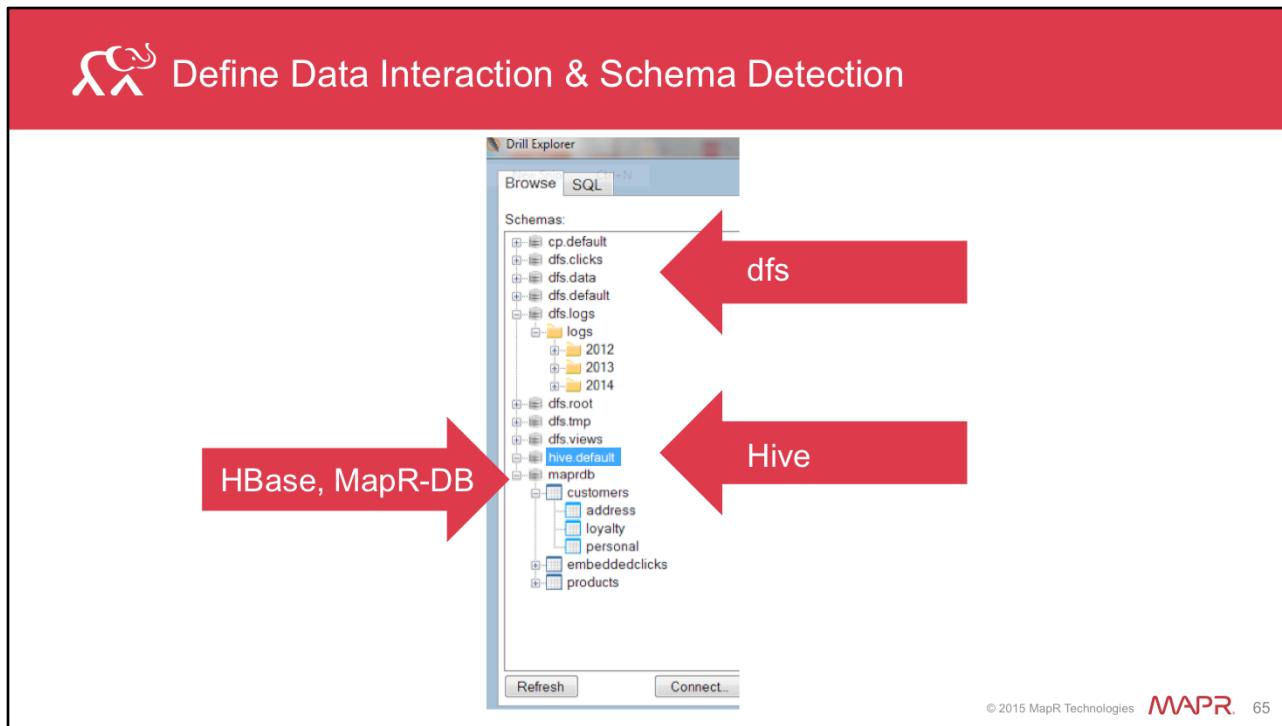
COLUMN_NAME	DATA_TYPE	IS_NULLABLE
order_id	BIGINT	YES
month	VARCHAR	YES
cust_id	BIGINT	YES
state	VARCHAR	YES
prod_id	BIGINT	YES

**Data Preview:** A table titled "Data Preview" showing the first 11 rows of the "orders" table. The columns are: order\_id, month, cust\_id, state, prod\_id, and order\_total. The data rows are:

	order_id	month	cust_id	state	prod_id	order_total
1	67212	June	10001	ca	909	13
2	70302	June	10004	ga	420	11
3	69090	June	10011	fl	44	76
4	68834	June	10012	ar	0	81
5	71220	June	10018	az	411	24
6	61287	June	1001	nj	104	134
7	68553	June	10021	ca	117	67
8	68109	June	10022	tx	337	10
9	68526	June	10025	mi	11	63
10	69362	June	10028	tx	430	65
11	68624	June	10030	fl	808	51

At the bottom of the interface, there are buttons for "Sample", "100", and "Rows", along with a "Close" button.

Looking at the Drill explorer, we can see the different data sources that Drill can query in this way.



The storage system type can be a data base like Hive, HBase or MapR-DB, or a distributed filesystem like HDFS or MapR-FS.

The screenshot shows the Drill Explorer interface. On the left, there's a tree view of schemas and databases. A red arrow points from the text "Filesystem subdirectory" to the "dfs.logs" entry under the "hive.default" database. The main panel displays the metadata for the "orders" table in the "hive.default" database. It shows the column names, data types, and whether they are nullable. Below the metadata is a preview of 11 rows of data from the table.

Filesystem subdirectory

COLUMN_NAME	DATA_TYPE	IS_NULLABLE
order_id	BIGINT	YES
month	VARCHAR	YES
cust_id	BIGINT	YES
	VARCHAR	YES
	BIGINT	YES

order_id	month	cust_id	state	prod_id	order_total
1	June	10001	ca	909	13
2	June	10004	ga	420	11
3	June	10011	fl	44	76
4	June	10012	ar	0	81
5	June	10018	az	411	24
6	June	1001	nj	104	134
7	June	10021	ca	117	67
8	June	10022	tx	337	10
9	June	10025	mi	11	63
10	June	10028	tx	430	65
11	June	10030	fl	808	51

© 2015 MapR Technologies **MAPR** 66

We can include a workspace, which can be a hive database, an HBase namespace, or filesystem subdirectory.

 Define Data Interaction & Schema Detection

```
SELECT `month`, SUM(order_total) as sales
FROM hive.orders GROUP BY `month` ORDER BY sales desc;
```

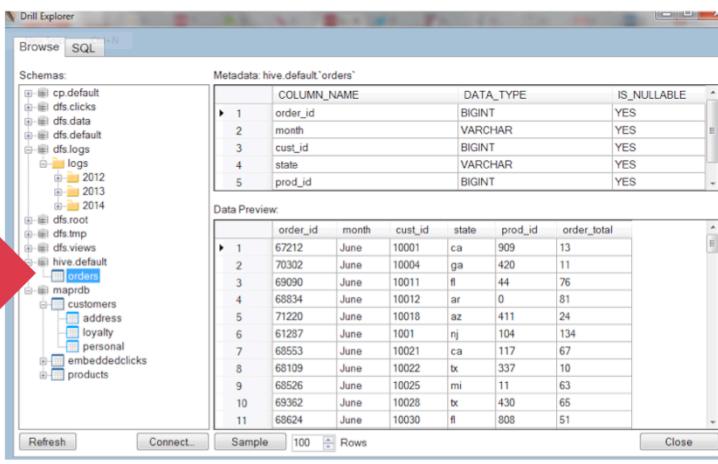


Table Name

© 2015 MapR Technologies  67

and finally we include the path to the table or file that we want to query.



The screenshot shows the Drill Explorer application window. At the top, there's a red header bar with the title "Define Data Interaction & Schema Detection" and the MapR logo. Below the header, the main interface has two main sections: "Schemas" on the left and "Metadata" on the right.

**Schemas:** A tree view showing various HDFS paths and databases. The "orders" table is selected under the "hive.default" database.

**Metadata:** A table titled "Metadata: hive.default.orders". It lists the columns: order\_id, month, cust\_id, state, prod\_id, and order\_total. The table includes column names, data types (e.g., BIGINT, VARCHAR), and nullability (e.g., YES).

**Data Preview:** A table showing 11 rows of sample data from the "orders" table. The columns correspond to the schema defined above. The data includes various order IDs, months (e.g., June), customer IDs, states (e.g., ca, ga, fl, ar, az, nj, ca, tx, mi), product IDs, and order totals.

At the bottom of the window, there are buttons for "Refresh", "Connect...", "Sample", "100 Rows", and "Close".

© 2015 MapR Technologies MAPR 68

The Drill Explorer gives us some further insight into how Drill discovers the schema of the data available to us. In the previous lesson, we introduced the idea of exploring data. Exploring data is a very powerful tool for our data analysts when looking at unknown tables, or data streams with dynamic schemas.



The screenshot shows the Drill Explorer application window. On the left, a tree view of schemas and tables is displayed, including 'cp.default', 'dfs.clicks', 'dfs.data', 'dfs.default', 'dfs.logs', 'dfs.root', 'dfs.tmp', 'dfs.views', 'hive.default', 'maprdb', and 'orders'. The 'orders' table is selected. In the center, two panes are shown: 'Metadata: hive.default.orders' (schema information) and 'Data Preview' (sample data). The 'Metadata' pane shows columns: order\_id (BIGINT), month (VARCHAR), cust\_id (BIGINT), state (VARCHAR), prod\_id (BIGINT), and order\_total (BIGINT). The 'Data Preview' pane shows 11 rows of sample data.

	COLUMN_NAME	DATA_TYPE	IS_NULLABLE
1	order_id	BIGINT	YES
2	month	VARCHAR	YES
3	cust_id	BIGINT	YES
4	state	VARCHAR	YES
5	prod_id	BIGINT	YES

	order_id	month	cust_id	state	prod_id	order_total
1	67212	June	10001	ca	909	13
2	70302	June	10004	ga	420	11
3	69090	June	10011	fl	44	76
4	68834	June	10012	ar	0	81
5	71220	June	10018	az	411	24
6	61287	June	1001	nj	104	134
7	68553	June	10021	ca	117	67
8	68109	June	10022	tx	337	10
9	68526	June	10025	mi	11	63
10	69362	June	10028	tx	430	65
11	68624	June	10030	fl	808	51

© 2015 MapR Technologies **MAPR** 69

For example, if we have unknown structured data, our data analysts can first use the Drill explorer to learn about the data before writing queries. When we look at our orders table stored in Hive, we can see that Drill is able to learn the column names, the data types and whether the data is nullable. We did not tell Drill the table schema, and normally we would have to execute HiveQL statements to learn the schema of a table.

The screenshot shows the Drill Explorer application window. On the left, there's a tree view of schemas and tables. The 'hive.default' schema is expanded, showing the 'orders' table. The 'orders' table is selected, and its metadata is displayed in a table on the right. This table has columns: COLUMN\_NAME, DATA\_TYPE, and IS\_NULLABLE. The data preview section below shows 11 rows of sample data with columns: order\_id, month, cust\_id, state, prod\_id, and order\_total.

COLUMN_NAME	DATA_TYPE	IS_NULLABLE
order_id	BIGINT	YES
month	VARCHAR	YES
cust_id	BIGINT	YES
state	VARCHAR	YES
prod_id	BIGINT	YES

	order_id	month	cust_id	state	prod_id	order_total
1	67212	June	10001	ca	909	13
2	70302	June	10004	ga	420	11
3	69090	June	10011	fl	44	76
4	68834	June	10012	ar	0	81
5	71220	June	10018	az	411	24
6	61287	June	1001	nj	104	134
7	68553	June	10021	ca	117	67
8	68109	June	10022	tx	337	10
9	68526	June	10025	mi	11	63
10	69362	June	10028	tx	430	65
11	68624	June	10030	fl	808	51

© 2015 MapR Technologies **MAPR** 70

Drill uses the HCatalog Connector to access the HCATALOG\_COLUMNS metadata. Drill discovers the schema and shows us the table names and data types. Drill does the work for us, showing us the data structure and data types, so that we can then write the queries that we need.



cust_id	name	state	gender	age	agg_rev	membership	reviews
16841	Lisa Wells	tx	FEMALE	26-35	40	basic	NULL
19905	Neely Nova	va	MALE	15-20	45	basic	NULL
17718	William Bush	il	FEMALE	51-100	29	basic	NULL
21975	Gerald Staples	ma	MALE	26-35	28	basic	TiqA9ybgewk

```

{"trans_id":31920,"date":"2014-04-26","time":"12:17:12","user_info":{"cust_id":22526,"device":"I055","state":"il"},"trans_info":{"prod_id":[174,2],"purch_flag":"false"}}
{"trans_id":31826,"date":"2014-04-26","time":"13:59:29","user_info":{"cust_id":16391,"device":"A054,2","state":"nc"},"trans_info":{"prod_id":[174,1],"purch_flag":"false"}}
{"trans_id":31827,"date":"2014-04-26","time":"14:00:19","user_info":{"cust_id":16391,"device":"A054,2","state":"nc"},"trans_info":{"prod_id":[174,1],"purch_flag":"false"}}
{"trans_id":32383,"date":"2014-04-18","time":"08:27:47","user_info":{"cust_id":28223,"device":"I055","state":"oh"},"trans_info":{"prod_id":[178,47],"purch_flag":"false"}}
{"trans_id":32359,"date":"2014-04-19","time":"12:13:25","user_info":{"cust_id":15360,"device":"I055","state":"ca"},"trans_info":{"prod_id":[0,8,170,173,1,124,46,764,38,711,0,3,25],"purch_flag":"true"}}
{"trans_id":30422,"date":"2014-04-23","time":"01:46:05","user_info":{"cust_id":15957,"device":"I057","state":"sc"},"trans_info":{"prod_id":[628], "purch_flag":"false"}}
{"trans_id":35898,"date":"2014-04-18","time":"13:28:56","user_info":{"cust_id":28677,"device":"I057","state":"ny"},"trans_info":{"prod_id":[1],"purch_flag":"false"}}
{"trans_id":35899,"date":"2014-04-18","time":"13:28:56","user_info":{"cust_id":28677,"device":"I057","state":"ny"},"trans_info":{"prod_id":[127], "purch_flag":"false"}}
{"trans_id":36443,"date":"2014-04-21","time":"01:34:09","user_info":{"cust_id":16122,"device":"I056","state":"il"},"trans_info":{"prod_id": [26,594,1,47,21,116,117,2,7,7,110],"purch_flag":"false"}}
 {"trans_id":35373,"date":"2014-04-02","time":"08:53:28","user_info":{"cust_id":15342,"device":"I055","state":"ms"}, "trans_info":{"prod_id": [15,8], "purch_flag":"false"}}
 {"trans_id":30778,"date":"2014-04-13","time":"01:28:05","user_info":{"cust_id":16996,"device":"A054,2","state":"oh"}, "trans_info":{"prod_id": [148,499,15], "purch_flag":"true"}}
 {"trans_id":32120,"date":"2014-04-28","time":"06:58:19","user_info":{"cust_id":21482,"device":"I055","state":"nm"}, "trans_info":{"prod_id": [], "purch_flag":"false"}}
 {"trans_id":30705,"date":"2014-04-30","time":"21:00:15","user_info":{"cust_id":15344,"device":"A057","state":"ny"}, "trans_info":{"prod_id": [27], "purch_flag":"false"}}
 {"trans_id":30807,"date":"2014-04-15","time":"11:51:05","user_info":{"cust_id":15492,"device":"A054,2","state":"md"}, "trans_info":{"prod_id": [181,0,33,768,92], "purch_flag":"false"}}
 {"trans_id":32955,"date":"2014-04-15","time":"12:41:59","user_info":{"cust_id":15492,"device":"A054,2","state":"wi"}, "trans_info":{"prod_id": [242,0,2,3,166,26,0,0], "purch_flag":"false"}}

```

© 2015 MapR Technologies  71

At the Big Office Supply Company, our data sources also include data streams that have a dynamic schema. Our HBase tables and JSON click data are both examples of data that is dynamic, as it changes its schema over time.





## Define Data Interaction & Schema Detection

<b>cust_id</b>	<b>name</b>	<b>state</b>	<b>gender</b>	<b>age</b>	<b>agg_rev</b>	<b>membership</b>
16841	Lisa Wells	tx	FEMALE	26-35	40	basic
19905	Neely Nova	va	MALE	15-20	45	basic
17718	William Bush	il	FEMALE	51-100	29	basic
21975	Gerald Staples	ma	MALE	26-35	28	basic

Data Preview:

	column_0	column_1	column_2	column_3	column_4	column_5
▶ 1	16841	"Lisa Wells"	"tx"	"FEMALE"	"26-35"	40
2	19905	"Neely Nova"	"va"	"MALE"	"15-20"	45
3	17718	"William Bush"	"il"	"FEMALE"	"51-100"	29
4	21975	"Gerald Staples"	"ma"	"MALE"	"26-35"	28
5	20468	"William Conners"	"de"	"MALE"	"36-50"	32
6	16670	"John Rubio"	"oh"	"MALE"	"15-20"	42

© 2015 MapR Technologies  72

When exploring HBase or MapR-DB, Drill reads the table column headers, which are saved as text. We can see all of the column headers currently in the table when using the Drill Explorer. Since these columns are read at the time we select the data source, they are always current. As with the Hive table, Drill is doing the work for us, and we do not need to execute any code to view the dynamic schema.





## Define Data Interaction &amp; Schema Detection

```
[{"trans_id":31920,"date":"2014-04-26","time":"12:17:12","user_info":{"cust_id":22526}, {"trans_id":31026,"date":"2014-04-20","time":"13:50:29","user_info":{"cust_id":16368}, {"trans_id":33848,"date":"2014-04-10","time":"04:44:42","user_info":{"cust_id":21449}, {"trans_id":32383,"date":"2014-04-18","time":"06:27:47","user_info":{"cust_id":20323}, {"trans_id":32359,"date":"2014-04-19","time":"23:13:25","user_info":{"cust_id":15360}, {"trans_id":30422,"date":"2014-04-23","time":"01:46:05","user_info":{"cust_id":15957}, {"trans_id":35898,"date":"2014-04-18","time":"13:28:56","user_info":{"cust_id":20677}, {"trans_id":32421,"date":"2014-04-15","time":"11:00:53","user_info":{"cust_id":23599},
```

Data Description: "cust\_id"  
Data Content: :15360

© 2015 MapR Technologies 73

With text based docs like our JSON click logs, Drill reads the schema from the document itself. JSON is self describing data, meaning that each data element contains the data description preceding the data content. Drill will read these data description elements, and present them to us as schema metadata.



The screenshot shows the Drill Explorer interface. On the left, a tree view of schemas is displayed, including 'cp default', 'dfs clicks', 'dfs data', 'dfs default', 'dfs logs', 'dfs root', 'apps', 'data' (with subfolders 'flat', 'nested', 'orders', 'products', 'customers all csv'), 'drill-beta-demo', 'hbase', 'oozie', 'tables', 'tmp', 'user', 'var', 'dfs tmp', 'dfs views', 'hive default', 'maprdb', 'customers', and 'embeddedclicks'. The 'data' folder is expanded. On the right, there are two panes: 'Metadata' and 'Data Preview'. The 'Metadata' pane shows a table with columns 'Column\_Index', 'Data\_Type', and 'Value\_Width'. The 'Data Preview' pane shows a sample of 12 rows from a table with columns 'column\_0', 'column\_1', 'column\_2', 'column\_3', 'column\_4', and 'column\_5'. The data includes names like Lisa Wells, Neely Nova, William Bush, Gerald Staples, etc., along with gender and age ranges.

Column_Index	Data_Type	Value_Width
1	VARCHAR(5)	5
2	VARCHAR(22)	22
3	VARCHAR(4)	4
4	VARCHAR(8)	8
5	VARCHAR(8)	8
6	VARCHAR(2)	2

column_0	column_1	column_2	column_3	column_4	column_5
16841	"Lisa Wells"	"tx"	"FEMALE"	"26-35"	40
2	"Neely Nova"	"va"	"MALE"	"15-20"	45
3	"William Bush"	"il"	"FEMALE"	"51-100"	29
4	"Gerald Staples"	"ma"	"MALE"	"26-35"	28
5	"William Conners"	"de"	"MALE"	"36-50"	32
6	"John Rubio"	"oh"	"MALE"	"15-20"	42
7	"Jay Holland"	"f"	"FEMALE"	"21-25"	43
8	"James Hogsett"	"ne"	"FEMALE"	"15-20"	30
9	"Florence Black"	"pa"	"FEMALE"	"26-35"	24
10	"Judith Stuart"	"ga"	"MALE"	"21-25"	25
11	"Dale Johnson"	"ga"	"FEMALE"	"15-20"	24
12	"Patrick Cote"	"ms"	"MALE"	"36-50"	45

© 2015 MapR Technologies **MAPR** 74

Exploring the data with Drill Explorer gives us the data schema information that we need to make effective queries of data that is unknown or dynamic in structure.



## Knowledge Check

**Match the following definitions**

- A. `hive.orders`
- B. `Dfs.'data/nested/clicks/clicks.json'`
- C. `HCATALOG_COLUMNS`
- D. `"user_info": {"cust_id": 5324, "device": "AOS4.3", "state": "ca"}`

- Tells Drill what table to query
- Metadata used to discover the schema of a Hive table
- Tells Drill what file to query
- Self describing JSON data

A  
C  
B  
D



SQL Support in Drill		
	Fixed Schema	Dynamic Schema
Simple	<b>Structured Tables:</b> <ul style="list-style-type: none"><li>Hive</li><li>CSV</li><li>TSV</li></ul>	<b>Semi-Structured Table :</b> <ul style="list-style-type: none"><li>Hbase</li><li>MapR-DB</li></ul>
Complex	<b>Structured File:</b> <ul style="list-style-type: none"><li>Parquet</li></ul>	<b>Semi-Structured File:</b> <ul style="list-style-type: none"><li>JSON</li><li>MongoDB</li></ul>

© 2015 MapR Technologies **MAPR.** 76

Drill makes it easy for us to query a variety of simple, complex, unknown and dynamic data sources. We can use Drill to explore our data, keeping our queries current and efficient as our data changes.





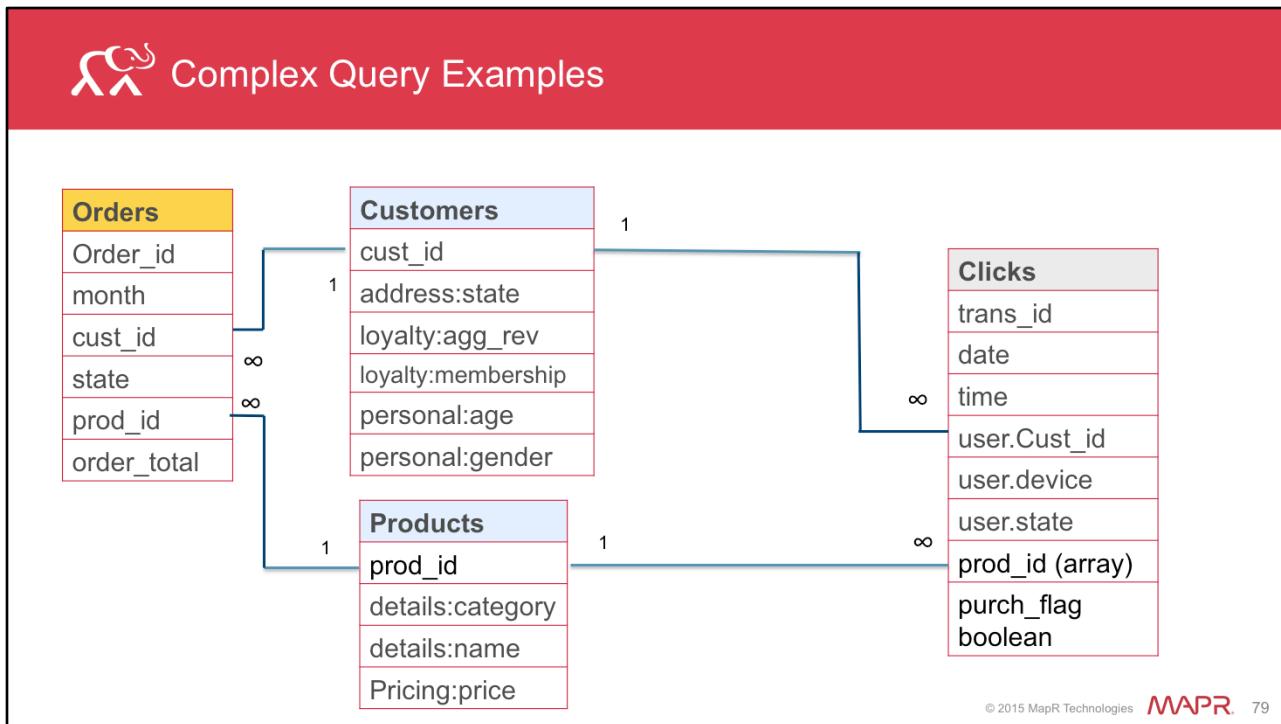
- Drill supports ANSI SQL – theory on the query we are about to show
- List key data types
- Key operators – joins, aggregators, functions
- Show a couple of sample queries, doing joins, but without complex data





- Extending ANSI SQL to work with complex types such as arrays and maps
- Do an example with flatten and KV Gen
- Show convert to/from – steer away from cast





Some of the most useful insights to our company, however, are gained when we combine data across multiple sources, such as combining our click log data with customer locations or purchasing habits.

We store all of these different data sources in different formats, to best suit the type of data in them. They also all have different schemas, some of which are known and fixed, while others are dynamic and constantly changing.



## Complex Query Examples

```
SELECT cust_clk.cust_name, prod.name as prod_name, cust_clk.`date` FROM
hive.products as prod,
(SELECT cast(cust.personal.name as varchar(40)) as cust_name, clk.prod_id,
clk.`date`
FROM maprdb.customers as cust, (SELECT cast(clicks.user_info.cust_id as bigint)
as cust_id, cast(FLATTEN(clicks.trans_info.prod_id) as bigint) as prod_id,
clicks.trans_info.purch_flag as purch_flag, to_date(clicks.`date`, 'yyyy-mm-dd')
as `date` FROM dfs.clicks.`/clicks/clicks.json` as clicks) as clk
WHERE cast(cust.row_key as bigint) = clk.cust_id and clk.purch_flag = 'false')
as cust_clk
WHERE cust_clk.prod_id = prod.prod_id
and cust_clk.`date` between '2014-01-01' and '2014-03-31';
```

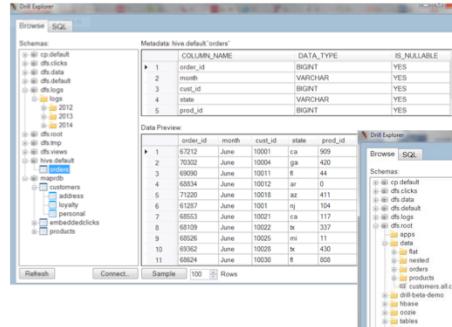
© 2015 MapR Technologies 80

In our final query in Lesson 1, we joined our structured Hive orders table, with our semi structured HBase customers table and JSON click logs.

When we examine this query, we can see how Drill provides us with the information we need to first build the query, and then the power to execute a single query across all of these different types of data.

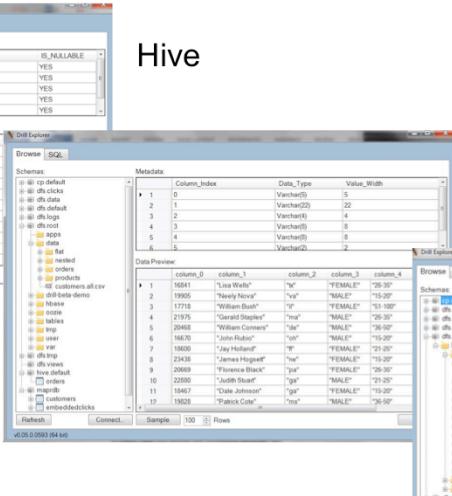


### Hive



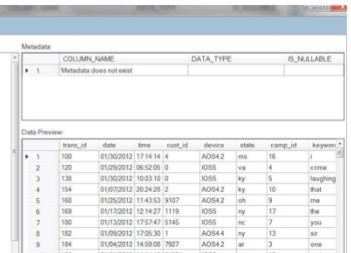
The Hive schema for the 'orders' table is displayed, showing columns: order\_id, month, cust\_id, state, prod\_id. Data preview shows rows from June 2001 to June 2008.

### HBase



The HBase schema for the 'people' table is displayed, showing columns: column\_0, column\_1, column\_2, column\_3, column\_4. Data preview shows rows for various people with their names and gender.

### JSON



The JSON schema for the 'log.json' file is displayed, showing a single column: log. Data preview shows rows of log entries.

© 2015 MapR Technologies  81

We knew that we wanted to perform a query that combined information from three different sources of data. Prior to writing the query, we used the Drill Explorer to view these three data sources, learning their current schema and how the data is stored.





## Complex Query Examples

```
SELECT cust_clk.cust_name, prod.name as prod_name, cust_clk.`date` FROM
hive.products as prod,
(SELECT cast(cust.personal.name as varchar(40)) as cust_name, clk.prod_id,
clk.`date`
FROM maprdb.customers as cust, (SELECT cast(clicks.user_info.cust_id as bigint)
as cust_id, cast(FLATTEEN(clicks.trans_info.prod_id) as bigint) as prod_id,
clicks.trans_info.purch_flag as purch_flag, to_date(clicks.`date`, 'yyyy-mm-dd')
as `date` FROM dfs.clicks.`/clicks/clicks.json` as clicks) as clk
WHERE cast(cust.row_key as bigint) = clk.cust_id and clk.purch_flag = 'false')
as cust_clk
WHERE cust_clk.prod_id = prod.prod_id
and cust_clk.`date` between '2014-01-01' and '2014-03-31';
```

© 2015 MapR Technologies 82

Once we learned how to access the data, and what types of data we were querying, we can then build a single query that joins all of these sources; simple, complex, structured and dynamic, into this single query.





## Complex Query Examples

```
SELECT cust_clk.cust_name, prod.name as prod_name, cust_clk.`date`
FROM hive.products as prod
```

The screenshot shows a data exploration interface with the following sections:

- Schemas:** A tree view of schemas including cp.default, dfs.clicks, dfs.data, dfs.default, dfs.logs, logs (with subfolders 2012, 2013, 2014), dfs.root, dfs.tmp, dfs.views, hive.default, orders, products, maprdb, customers, address, loyalty, personal, and embeddedclicks.
- Metadata:** A table titled "Metadata: hive.default.orders" showing columns: COLUMN\_NAME, DATA\_TYPE, and IS\_NULLABLE. The data includes:
 

COLUMN_NAME	DATA_TYPE	IS_NULLABLE
order_id	BIGINT	YES
month	VARCHAR	YES
cust_id	BIGINT	YES
state	VARCHAR	YES
prod_id	BIGINT	YES
- Data Preview:** A table titled "Data Preview" showing 10 rows of data from the 'orders' table:
 

	order_id	month	cust_id	state	prod_id	order_total
1	67212	June	10001	ca	909	13
2	70302	June	10004	ga	420	11
3	69090	June	10011	fl	44	76
4	68834	June	10012	ar	0	81
5	71220	June	10018	az	411	24
6	61287	June	1001	nj	104	134
7	68553	June	10021	ca	117	67
8	68109	June	10022	tx	337	10
9	68526	June	10025	mi	11	63
10	69362	June	10028	tx	430	65

© 2015 MapR Technologies 83

**Highlight 3 Select statements – We nested the SELECT statements, each targeting one of our data sources. Once we learned where each piece of data is stored, we can then map how we will join our data sources.**





## Complex Query Examples

```
SELECT cast(cust.personal.name as varchar(40)) as cust_name,
clk.prod_id, clk.`date`
FROM maprdb.customers as cust
```

Schemas:

- cp default
- dfs clicks
  - clicks
    - clicks.campaign.json
    - clicks.json
- dfs default
- dfs logs
- dfs root
- dfs tmp
- dfs views
  - custview
  - tradesflatview
  - tradesview
- hive default
- orders
  - address
  - loyalty
  - personal
- maprdb
  - customers
    - address
    - loyalty
    - personal
- embeddeddicks
- products
- test
- test2
- trades\_flat
- trades\_tall

Metadata: 'maprdb'.customers'

	COLUMN_NAME	DATA_TYPE
1	row_key	ANY
2	address	[VARCHAR(1), ANY] MAP
3	loyalty	[VARCHAR(1), ANY] MAP
4	personal	[VARCHAR(1), ANY] MAP

Data Preview

	row_key	address	loyalty	personal
1	0x3130303031	{ "state": "InZhlg=="} { "agg_rev": "MTK3", "membership": "InNpbHZlcil="}	{ "age": "iE1LTiWlg==", "i	
2	0x3130303035	{ "state": "Imulg=="} { "agg_rev": "MjMw", "membership": "InNpbHZlcil="}	{ "age": "ij2LTM1Ig==", "i	
3	0x3130303036	{ "state": "imNhlg=="} { "agg_rev": "MJuW", "membership": "InNpbHZlcil="}	{ "age": "ij2LTM1Ig==", "i	
4	0x3130303037	{ "state": "im1Ilg=="} { "agg_rev": "Myz", "membership": "InNpbHZlcil="}	{ "age": "iJULTEwMCi=", "i	
5	0x3130303130	{ "state": "Imulg=="} { "agg_rev": "MjAy", "membership": "InNpbHZlcil="}	{ "age": "ijUxLTewMCi=", "i	
6	0x3130303131	{ "state": "imhplg=="} { "agg_rev": "MTV5", "membership": "InNpbHZlcil="}	{ "age": "iJULTEwMCi=", "i	
7	0x3130303132	{ "state": "Im8olg=="} { "agg_rev": "MTU1", "membership": "InNpbHZlcil="}	{ "age": "iJxLTHlg==", "gi	
8	0x3130303134	{ "state": "Im5qlg=="} { "agg_rev": "MTK5", "membership": "InNpbHZlcil="}	{ "age": "iE1LTiWlg==", "gi	
9	0x3130303138	{ "state": "Im55lg=="} { "agg_rev": "MjQ4", "membership": "InNpbHZlcil="}	{ "age": "iJULTEwMCi=", "i	
10	0x3130303032	{ "state": "ImN0lg=="} { "agg_rev": "MTA5MQ==", "membership": "ImdvbGQi"} { "age": "ijxLTiIg==", "gi		

© 2015 MapR Technologies 84

**Highlight 3 Select statements – We nested the SELECT statements, each targeting one of our data sources. Once we learned where each piece of data is stored, we can then map how we will join our data sources.**





## Complex Query Examples

```
SELECT cast(clicks.user_info.cust_id as bigint) as cust_id, cast(FLATTEN(clicks.trans_info.prod_id) as bigint) as prod_id, clicks.trans_info.purch_flag as purch_flag, to_date(clicks.`date`, 'yyyy-mm-dd') as `date` FROM dfs.clicks.`/clicks/clicks.json` as clicks
```

The screenshot shows a user interface for managing data schemas and metadata. On the left, there is a 'Schemas' tree view with nodes like 'cp.default', 'dfs.clicks', 'dfs.logs', 'dfs.root', 'dfs.tmp', and 'dfs.views'. Under 'dfs.clicks', there are sub-nodes: 'clicks.campaign.json' and 'clicks.json'. A red arrow points from the text above to the 'clicks.json' node. To the right of the schema tree is a 'Metadata' panel titled 'hive.default.orders'. It contains a single row with the column name 'COLUMN\_NAME' and value 'Metadata does not exist'. Below the metadata panel is a 'Data Preview' section which is currently empty.

© 2015 MapR Technologies 85

Highlight 3 Select statements – We nested the SELECT statements, each targeting one of our data sources. Once we learned where each piece of data is stored, we can then map how we will join our data sources.





## Complex Query Examples

```
cast(cust.personal.name as varchar(40))
```

Metadata: `maprdb`.`customers`

	COLUMN_NAME	DATA_TYPE	IS_NULLABLE
▶ 1	row_key	ANY	NO
2	address	(VARCHAR(1), ANY) MAP	NO
3	loyalty	(VARCHAR(1), ANY) MAP	NO
4	personal	(VARCHAR(1), ANY) MAP	NO

© 2015 MapR Technologies 86

Highlight ‘cast’ functions – When we explored the data before writing this query, we were able to determine the type of data in each part of the query. Based on this information, we are able to cast each part of our query into the appropriate type.





## Complex Query Examples

```
cast(clicks.user_info.cust_id as bigint)
```

Metadata: hive.default.`orders`

	COLUMN_NAME	DATA_TYPE	IS_NULLABLE
1	order_id	BIGINT	YES
2	month	VARCHAR	YES
3	cust_id	BIGINT	YES
4	state	VARCHAR	YES
5	prod_id	BIGINT	YES

Data Preview:

	order_id	month	cust_id	state	prod_id	order_total
1	67212	June	10001	ca	909	13
2	70302	June	10004	ga	420	11
3	69090	June	10011	fl	44	76
4	68834	June	10012	ar	0	81
5	71220	June	10018	az	411	24
6	61297	June	1001	ri	104	124

© 2015 MapR Technologies 87

Highlight ‘cast’ functions – When we explored the data before writing this query, we were able to determine the type of data in each part of the query. Based on this information, we are able to cast each part of our query into the appropriate type.





## Complex Query Examples

```
cast(FLATTEN(clicks.trans_info.prod_id) as bigint)
```

Metadata: hive.default.`orders`

	COLUMN_NAME	DATA_TYPE	IS_NULLABLE
1	order_id	BIGINT	YES
2	month	VARCHAR	YES
3	cust_id	BIGINT	YES
4	state	VARCHAR	YES
5	prod_id	BIGINT	YES

Data Preview:

	order_id	month	cust_id	state	prod_id	order_total
1	67212	June	10001	ca	909	13
2	70302	June	10004	ga	420	11
3	69090	June	10011	fl	44	76
4	68834	June	10012	ar	0	81
5	71220	June	10018	az	411	24
c	61987	June	1001	ri	104	124

© 2015 MapR Technologies 88

Highlight ‘cast’ functions – When we explored the data before writing this query, we were able to determine the type of data in each part of the query. Based on this information, we are able to cast each part of our query into the appropriate type.





## Complex Query Examples

```
(FLATTEN(clicks.trans_info.prod_id) as bigint) as prod_id,
clicks.trans_info.purch_flag as purch_flag, to_date(clicks.`date`, 'yyyy-mm-dd') as `date` FROM dfs.clicks.`/clicks@clicks.json` as clicks) as clk
```

	trans_info
:22526	"device": "IOS5", "state": "il"} { "prod_id": [174, 2], "purch_flag": "false"}
:16368	"device": "AOS42", "state": "nc"} { "prod_id": [], "purch_flag": "false"}
:21449	"device": "IOS6", "state": "oh"} { "prod_id": [582], "purch_flag": "false"}
:20323	"device": "IOS5", "state": "oh"} { "prod_id": [710, 47], "purch_flag": "false"}
:15360	"device": "IOS5", "state": "ca"} { "prod_id": [0, 8, 170, 173, 1, 124, 46, 764, 30, 711, 0, 3, 25], "purch_flag": "false"}
:15957	"device": "IOS7", "state": "sc"} { "prod_id": [628], "purch_flag": "false"}
:20677	"device": "IOS7", "state": "ny"} { "prod_id": [], "purch_flag": "false"}
:23599	"device": "IOS5", "state": "n"} { "prod_id": [1, 0, 87, 170, 445, 6, 0, 2, 7, 110], "purch_flag": "false"}
:16122	"device": "IOS6", "state": "il"} { "prod_id": [26, 504, 1, 47, 31, 116, 167, 71, 20, 305, 554, 0, 189, 384], "purch_flag": "false"}
:15342	"device": "IOS5", "state": "ms"} { "prod_id": [15, 0], "purch_flag": "false"}
:16996	"device": "AOS42", "state": "oh"} { "prod_id": [40, 499, 15], "purch_flag": "true"}
:21402	"device": "IOS5", "state": "nm"} { "prod_id": [], "purch_flag": "false"}
:15344	"device": "IOS7", "state": "ny"} { "prod_id": [27], "purch_flag": "false"}
:16841	"device": "AOS42", "state": "mo"} { "prod_id": [181, 0, 33, 708, 92], "purch_flag": "false"}
:15492	"device": "AOS42", "state": "wa"} { "prod_id": [242, 0, 2, 3, 166, 96, 0, 0], "purch_flag": "false"}

© 2015 MapR Technologies 89

Highlight FLATTEN statement – with Drill, we can perform a join that includes complex data types like arrays, maps and JSON blobs. When we explore the data with Drill explorer, we can see that we have a complex data type, and flatten it in our final query





## Complex Query Examples

```
SELECT cust_clk.cust_name, prod.name as prod_name, cust_clk.`date` FROM
hive.products as prod,
(SELECT cast(cust.personal.name as varchar(40)) as cust_name, clk.prod_id, clk.`date` 
FROM maprdb.customers as cust, (SELECT cast(clicks.user_info.cust_id as bigint) as
cust_id,
cast(FLATTEN(clicks.trans_info.prod_id) as bigint) as prod_id,
clicks.trans_info.purch_flag as purch_flag, to_date(clicks.`date`, 'yyyy-mm-dd') as
`date`
FROM dfs.clicks.`/clicks/clicks.json` as clicks) as clk
WHERE cast(cust.row_key as bigint) = clk.cust_id and clk.purch_flag = 'false') as
cust_clk
WHERE cust_clk.prod_id = prod.prod_id
and cust_clk.`date` between '2014-01-01' and '2014-03-31';
```

© 2015 MapR Technologies 90

Color code each sub query – Drill does most of our work for us. When we explore the data, we can view the SQL code needed to access each piece of data individually. Once we can access each piece on its own, we just need to join them together into the final query. Drill is then able to perform this single query on all of these data sources, and give us the results we need



## Complex Query Examples

```

SELECT cust_clk.cust_name, prod.name as prod_name, cust_clk.`date`  

FROM hive.products as prod,  

WHERE cust_clk.prod_id = prod.prod_id  

and cust_clk.`date` between '2014-01-01' and '2014-03-31';

```

**Query**

```

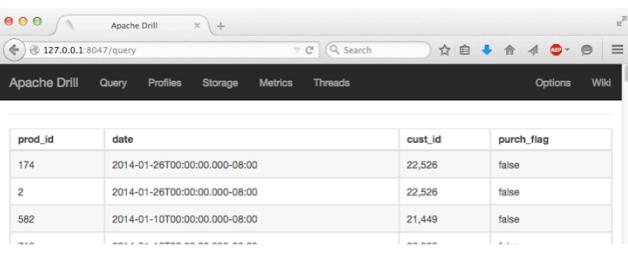
SELECT cast(clicks.user_info.cust_id as bigint) as cust_id,  

cast(FLATTEN(clicks.trans_info.prod_id) as bigint) as prod_id, clicks.trans_info.purch_flag as  

purch_flag, to_date(clicks.`date`, 'yyyy-mm-dd') as `date`  

FROM dfs.clicks."/clicks/clicks.json` as clicks

```



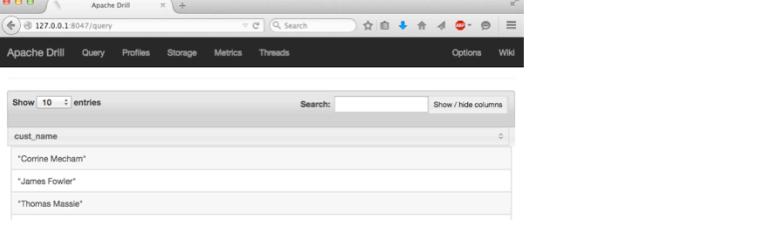
prod_id	date	cust_id	purch_flag
174	2014-01-26T00:00:00-08:00	22,526	false
2	2014-01-26T00:00:00-08:00	22,526	false
582	2014-01-10T00:00:00-08:00	21,449	false

© 2015 MapR Technologies **MAPR** 91

Color code each sub query – Drill does most of our work for us. When we explore the data, we can view the SQL code needed to access each piece of data individually. Once we can access each piece on its own, we just need to join them together into the final query. Drill is then able to perform this single query on all of these data sources, and give us the results we need

## Complex Query Examples

```
(SELECT cast(cust.personal.name as varchar(40)) as cust_name, clk.prod_id, clk.`date`
FROM maprdb.customers as cust,
WHERE cast(cust.row_key as bigint) = clk.cust_id and clk.purch_flag = 'false') as
cust_clk
Query
select cast(cust.personal.name as varchar(40)) as cust_name
from maprdb.customers as cust, dfs.clicks.`clicks/clicks.json` as clk
where cast(cust.row_key as bigint) = cast(clk.user_info.cust_id as bigint)
and cast(clk.trans_info.purch_flag as varchar(20)) = 'false' and
clk.trans_info.prod_id[0] is not null limit 10
```



The screenshot shows the Apache Drill interface running on port 8047. The query results are displayed in a table with one column, 'cust\_name', containing three rows of data: "Corrine Mecham", "James Fowler", and "Thomas Massie".

© 2015 MapR Technologies  92

Color code each sub query – Drill does most of our work for us. When we explore the data, we can view the SQL code needed to access each piece of data individually. Once we can access each piece on its own, we just need to join them together into the final query. Drill is then able to perform this single query on all of these data sources, and give us the results we need



## Complex Query Examples

```
SELECT cast(clicks.user_info.cust_id as bigint) as cust_id,
       cast(FLATEN(clicks.trans_info.prod_id) as bigint) as prod_id,
       clicks.trans_info.purch_flag as purch_flag, to_date(clicks.`date`, 'yyyy-mm-dd') as `date`
  FROM dfs.clicks.`/clicks/clicks.json` as clicks as clk
```

### Query

```
select prod.name as prod_name
  from hive.products as prod, dfs.clicks.`/clicks/clicks.json` as clk
 where cast(clk.trans_info.prod_id[0] as bigint) = prod.prod_id and cast(clk.trans_info.purch_flag
 as varchar(20)) = 'false' and clk.trans_info.prod_id[0] is not null limit 10
```

The screenshot shows the Apache Drill interface running on port 8047. The main window displays a table with one column, 'prod\_name', containing three rows: 'Sony notebook'. At the top, there's a navigation bar with tabs for 'Apache Drill', 'Query', 'Profiles', 'Storage', 'Metrics', 'Threads', 'Options', and 'Help'. Below the navigation bar is a search bar and a 'Show 10 : entries' button. The bottom right corner of the interface shows the copyright notice: '© 2015 MapR Technologies' and the MAPR logo.

© 2015 MapR Technologies **MAPR** 93

Color code each sub query – Drill does most of our work for us. When we explore the data, we can view the SQL code needed to access each piece of data individually. Once we can access each piece on its own, we just need to join them together into the final query. Drill is then able to perform this single query on all of these data sources, and give us the results we need





## Knowledge Check

Drill can join which of the following types into a single query?:

- Structured table data
- Semi structured table data
- Structured text files
- Semi structured text files
- Any combination of these data types

**Any combination of these data types**





- Add tables into views section
- Look at sales PPT for views information



The screenshot shows the Drill Explorer interface with a red header bar containing the 'Views' logo and the word 'Views'. Below the header is a 'Drill Explorer' window. The window has tabs for 'Browse' and 'SQL', with 'SQL' selected. The SQL code in the editor is:

```

SELECT cust_clk.cust_name, prod.name as prod_name, cust_clk.`date` FROM
hive.products as prod,
(SELECT cast(cust.personal.name as varchar(40)) as cust_name, clk.prod_id,
clk.`date` FROM maprdb.customers as cust, (SELECT cast(clicks.user_info.cust_id
as bigint) as cust_id, cast(FLATTEN(clicks.trans_info.prod_id) as bigint)
as prod_id, clicks.trans_info.purch_flag as purch_flag, to_date(clicks.`date`,
'yyyy-mm-dd') as `date` FROM dfs.clicks ./clicks.clicks.json` as clicks) as clk
WHERE cast(cust.row_key as bigint) = clk.cust_id and clk.purch_flag = 'false')
as cust_clk WHERE cust_clk.prod_id = prod.prod_id
and cust_clk.`date` between '2014-01-01' and '2014-03-31';

```

Below the code, it says 'Total Number of Records: 47,629'. A preview table shows the following data:

	cust_name	date	prod_name
1	"Jerry Rackley"	2014-01-29T00:00:00.000-08:00	"Sony VAIO Duo 13"
2	"Francis Diaz"	2014-01-29T00:00:00.000-08:00	"Sony VAIO Duo 13"
3	"Lucio Goods"	2014-01-05T00:00:00.000-08:00	"Sony VAIO Duo 13"
4	"Thelma Mcnair"	2014-01-04T00:00:00.000-08:00	"HP Spectre 13 X2"
5	"Kayla Jones"	2014-01-22T00:00:00.000-08:00	"HP Spectre 13 X2"
6	"Carmen Polk"	2014-01-30T00:00:00.000-08:00	"Lenovo ThinkPad S431 Ultrabook"
7	"Regina Moriarty"	2014-01-03T00:00:00.000-08:00	"Lenovo ThinkPad S431 Ultrabook"
8	"Lisa Sprague"	2014-01-05T00:00:00.000-08:00	"Alienware 14"
9	"Tony Hendrix"	2014-01-30T00:00:00.000-08:00	"Alienware 14"
10	"Andre Jude"	2014-01-29T00:00:00.000-08:00	"Alienware 14"

At the bottom right of the interface, it says '© 2015 MapR Technologies MAPR 96'.

As we explore data using Drill, we will often find some interesting connection we did not know before, or we may have a relationship in our data that we want to return to and review frequently.

For example, when exploring the data for our recent promotion, we examined our sales and click data over time and geographic region, and were able to target our promotion based on customer views. Now, we want to watch the click data to see how our promotion affects spending in the regions that we targeted.

The screenshot shows the Drill Explorer interface with a red header bar containing a logo and the word "Views". Below the header is a toolbar with "Browse" and "SQL" buttons, where "SQL" is selected. A large text area labeled "View Definition SQL:" contains the following complex multi-table query:

```
SELECT cust_clk.cust_name, prod.name as prod_name, cust_clk.`date` FROM
hive.products as prod,
(SELECT cast(cust.personal.name as varchar(40)) as cust_name, clk.prod_id,
clk.`date` FROM maprdb.customers as cust, (SELECT cast(clicks.user_info.cust_id
as bigint) as cust_id, cast(FLATTEN(clicks.trans_info.prod_id) as bigint)
as prod_id, clicks.trans_info.purch_flag as purch_flag, to_date(clicks.`date`,
'yyyy-mm-dd') as `date` FROM dfs.clicks.`/clicks/clicks.json` as clicks) as clk
WHERE cast(cust.row_key as bigint) = clk.cust_id and clk.purch_flag = 'false')
as cust_clk WHERE cust_clk.prod_id = prod.prod_id
and cust_clk.`date` between '2014-01-01' and '2014-03-31';
```

At the bottom right of the interface, there is a copyright notice: "© 2015 MapR Technologies MAPR 97".

In Drill Explorer, we select the data that we want to view in the browser, or copy our SQL directly into the View Definition SQL field, and then select preview to look at the data that is returned from that query. If we find this query useful to us, we can save it as a view. A view saves the SQL and data source information for us, so that we can instantly reuse the query in the Drill Explorer and even share it with other BI tools through the ODBC interface.

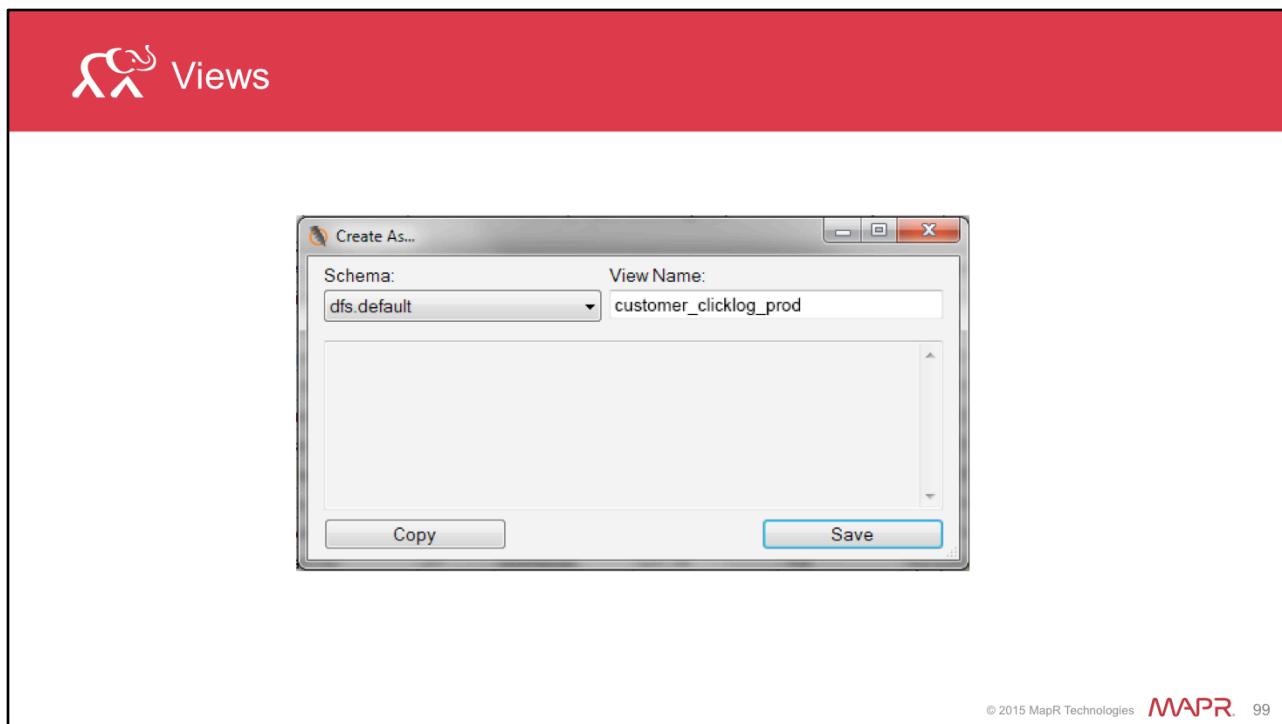


The screenshot shows the Drill Explorer interface with a red header bar containing the 'Views' logo and the word 'Views'. Below the header is a 'Drill Explorer' window. The window has tabs for 'Browse' and 'SQL', with 'SQL' selected. A code editor displays a complex SQL query. To the right of the code editor are two buttons: 'Preview' and 'Create As...', with 'Create As...' highlighted by a red box. Below the code editor, a message says 'Total Number of Records: 47,629'. A table below the message shows 10 rows of data with columns 'cust\_name', 'date', and 'prod\_name'. The data is as follows:

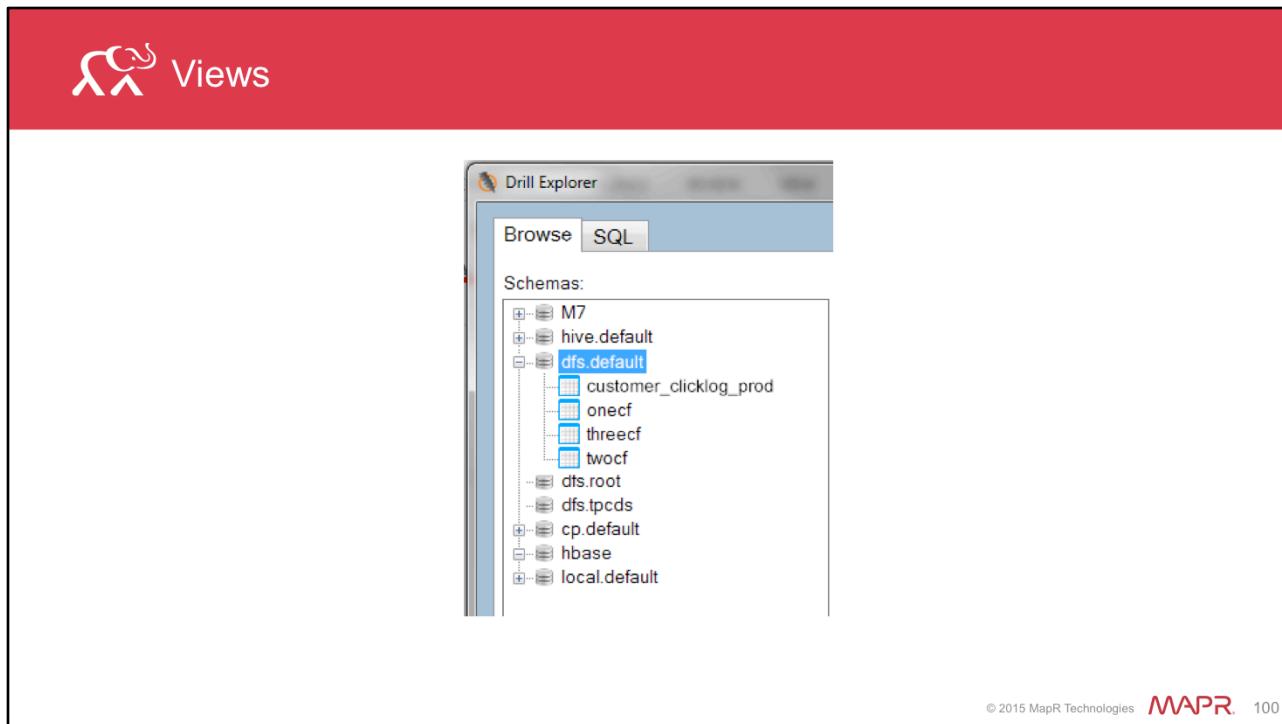
	cust_name	date	prod_name
1	"Jerry Rackley"	2014-01-29T00:00:00.000-08:00	"Sony VAIO Duo 13"
2	"Francis Diaz"	2014-01-29T00:00:00.000-08:00	"Sony VAIO Duo 13"
3	"Lucio Goods"	2014-01-05T00:00:00.000-08:00	"Sony VAIO Duo 13"
4	"Thelma Mcnair"	2014-01-04T00:00:00.000-08:00	"HP Spectre 13 X2"
5	"Kayla Jones"	2014-01-22T00:00:00.000-08:00	"HP Spectre 13 X2"
6	"Carmen Polk"	2014-01-30T00:00:00.000-08:00	"Lenovo ThinkPad S431 Ultrabook"
7	"Regina Moriarty"	2014-01-03T00:00:00.000-08:00	"Lenovo ThinkPad S431 Ultrabook"
8	"Lisa Sprague"	2014-01-05T00:00:00.000-08:00	"Alienware 14"
9	"Tony Hendrix"	2014-01-30T00:00:00.000-08:00	"Alienware 14"
10	"Andre Jude"	2014-01-29T00:00:00.000-08:00	"Alienware 14"

© 2015 MapR Technologies **MAPR** 98

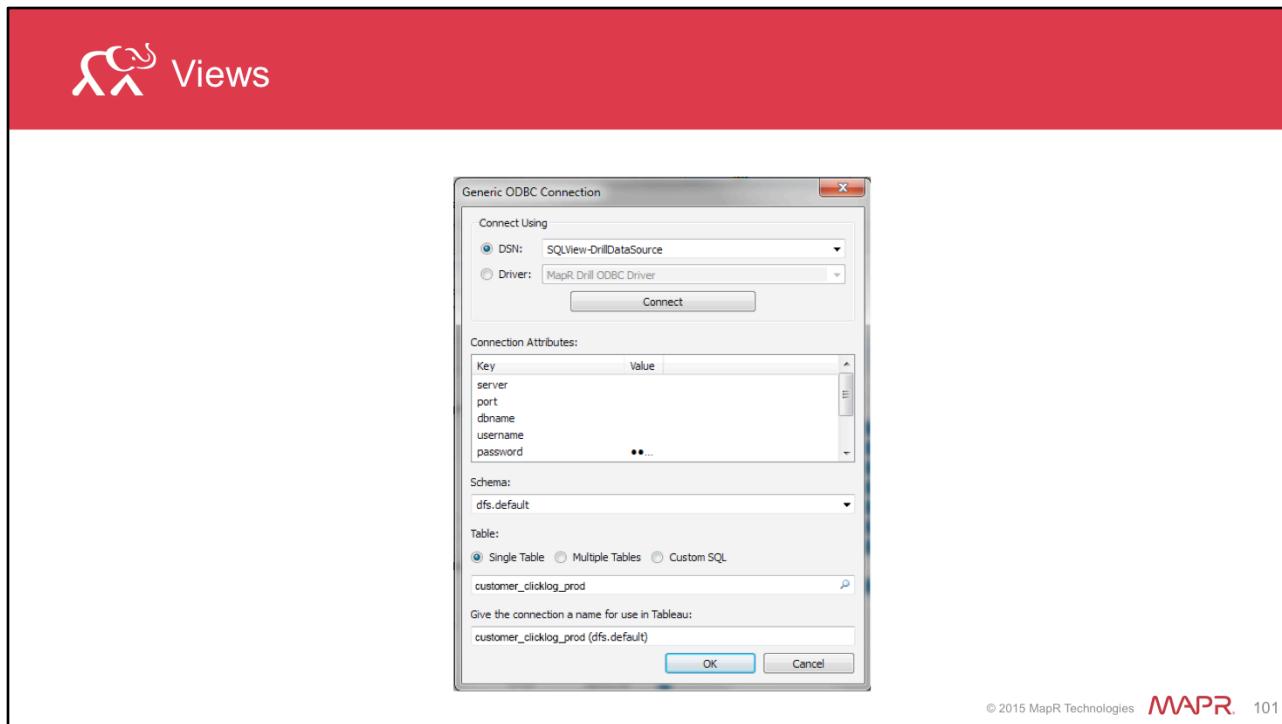
We create our new view by selecting the “Create As...” button.



we give our view a useful name describing the data, select where to store it and hit save.



We can see our new view in the data sources. Selecting this view will fill the Data Preview window with the results of our saved query.



© 2015 MapR Technologies **MAPR**. 101

With our view saved, we can share this information with any BI tool we have that uses an ODBC interface. We simply open our BI tool, point to the view, and then visualize the query in the interface.

Tableau - Book2

File Data Worksheet Dashboard Story Analysis Map Format Server Window Help

Pages Columns Descr

Rows SUM(Number of Records)

Marks Automatic

Color Size Label

Detail Tooltip

Number of Records

Sheet 1 SUM(Number of Records): 27,076

© 2015 MapR Technologies 102

With our view saved, we can share this information with any BI tool we have that uses an ODBC interface. We simply open our BI tool, point to the view, and then visualize the query in the interface.





## Knowledge Check

**Drill Explorer views can be visualized with BI tools using:**

- The ODBC interface
- The JDBC interface
- Either the ODBC or JDBC interface

**the ODBC interface**





## Next Steps



**Congratulations!**

© 2015 MapR Technologies **MAPR** 104

Congratulations, you have completed the Drill Essentials on demand course! You now have a good background on Apache Drill, and how it is a powerful new tool for exploring and visualizing structured and semi structured data.

Explore additional courses at the MapR training academy and the other resources at [doc.mapr.com](http://doc.mapr.com) and [answers.mapr.com](http://answers.mapr.com) to learn more about Hadoop and the ecosystem of components (*Elephants playing poker*)

