Drill

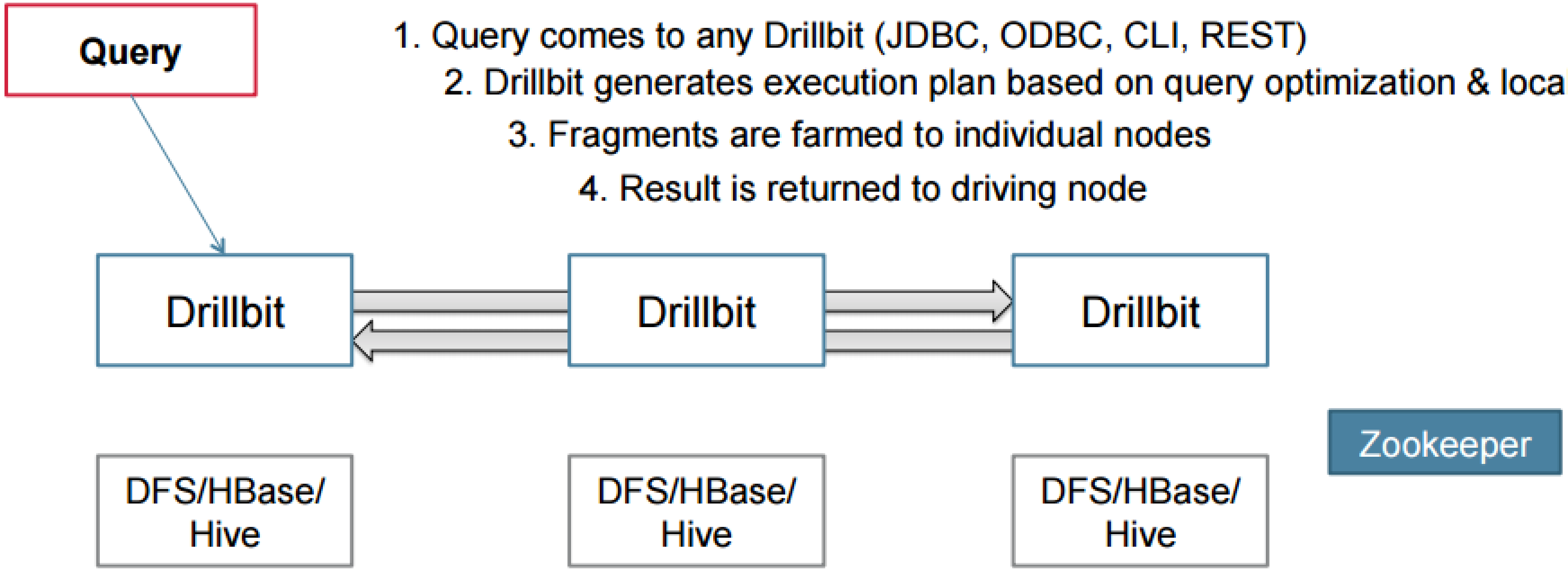
Definitions

- Drill is an Apache open-source SQL query engine for Big Data exploration.
- 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 ANSI SQL, the industry-standard query language
- Apache Drill is inspired by Google's Dremel, Drill is designed to scale to several thousands of nodes and query petabytes of data at interactive speeds that BI/Analytics environments require.

Design

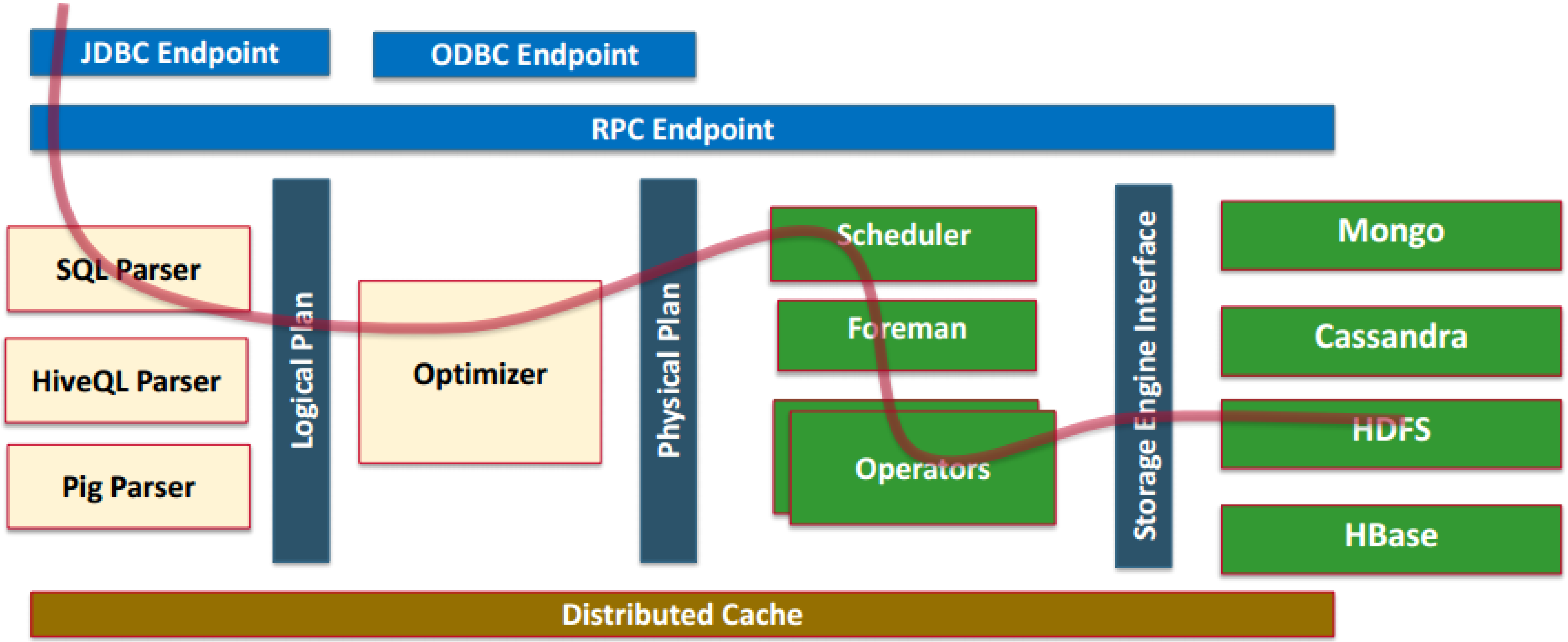
- Schema Free
- Uniformity in data sources
- Cluster of commodity servers
  - Daemon (drillbit) on each node
- ZooKeeper maintains ephemeral cluster membership information
  - Drillbit uses ZooKeeper to find other drillbits in the cluster
  - Client uses ZooKeeper to find drillbits
- Built-in, optimistic query execution engine. Doesn't require a particular storage or execution system (MapReduce, Spark, Tez)
  - Better performance and manageability
- Data processing unit is columnar record batches – Enables schema flexibility with negligible performance impact
  - Designed for Extensibility at all layers

Architecture

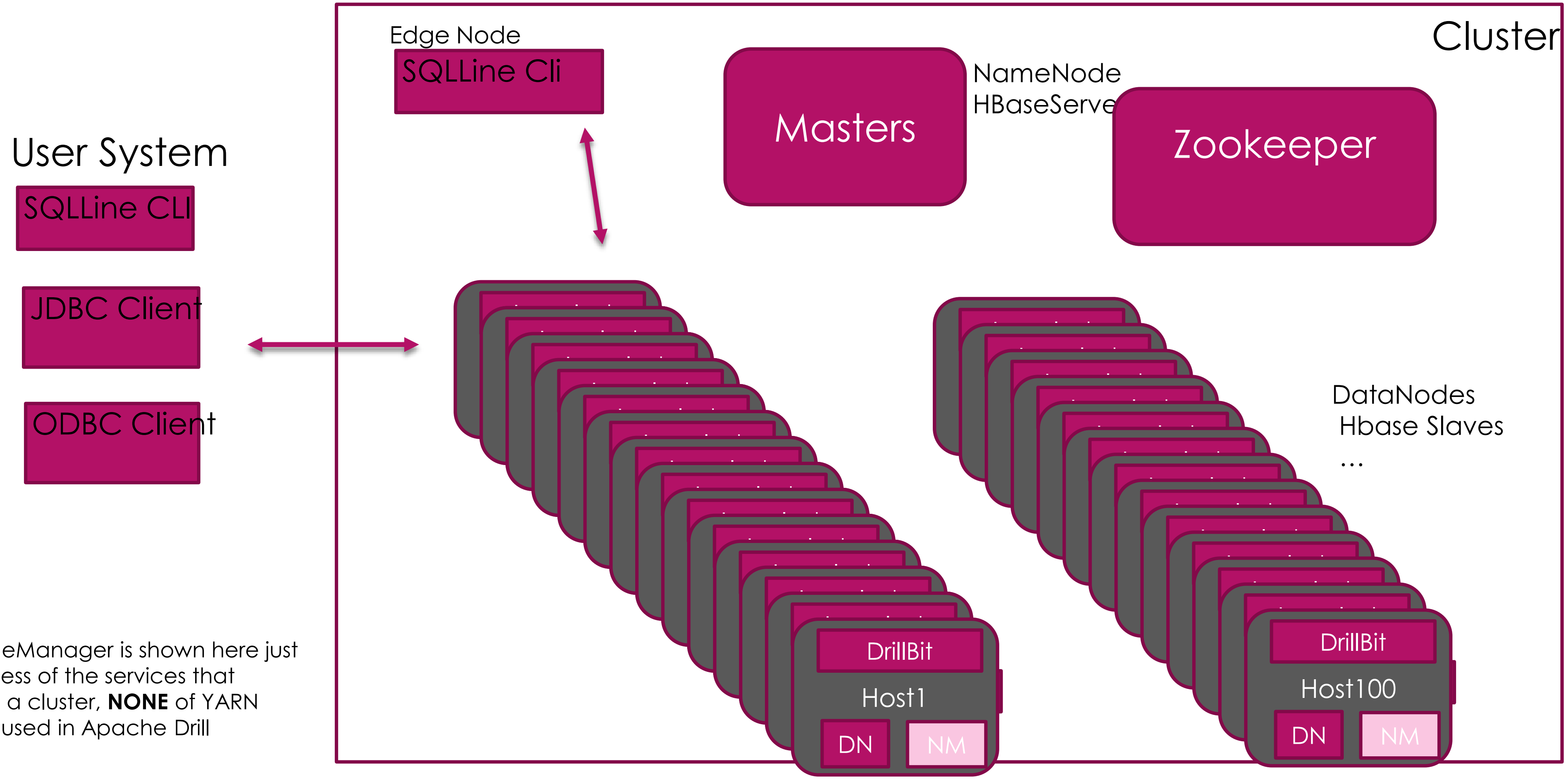


1. Query comes to any Drillbit (JDBC, ODBC, CLI, REST)
2. Drillbit generates execution plan based on query optimization & locality
3. Fragments are farmed to individual nodes
4. Result is returned to driving node

Architecture



Drill Physical Architecture



User System

SQLLine CLI

JDBC Client

ODBC Client

Cluster

Edge Node

SQLLine Cli

Masters

NameNode
HBaseServe

Zookeeper

DataNodes
Hbase Slaves
…

DrillBit
Host1
DN NM

DrillBit
Host100
DN NM

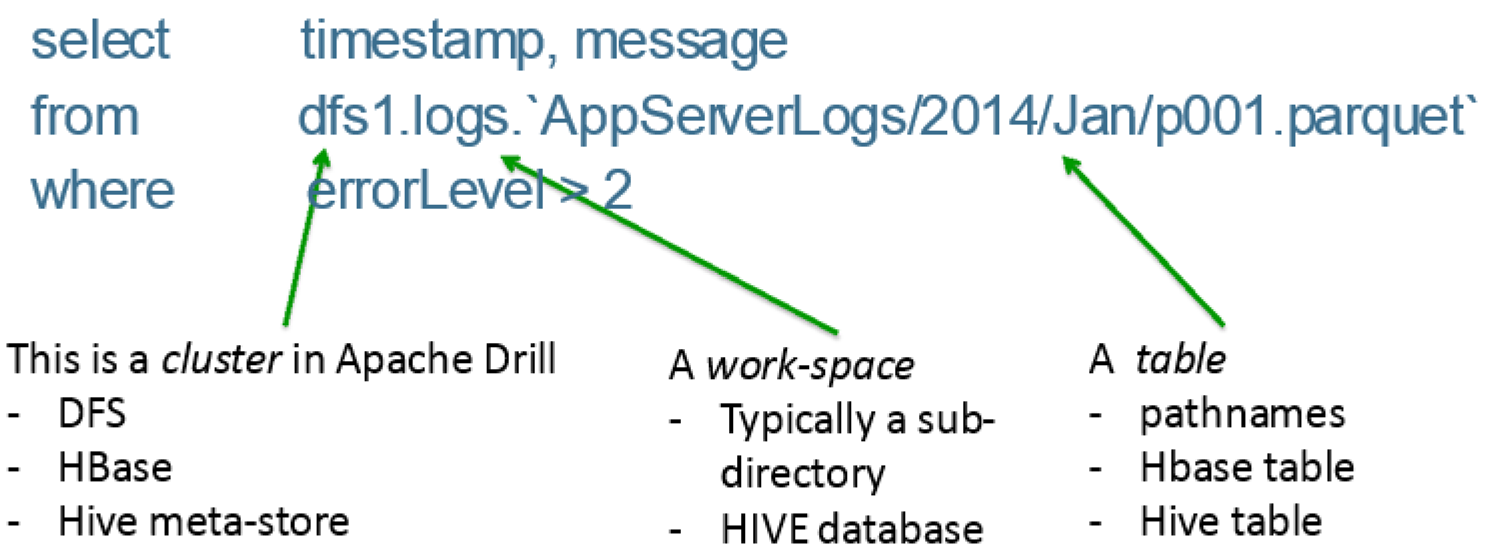
Note that NodeManager is shown here just for completeness of the services that typically run in a cluster, **NONE** of YARN processes are used in Apache Drill

Unified Datasource Access

- JSON
- CSV
- ORC (ie, all Hive types)
- Parquet
- HBase tables
- ... can combine them

```
Select  USERS.name,  PROF.emails.work
from
  dfs.logs.`/data/logs`  LOGS,
  dfs.users.`/profiles.json`  USERS,
where
   LOGS.uid = USERS.uid   and
   errorLevel > 5
order by  count(*);
```

Datasources in the Query



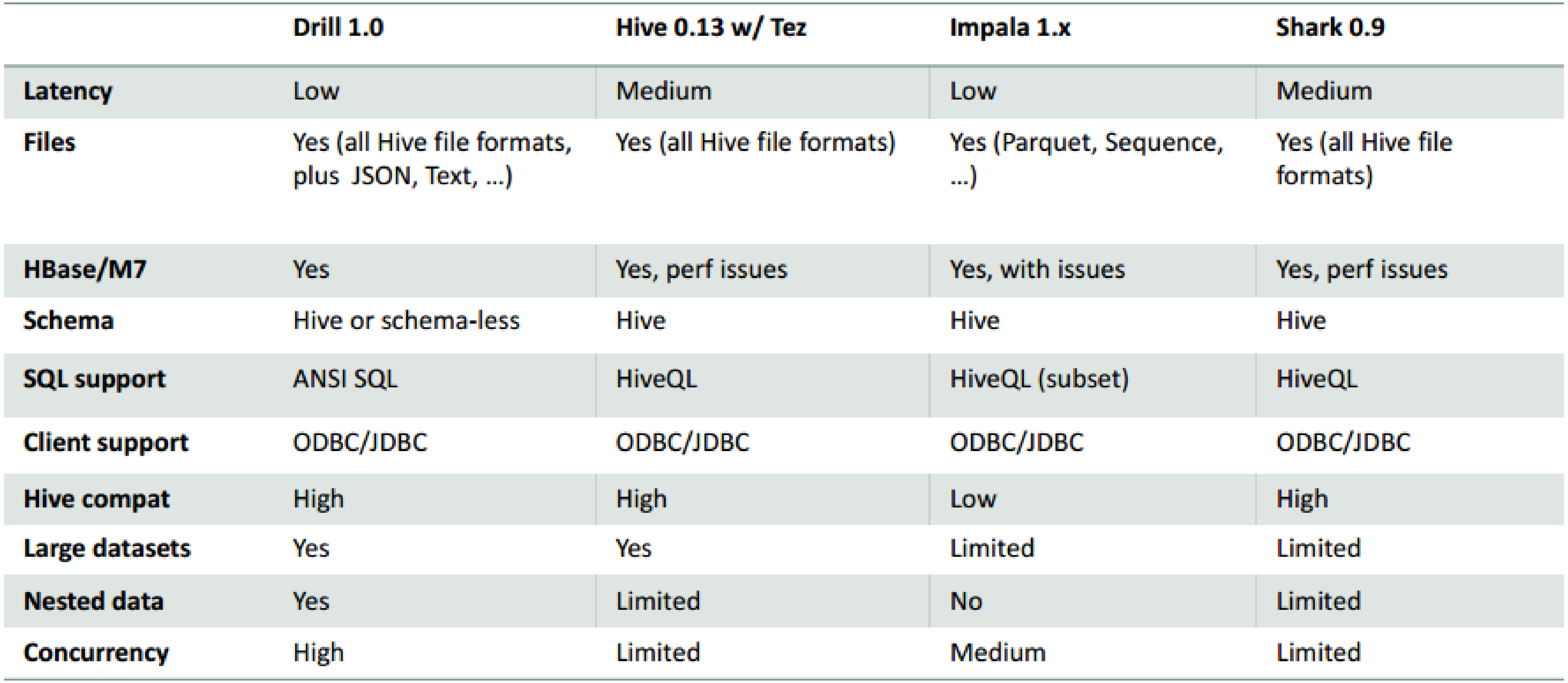
```
select      timestamp, message
from        dfs1.logs.`AppServerLogs/2014/Jan/p001.parquet`
where       errorLevel > 2
```

This is a *cluster* in Apache Drill
- DFS
- HBase
- Hive meta-store

A *work-space*
- Typically a sub-directory
- HIVE database

A *table*
- pathnames
- Hbase table
- Hive table

Comparision



	Drill 1.0	Hive 0.13 w/ Tez	Impala 1.x	Shark 0.9
Latency	Low	Medium	Low	Medium
Files	Yes (all Hive file formats, plus JSON, Text, …)	Yes (all Hive file formats)	Yes (Parquet, Sequence, …)	Yes (all Hive file formats)
HBase/M7	Yes	Yes, perf issues	Yes, with issues	Yes, perf issues
Schema	Hive or schema-less	Hive	Hive	Hive
SQL support	ANSI SQL	HiveQL	HiveQL (subset)	HiveQL
Client support	ODBC/JDBC	ODBC/JDBC	ODBC/JDBC	ODBC/JDBC
Hive compat	High	High	Low	High
Large datasets	Yes	Yes	Limited	Limited
Nested data	Yes	Limited	No	Limited
Concurrency	High	Limited	Medium	Limited

In Action

- Current version 1.5
- Download from Apache site
- Untar
- Set DRILL_HOME
- Also, for convenience set PATH to DRILL_HOME/bin
- Copy the movies.json and business.json datasets to your test host
- Start drill in Embedded mode by running the command “drill-embedded”

Movies DataSet

```
select * from dfs.`/home/hdtester/movies.json`;

select title, `year`, country  from dfs.`/home/hdtester/movies.json` ;

select tbl.title, tbl.`year` from dfs.`/home/hdtester/movies.json` as tbl ;

select tbl.title, tbl.country, tbl.genre,tbl.director.id,tbl.director.year_of_birth  from dfs.`/home/hdtester/movies.json` as tbl ;

select CONCAT(CONCAT(tbl.director.first_name, ` `),tbl.director.last_name)  from dfs.`/home/hdtester/movies.json` as tbl ;

select tbl.director.first_name, COUNT(*) NUM_MOVIES_DIRECTED from dfs.`/home/hdtester/movies.json` as tbl
group by tbl.director.first_name
order by NUM_MOVIES_DIRECTED desc;
```

Business Dataset

- select * from dfs.`/home/hdtester/business.json` limit 10;

- select state, count(review_count) as Reviews from dfs.`/home/hdtester/business.json` group by state;

- use `dfs.tmp`;
- create view bv as select state, count(review_count) as Reviews from dfs.`/home/hdtester/business.json` group by state;