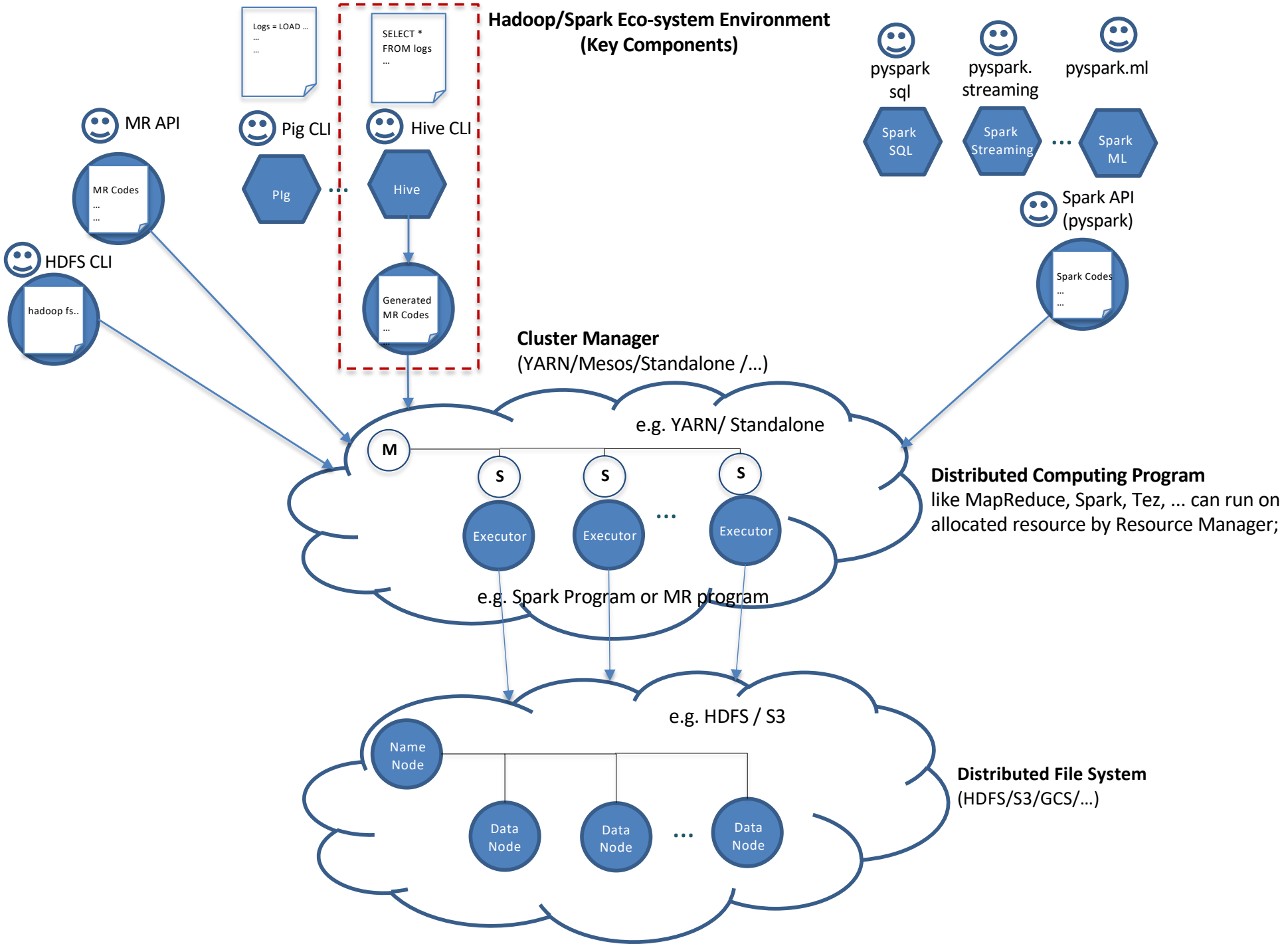
Hive Introduction


Hadoop/Spark Eco-system Environment
(Key Components)
Logs = LOAD ...
...
...
SELECT *
FROM logs
...
MR API
Pig CLI
Hive CLI
MR Codes
...
...
Pig
Hive
HDFS CLI
hadoop fs..
Generated
MR Codes
...
pyspark
sql
pyspark.
streaming
pyspark.ml
Spark
SQL
Spark
Streaming
Spark
ML
Spark API
(pyspark)
Spark Codes
...
...
Cluster Manager
(YARN/Mesos/Standalone /...)
e.g. YARN/ Standalone
M
S
S
S
Executor
Executor
Executor
e.g. Spark Program or MR program
Distributed Computing Program
like MapReduce, Spark, Tez, ... can run on
allocated resource by Resource Manager;
e.g. HDFS / S3
Name
Node
Data
Node
Data
Node
Data
Node
Distributed File System
(HDFS/S3/GCS/...)

Hive history

- Started at Facebook
  - MR development is time consuming
  - Requires detailed and thorough knowledge of MR
  - Limited resources with required expertise
  - No schema to help understand data in HDFS
- Mainly use mapreduce as underlying execution engine
  - After 0.13, Hive can use Tez
  - After 1.1, Hive can run on Spark (still under active development)

Get average temperature from BIG weather data

200701hourly.txt

```
Wban Number, YearMonthDay, Time, Station Type, Maintenance Indicator, Sky Conditions, Visibility, Weather Type, Dry Bulb Temp
03011,20070101,0050,AO2 ,-,CLR                  ,10SM   ,-,30,1,23,29  , 5   ,150,-,0  ,30.14,-,-,AA,
03011,20070101,0150,AO2 ,-,CLR                  ,10SM   ,-,28,1,21,31  , 6   ,160,-,0  ,30.13,-,-,AA,
03011,20070101,0250,AO2 ,-,SCT120               ,10SM   ,-,30,1,23,29  , 4   ,150,-,0  ,30.13,-,-,AA,
03011,20070101,0350,AO2 ,-,CLR                  ,10SM   ,-,30,1,23,29  , 3   ,140,-,0  ,30.12,-,-,AA,
03011,20070101,0450,AO2 ,-,CLR                  ,10SM   ,-,30,5,23,34  , 5   ,150,-,0  ,30.10,-,-,AA,
03011,20070101,0550,AO2 ,-,CLR                  ,10SM   ,-,32,1,25,26  , 4   ,130,-,0  ,30.10,-,-,AA,
03011,20070101,0650,AO2 ,-,CLR                  ,10SM   ,-,32,1,25,26  , 0   ,000,-,0  ,30.11,-,-,AA,
03011,20070101,0750,AO2 ,-,CLR                  ,10SM   ,-,28,5,21,37  , 0   ,000,-,0  ,30.13,-,-,AA,
03011,20070101,0850,AO2 ,-,CLR                  ,10SM   ,-,28,5,21,37  , 0   ,000,-,0  ,30.14,-,-,AA,
03011,20070101,0950,AO2 ,-,CLR                  ,10SM   ,-,32,3,25,29  , 0   ,000,-,0  ,30.14,-,-,AA,
03011,20070101,1050,AO2 ,-,CLR                  ,10SM   ,-,32,3,25,29  , 4   ,290,-,0  ,30.14,-,-,AA,
03011,20070101,1150,AO2 ,-,CLR                  ,10SM   ,-,36,1,27,22  , 4   ,270,-,0  ,30.11,-,-,AA,
03011,20070101,1250,AO2 ,-,CLR                  ,10SM   ,-,37,1,27,22  , 5   ,270,-,0  ,30.09,-,-,AA,
03011,20070101,1350,AO2 ,-,CLR                  ,10SM   ,-,37,1,27,22  , 5   ,280,-,0  ,30.09,-,-,AA,
03011,20070101,1450,AO2 ,-,CLR                  ,10SM   ,-,36,5,27,27  , 6   ,270,-,0  ,30.10,-,-,AA,
03011,20070101,1550,AO2 ,-,CLR                  ,10SM   ,-,34,5,27,29  , 6   ,280,-,0  ,30.12,-,-,AA,
03011,20070101,1630,AO2 ,-,CLR                  ,10SM   ,-,34,7,27,32  , 5   ,280,-,0  ,30.13,-,-,AA,
03011,20070101,1750,AO2 ,-,CLR                  ,10SM   ,-,32,7,25,35  , 0   ,000,-,0  ,30.15,-,-,AA,
03011,20070101,1850,AO2 ,-,CLR                  ,10SM   ,-,32,9,25,38  , 3   ,140,-,0  ,30.17,-,-,AA,
03011,20070101,1950,AO2 ,-,CLR                  ,10SM   ,-,28,10,23,47  , 4   ,120,-,0  ,30.18,-,-,AA
03011,20070101,2050,AO2 ,-,CLR                  ,10SM   ,-,28,9,23,45  , 0   ,000,-,0  ,30.19,-,-,AA,
03011,20070101,2150,AO2 ,-,CLR                  ,10SM   ,-,27,9,21,47  , 5   ,150,-,0  ,30.19,-,-,AA,
03011,20070101,2250,AO2 ,-,CLR                  ,10SM   ,-,27,9,21,47  , 0   ,000,-,0  ,30.21,-,-,AA,
03011,20070101,2350,AO2 ,-,CLR                  ,10SM   ,-,27,7,21,43  , 5   ,140,-,0  ,30.21,-,-,AA,
03011,20070102,0050,AO2 ,-,CLR                  ,10SM   ,-,23,9,19,55  , 3   ,100,-,0  ,30.21,-,-,AA,
03011,20070102,0150,AO2 ,-,CLR                  ,10SM   ,-,23,7,19,50  , 5   ,140,-,0  ,30.23,-,-,AA,
```

In MapReduce

The driver (main)

```java
package iii.mr101;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;


public class AvgTemp {

    public static void main(String[] args) throws Exception {

        JobConf conf = new JobConf(AvgTemp.class);
        conf.setJobName("Avg Temp");

        conf.setInputFormat(SequenceFileInputFormat.class);
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);

        conf.setMapperClass(AvgTempMapper.class);
        conf.setReducerClass(AvgTempReducer.class);

        FileInputFormat.addInputPath(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));

        JobClient.runJob(conf);
    }
}
```

And Build, package, deploy,…

mapper

```java
package iii.mr101;

import java.io.*;

import org.apache.commons.lang.StringUtils;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class AvgTempMapper extends MapReduceBase
implements Mapper<LongWritable, Text, Text, IntWritable>  {

    public void map(LongWritable key, Text value,
            OutputCollector<Text, IntWritable> output, Reporter reporter)
    throws IOException {

        String[] line = value.toString().split(",");

        String dataPart = line[1];
        String temp = line[10];

        if (StringUtils.isNumeric(temp) && !temp.equals("") ){
            output.collect(new Text(dataPart), new IntWritable(Integer.parseInt(temp)));
        }
    }
}
```

reducer

```java
package iii.mr101;

import java.io.*;
import java.util.*;

import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public  class AvgTempReducer extends MapReduceBase
    implements Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text key, Iterator<IntWritable> values,
        OutputCollector<Text, IntWritable> output, Reporter reporter)
    throws IOException {

        int sumTemps = 0;
        int numItems = 0;

        while (values.hasNext()) {
            sumTemps += values.next().get();
            numItems +=1;
        }
        output.collect(key,  new IntWritable(sumTemps / numItems));
    }
}
```

In Spark

```
from pyspark import SparkConf, SparkContext

def is_good(record):
    try:
        temp = int(record.split(",")[10])
    except ValueError:
        return False
    return True


if __name__ == "__main__":
    sc = SparkContext()

    records = sc.textFile("hdfs://localhost/user/cloudera/spark101/avg_temperature/weather")

    good_records = records.filter(is_good)

    day_temp = good_records.map(lambda x: (x.split(",")[1], int(x.split(",")[10])))

    result = day_temp.combineByKey(lambda v: (v, 1), lambda acc, v: (acc[0] + v, acc[1] + 1),
                                   lambda acc1, acc2: (acc1[0] + acc2[0], acc1[1] + acc2[1])) \
        .map(lambda x: (x[0], x[1][0] / x[1][1]))

    for line in result.collect():
        print(line)
```

In Hive or Spark SQL (CLI)

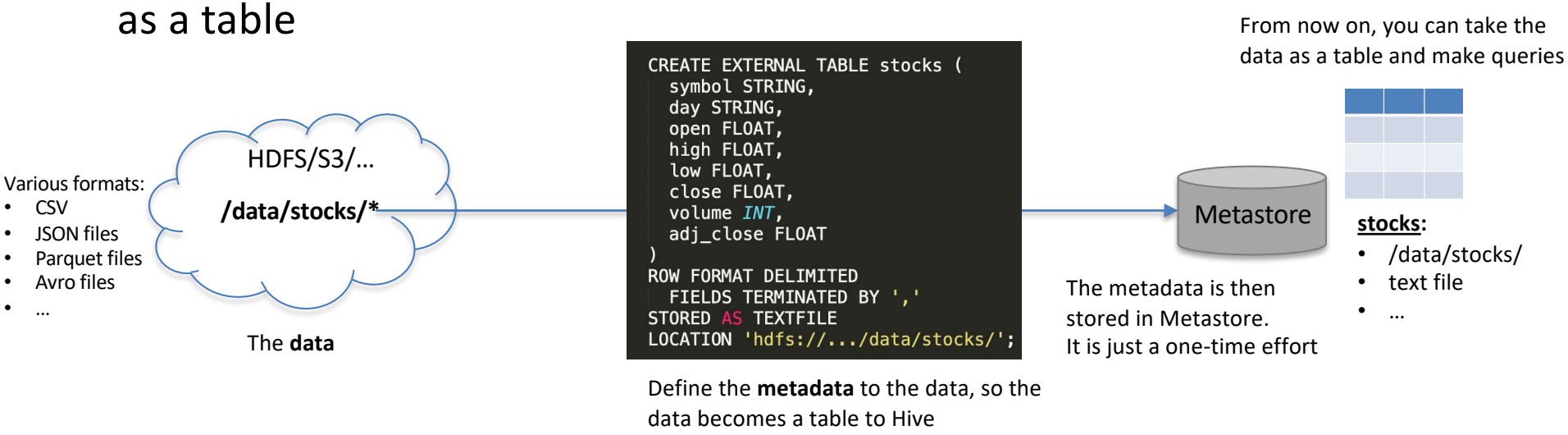
```
SELECT year, AVG(temperature)
FROM records
GROUP BY year;
```

Major steps to use Hive

Using **HiveQL** (SQL-compliant) **+ CLI** (hive or beeline commands)

1. Define the data as a table

- The dataset is on HDFS (or S3, GCS,…)
- Define metadata (i.e. schema) for the dataset to be interpreted by Hive as a table



2. Making quires in HiveQL

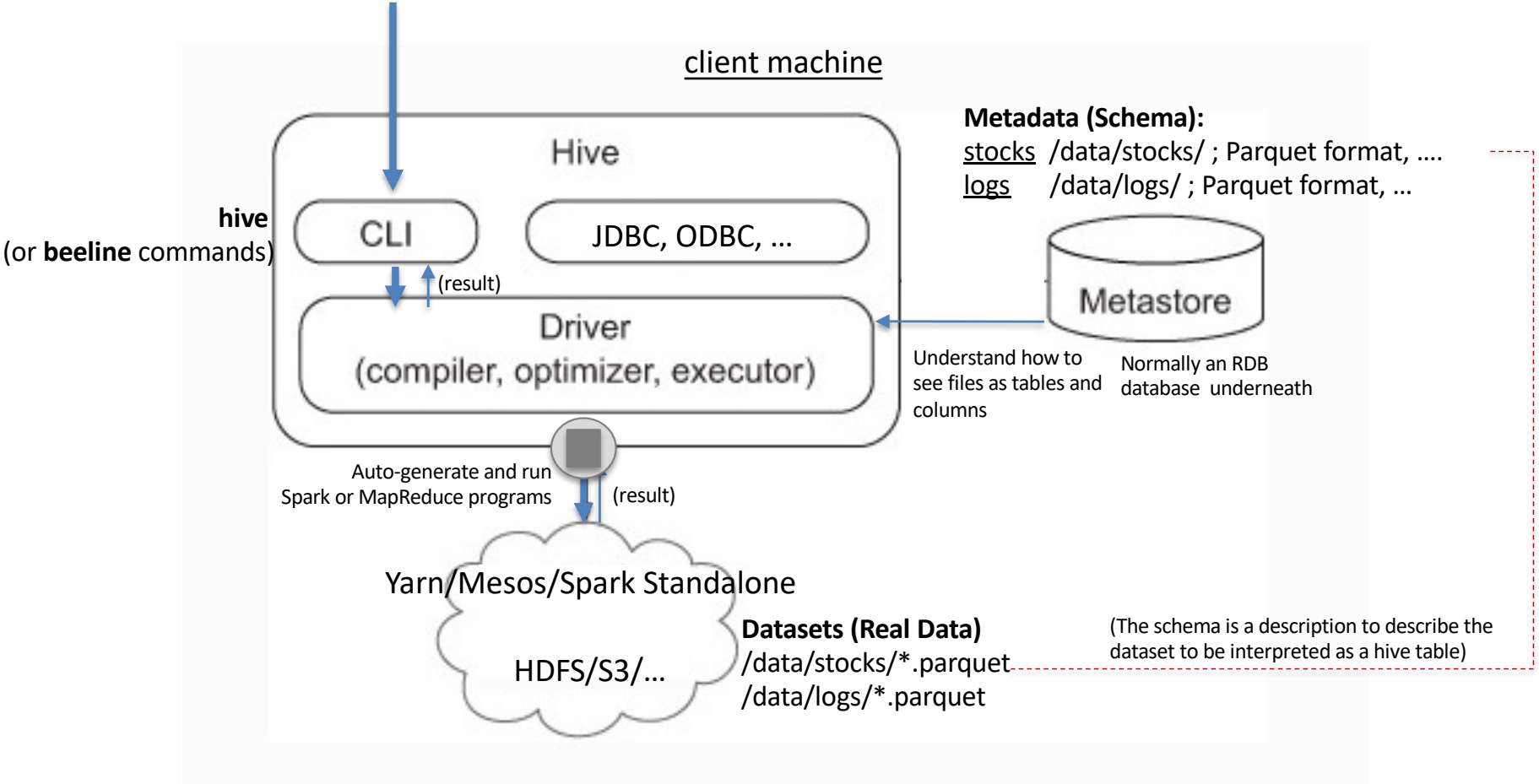
```
SELECT symbol, count(*), AVG(open), Max(open)
FROM stocks
GROUP BY symbol;
```

The HiveQL is translated by Hive engine as a MR or Spark program which is run automatically by Hive

A Query's Run-time Execution Flow

HiveQL
(SQL standard compatible)

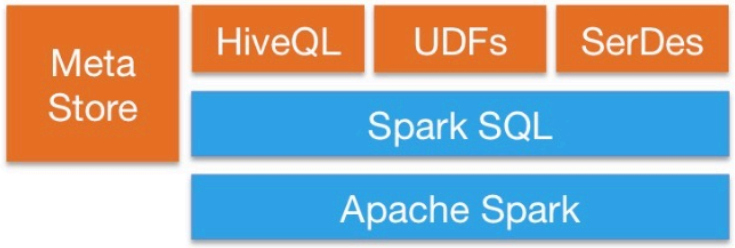
```
SELECT symbol, count(*) AS num_records, AVG(open) AS avg_open , Max(open) AS max_open
FROM stocks
WHERE symbol != 'CSCO'
GROUP BY symbol
HAVING avg_open >= 100
ORDER BY symbol;
```



Metastore is currently supported by several platforms

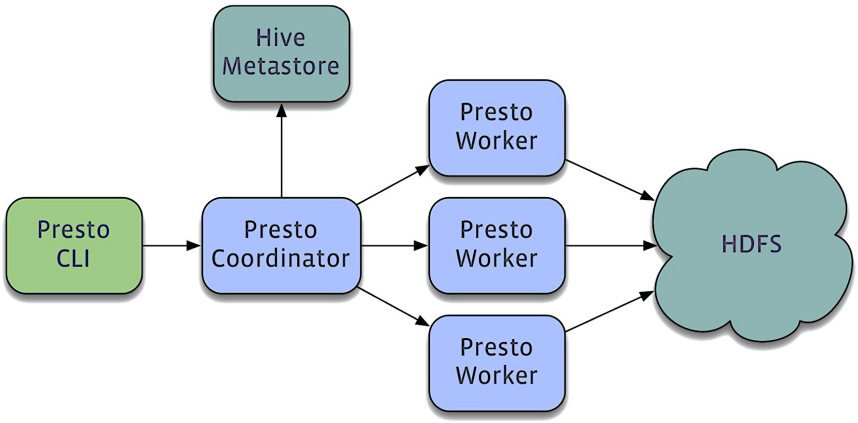
- Spark SQL
- Presto
- Impala
- Pig
- …

Spark SQL

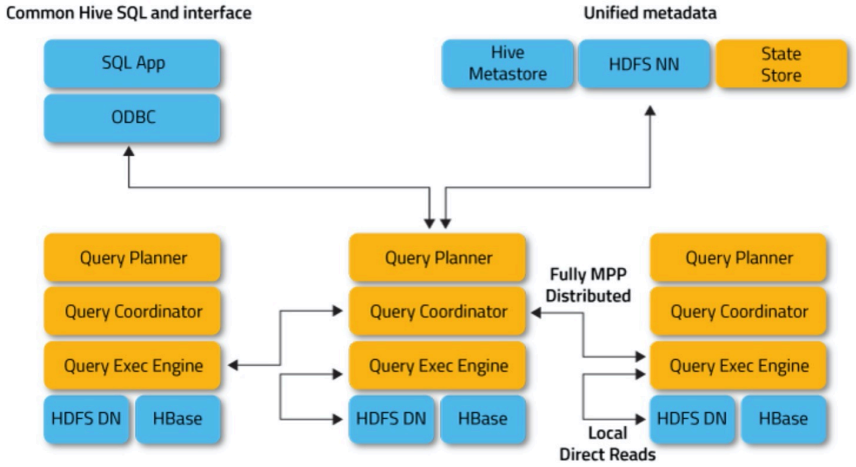


Spark SQL can use existing Hive metastores, SerDes, and UDFs.

Presto

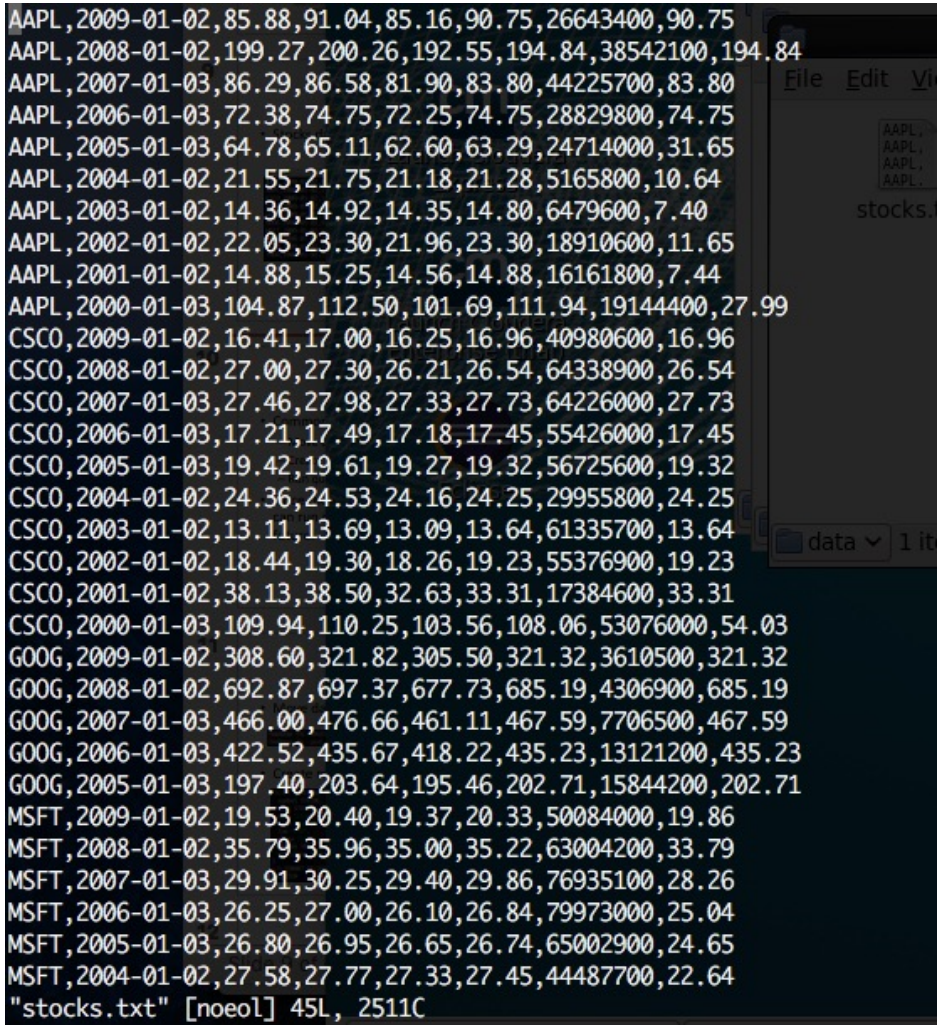


Impala



Demo: Dive into HiveQL

- Process Stocks data
  - Put stocks data into HDFS
  - Create table
  - Run queries



Hive Shell

Hive shell (interactive)

```
[cloudera@localhost hive101]$ hive
Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-common-0.10.0-cdh4.7.0.jar!/hi
ve-log4j.properties
Hive history file=/tmp/cloudera/hive_job_log_ba9c6981-ec65-4818-a4c9-bbee98f4eb5f_1601337972.txt
hive> show tables;
OK
audi_report_input
audi_report_input_staging
good_records_by_year
logs
```

Hive shell (in batch)

```
[cloudera@localhost hive101]$ hive -f max_temp.hive
Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-common-0.10.0-cdh4.7.0.jar!/hi
ve-log4j.properties
Hive history file=/tmp/cloudera/hive_job_log_a15741e8-6c40-4c05-9444-14c8b3f7d533_228130965.txt
OK
Time taken: 0.647 seconds
```

Hive Shell - Commands

```
hive> show tables;
OK
stocks
Time taken: 0.526 seconds, Fetched: 1 row(s)
hive> describe stocks;
OK
symbol                  string
date                    string
open                    float
high                    float
low                     float
close                   float
volume                  int
adj_close               float
Time taken: 0.139 seconds, Fetched: 8 row(s)
hive> describe extended stocks;
Detailed Table Information      Table(tableName:stocks, dbName:default, owner:cloudera, createTime:14459289
29, lastAccessTime:0, retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:symbol, type:string, comment
:null), FieldSchema(name:date, type:string, comment:null), FieldSchema(name:open, type:float, comment:null)
, FieldSchema(name:high, type:float, comment:null), FieldSchema(name:low, type:float, comment:null), FieldS
chema(name:close, type:float, comment:null), FieldSchema(name:volume, type:int, comment:null), FieldSchema(
name:adj_close, type:float, comment:null)], location:hdfs://quickstart.cloudera:8020/user/cloudera/hive101/
stocks/data, inputFormat:org.apache.hadoop.mapred.TextInputFormat, outputFormat:org.apache.hadoop.hive.ql.i
o.HiveIgnoreKeyTextOutputFormat, compressed:false, numBuckets:-1, serdeInfo:SerDeInfo(name:null, serializat
ionLib:org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, parameters:{serialization.format=,, field.delim=
,}), bucketCols:[], sortCols:[], parameters:{}, skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[]
, skewedColValueLocationMaps:{}), storedAsSubDirectories:false), partitionKeys:[], parameters:{EXTERNAL=TRU
E, transient_lastDdlTime=1445928929}, viewOriginalText:null, viewExpandedText:null, tableType:EXTERNAL_TABL
E)
Time taken: 0.091 seconds, Fetched: 10 row(s)
```

Hive Shell - Commands

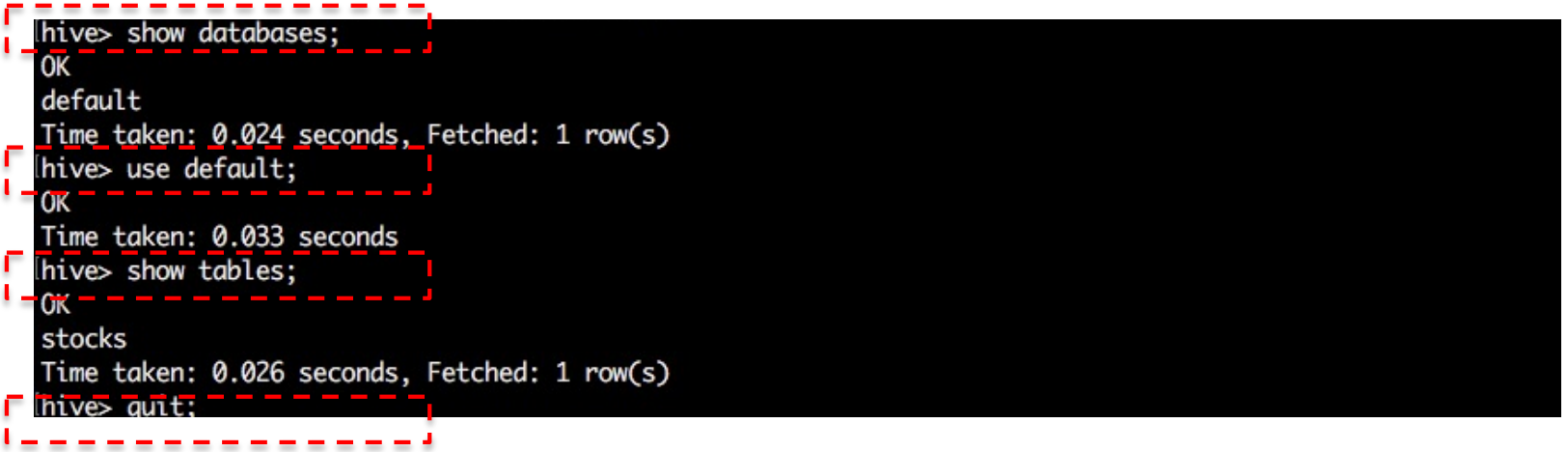
```
hive> describe formatted stocks;
OK
# col_name              data_type               comment

symbol                  string
date                    string
open                    float
high                    float
low                     float
close                   float
volume                  int
adj_close               float

# Detailed Table Information
Database:               default
Owner:                  cloudera
CreateTime:             Mon Oct 26 23:55:29 PDT 2015
LastAccessTime:         UNKNOWN
Protect Mode:           None
Retention:              0
Location:               hdfs://quickstart.cloudera:8020/user/cloudera/hive101/stocks/data
Table Type:             EXTERNAL_TABLE
Table Parameters:
        EXTERNAL                TRUE
        transient_lastDdlTime   1445928929

# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            org.apache.hadoop.mapred.TextInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:             No
Num Buckets:            -1
Bucket Columns:         []
Sort Columns:           []
Storage Desc Params:
        field.delim             ,
        serialization.format    ,
Time taken: 0.628 seconds, Fetched: 35 row(s)
```

Hive Shell - Commands



Hive v.s. RDBMS

- Hive
  - OLAP
  - Schema on Read
    - doesn't verify the data when it is loaded
    - very fast initial load
    - There are many scenarios where the schema is not known at load time
    - Accessed by multiple tools
- RDBMS
  - OLTP and OLAP
  - Schema on Write
    - makes query time performance faster
    - it takes longer to load data into the database

HiveQL

- A SQL-like language in which data is manipulated through relational operations
- HiveQL consists of
  - Data Definition Language (DDL)
    - Create table, Create database,..
    - Drop table, Drop database,..
    - Alter table, Alter database,…
  - Data Manipulation Language (DML)
    - Load data
    - Save result into a table
    - Select statement in HiveQL and SQL

Create database

```
CREATE (DATABASE|SCHEMA) [IF NOT EXISTS] database_name
  [WITH DBPROPERTIES (property_name=property_value, ...)];
```

Create table

```
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name
  [(col_name data_type [COMMENT col_comment], ...)]
   [ROW FORMAT row_format]
   [STORED AS file_format]
     | STORED BY 'storage.handler.class.name' [WITH SERDEPROPERTIES (...)]
  ]
  [LOCATION hdfs_path]
```

Create Table – Table types

- External table and managed table
- Managed table
  - Hive manages the complete lifecycle of your data, including the deletion
- External table
  - Hive only manage the metadata to the data.
    - When you drop a table, the data is unaffected.
  - Consider using External tables if data is already in HDFS

Table metadata in Metastore

```
hive> describe formatted stocks;
OK
# col_name              data_type               comment

symbol                  string
date                    string
open                    float
high                    float
low                     float
close                   float
volume                  int
adj_close               float

# Detailed Table Information
Database:               default
Owner:                  cloudera
CreateTime:             Mon Oct 26 23:55:29 PDT 2015
LastAccessTime:         UNKNOWN
Protect Mode:           None
Retention:              0
Location:               hdfs://quickstart.cloudera:8020/user/cloudera/hive101/stocks/data
Table Type:             EXTERNAL_TABLE
Table Parameters:
        EXTERNAL                TRUE
        transient_lastDdlTime   1445928929

# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            org.apache.hadoop.mapred.TextInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:             No
Num Buckets:            -1
Bucket Columns:         []
Sort Columns:           []
Storage Desc Params:
        field.delim             ,
        serialization.format    ,
Time taken: 0.628 seconds, Fetched: 35 row(s)
```

Demo

- External and Managed tables

Crate table – Column types

Primitive column types

Numeric Types

- `TINYINT` (1-byte signed integer, from `-128` to `127`)
- `SMALLINT` (2-byte signed integer, from `-32,768` to `32,767`)
- `INT` (4-byte signed integer, from `-2,147,483,648` to `2,147,483,647`)
- `BIGINT` (8-byte signed integer, from `-9,223,372,036,854,775,808` to `9,223,372,036,854,775,807`)
- `FLOAT` (4-byte single precision floating point number)
- `DOUBLE` (8-byte double precision floating point number)
- `DECIMAL`
  - Introduced in Hive 0.11.0 with a precision of 38 digits
  - Hive 0.13.0 introduced user definable precision and scale

Date/Time Types

- `TIMESTAMP` (Note: Only available starting with Hive 0.8.0)
- `DATE` (Note: Only available starting with Hive 0.12.0)

String Types

- `STRING`
- `VARCHAR` (Note: Only available starting with Hive 0.12.0)
- `CHAR` (Note: Only available starting with Hive 0.13.0)

Misc Types

- `BOOLEAN`
- `BINARY` (Note: Only available starting with Hive 0.8.0)

```
1  CREATE EXTERNAL TABLE stocks (
2    symbol STRING,
3    day STRING,
4    open FLOAT,
5    high FLOAT,
6    low FLOAT,
7    close FLOAT,
8    volume INT,
9    adj_close FLOAT
10 )
11 ROW FORMAT DELIMITED
12   FIELDS TERMINATED BY ','
13 STORED AS TEXTFILE
14 LOCATION 'hdfs://devenv/user/spark/hive101/stocks/data';
```

Complex column types

- Array
- Struct

Create table
ROW FORMAT…STORED AS...

```
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name
  [(col_name data_type [COMMENT col_comment], ...)]
   [ROW FORMAT row_format]
   [STORED AS file_format]
     | STORED BY 'storage.handler.class.name' [WITH SERDEPROPERTIES (...)]
  ]
  [LOCATION hdfs_path]
```

Create table ROW FORMAT…STORED AS...

- ROW FORMAT…STORED AS...is used to specify a kind of loader to interpret data into a table structure.
- Schedema on read
- Hive supports a varlity of loaders for common data and file types e.g.
  - Text file :
    - In cvs, tsv, regular expression, …
  - Sequencefile
  - Avro
  - Parquet
  - …

Create table - location

- Default location was set by
  - hive.metastore.warehouse.dir
  - /user/hive/warehouse
- If location is not specified, hive will look for data in /user/hive/warehouse/<table_name>

```
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name
  [(col_name data_type [COMMENT col_comment], ...)]
   [ROW FORMAT row_format]
   [STORED AS file_format]
     | STORED BY 'storage.handler.class.name' [WITH SERDEPROPERTIES (...)]
  ]
  [LOCATION hdfs_path]
```

Concept: Schema on Read

- The schema of the data is defined at read time
- Allows extremely agility while dealing with complex data format and excising data
- Hadoop eco-system tools such as Pig, hive, impala, Spark are in schema on read manner

```
CREATE EXTERNAL TABLE stocks (
  symbol STRING,
  day STRING,
  open FLOAT,
  high FLOAT,
  low FLOAT,
  close FLOAT,
  volume INT,
  adj_close FLOAT
)
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION 'hdfs://devenv/user/spark/hive101/stocks/data';
```

HiveQL

- A SQL-like language in which data is manipulated through relational operations
- HiveQL consists of
  - Data Definition Language (DDL)
    - Create table, Create database,..
    - Drop table, Drop database,..
    - Alter table, Alter database,…
  - Data Manipulation Language (DML)
    - Load data
    - Save result into a table
    - Select statement in HiveQL and SQL

Load data

```
LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename
```

```
CREATE TABLE records2 (station STRING, year STRING, temperature INT, quality INT)
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY '\t';

LOAD DATA LOCAL INPATH 'input/ncdc/micro-tab/sample2.txt'
OVERWRITE INTO TABLE records2;
```

Save result into a table

If the output table does not exist:

```
CREATE TABLE tablename1
   ROW FORMAT ...
   STORED AS ...
   AS
SELECT ...
FROM ...
```

If the output table exists:

```
INSERT OVERWRITE TABLE tablename1
SELECT ...
FROM ...

INSERT INTO TABLE tablename1
SELECT ...
FROM ...
```

Demo

- Save result into a table

Make Quires using SELECT statements

- Use any SQL syntax you leaned before.

```
SELECT symbol, count(*) AS num_records, AVG(open) AS avg_open , Max(open) AS max_open
FROM stocks
WHERE symbol != 'CSCO'
GROUP BY symbol
HAVING avg_open >= 100
ORDER BY symbol;
```

- Some syntaxes are extended by HiveQL

For example:

```
SELECT TRANSFORM (src.userid, src.logtime) USING './timeonsite.py' AS (userid, timeonsite)
FROM (
  SELECT userid, logtime
  FROM page_views
  ORDER BY userid, logtime
) src;
```

Built-in Functions

Mathematical Functions

The following built-in mathematical functions are supported in Hive; most return NULL when the argument(s) are NULL:

Return Type	Name (Signature)	Description
DOUBLE	round(DOUBLE a)	Returns the rounded `BIGINT` value of `a`.
DOUBLE	round(DOUBLE a, INT d)	Returns `a` rounded to `d` decimal places.
DOUBLE	bround(DOUBLE a)	Returns the rounded BIGINT value of `a` using HALF_EVEN rounding mode (as of Hive 1.3.0, 2.0.0). Also known as Gaussian rounding or bankers' rounding. Example: bround(2.5) = 2, bround(3.5) = 4.
DOUBLE	bround(DOUBLE a, INT d)	Returns `a` rounded to `d` decimal places using HALF_EVEN rounding mode (as of Hive 1.3.0, 2.0.0). Example: bround(8.25, 1) = 8.2, bround(8.35, 1) = 8.4.
BIGINT	floor(DOUBLE a)	Returns the maximum `BIGINT` value that is equal to or less than `a`.
BIGINT	ceil(DOUBLE a), ceiling(DOUBLE a)	Returns the minimum BIGINT value that is equal to or greater than `a`.
DOUBLE	rand(), rand(INT seed)	Returns a random number (that changes from row to row) that is distributed uniformly from 0 to 1. Specifying the seed will make sure the generated random number sequence is deterministic.
DOUBLE	exp(DOUBLE a), exp(DECIMAL a)	Returns e^a where `e` is the base of the natural logarithm. Decimal version added in Hive 0.13.0.

More on https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF

Built-in Functions

String Functions

The following built-in String functions are supported in Hive:

Return Type	Name(Signature)	Description
int	ascii(string str)	Returns the numeric value of the first character of str.
string	base64(binary bin)	Converts the argument from binary to a base 64 string (as of Hive 0.12.0).
string	concat(string|binary A, string|binary B...)	Returns the string or bytes resulting from concatenating the strings or bytes passed in as parameters in order. For example, concat('foo', 'bar') results in 'foobar'. Note that this function can take any number of input strings.
array<struct<string,double>>	context_ngrams(array<array<string>>, array<string>, int K, int pf)	Returns the top-k contextual N-grams from a set of tokenized sentences, given a string of "context". See StatisticsAndDataMining for more information.
string	concat_ws(string SEP, string A, string B...)	Like concat() above, but with custom separator SEP.
string	concat_ws(string SEP, array<string>)	Like concat_ws() above, but taking an array of strings. (as of Hive 0.9.0)
string	decode(binary bin, string charset)	Decodes the first argument into a String using the provided character set (one of 'US-ASCII', 'ISO-8859-1', 'UTF-8', 'UTF-16BE', 'UTF-16LE', 'UTF-16'). If either argument is null, the result will also be null. (As of Hive 0.12.0.)
binary	encode(string src, string charset)	Encodes the first argument into a BINARY using the provided character set (one of 'US-ASCII', 'ISO-

More on https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF

Built-in Functions

Date Functions

The following built-in date functions are supported in Hive:

Return Type	Name(Signature)	Description
string	from_unixtime(bigint unixtime[, string format])	Converts the number of seconds from unix epoch (1970-01-01 00:00:00 UTC) to a string representing the timestamp of that moment in the current system time zone in the format of "1970-01-01 00:00:00".
bigint	unix_timestamp()	Gets current Unix timestamp in seconds.
bigint	unix_timestamp(string date)	Converts time string in format `yyyy-MM-dd HH:mm:ss` to Unix timestamp (in seconds), using the default timezone and the default locale, return 0 if fail: unix_timestamp('2009-03-20 11:30:01') = 1237573801
bigint	unix_timestamp(string date, string pattern)	Convert time string with given pattern (see [http://docs.oracle.com/javase/tutorial/i18n/format/simpleDateFormat.html]) to Unix time stamp (in seconds), return 0 if fail: unix_timestamp('2009-03-20', 'yyyy-MM-dd') = 1237532400.
string	to_date(string timestamp)	Returns the date part of a timestamp string: to_date("1970-01-01 00:00:00") = "1970-01-01".
int	year(string date)	Returns the year part of a date or a timestamp string: year("1970-01-01 00:00:00") = 1970, year("1970-01-01") = 1970.
int	quarter(date/timestamp/string)	Returns the quarter of the year for a date, timestamp, or string in the range 1 to 4 (as of Hive 1.3.0). Example: quarter('2015-04-08') = 2.
int	month(string date)	Returns the month part of a date or a timestamp string: month("1970-11-01 00:00:00") = 11, month("1970-11-01") = 11.
int	day(string date) dayofmonth(date)	Returns the day part of a date or a timestamp string: day("1970-11-01 00:00:00") = 1, day("1970-11-01") = 1.

More on https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF

Built-in Functions

Conditional Functions

Return Type	Name(Signature)	Description
T	if(boolean testCondition, T valueTrue, T valueFalseOrNull)	Returns valueTrue when testCondition is true, returns valueFalseOrNull otherwise.
boolean	isnull(a)	Returns true if a is NULL and false otherwise.
boolean	isnotnull (a)	Returns true if a is not NULL and false otherwise.
T	nvl(T value, T default_value)	Returns default value if value is null else returns value (as of HIve 0.11).
T	COALESCE(T v1, T v2, ...)	Returns the first v that is not NULL, or NULL if all v's are NULL.
T	CASE a WHEN b THEN c [WHEN d THEN e]* [ELSE f] END	When a = b, returns c; when a = d, returns e; else returns f.
T	CASE WHEN a THEN b [WHEN c THEN d]* [ELSE e] END	When a = true, returns b; when c = true, returns d; else returns e.

More on https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF

Demo & Exercise

- Load data and Join operations

Create an UDF function

1. Import necessary packages
   - `import org.apache.hadoop.hive.ql.exec.UDF;`
   - `import anything you will use`
2. Create a class that extends the UDF abstract class
   - Implement evaluate() method

Create an UDF

```java
package com.iii.udf;

import org.apache.hadoop.hive.ql.exec.UDF;
import org.apache.hadoop.io.Text;
import java.util.*;


public final class MyReverse extends UDF {

    public Text evaluate(Text s) {
        if (s == null) {
            return null;
        }
        String reverse = new StringBuilder(s.toString()).reverse().toString();
        return new Text(reverse);
    }
}
```

Use the UDF in Hive

Inside HIVE environment:

1. Registration:

```
add jar /home/spark/IdeaProjects/proj_hive_udf/hive_udf.jar;
CREATE TEMPORARY FUNCTION revs as 'com.iii.udf.MyReverse’;
```

2.Use it:

```
SELECT agent, revs(agent) FROM page_views limit 100;
```

Demo

- Create a UDF function and use it

.hiverc file

- Put commonly used UDF registrations in the .hiverc file
- .hiverc file can be in
  - User's home directory
  - Hive's bin directory

Demo: Partition table

- Introduce ”Partition table” through the examples