

# Exercises 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 (**takes some time!**), wait till you see:

```
hive/bin/hiveserver2
```

```
2021-02-21 16:43:55: 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 = ae41ac72-4dbd-4115-9863-59c3859c3db6
```

```
Hive Session ID = 17f9f63b-4018-4976-bb7d-15fbf1bc8042
```

```
Hive Session ID = 83b2ad76-c248-46a1-91d4-f2ad289614ee
```

```
Hive Session ID = b9ff1fd3-ccb1-4254-abc7-4c696d8ff8a1
```

```
[...]
```

# Connect To HiveServer2 via JDBC

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

Mac OSX:

```
wget https://dbeaver.io/files/dbeaver-ce-latest-macos.dmg
```

Linux (Debian):

```
wget https://dbeaver.io/files/dbeaver-ce_latest_amd64.deb
```

Linux (RPM):

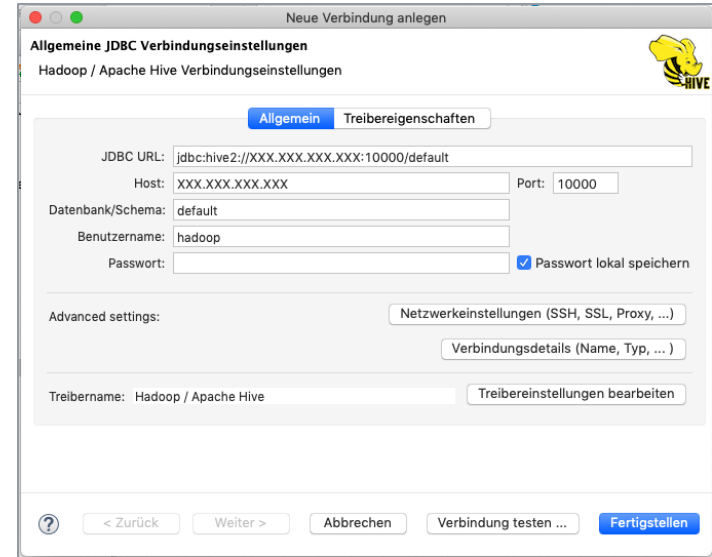
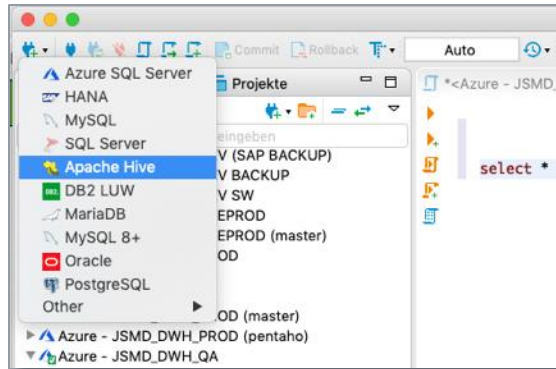
```
wget https://dbeaver.io/files/dbeaver-ce-latest-stable.x86_64.rpm
```

Windows:

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

# Connect To HiveServer2 via JDBC

## 2. Configure Connection To Hive Server:



# Let's 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 it 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:

The screenshot shows a Hive SQL editor interface. On the left, a project tree lists various databases and tables. The 'default' database is selected, and the 'title\_basics' table is highlighted. The main editor displays the SQL code for creating three external tables:

```
CREATE EXTERNAL TABLE IF NOT EXISTS title_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/title_ratings'  
TBLPROPERTIES ('skip.header.line.count'='1');
```

[https://github.com/marcelmittelstaedt/BigData/blob/master/exercises/winter\\_semester\\_2025-2026/03\\_hive-ql\\_partitioning\\_hive-server/title\\_ratings.sql](https://github.com/marcelmittelstaedt/BigData/blob/master/exercises/winter_semester_2025-2026/03_hive-ql_partitioning_hive-server/title_ratings.sql)

```
CREATE EXTERNAL TABLE IF NOT EXISTS title_basics (  
  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/title_basics'  
TBLPROPERTIES ('skip.header.line.count'='1');
```

[https://github.com/marcelmittelstaedt/BigData/blob/master/exercises/winter\\_semester\\_2025-2026/03\\_hive-ql\\_partitioning\\_hive-server/title\\_basics.sql](https://github.com/marcelmittelstaedt/BigData/blob/master/exercises/winter_semester_2025-2026/03_hive-ql_partitioning_hive-server/title_basics.sql)

```
CREATE EXTERNAL TABLE IF NOT EXISTS name_basics(  
  nconst STRING,  
  primary_name STRING,  
  birth_year INT,  
  death_year STRING,  
  primary_profession STRING,  
  known_for_titles STRING  
) COMMENT 'IMDb Actors' ROW FORMAT DELIMITED FIELDS TERMINATED BY '\c' STORED AS TEXTFILE LOCATION '/user/hadoop/imdb/name_basics'  
TBLPROPERTIES ('skip.header.line.count'='1');
```

[https://github.com/marcelmittelstaedt/BigData/blob/master/exercises/winter\\_semester\\_2025-2026/03\\_hive-ql\\_partitioning\\_hive-server/name\\_basics.sql](https://github.com/marcelmittelstaedt/BigData/blob/master/exercises/winter_semester_2025-2026/03_hive-ql_partitioning_hive-server/name_basics.sql)



# 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;
```

Ergebnisse 1 X

SELECT m.tconst, m.original\_title, m.start\_year, 123 average\_rating, 123 num\_votes | Geben Sie einen SQL-Ausdruck ein, um die Ergebnisse zu filtern (verwenden Sie ...)

	abc tconst	abc original_title	123 start_year	123 average_rating	123 num_votes
1	tt1375666	Inception	2.010	8,8	2.738.181
2	tt0816692	Interstellar	2.014	8,7	2.415.719
3	tt23849204	12th Fail	2.023	8,7	157.437
4	tt15097216	Jai Bhim	2.021	8,6	229.927
5	tt10189514	Soorarai Pottru	2.020	8,6	129.952
6	tt1853728	Django Unchained	2.012	8,5	1.828.304
7	tt6751668	Gisaengchung	2.019	8,5	1.105.714
8	tt2582802	Whiplash	2.014	8,5	1.104.248
9	tt1675434	Intouchables	2.011	8,5	997.037
10	tt15239678	Dune: Part Two	2.024	8,5	671.701

# Break

TIME FOR  
A  
BREAK

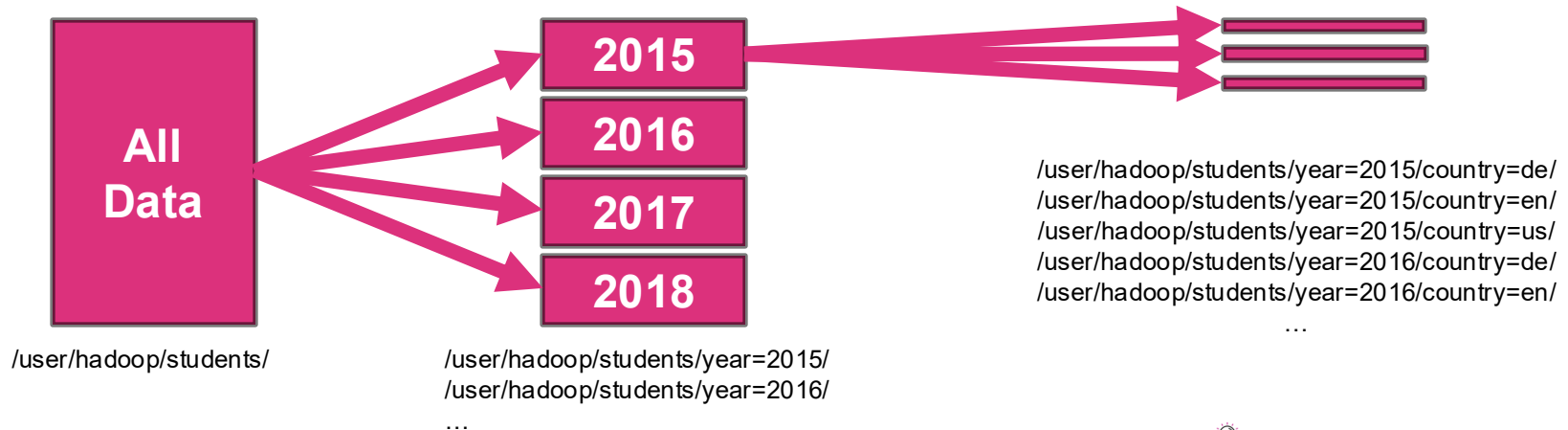


# HandsOn – Data Partitioning with HDFS and Hive (via JDBC)



# 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 PARQUET LOCATION '/user/hadoop/imdb/ratings_partitioned';
```

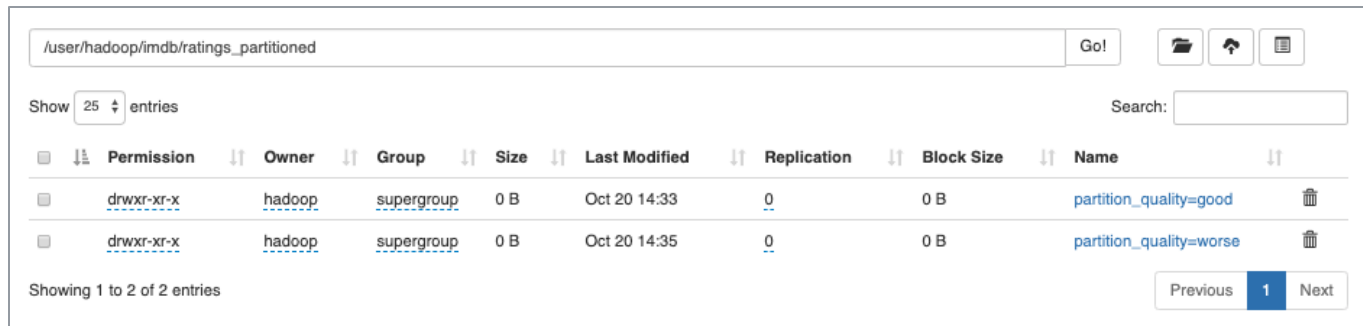
# Static Partitioning – INSERT Into Table via Hive



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:



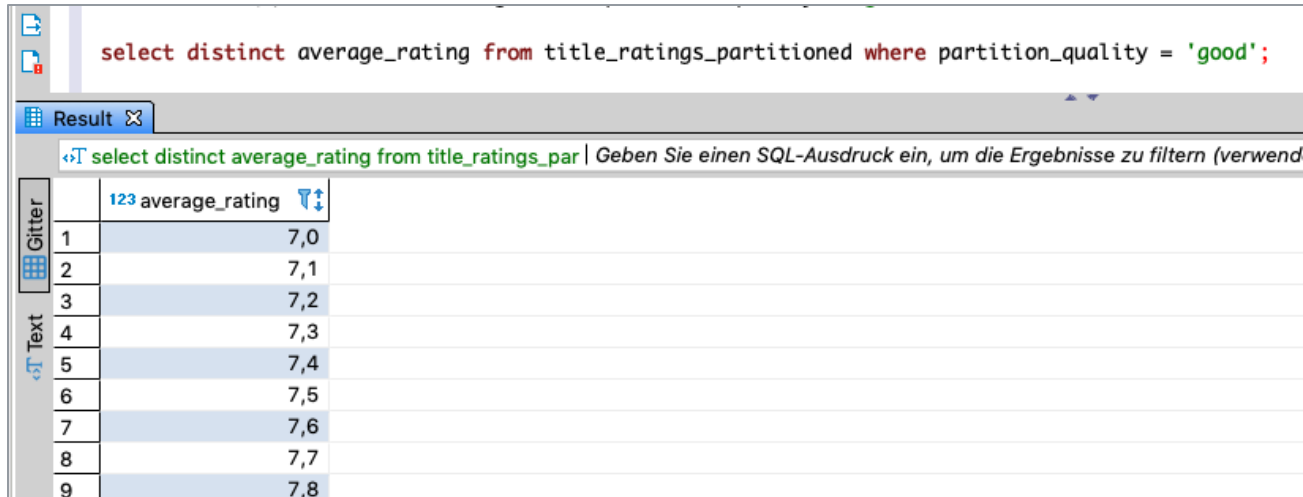
	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
<input type="checkbox"/>	drwxr-xr-x	<a href="#">hadoop</a>	<a href="#">supergroup</a>	0 B	Oct 20 14:33	<a href="#">0</a>	0 B	<a href="#">partition_quality=good</a> 
<input type="checkbox"/>	drwxr-xr-x	<a href="#">hadoop</a>	<a href="#">supergroup</a>	0 B	Oct 20 14:35	<a href="#">0</a>	0 B	<a href="#">partition_quality=worse</a> 

Showing 1 to 2 of 2 entries

Previous **1** Next

# Static Partitioning – INSERT Into Table via Hive

## 3. Check Success via Hive:



The screenshot shows a Hive query interface. At the top, a SQL query is entered: `select distinct average_rating from title_ratings_partitioned where partition_quality = 'good';`. Below the query, a tab labeled "Result" is active. Underneath the tab, a filter bar shows the query: `select distinct average_rating from title_ratings_par` followed by a German instruction: *Geben Sie einen SQL-Ausdruck ein, um die Ergebnisse zu filtern (verwend*. Below the filter bar, a table displays the results. The table has two columns: an index from 1 to 9 and the `average_rating` values. The ratings are 7,0, 7,1, 7,2, 7,3, 7,4, 7,5, 7,6, 7,7, and 7,8 respectively. On the left side of the table, there are icons for "Gitter" (grid) and "Text", with the "Text" icon being selected.

	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 PARQUET L  
OCATION '/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:

Runtime: 28 seconds

SELECT count(*) FROM title_basics_partitioned tb WHERE tb.start_year = 2021	
Ergebnisse 1 ×	
SELECT count(*) FROM title_basics_partitioned   Geben Sie einen SQL-Ausdruck ein, um die Ergebnisse	
123_c0	518.712

Runtime: 28 seconds

SELECT count(*) FROM title_basics tb WHERE tb.start_year = 2021	
ergebnisse 1 ×	
SELECT count(*) FROM title_basics tb WHERE tb.start_year = 2021   Geben Sie einen SQL-Ausdruck ein, um die Ergebnisse zu	
123_c0	518.712

Runtime: <1 second

SELECT count(*) FROM title_basics_partitioned tb WHERE tb.partition_year = 2021;	
ergebnisse 1 ×	
SELECT count(*) FROM title_basics_partitioned   Geben Sie einen SQL-Ausdruck ein, um die Ergebnisse zu	
123_c0	518.712



# Dynamic Partitioning – INSERT Into Table via Hive

## 3. Check Success on HDFS:

```
hadoop fs -ls /user/hadoop/imdb/title_basics_partitioned
```

Found 149 items

```
drwxr-xr-x - hadoop supergroup 0 2021-02-21 17:14 /user/hadoop/imdb/title_basics_partitioned/partition_year=1874
drwxr-xr-x - hadoop supergroup 0 2021-02-21 17:23 /user/hadoop/imdb/title_basics_partitioned/partition_year=1878
drwxr-xr-x - hadoop supergroup 0 2021-02-21 17:13 /user/hadoop/imdb/title_basics_partitioned/partition_year=1881
drwxr-xr-x - hadoop supergroup 0 2021-02-21 17:14 /user/hadoop/imdb/title_basics_partitioned/partition_year=1883
drwxr-xr-x - hadoop supergroup 0 2021-02-21 17:13 /user/hadoop/imdb/title_basics_partitioned/partition_year=1885
drwxr-xr-x - hadoop supergroup 0 2021-02-21 17:23 /user/hadoop/imdb/title_basics_partitioned/partition_year=1887
drwxr-xr-x - hadoop supergroup 0 2021-02-21 17:23 /user/hadoop/imdb/title_basics_partitioned/partition_year=1888
drwxr-xr-x - hadoop supergroup 0 2021-02-21 17:23 /user/hadoop/imdb/title_basics_partitioned/partition_year=1889
drwxr-xr-x - hadoop supergroup 0 2021-02-21 17:23 /user/hadoop/imdb/title_basics_partitioned/partition_year=1890
drwxr-xr-x - hadoop supergroup 0 2021-02-21 17:23 /user/hadoop/imdb/title_basics_partitioned/partition_year=1891
drwxr-xr-x - hadoop supergroup 0 2021-02-21 17:14 /user/hadoop/imdb/title_basics_partitioned/partition_year=1892
drwxr-xr-x - hadoop supergroup 0 2021-02-21 17:23 /user/hadoop/imdb/title_basics_partitioned/partition_year=1893
drwxr-xr-x - hadoop supergroup 0 2021-02-21 17:23 /user/hadoop/imdb/title_basics_partitioned/partition_year=1894
drwxr-xr-x - hadoop supergroup 0 2021-02-21 17:23 /user/hadoop/imdb/title_basics_partitioned/partition_year=1895
drwxr-xr-x - hadoop supergroup 0 2021-02-21 17:23 /user/hadoop/imdb/title_basics_partitioned/partition_year=1896
drwxr-xr-x - hadoop supergroup 0 2021-02-21 17:23 /user/hadoop/imdb/title_basics_partitioned/partition_year=1897
drwxr-xr-x - hadoop supergroup 0 2021-02-21 17:23 /user/hadoop/imdb/title_basics_partitioned/partition_year=1898
drwxr-xr-x - hadoop supergroup 0 2021-02-21 17:22 /user/hadoop/imdb/title_basics_partitioned/partition_year=1899
drwxr-xr-x - hadoop supergroup 0 2021-02-21 17:22 /user/hadoop/imdb/title_basics_partitioned/partition_year=1900
drwxr-xr-x - hadoop supergroup 0 2021-02-21 17:22 /user/hadoop/imdb/title_basics_partitioned/partition_year=1901
[...]
```

# Dynamic Partitioning – INSERT Into Table via Hive

## 4. Check Success via HDFS Web Browser (<http://X.X.X.X:9870/>)

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities ▾									
Browse Directory									
<input type="text" value="/user/hadoop/ imdb/title_basics_partitioned"/>						<input type="button" value="Go!"/>	<input type="button" value="📁"/>	<input type="button" value="📄"/>	<input type="button" value="🗑"/>
Show	25	entries	Search: <input type="text"/>						
<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	<input type="checkbox"/>
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Feb 21 18:14	0	0 B	partition_year=1874	<input type="checkbox"/>
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Feb 21 18:23	0	0 B	partition_year=1878	<input type="checkbox"/>
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Feb 21 18:13	0	0 B	partition_year=1881	<input type="checkbox"/>
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Feb 21 18:14	0	0 B	partition_year=1883	<input type="checkbox"/>
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Feb 21 18:13	0	0 B	partition_year=1885	<input type="checkbox"/>
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Feb 21 18:23	0	0 B	partition_year=1887	<input type="checkbox"/>
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Feb 21 18:23	0	0 B	partition_year=1888	<input type="checkbox"/>
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Feb 21 18:23	0	0 B	partition_year=1889	<input type="checkbox"/>
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Feb 21 18:23	0	0 B	partition_year=1890	<input type="checkbox"/>
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Feb 21 18:23	0	0 B	partition_year=1891	<input type="checkbox"/>
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Feb 21 18:14	0	0 B	partition_year=1892	<input type="checkbox"/>
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Feb 21 18:23	0	0 B	partition_year=1893	<input type="checkbox"/>
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Feb 21 18:23	0	0 B	partition_year=1894	<input type="checkbox"/>
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Feb 21 18:23	0	0 B	partition_year=1895	<input type="checkbox"/>



# Break

TIME FOR  
A  
BREAK



# Exercise

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`

# Well Done

WE'RE DONE  
FOR  
...TODAY





# Stop Your VM Instances

**DON'T FORGET TO  
STOP YOUR VM  
INSTANCE!**



```
gcloud compute instances stop big-data
```

