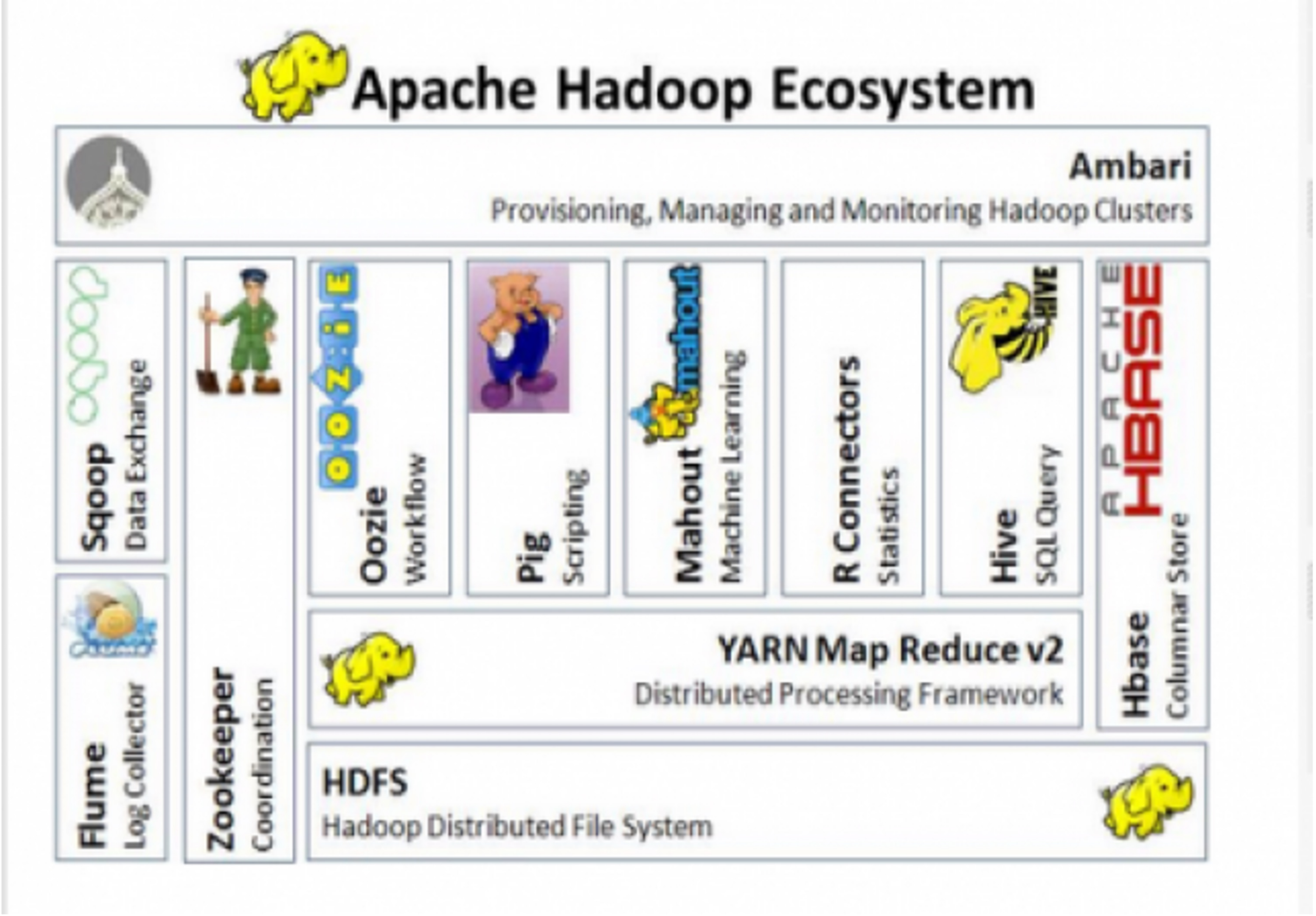
Data Retrieval



Source: https://opensource.com/life/14/8/intro-apache-hadoop-big-data

Hive Introduction

- Data Warehousing Solution built on top of Hadoop
- Provides SQL-like query language named HiveQL
- Minimal learning curve for people with SQL expertise
- Data analysts are target audience

- Early Hive development work started at Facebook in 2007
- Today Hive is an Apache project under Hadoop

— http://hive.apache.org

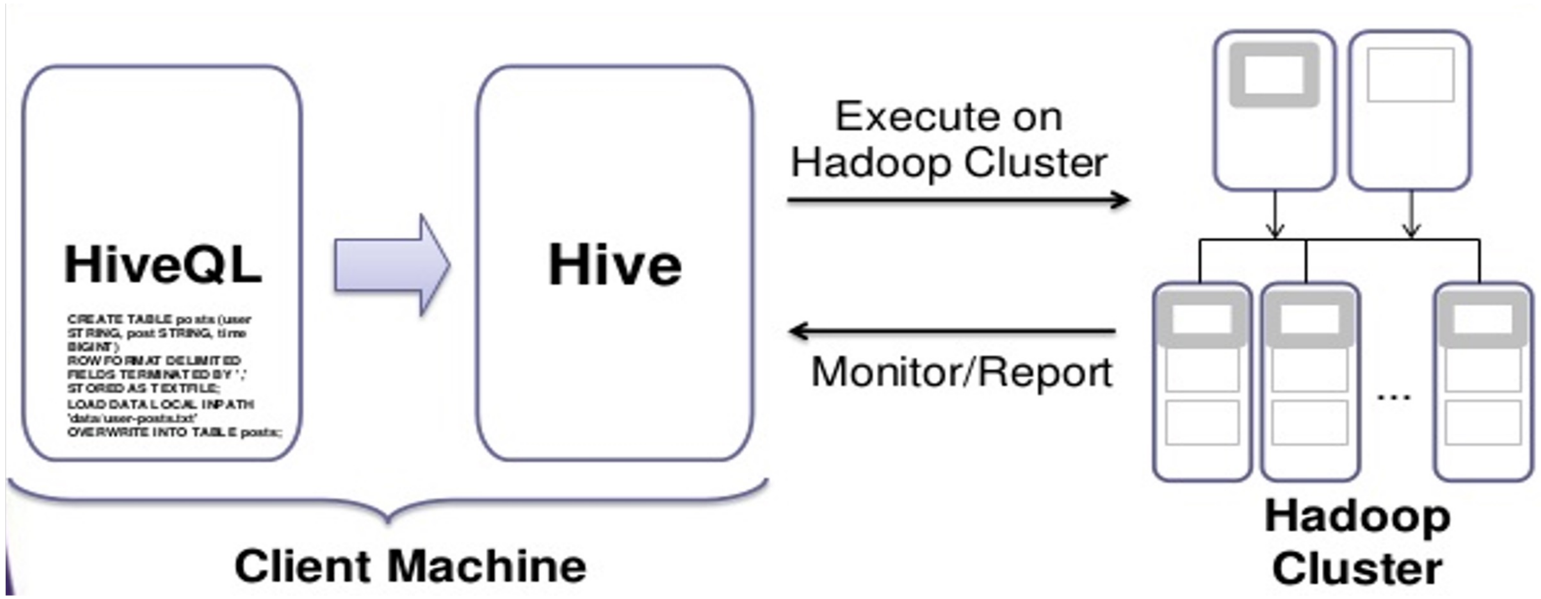
Hive Provides

- Ability to bring structure to various data Formats
- Simple interface for ad hoc querying, analyzing and summarizing large amounts of data
- Access to files on various data stores such as HDFS and Hbase
- Hive is not designed for online transaction processing and does not offer real-time queries and row level updates. It is best used for batch jobs over large sets of immutable data (like web logs).

- https://cwiki.apache.org/confluence/display/Hive/Tutorial

Hive

Translates HiveQL statements into a set of MapReduce Jobs which are then executed on a Hadoop Cluster



```
CREATE TABLE posts (user
STRING, post STRING, time
BIGINT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
LOAD DATA LOCAL INPATH
'data/user-posts.txt'
OVERWRITE INTO TABLE posts;
```

Hive Metastore

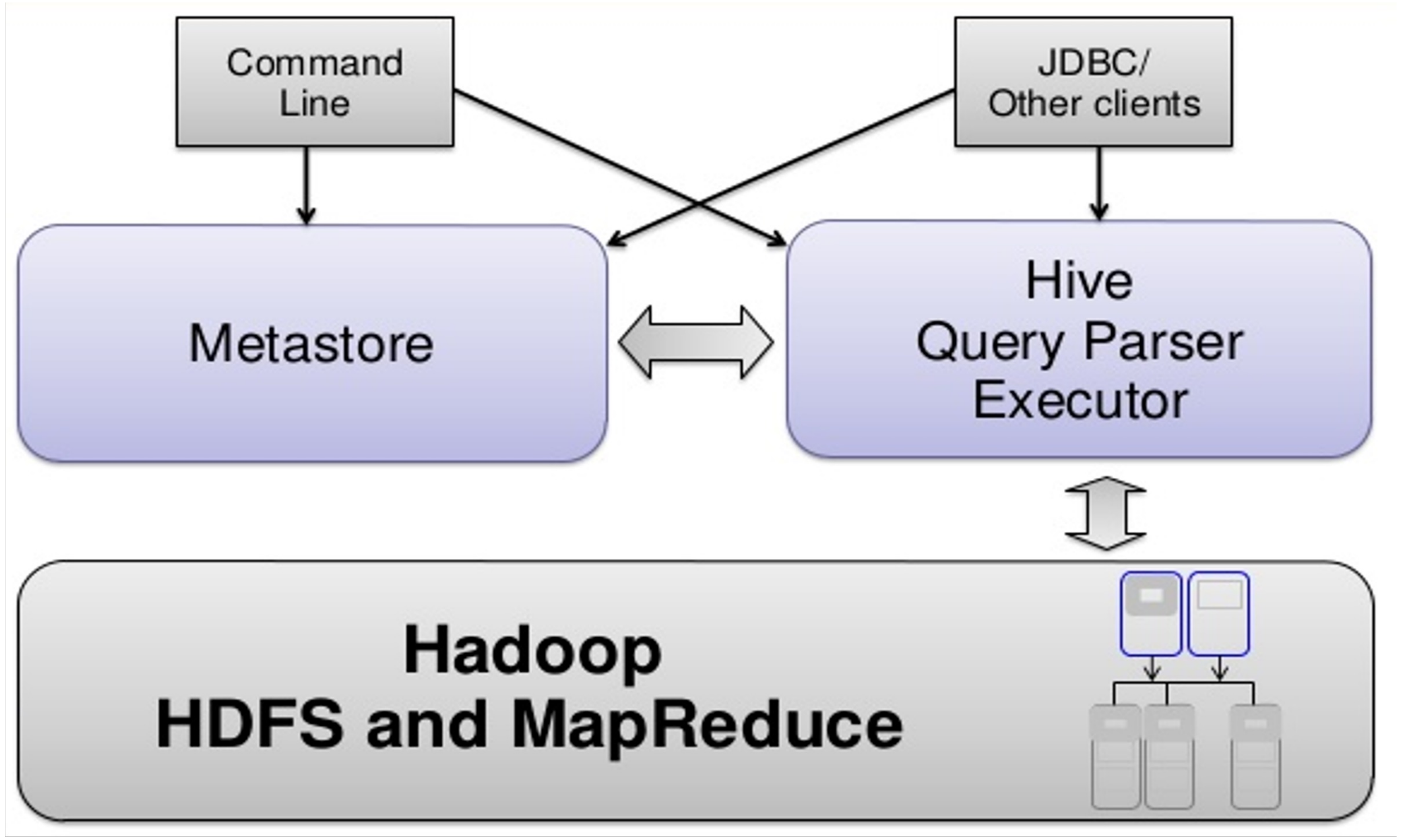
To support features like schema(s) and data partitioning Hive keeps its metadata in a Relational Database

Packaged with Derby, a lightweight embedded SQL DB

- Schema is not shared between users as each user has their own instance of embedded Derby
- Stored in metastore_db directory which resides in the directory that hive was started from

Can easily switch another SQL installation such as MySQL

Hive Architecture



Hive Architecture

Metastore: stores system catalog

Driver: manages life cycle of HiveQL query as it moves thru' HIVE; also manages session handle and session statistics

Query compiler: Compiles HiveQL into a directed acyclic graph of map/reduce tasks

Execution engines: The component executes the tasks in proper dependency order; interacts with Hadoop

HiveServer: provides Thrift interface and JDBC/ODBC for integrating other applications.

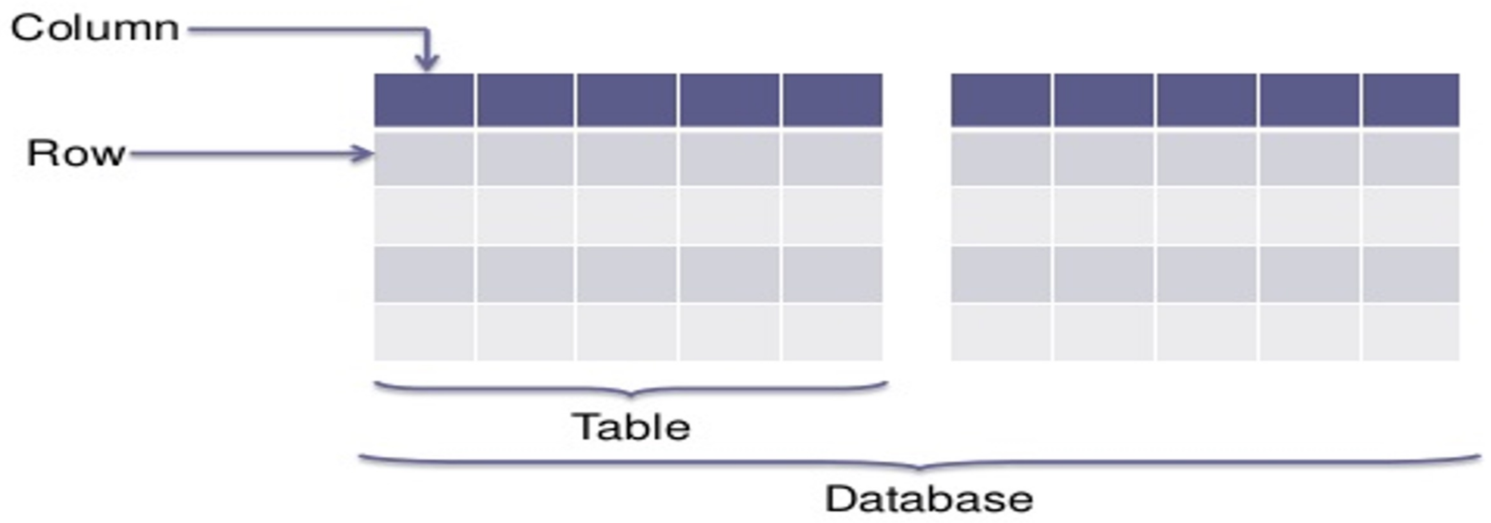
Client components: CLI, web interface, jdbc/odbc interface,

Extensibility interface: include SerDe, User Defined Functions and User Defined Aggregate Function.

Hive Concepts

Re-used from Relational Databases

– **Database**: Set of Tables, used for name conflicts resolution
– **Table**: Set of Rows that have the same schema (same columns)
– **Row**: A single record; a set of columns
– **Column**: provides value and type for a single value



HiveQL - Language Capabilities

- Ability to filter rows from a table using a where clause.
- Ability to select certain columns from the table using a select clause.
- Ability to do equi-joins between two tables.
- Ability to evaluate aggregations on multiple "group by" columns for the data stored in a table.
- Ability to store the results of a query into another table.
- Ability to download the contents of a table to a local (e.g., nfs) directory.
- Ability to store the results of a query in a hadoop dfs directory.
- Ability to manage tables and partitions (create, drop and alter).
- Ability to plug in custom scripts in the language of choice for custom map/reduce jobs.

Hive Installation

- **Set $HADOOP_HOME environment variable**
  – Was done as a part of HDFS installation

- **Set $HIVE_HOME and add hive to the PATH**

```
export HIVE_HOME=$<Hive_path>
export PATH=$PATH:$HIVE_HOME/bin
```

- **Hive will store its tables on HDFS**

```
$ hdfs dfs -mkdir /tmp
$ hdfs dfs -mkdir /user/hive/warehouse
$ hdfs dfs -chmod g+w /tmp
$ hdfs dfs -chmod g+w /user/hive/warehouse
```

Run Hive

```
$ hive

Logging initialized using configuration in jar:file:/usr/local/hadoop/lib/hive-common-0.13.1.jar!/hive-log4j.properties

hive>

<Hive’s Interactive Command Line Interface (CLI)>
```

HiveQL (Hive Query Language) I

- Data Units
  - Databases
  - Tables
  - Partitions
  - Buckets (or Clusters)
- Hive Data Types
- Complex Types
- Built In Operators and Functions
- All Hive keywords are case-insensitive

- https://cwiki.apache.org/confluence/display/Hive/Tutoria

Creating, Showing, Altering, and Dropping Tables

```
CREATE TABLE page_view1(viewTime INT, userid BIGINT,
page_url STRING, referrer_url STRING,

ip STRING COMMENT 'IP Address of the User')

COMMENT 'This is the page view table'

PARTITIONED BY(dt STRING, country STRING)

CLUSTERED BY(userid) SORTED BY(viewTime) INTO 32 BUCKETS

ROW FORMAT DELIMITED
FIELDS TERMINATED BY '1'

COLLECTION ITEMS TERMINATED BY '2'

MAP KEYS TERMINATED BY '3'

STORED AS SEQUENCEFILE;
```

Creating, Showing, Altering, and Dropping Tables

- Browsing Tables and Partitions

–SHOW TABLES;

–SHOW TABLES 'page.*';

–SHOW PARTITIONS page_view;

–DESCRIBE page_view;

- Altering Tables

–ALTER TABLE old_table_name RENAME TO new_table_name;

–ALTER TABLE tab1 ADD COLUMNS (c1 INT COMMENT 'a new int column', c2 STRING DEFAULT 'def val');

- Dropping Tables and Partitions

–DROP TABLE pv_users;

–ALTER TABLE pv_users DROP PARTITION (dt='2008-08-08')

Example 1

- hadoop jar /home/hadoop/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.0.jar grep /user/bible /user/bible_out2 '\w+'
- $hadoop dfs -rmr /user/jaiprakash/shakespeare_freq/_logs
- Hive>CREATE TABLE bible_table1 (freq INT, word STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE;
- hive> load data inpath “bible_out1" into table bible_table1 ;
- hive> select * from bible_table1 where freq > 20 sort by freq asc limit 10;

Example 1 (Cont...)

- hive> select freq, count(1) as f2 from bible_table group by freq sort by f2 desc limit 10;

- hive> explain select freq, count(1) as f2 from bible_table group by freq sort by f2 desc limit 10;

HiveQL with Movie Recommendation Dataset

- CREATE TABLE IF NOT EXISTS movie_rating(UserID STRING,MovieID STRING,rating STRING,Timestamp1 STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE;

- load data inpath "/user/u.data" into table movie_rating;

- select * from movie_rating where rating > 4 sort by rating asc limit 10;
- Select MovieID, avg(rating) from movie_rating group by MovieID;
- Select MovieID, Max(rating), Min(rating) from movie_rating group by MovieID sort by Max(rating) desc limit 5;

- Questions ?