HandsOn – **MapReduce**

Quick Introduction To Java and MapReduce

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 MapRedcue 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
"Mein     1
"Nur      1
[…]
```


HandsOn – **HiveServer2**

Quick Introduction To HiveServer2

Exercises III Preparation

Setup HiveServer2 For Remote Connections

Start Gcloud VM and Connect

1. Start Gcloud Instance:

```
gcloud compute instances start big-data
```

2. Connect to Gcloud instance via SSH (on Windows using Putty):

```
ssh hans.wurst@XXX.XXX.XXX.XXX
```


Pull and Start Docker Container

1. Pull Docker Image:

```
docker pull marcelmittelstaedt/hiveserver_base:latest
```

2. Start Docker Image:

```
docker run -dit --name hiveserver_base_container \
    -p 8088:8088 -p 9870:9870 -p 9864:9864 \
    -p 10000:10000 -p 9000:9000 \
    marcelmittelstaedt/hiveserver_base:latest
```

3. Wait till first Container Initialization finished:

```
docker logs hiveserver_base_container

[…]
Stopping nodemanagers
Stopping resourcemanager
Container Startup finished.
```


Start Hadoop Cluster

1. Get into Docker container:

```
docker exec -it hiveserver_base_container bash
```

2. Switch to hadoop user:

```
sudo su hadoop
```

```
cd
```

3. Start Hadoop Cluster:

```
start-all.sh
```


Start HiveServer2

1. 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/slf4
j/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
```

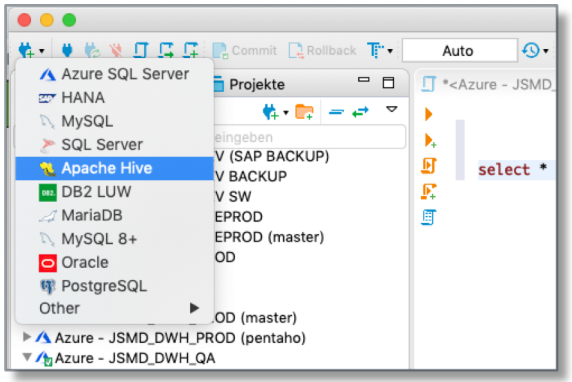
Linux (Debian):
```
wget https://dbeaver.io/files/dbeaver-ce_latest_amd64.deb
```

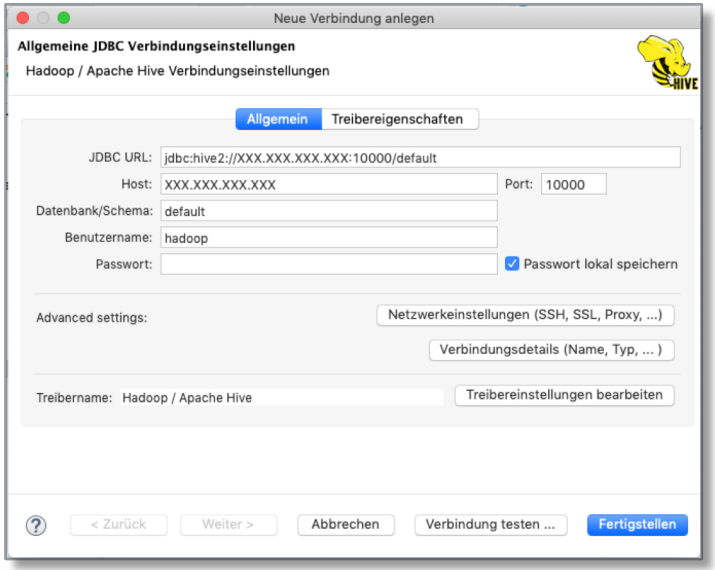
Linux (RPM):
```
wget https://dbeaver.io/files/dbeaver-ce-latest-stable.x86_64.rpm
```

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


Connect To HiveServer2 via JDBC

2. Configure Connection To Hive Server:





Get some data…

1. Get some IMDb data:

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

2. Put them into HDFS:

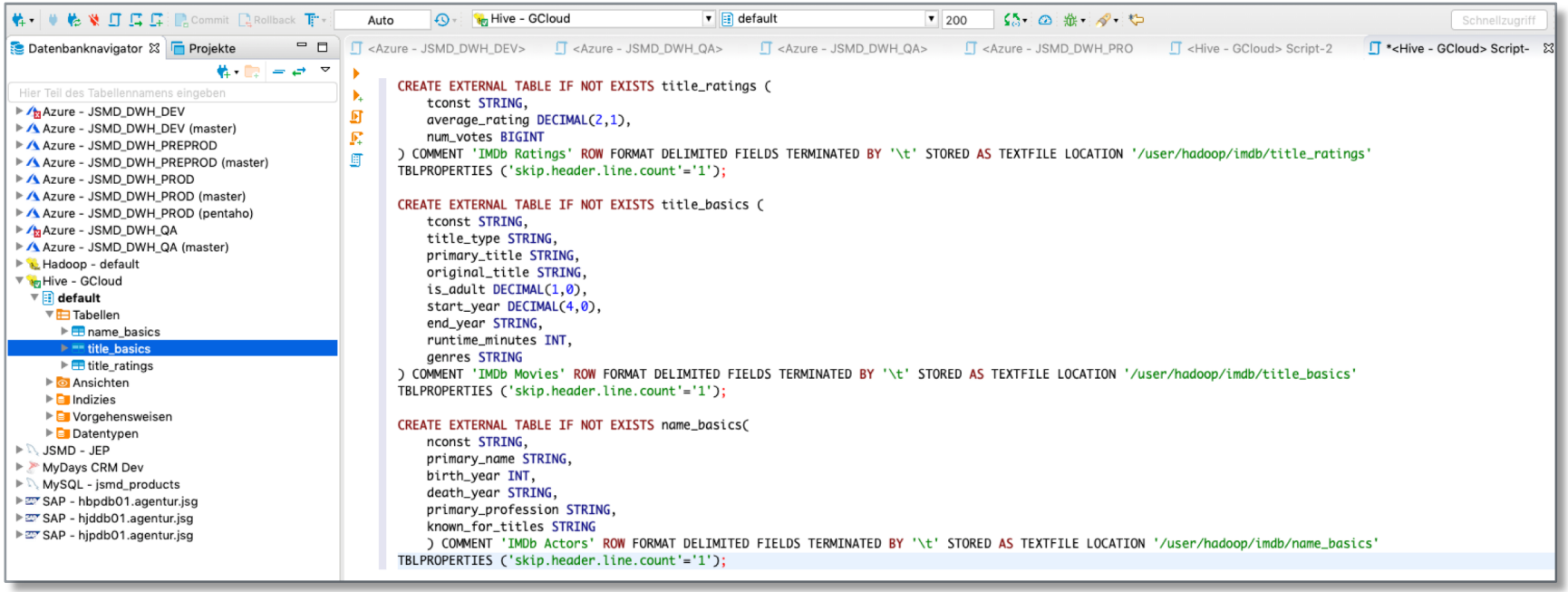
```
hadoop fs -mkdir /user/hadoop/imdb
```

```
hadoop fs -mkdir /user/hadoop/imdb/title_basics && hadoop fs -mkdir /user/hadoop/imdb/title_r
atings && hadoop fs -mkdir /user/hadoop/imdb/name_basics
```

```
hadoop fs -put title.basics.tsv /user/hadoop/imdb/title_basics/title.basics.tsv && hadoop fs
-put title.ratings.tsv /user/hadoop/imdb/title_ratings/title.ratings.tsv && hadoop fs -put na
me.basics.tsv /user/hadoop/imdb/name_basics/name.basics.tsv
```


Create some external tables…

1. Create some tables on top of files:



Query some data…

1. Query some data:

```
SELECT
    m.tconst,
    m.original_title,
    m.start_year,
    r.average_rating,
    r.num_votes
FROM
    title_basics m
    JOIN title_ratings r on (m.tconst = r.tconst)
WHERE
    r.average_rating >= 8.1 and m.start_year >= 2010
    and m.title_type = 'movie' and r.num_votes > 100000
ORDER BY r.average_rating desc, r.num_votes DESC;
```

Result

SELECT m.tconst, m.original_title, m.start_year, r.a | *Geben Sie einen SQL-Ausdruck ein, um die Ergebnis*

Gitter | Text

	tconst	original_title	start_year	average_rating	num_votes
1	tt5813916	Dag II	2.016	9,0	101.581
2	tt7286456	Joker	2.019	8,9	331.283
3	tt1375666	Inception	2.010	8,8	1.883.414
4	tt0816692	Interstellar	2.014	8,6	1.339.327
5	tt4154756	Avengers: Infinity War	2.018	8,5	717.106
6	tt1675434	Intouchables	2.011	8,5	694.378
7	tt2582802	Whiplash	2.014	8,5	637.001
8	tt4154796	Avengers: Endgame	2.019	8,5	582.173
9	tt1345836	The Dark Knight Rises	2.012	8,4	1.414.773
10	tt1853728	Django Unchained	2.012	8,4	1.243.114

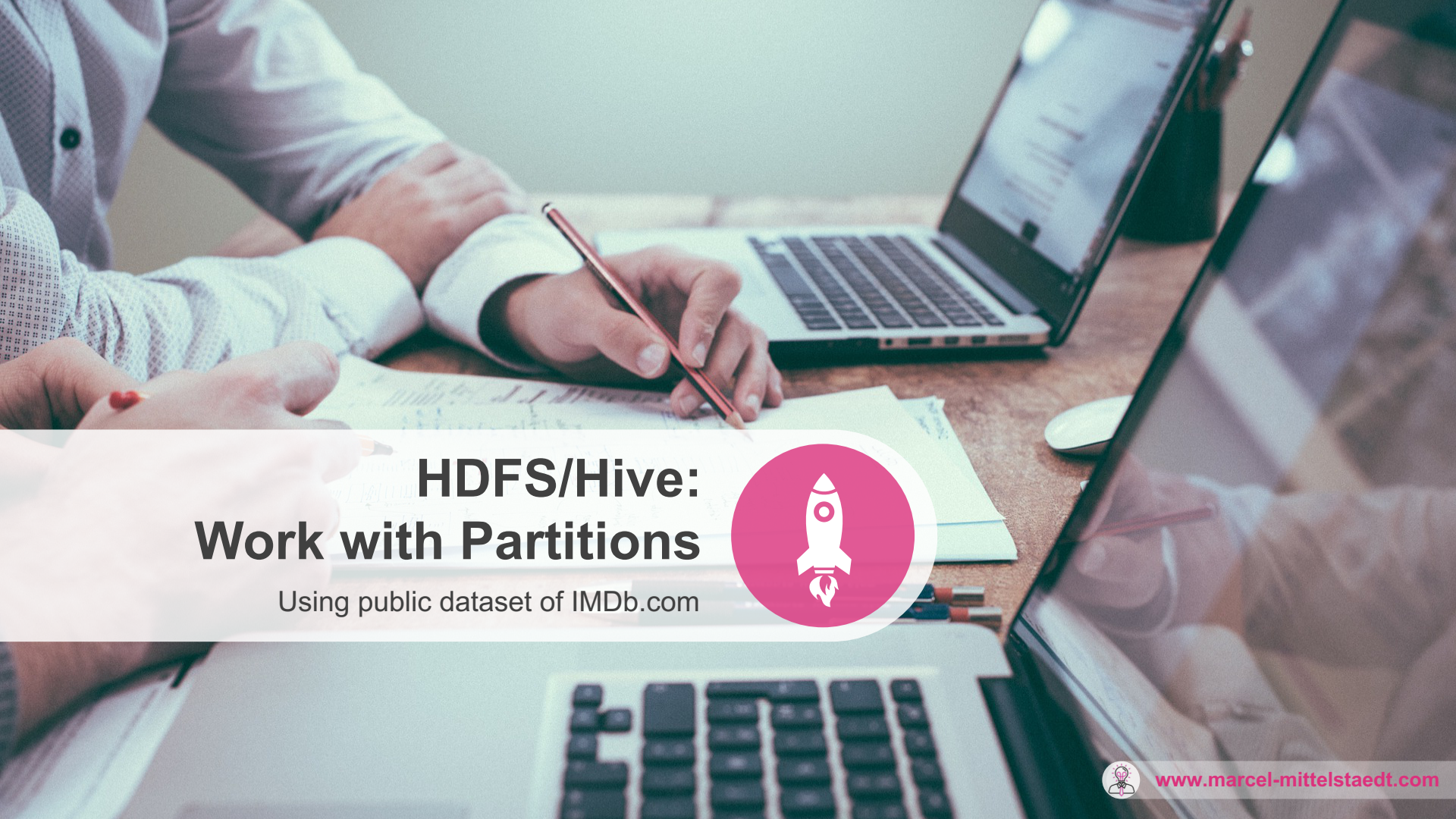
Exercises III

HiveServer2 For Remote Connections

HiveServer2 Exercises

1. Execute Tasks of previous HandsOn Slides.

HandsOn – HDFS/Hive Partitioning and HiveServer2

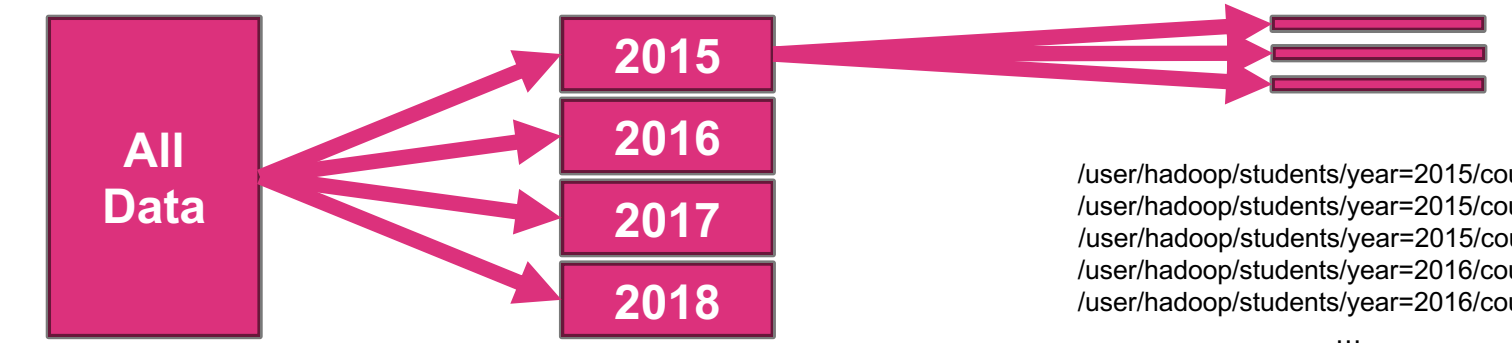


HDFS/Hive: Work with Partitions

Using public dataset of IMDb.com

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**



Static Partitioning – **Create Partitioned Table**

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

```
CREATE TABLE IF NOT EXISTS title_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';
```

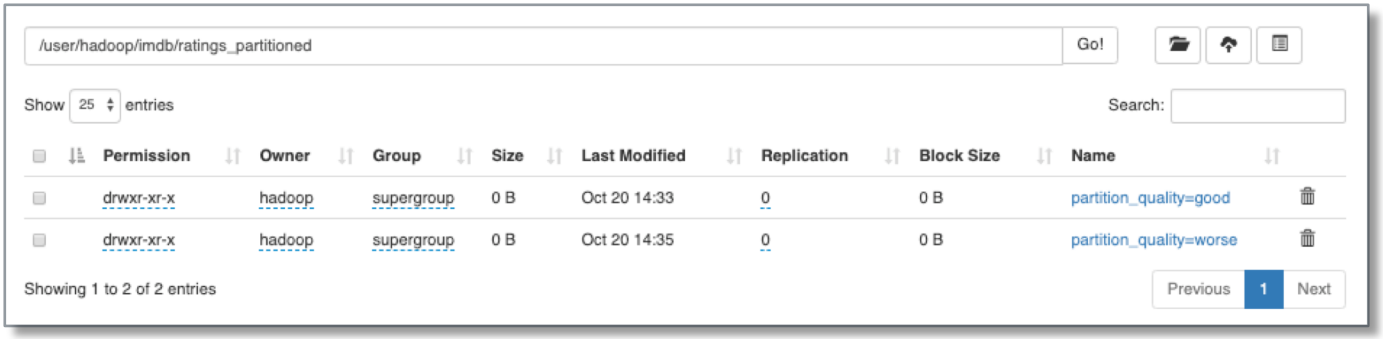

Static Partitioning – INSERT Into Table via Hive

1. Migrate and partition data of table `title_ratings` to table **`title_ratings_partitioned`**:

```
INSERT OVERWRITE TABLE title_ratings_partitioned PARTITION(partition_quality='good')
SELECT r.tconst, r.average_rating, r.num_votes FROM title_ratings r WHERE r.average_rating >= 7;

INSERT OVERWRITE TABLE title_ratings_partitioned PARTITION(partition_quality='worse')
SELECT r.tconst, r.average_rating, r.num_votes FROM title_ratings r WHERE r.average_rating < 7;
```

2. Check Success on HDFS:



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

3. Check Success via Hive:

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

Result

select distinct average_rating from title_ratings_par | *Geben Sie einen SQL-Ausdruck ein, um die Ergebnisse zu filtern (verwend*

Gitter | Text

	123 average_rating
1	7,0
2	7,1
3	7,2
4	7,3
5	7,4
6	7,5
7	7,6
8	7,7
9	7,8

Dynamic Partitioning – **Create Partitioned Table**

1. Create partitioned version of table `title_basics`: **`title_basics_partitioned`**:

```
CREATE TABLE IF NOT EXISTS title_basics_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 DECIMAL(4,0)) STORED AS ORCFILE
LOCATION '/user/hadoop/imdb/title_basics_partitioned';
```

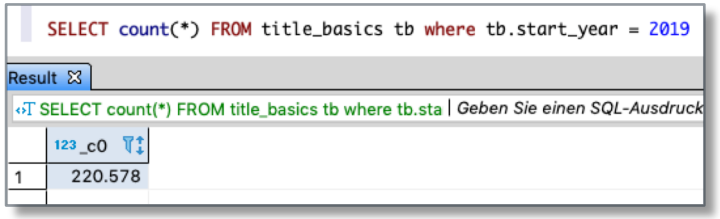

Dynamic Partitioning – INSERT Into Table via Hive

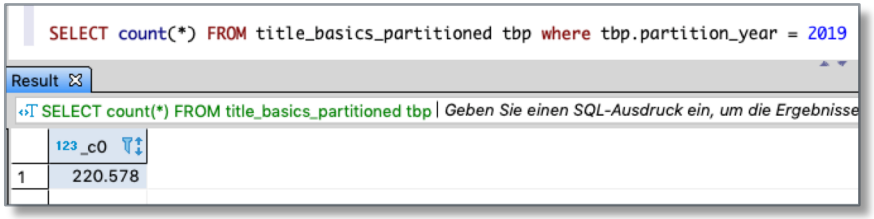
1. Migrate and partition data of table `title_basics` to table **`title_basics_partitioned`**:

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

INSERT OVERWRITE TABLE title_basics_partitioned partition(partition_year)
SELECT t.tconst, t.title_type, t.primary_title, t.original_title, t.is_adult,
t.start_year, t.end_year, t.runtime_minutes, t.genres,
t.start_year -- last column = partition column
FROM title_basics t;
```

2. Check Success via Hive:





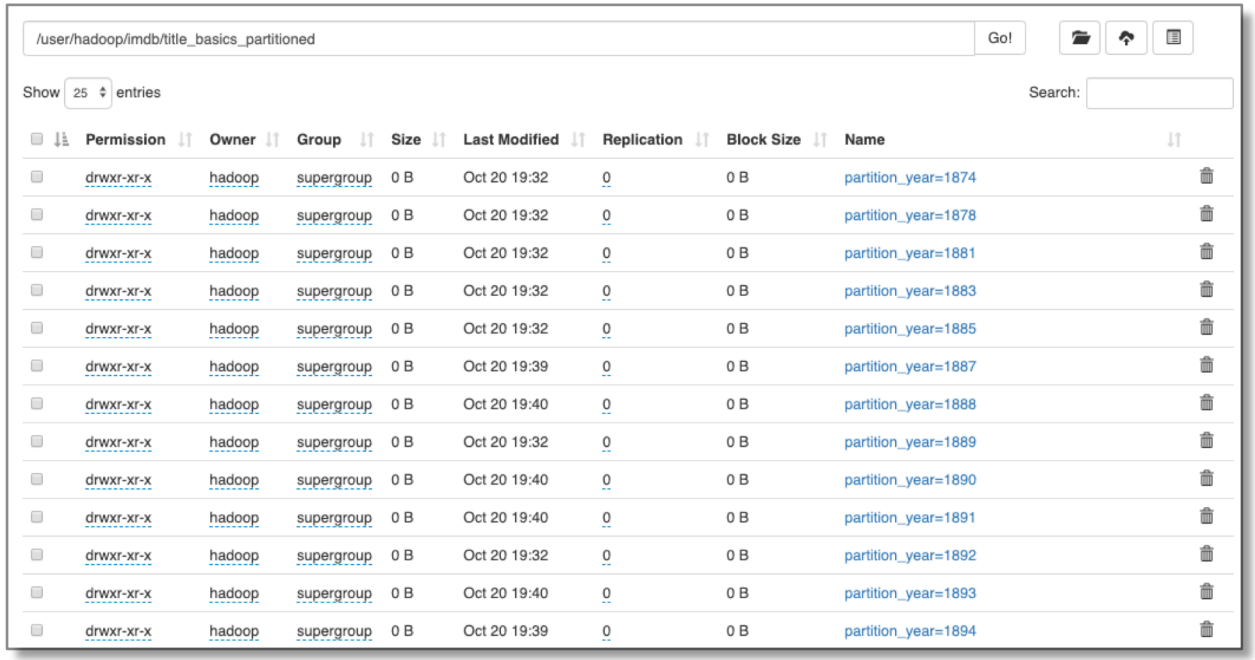
Dynamic Partitioning – **INSERT Into Table via Hive**

3. Check Success on HDFS:

```
hadoop fs -ls /user/hadoop/imdb/title_basics_partitioned
[…]
drwxr-xr-x   - hadoop supergroup          0 2019-10-20 17:32 /user/hadoop/imdb/title_basics_partitioned/partition_year=1874
drwxr-xr-x   - hadoop supergroup          0 2019-10-20 17:32 /user/hadoop/imdb/title_basics_partitioned/partition_year=1878
drwxr-xr-x   - hadoop supergroup          0 2019-10-20 17:32 /user/hadoop/imdb/title_basics_partitioned/partition_year=1881
drwxr-xr-x   - hadoop supergroup          0 2019-10-20 17:32 /user/hadoop/imdb/title_basics_partitioned/partition_year=1883
drwxr-xr-x   - hadoop supergroup          0 2019-10-20 17:32 /user/hadoop/imdb/title_basics_partitioned/partition_year=1885
drwxr-xr-x   - hadoop supergroup          0 2019-10-20 17:39 /user/hadoop/imdb/title_basics_partitioned/partition_year=1887
drwxr-xr-x   - hadoop supergroup          0 2019-10-20 17:40 /user/hadoop/imdb/title_basics_partitioned/partition_year=1888
drwxr-xr-x   - hadoop supergroup          0 2019-10-20 17:32 /user/hadoop/imdb/title_basics_partitioned/partition_year=1889
drwxr-xr-x   - hadoop supergroup          0 2019-10-20 17:40 /user/hadoop/imdb/title_basics_partitioned/partition_year=1890
drwxr-xr-x   - hadoop supergroup          0 2019-10-20 17:40 /user/hadoop/imdb/title_basics_partitioned/partition_year=1891
drwxr-xr-x   - hadoop supergroup          0 2019-10-20 17:32 /user/hadoop/imdb/title_basics_partitioned/partition_year=1892
drwxr-xr-x   - hadoop supergroup          0 2019-10-20 17:40 /user/hadoop/imdb/title_basics_partitioned/partition_year=1893
drwxr-xr-x   - hadoop supergroup          0 2019-10-20 17:39 /user/hadoop/imdb/title_basics_partitioned/partition_year=1894
drwxr-xr-x   - hadoop supergroup          0 2019-10-20 17:39 /user/hadoop/imdb/title_basics_partitioned/partition_year=1895
drwxr-xr-x   - hadoop supergroup          0 2019-10-20 17:39 /user/hadoop/imdb/title_basics_partitioned/partition_year=1896
drwxr-xr-x   - hadoop supergroup          0 2019-10-20 17:39 /user/hadoop/imdb/title_basics_partitioned/partition_year=1897
drwxr-xr-x   - hadoop supergroup          0 2019-10-20 17:39 /user/hadoop/imdb/title_basics_partitioned/partition_year=1898
drwxr-xr-x   - hadoop supergroup          0 2019-10-20 17:39 /user/hadoop/imdb/title_basics_partitioned/partition_year=1899
drwxr-xr-x   - hadoop supergroup          0 2019-10-20 17:39 /user/hadoop/imdb/title_basics_partitioned/partition_year=1900
drwxr-xr-x   - hadoop supergroup          0 2019-10-20 17:39 /user/hadoop/imdb/title_basics_partitioned/partition_year=1901
[…]
```


Dynamic Partitioning – INSERT Into Table via Hive

4. Check Success via HDFS Web Browser:



Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hadoop	supergroup	0 B	Oct 20 19:32	0	0 B	partition_year=1874
drwxr-xr-x	hadoop	supergroup	0 B	Oct 20 19:32	0	0 B	partition_year=1878
drwxr-xr-x	hadoop	supergroup	0 B	Oct 20 19:32	0	0 B	partition_year=1881
drwxr-xr-x	hadoop	supergroup	0 B	Oct 20 19:32	0	0 B	partition_year=1883
drwxr-xr-x	hadoop	supergroup	0 B	Oct 20 19:32	0	0 B	partition_year=1885
drwxr-xr-x	hadoop	supergroup	0 B	Oct 20 19:39	0	0 B	partition_year=1887
drwxr-xr-x	hadoop	supergroup	0 B	Oct 20 19:40	0	0 B	partition_year=1888
drwxr-xr-x	hadoop	supergroup	0 B	Oct 20 19:32	0	0 B	partition_year=1889
drwxr-xr-x	hadoop	supergroup	0 B	Oct 20 19:40	0	0 B	partition_year=1890
drwxr-xr-x	hadoop	supergroup	0 B	Oct 20 19:40	0	0 B	partition_year=1891
drwxr-xr-x	hadoop	supergroup	0 B	Oct 20 19:32	0	0 B	partition_year=1892
drwxr-xr-x	hadoop	supergroup	0 B	Oct 20 19:40	0	0 B	partition_year=1893
drwxr-xr-x	hadoop	supergroup	0 B	Oct 20 19:39	0	0 B	partition_year=1894

Exercises IV

HDFS/Hive: Work with Partitions

HDFS/Hive Partitioning Exercises - **IMDB**

1. Execute Tasks of previous HandsOn Slides

2. Create a (*statically*) partitioned table **`name_basics_partitioned`**, which:
   - contains all columns of table `name_basics`
   - 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 `name_basics` into table **`name_basics_partitioned`**

3. Create a (*dynamically*) partitioned table **`imdb_movies_and_ratings_partitioned`**, which:
   - contains all columns of the two tables `title_basics` and `title_ratings` and
   - is partitioned by start year of movie (create and add column `partition_year`).

   Load all data of `title_basics` and `title_ratings` into table:
   `imdb_movies_and_ratings_partitioned`