

HandsOn – MapReduce

Quick Introduction To Java and MapReduce



www.marcel-mittelstaedt.com

WordCount – Mapper

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

    public static class TokenizerMapper
        extends Mapper<Object, Text, Text, IntWritable> {

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text value, Context context
                       ) throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }
}
```



WordCount – Reducer

```
public static class IntSumReducer
    extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                      Context context
                     ) throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}

public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```



Run Java MapReduce Job

1. Compile Java file:

```
hadoop/bin/hadoop com.sun.tools.javac.Main WordCount.java
```

2. Create Jar File:

```
jar cf WordCount.jar WordCount*.class
```

3. Execute MapReduce Job:

```
hadoop jar WordCount.jar WordCount /user/hadoop/Faust_1.txt  
/user/hadoop/wordcount_output
```



Run Java MapReduce Job

4. Take a look at the results:

```
hadoop fs -cat /user/hadoop/wordcount_output/part-r-00000 | head -10
"Allein" 1
"Alles" 1
"Als" 1
"Der" 1
"Die" 2
"Er" 2
"ICH" 4
"Im" 1
"Myein" 1
"Nur" 1
[...]
```





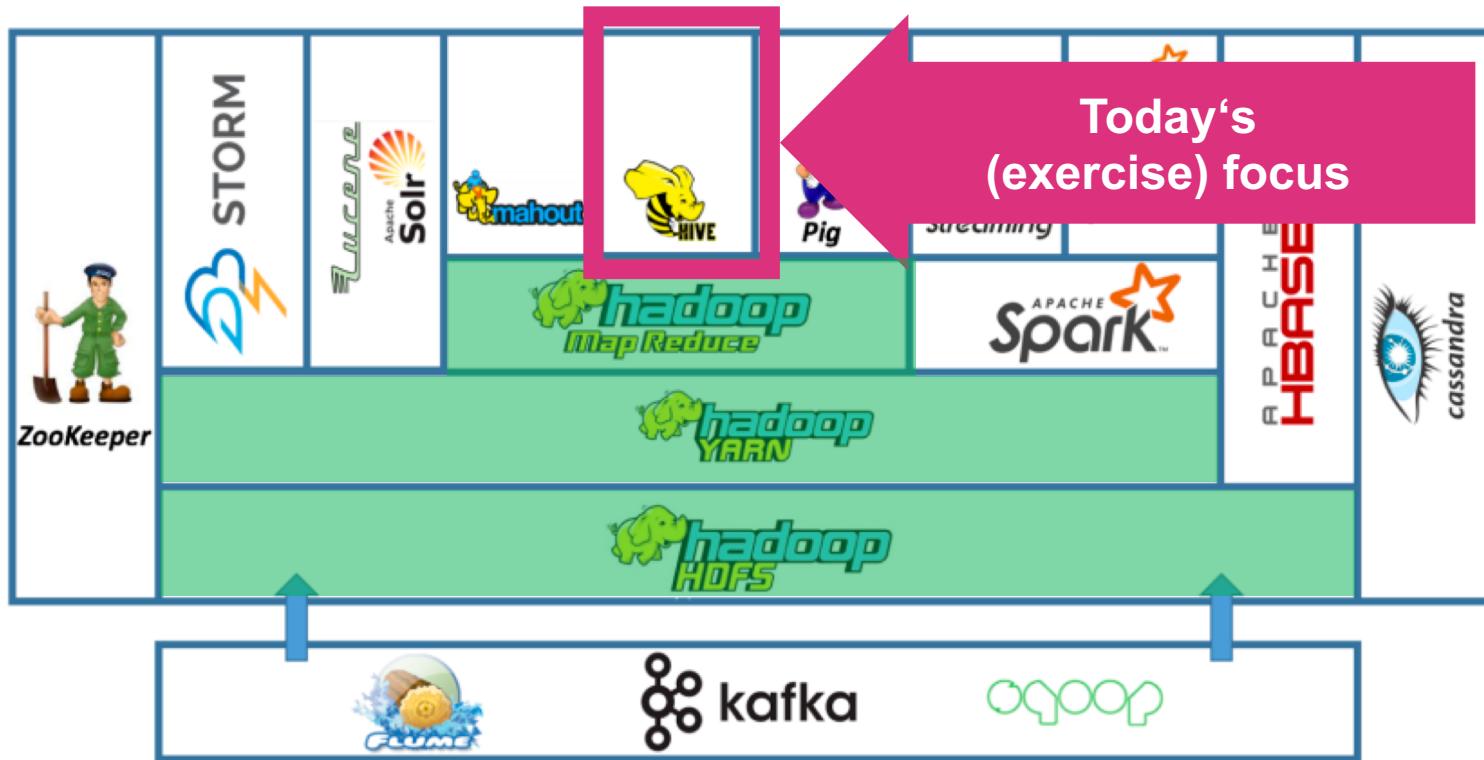
HandsOn – Hive and HiveQL

Quick Introduction To Hive and HiveQL



www.marcel-mittelstaedt.com

The Hadoop Ecosystem



Apache Hive



Apache Hive

- Initial Release in **October 2010**
- written in **Java**
- current Version: **3.1.0**

<http://hive.apache.org>

- **What is Hive:**

- **easy to use** (HiveQL)
- **based on Hadoop** (HDFS, YARN)
- **good scale-out capabilities** (e.g. by partitioning and underlying HDFS and YARN)
- **best used for:**
 - (BigData) **datawarehousing tasks**
 - a **DataLake**
 - **interface** for Analysts, DataScientists, Developers
 - **adhoc (batch) querying, aggregation and analysis** of large amounts of data (**PB!**) and hundreds of nodes

- **What Hive is NOT:**

- **transactional database**
- **highly responsive**



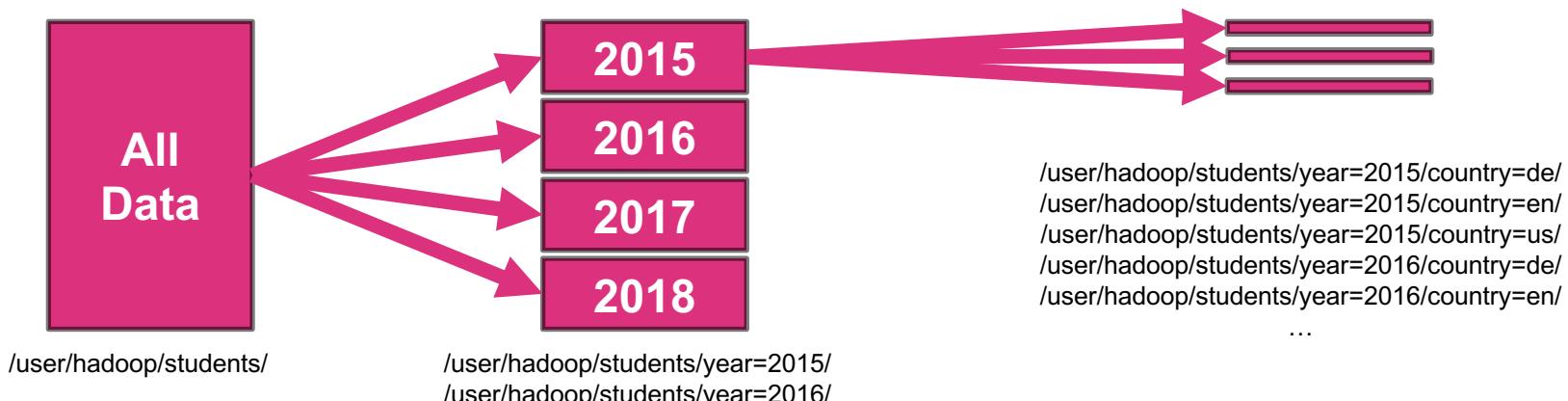
HiveQL

- SQL like **query language**
- Supported by: Hive CLI (*deprecated*), Beeline CLI and most JDBC clients
- Hive supported HDFS file formats:
 - **TextFile** (even compressed by gzip or bzip2)
 - **SequenceFile**
 - **RCFile**
 - **ORC**
 - **Parquet**
 - **Avro**



HDFS/Hive - Partitioning

- Partitioning of data distributes load and speeds up data processing
- A table can have one or more partition columns, defined by the time of creating a table (CREATE TABLE student(id Int, name STRING) PARTITIONED BY (year STRING) ... STORED AS TEXTFILE LOCATION '/user/hadoop/students';)
- partitioning can be done either **static** or **dynamic**
- each distinct value of a partition column is represented by a **HDFS directory**



HDFS/Hive - Wordcount

- Previous WordCount example achieved by using Hive and HiveQL:

```
CREATE TABLE faust (line STRING);

LOAD DATA INPATH '/user/hadoop/faust' OVERWRITE INTO TABLE faust;

CREATE TABLE word_counts AS
SELECT word, count(1) AS count FROM
(SELECT explode(split(line, '\\s')) AS word FROM faust) temp
GROUP BY word ORDER BY word;
```

```
SELECT * from word_counts ORDER BY count DESC LIMIT 10;
```



| word | count |
|-------|-------|
| und | 509 |
| die | 463 |
| der | 440 |
| ich | 435 |
| Und | 400 |
| nicht | 346 |
| zu | 319 |
| [...] | |

Exercises Preparation A

Install and Setup Hive



www.marcel-mittelstaedt.com

Install and Setup Hive

1. Get Hive:

```
wget http://ftp.halifax.rwth-aachen.de/apache/hive/hive-3.1.0/apache-hive-3.1  
.0-bin.tar.gz
```

2. Uncompress Hive:

```
tar -xzf apache-hive-3.1.0-bin.tar.gz
```

3. Move Binaries:

```
mv apache-hive-3.1.0-bin /home/hadoop/hive
```



Install and Setup Hive

4. Setup **UNIX** Environment Parameters:

```
vi .bashrc
```



```
export HIVE_HOME=/home/hadoop/hive  
export PATH=$PATH:/home/hadoop/hive/bin
```



```
source .bashrc
```

Install and Setup Hive

5. Verify Hive Installation:

```
hive --version
```



```
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in [jar:file:/home/hadoop/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation  
.   
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]  
Hive 3.1.0  
Git git://vgargwork.local/Users/vgarg/repos/hive.apache.master -r bcc7df9582483  
1a8d2f1524e4048dfc23ab98c19  
Compiled by vgarg on Mon Jul 23 16:02:03 PDT 2018  
From source with checksum 147d8070369f8c672407753089777fd
```



Install and Setup Hive

6. Create Hive directories within *HDFS*:

```
hadoop fs -mkdir -p /user/hive/warehouse  
hadoop fs -chmod g+w /user/hive/warehouse
```

```
hadoop fs -mkdir -p /tmp  
hadoop fs -chmod g+w /tmp
```

7. Create Hive Config SH:

```
cp hive/conf/hive-env.sh.template hive/conf/hive-env.sh  
vi hive/conf/hive-env.sh
```



```
export HADOOP_HOME=/home/hadoop/hadoop  
export HIVE_CONF_DIR=/home/hadoop/hive/conf
```

Install and Setup Hive

8. Create Hive Config (*hive-site.xml*):

```
vi hive/conf/hive-site.xml
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xmlelement type="text/xsl" href="configuration.xsl"?><!--
Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements. See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<configuration>
    <property>
        <name>javax.jdo.option.ConnectionURL</name>
        <value>jdbc:derby:;databaseName=/home/hadoop/hive/metastore_db;create=true</value>
        <description>
            JDBC connect string for a JDBC metastore. To use SSL to encrypt/authenticate the connection,
            provide database-specific SSL flag in the connection URL.
            For example, jdbc:postgresql://myhost/db?ssl=true for postgres database.
        </description>
    </property>
```



Install and Setup Hive

```
<property>
    <name>hive.metastore.warehouse.dir</name>
    <value>/user/hive/warehouse</value>
    <description>location of default database for the warehouse
    </description>
</property>
<property>
    <name>hive.metastore.uris</name>
    <value/>
    <description>Thrift URI for the remote metastore. Used by metastore cli
    ent to connect to remote metastore.
    </description>
</property>
<property>
    <name>javax.jdo.option.ConnectionDriverName</name>
    <value>org.apache.derby.jdbc.EmbeddedDriver</value>
    <description>Driver class name for a JDBC metastore</description>
</property>
<property>
    <name>javax.jdo.PersistenceManagerFactoryClass</name>
    <value>org.datanucleus.api.jdo.JDOPersistenceManagerFactory</value>
    <description>class implementing the jdo persistence</description>
</property>
</configuration>
```



Install and Setup Hive

9. Edit YARN MapRed Config (*mapred-site.xml*) for higher memory usage:

```
vi hadoop/etc/hadoop/mapred-site.xml
```

```
<property>
    <name>mapreduce.map.memory.mb</name>
    <value>3072</value>
</property>
<property>
    <name>mapreduce.reduce.memory.mb</name>
    <value>3072</value>
</property>
```

10. Restart DFS and YARN:

```
stop-all.sh
start-dfs.sh
Start-yarn.sh
```



Install and Setup Hive

11. By default Hive makes use of *Derby* as a metastore database, so we need to initialize *Derby*:

```
hive/bin/schematool -initSchema -dbType derby
```



```
[...]
Starting metastore schema initialization to 3.1.0
Initialization script hive-schema-3.1.0.derby.sql
[...]
Initialization script completed
schemaTool completed
```

Install and Setup Hive

12. Test if Hive Installation and Configuration is successful. Start Hive:

```
hive
```



```
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in [jar:file:/home/hadoop/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/  
impl/StaticLoggerBinder.class]  
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7  
.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.  
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]  
Hive Session ID = c120d0b1-9025-43db-96e4-48ccfb875f1a  
  
Logging initialized using configuration in jar:file:/home/hadoop/hive/lib/hive-common-3.1.0.jar  
!/hive-log4j2.properties Async: true  
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider us  
ing a different execution engine (i.e. spark, tez) or using Hive 1.X releases.  
Hive Session ID = fdd6f06a-d4e4-48e6-8971-997e8a0a8e2c  
hive>
```



Install and Setup Hive

13. Execute First SQL Query:

```
hive> show databases;  
OK  
default  
Time taken: 0.083 seconds, Fetched: 1 row(s)  
hive>
```





Hive: Create and Work with External Tables

Using public dataset of IMDb.com



www.marcel-mittelstaedt.com

Get IMDb Data And Move It To HDFS

1. Get **IMDb Data** (<https://www.imdb.com/interfaces/>):

```
wget https://datasets.imdbws.com/title.basics.tsv.gz  
wget https://datasets.imdbws.com/title.ratings.tsv.gz
```

2. Uncompress IMDb Data:

```
gunzip title.basics.tsv.gz  
gunzip title.ratings.tsv.gz
```

3. Create HDFS Directories for IMDb Data:

```
hadoop fs -mkdir /user/hadoop/imdb  
hadoop fs -mkdir /user/hadoop/imdb/name  
hadoop fs -mkdir /user/hadoop/imdb/ratings
```



Create External Tables In Hive

4. Transfer IMDb data files to HDFS:

```
hadoop fs -put title.ratings.tsv /user/hadoop/imdb/ratings/ratings.tsv  
hadoop fs -put name.basics.tsv /user/hadoop/imdb/name/title.tsv
```

5. Create External Table **imdb_ratings** (file *title.ratings.tsv*) in Hive:

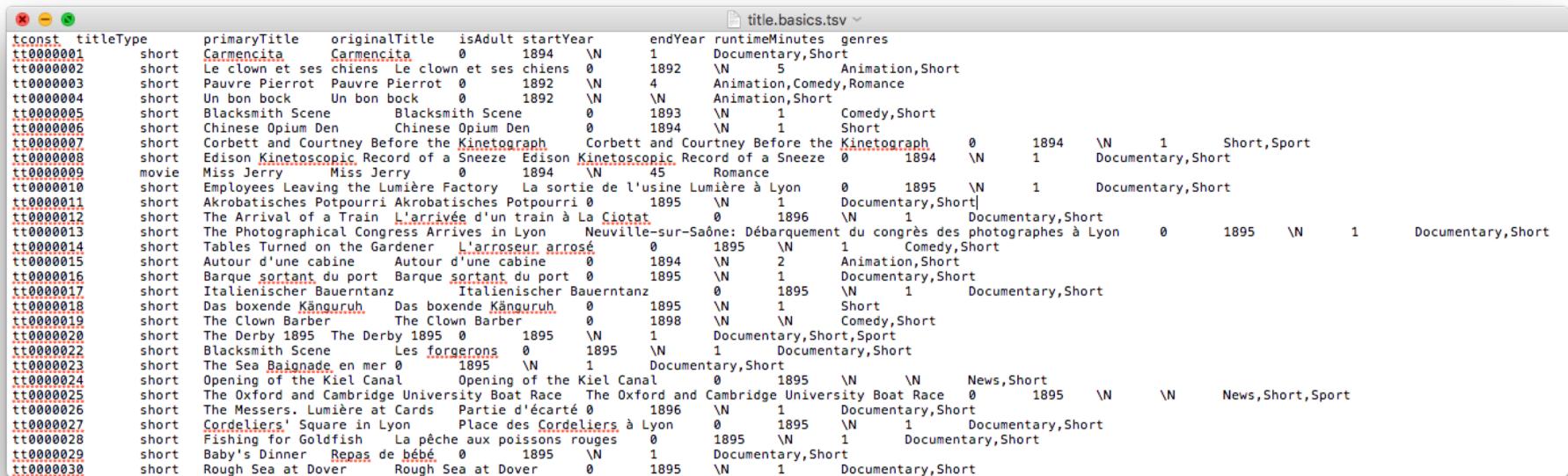
```
hive > CREATE EXTERNAL TABLE IF NOT EXISTS imdb_ratings(  
        tconst STRING,  
        average_rating DECIMAL(2,1),  
        num_votes BIGINT  
) COMMENT 'IMDb Ratings'  
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS  
TEXTFILE LOCATION '/user/hadoop/imdb/ratings';
```

| tconst | averageRating | numVotes |
|-----------|---------------|----------|
| tt0000001 | 5.8 | 1416 |
| tt0000002 | 6.4 | 167 |
| tt0000003 | 6.6 | 1013 |
| tt0000004 | 6.4 | 100 |
| tt0000005 | 6.2 | 1712 |
| tt0000006 | 5.6 | 87 |
| tt0000007 | 5.5 | 571 |
| tt0000008 | 5.6 | 1520 |
| tt0000009 | 5.6 | 68 |
| tt0000010 | 6.9 | 5075 |
| tt0000011 | 5.4 | 208 |
| tt0000012 | 7.4 | 8479 |
| tt0000013 | 5.7 | 1297 |
| tt0000014 | 7.2 | 3683 |
| tt0000015 | 6.2 | 642 |



Create External Tables In Hive

6. Create External Table `imdb_movies` for file `title.basics.tsv` in Hive:



| tconst | titleType | primaryTitle | originalTitle | isAdult | startYear | endYear | runtimeMinutes | genres |
|-----------|-----------|---|---|---------|-----------|---------|-------------------|--------------------------|
| tt0000001 | short | Carmencita | Carmencita | 0 | 1894 | \N | 1 | Documentary,Short |
| tt0000002 | short | Le clown et ses chiens | Le clown et ses chiens | 0 | 1892 | \N | 5 | Animation,Short |
| tt0000003 | short | Pauvre Pierrot | Pauvre Pierrot | 0 | 1892 | \N | 4 | Animation,Comedy,Romance |
| tt0000004 | short | Un bon bock | Un bon bock | 0 | 1892 | \N | 1 | Animation,Short |
| tt0000005 | short | Blacksmith Scene | Blacksmith Scene | 0 | 1893 | \N | 1 | Comedy,Short |
| tt0000006 | short | Chinese Opium Den | Chinese Opium Den | 0 | 1894 | \N | 1 | Short |
| tt0000007 | short | Corbett and Courtney Before the Kinetograph | Corbett and Courtney Before the Kinetograph | 0 | 1894 | \N | 1 | Short,Sport |
| tt0000008 | short | Edison Kinetoscopic Record of a Sneeze | Edison Kinetoscopic Record of a Sneeze | 0 | 1894 | \N | 1 | Documentary,Short |
| tt0000009 | movie | Miss Jerry | Miss Jerry | 0 | 1894 | \N | 45 | Romance |
| tt0000010 | short | Employees Leaving the Lumière Factory | La sortie de l'usine Lumière à Lyon | 0 | 1895 | \N | 1 | Documentary,Short |
| tt0000011 | short | Akrobatisches Potpourri | Akrobatisches Potpourri | 0 | 1895 | \N | 1 | Documentary,Short |
| tt0000012 | short | The Arrival of a Train | L'arrivée d'un train à La Ciotat | 0 | 1896 | \N | 1 | Documentary,Short |
| tt0000013 | short | The Photographical Congress Arrives in Lyon | Neuville-sur-Saône: Débarquement du congrès des photographes à Lyon | 0 | 1895 | \N | 1 | Documentary,Short |
| tt0000014 | short | Tables Turned on the Gardener | L'arroseur arrosé | 0 | 1895 | \N | 1 | Comedy,Short |
| tt0000015 | short | Autour d'une cabine | Autour d'une cabine | 0 | 1894 | \N | 2 | Animation,Short |
| tt0000016 | short | Barque sortant du port | Barque sortant du port | 0 | 1895 | \N | 1 | Documentary,Short |
| tt0000017 | short | Italienischer Bauerntanz | Italienischer Bauerntanz | 0 | 1895 | \N | 1 | Documentary,Short |
| tt0000018 | short | Das boxende Känguru | Das boxende Känguru | 0 | 1895 | \N | 1 | Short |
| tt0000019 | short | The Clown Barber | The Clown Barber | 0 | 1898 | \N | 1 | Comedy,Short |
| tt0000020 | short | The Derby 1895 | The Derby 1895 | 0 | 1895 | \N | 1 | Documentary,Short,Sport |
| tt0000022 | short | Blacksmith Scene | Les forgerons | 0 | 1895 | \N | 1 | Documentary,Short |
| tt0000023 | short | The Sea Baignade en mer | 0 | 1895 | \N | 1 | Documentary,Short | |
| tt0000024 | short | Opening of the Kiel Canal | Opening of the Kiel Canal | 0 | 1895 | \N | 1 | News,Short |
| tt0000025 | short | The Oxford and Cambridge University Boat Race | The Oxford and Cambridge University Boat Race | 0 | 1895 | \N | 1 | News,Short,Sport |
| tt0000026 | short | The Messers. Lumière at Cards | Partie d'écarté | 0 | 1896 | \N | 1 | Documentary,Short |
| tt0000027 | short | Cordeliers' Square in Lyon | Place des Cordeliers à Lyon | 0 | 1895 | \N | 1 | Documentary,Short |
| tt0000028 | short | Fishing for Goldfish | La pêche aux poissons rouges | 0 | 1895 | \N | 1 | Documentary,Short |
| tt0000029 | short | Baby's Dinner | Repas de bébé | 0 | 1895 | \N | 1 | Documentary,Short |
| tt0000030 | short | Rough Sea at Dover | Rough Sea at Dover | 0 | 1895 | \N | 1 | Documentary,Short |



Create External Tables In Hive

6. Create External Table `imdb_movies` for file `title.basics.tsv` in Hive:

```
hive > CREATE EXTERNAL TABLE IF NOT EXISTS imdb_movies(
        tconst STRING,
        title_type STRING,
        primary_title STRING,
        original_title STRING,
        is_adult DECIMAL(1,0),
        start_year DECIMAL(4,0),
        end_year STRING,
        runtime_minutes INT,
        genres STRING
    ) COMMENT 'IMDb Movies' ROW FORMAT DELIMITED FIELDS TERMINATED
    BY '\t' STORED AS TEXTFILE LOCATION '/user/hadoop/imdb/name';
```





Exercises I

Hive: Create and Work with External Tables



HDFS and HiveQL Exercises - IMDB

1. Execute Tasks of previous HandsOn Slides
2. Download <https://datasets.imdbws.com/name.basics.tsv.gz>
3. Create HDFS Directory **/user/hadoop/imdb/actors/names.tsv** for file name.basics.tsv
4. Create External Hive Table **imdb_actors** for name.tsv
5. Use HiveQL to answer following questions:
 - a) How many movies are within the IMDB dataset?
 - b) Who is the oldest actor/writer/... within the dataset?
 - c) Create a list (`m.tconst, m.original_title, m.start_year, r.average_rating, r.num_votes`) of movies which are:
 - equal or newer than year 2000
 - have an average rating better than 8
 - have been voted more than 100.000 times
 - d) How many movies are in list of c)?



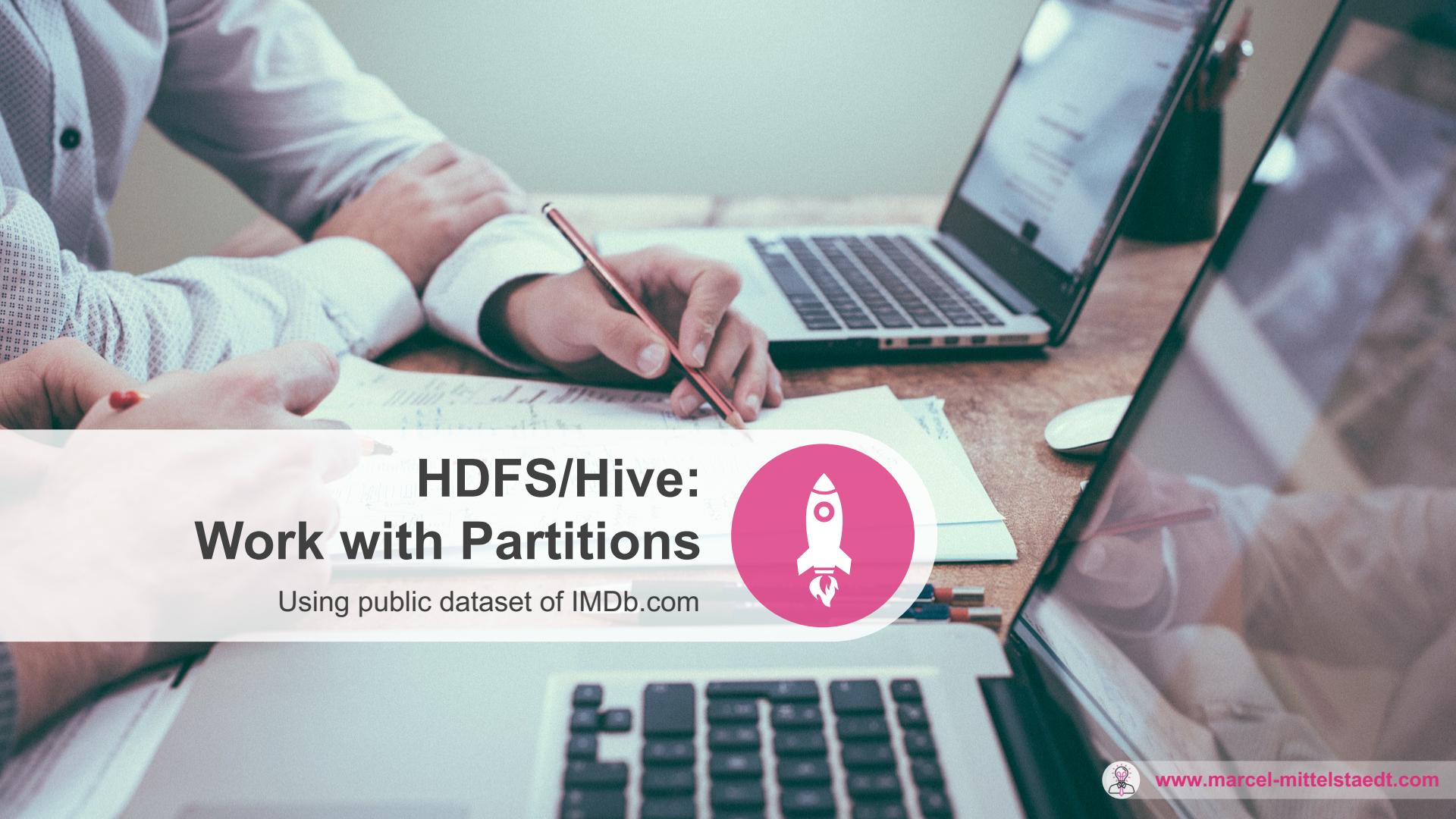
HDFS and HiveQL Exercises - IMDB

5. Use HiveQL to answer following questions:

e) We want to know which years have been great for cinema.

Create a list with one row per year and a related count of movies which:

- have an average rating better than 8
 - have been voted more than 100.000 times
- ordered descending by count of movies.



HDFS/Hive: Work with Partitions

Using public dataset of IMDb.com



www.marcel-mittelstaedt.com

Static Partitioning – Create Partitioned Table

1. Create partitioned version of table `imdb_ratings`: **`imdb_ratings_partitioned`**:

```
hive > CREATE TABLE IF NOT EXISTS imdb_ratings_partitioned(  
        tconst STRING,  
        average_rating DECIMAL(2,1),  
        num_votes BIGINT  
    ) PARTITIONED BY (partition_quality STRING)  
    STORED AS ORCFILE LOCATION '/user/hadoop/imdb/ratings_partitioned';
```



Static Partitioning – INSERT Into Table via Hive

1. Migrate and partition data of table `imdb_ratings` to table `imdb_ratings_partitioned`:

```
INSERT OVERWRITE TABLE imdb_ratings_partitioned partition(partition_quality='good')
SELECT r.tconst, r.average_rating, r.num_votes FROM imdb_ratings r WHERE r.average_rating >= 7;

INSERT OVERWRITE TABLE imdb_ratings_partitioned partition(partition_quality='worse')
SELECT r.tconst, r.average_rating, r.num_votes FROM imdb_ratings r WHERE r.average_rating < 7;
```

2. Check Success on HDFS:

| /user/hadoop/imdb/ratings_partitioned | | | | | | | | | Go! | | | |
|---------------------------------------|------------|--------|------------|------|---------------|-------------|------------|-------------------------|---------|--|--|--|
| Show 25 entries | | | | | | | | | Search: | | | |
| | Permission | Owner | Group | Size | Last Modified | Replication | Block Size | Name | | | | |
| <input type="checkbox"/> | drwxr-xr-x | hadoop | supergroup | 0 B | Oct 07 15:59 | 0 | 0 B | partition_quality=good | | | | |
| <input type="checkbox"/> | drwxr-xr-x | hadoop | supergroup | 0 B | Oct 07 16:01 | 0 | 0 B | partition_quality=worse | | | | |

Static Partitioning – INSERT Into Table via Hive

3. Check Success via Hive:

```
select distinct average_rating from imdb_ratings_partitioned where partition_quality = 'good';

7.0
7.1
7.2
7.3
7.4
7.5
7.6
7.7
[...]
9.0
9.1
9.2
9.3
9.4
9.5
9.6
9.7
9.8
9.9
```



Dynamic Partitioning – Create Partitioned Table

1. Create partitioned version of table `imdb_movies`: `imdb_movies_partitioned`:

```
hive > CREATE TABLE IF NOT EXISTS imdb_movies_partitioned(  
    tconst STRING,  
    title_type STRING,  
    primary_title STRING,  
    original_title STRING,  
    is_adult DECIMAL(1,0),  
    start_year DECIMAL(4,0),  
    end_year STRING,  
    runtime_minutes INT,  
    genres STRING  
) PARTITIONED BY (partition_year int)  
STORED AS ORCFILE  
LOCATION '/user/hadoop/imdb/name_partitioned';
```



Dynamic Partitioning – **INSERT** Into Table via Hive

1. Migrate and partition data of table `imdb_movies` to table `imdb_movies_partitioned`:

```
set hive.exec.dynamic.partition.mode=nonstrict; -- enable dynamic partitioning

INSERT OVERWRITE TABLE imdb_movies_partitioned partition(partition_year)
SELECT m.tconst, m.title_type, m.primary_title, m.original_title, m.is_adult,
m.start_year, m.end_year, m.runtime_minutes, m.genres,
m.start_year -- last column = partition column
FROM imdb_movies m
```

2. Check Success via Hive:

Result

```
select count(*) from imdb_movies m where m.start_year = 2018
```

| | 123_c0 |
|---|---------|
| 1 | 206.676 |

Result

```
select count(*) from imdb_movies_partitioned mp where mp.partition_year = 2018
```

| | 123_c0 |
|---|---------|
| 1 | 206.676 |



Dynamic Partitioning – INSERT Into Table via Hive

3. Check Success on HDFS:

```
hadoop fs -ls /user/hadoop/imdb/name_partitioned
[...]
drwxr-xr-x  - hadoop supergroup          0 2018-10-05 19:23 /user/hadoop/imdb/name_partitioned/partition_year=2011
drwxr-xr-x  - hadoop supergroup          0 2018-10-05 19:23 /user/hadoop/imdb/name_partitioned/partition_year=2012
drwxr-xr-x  - hadoop supergroup          0 2018-10-05 19:23 /user/hadoop/imdb/name_partitioned/partition_year=2013
drwxr-xr-x  - hadoop supergroup          0 2018-10-05 19:23 /user/hadoop/imdb/name_partitioned/partition_year=2014
drwxr-xr-x  - hadoop supergroup          0 2018-10-05 19:23 /user/hadoop/imdb/name_partitioned/partition_year=2015
drwxr-xr-x  - hadoop supergroup          0 2018-10-05 19:23 /user/hadoop/imdb/name_partitioned/partition_year=2016
drwxr-xr-x  - hadoop supergroup          0 2018-10-05 19:23 /user/hadoop/imdb/name_partitioned/partition_year=2017
drwxr-xr-x  - hadoop supergroup          0 2018-10-05 19:23 /user/hadoop/imdb/name_partitioned/partition_year=2018
drwxr-xr-x  - hadoop supergroup          0 2018-10-05 19:23 /user/hadoop/imdb/name_partitioned/partition_year=2019
drwxr-xr-x  - hadoop supergroup          0 2018-10-05 19:23 /user/hadoop/imdb/name_partitioned/partition_year=2020
drwxr-xr-x  - hadoop supergroup          0 2018-10-05 19:24 /user/hadoop/imdb/name_partitioned/partition_year=2021
drwxr-xr-x  - hadoop supergroup          0 2018-10-05 19:22 /user/hadoop/imdb/name_partitioned/partition_year=2022
drwxr-xr-x  - hadoop supergroup          0 2018-10-05 19:22 /user/hadoop/imdb/name_partitioned/partition_year=2023
drwxr-xr-x  - hadoop supergroup          0 2018-10-05 19:24 /user/hadoop/imdb/name_partitioned/partition_year=2024
drwxr-xr-x  - hadoop supergroup          0 2018-10-05 19:22 /user/hadoop/imdb/name_partitioned/partition_year=2025
drwxr-xr-x  - hadoop supergroup          0 2018-10-05 19:22 /user/hadoop/imdb/name_partitioned/partition_year=2115

hadoop fs -ls /user/hadoop/imdb/name_partitioned/partition_year=2018
Found 1 items
-rw-r--r--  1 hadoop supergroup  5371795 2018-10-05 19:23 /user/hadoop/imdb/name_partitioned/partition_year=2018/000
007_0
```



Dynamic Partitioning – INSERT Into Table via Hive

4. Check Success via HDFS Web Browser:

The screenshot shows a web browser window for the HDFS Web Browser at the URL `localhost:9870/explorer.html#/user/hadoop/imdb/name_partitioned/`. The page title is "Browse Directory". The address bar also contains the path `/user/hadoop/imdb/name_partitioned/`. There are buttons for "Go!", "Edit", "Upload", and "Delete". A search bar is present with the placeholder "Search: []". Below these controls, there is a table with the following columns: "Show 25 entries", "Permission", "Owner", "Group", "Size", "Last Modified", "Replication", "Block Size", and "Name". The table lists seven partitions named "partition_year=2001" through "partition_year=2007", each with permissions "drwxr-xr-x", owner "hadoop", group "supergroup", size "0 B", last modified "Oct 05 19:23", replication "0", block size "0 B", and name "partition_year=2001" through "partition_year=2007". Each row has a small trash can icon in the last column.

| Show 25 entries | Permission | Owner | Group | Size | Last Modified | Replication | Block Size | Name |
|-----------------|------------|--------|------------|------|---------------|-------------|------------|---------------------|
| | drwxr-xr-x | hadoop | supergroup | 0 B | Oct 05 19:23 | 0 | 0 B | partition_year=2001 |
| | drwxr-xr-x | hadoop | supergroup | 0 B | Oct 05 19:23 | 0 | 0 B | partition_year=2002 |
| | drwxr-xr-x | hadoop | supergroup | 0 B | Oct 05 19:23 | 0 | 0 B | partition_year=2003 |
| | drwxr-xr-x | hadoop | supergroup | 0 B | Oct 05 19:23 | 0 | 0 B | partition_year=2004 |
| | drwxr-xr-x | hadoop | supergroup | 0 B | Oct 05 19:23 | 0 | 0 B | partition_year=2005 |
| | drwxr-xr-x | hadoop | supergroup | 0 B | Oct 05 19:23 | 0 | 0 B | partition_year=2006 |
| | drwxr-xr-x | hadoop | supergroup | 0 B | Oct 05 19:23 | 0 | 0 B | partition_year=2007 |



Exercises II

HDFS/Hive: Work with Partitions



HDFS/Hive Partitioning Exercises - IMDB

1. Execute Tasks of previous HandsOn Slides

2. Create a (*statically*) partitioned table `imdb_actors_partitioned`, which:

- contains all columns of table `imdb_actors`
- is statically partitioned by `partition_is_alive`, containing:
 - „alive“ in case actor is still alive
 - „dead“ in case actor is already dead

Load all data from `imdb_actors` to table `imdb_actors_partitioned`

3. Create a (*dynamically*) partitioned table `imdb_movies_and_ratings_partitioned`, which:

- contains all columns of the two tables `imdb_movies` and `imdb_ratings` and
- is partitioned by start year of movie (create and add column `partition_year`).

Load all data of `imdb_movies` and `imdb_ratings` to table:

`imdb_movies_and_ratings_partitioned`



Exercises Preparation B

Setup HiveServer2 For Remote Connections



www.marcel-mittelstaedt.com

Setup HiveServer2

1. Edit Hive Config and add (*hive/conf/hive-site.xml*):

```
<property>
    <name>hive.server2.thrift.min.worker.threads</name>
    <value>3</value>
</property>
<property>
    <name>hive.server2.thrift.max.worker.threads</name>
    <value>5</value>
</property>
<property>
    <name>hive.server2.thrift.port</name>
    <value>10000</value>
</property>
<property>
    <name>hive.server2.thrift.bind.host</name>
    <value>localhost</value>
</property>
```



Setup HiveServer2

2. Update Hadoop Config and add (*hadoop/etc/hadoop/core-site.xml*):

```
<property>
    <name>hadoop.proxyuser.hadoop.hosts</name>
    <value>*</value>
</property>
<property>
    <name>hadoop.proxyuser.hadoop.groups</name>
    <value>*</value>
</property>
```

3. Restart DFS and YARN:

```
stop-all.sh
start-dfs.sh
Start-yarn.sh
```



Setup HiveServer2

4. Start HiveServer2:

```
hive/bin/hiveserver2

2018-10-02 16:19:08: Starting HiveServer2
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoop/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = b8d1efb3-fc8c-4ec8-bdf0-6a9a41e2ddaa
Hive Session ID = 32503981-a5fd-497e-b887-faf3ec1e686e
Hive Session ID = 00f7eab4-5a29-4ce4-ad97-e90904d9206f
Hive Session ID = 100e54c5-14c6-4acc-b398-040152b08ebf
[...]
```



Connect To HiveServer2 via JDBC

1. Download JDBC SQL Client, e.g. *DBeaver*:

Mac OSX: `wget https://dbeaver.io/files/dbeaver-ce-latest-installer.pkg`

Linux (Debian): `wget https://dbeaver.io/files/dbeaver-ce_latest_amd64.deb`

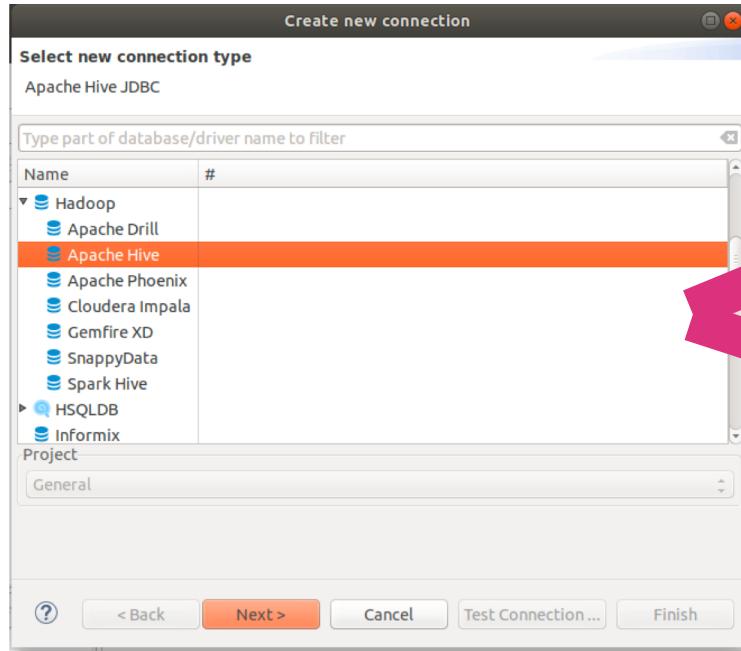
Linux (RPM): `wget https://dbeaver.io/files/dbeaver-ce-latest-stable.x86_64.rpm`

Windows: `wget https://dbeaver.io/files/dbeaver-ce-latest-x86_64-setup.exe`



Connect To HiveServer2 via JDBC

2. Configure Connection To Hive Server:



From inside VM:

JDBC URL: `jdbc:hive2://localhost:10000/default`

Host: localhost Port: 10000

Database/Schema: default

User name: hadoop

Password: [REDACTED] Passwort lokal speichern

From outside VM:

JDBC URL: `jdbc:hive2://marcel-Virtualbox:10000/default`

Host: marcel-Virtualbox Port: 10000

Database/Schema: default

User name: hadoop

Password: [REDACTED] Passwort lokal speichern

Connect To HiveServer2 via JDBC

3. Query something, e.g.:

The screenshot shows the DBeaver 5.2.1 interface. On the left, the Database browser displays a connection to 'Hadoop - default' with a schema named 'default'. Under 'Tables', there are entries for 'employee', 'imdb_ratings', 'names_text', 'Views', 'Indexes', 'Procedures', and 'Data Types'. In the center, a script tab contains the following SQL query:

```
select * from imdb_ratings r where r.average_rating > 5 ORDER BY r.average_rating desc limit 10
```

Below the script tab, the 'Result' tab displays the query results in a table:

| | r.tconst | r.average_rating | r.num_votes |
|----|-----------|------------------|-------------|
| 1 | tt0352593 | 9,9 | 7 |
| 2 | tt0339013 | 9,9 | 17 |
| 3 | tt9015750 | 9,9 | 17 |
| 4 | tt0398054 | 9,9 | 16 |
| 5 | tt0398053 | 9,9 | 12 |
| 6 | tt0060454 | 9,9 | 12 |
| 7 | tt0307114 | 9,9 | 9 |
| 8 | tt0487487 | 9,9 | 17 |
| 9 | tt8982612 | 9,9 | 15 |
| 10 | tt0057955 | 9,9 | 17 |





Exercises III

HiveServer2 For Remote Connections



HiveServer2 Exercises

1. Execute Tasks of previous HandsOn Slides.
2. Run any query.

