Solution – Exercise IV

HiveServer 2, HiveQL via JDBC,
Partitioning with HDFS and Hive

Solution

Prerequisites:

- Start Gcloud instance
- Pull and start Docker image (`marcelmittelstaedt/hiveserver_base:latest`)
- Start Hadoop Cluster
- Start HiveServer2
- Download, Install and Configure JDBC Rich-client:
    - e.g. DBeaver,
    - SquirrelSQL,
    - …
- Execute all preparation and example tasks of previous HandsOn slides of last lecture

Solution

Exercise IV:

2.1 Create table **name_basics_partitioned** partitioned by column partition_is_alive:

```
CREATE EXTERNAL TABLE IF NOT EXISTS name_basics_partitioned (
          nconst STRING,
          primary_name STRING,
          birth_year INT,
          death_year STRING,
          primary_profession STRING,
          known_for_titles STRING
) PARTITIONED BY (partition_is_alive STRING)
STORED AS PARQUET LOCATION '/user/hadoop/imdb/actors_partitioned';
```

Solution

Exercise IV:

2.2 Use static partitioning to create and fill partition ‘alive‘

```
INSERT OVERWRITE TABLE name_basics_partitioned
partition(partition_is_alive='alive')
SELECT
          a.nconst,
          a.primary_name,
          a.birth_year,
          a.death_year,
          a.primary_profession,
          a.known_for_titles
FROM name_basics a WHERE a.death_year IS NULL
```

Solution

Exercise IV:

2.3 Use static partitioning to create and fill partition ‘dead‘

```
INSERT OVERWRITE TABLE name_basics_partitioned
partition(partition_is_alive='dead')
SELECT
          a.nconst,
          a.primary_name,
          a.birth_year,
          a.death_year,
          a.primary_profession,
          a.known_for_titles
FROM name_basics a WHERE a.death_year IS NOT NULL
```

Solution

Exercise IV:

2.4 Check Results:

```
hadoop fs -ls /user/hadoop/imdb/actors_partitioned
drwxr-xr-x   - hadoop supergroup          0 2021-02-27 17:16 /user/hadoop/imdb/actors_partitioned/partition_is_alive=alive
drwxr-xr-x   - hadoop supergroup          0 2021-02-27 17:16 /user/hadoop/imdb/actors_partitioned/partition_is_alive=dead
```


Solution

Exercise IV:

2.4 Check Results:

```
SELECT * FROM name_basics_partitioned  WHERE partition_is_alive = 'dead' LIMIT 100
```

Result

SELECT * FROM name_basics_partitioned WH | Geben Sie einen SQL-Ausdruck ein, um die Ergebnisse zu filtern (verwenden Sie Strg+ Leertaste).

	nconst	primary_name	birth_year	death_year	primary_profession	known_for_titles	partition_is_alive
1	nm0000001	Fred Astaire	1.899	1987	soundtrack,actor,miscellaneous	tt0072308,tt0053137,tt0050419,tt0031983	dead
2	nm0000002	Lauren Bacall	1.924	2014	actress,soundtrack	tt0037382,tt0071877,tt0038355,tt0117057	dead
3	nm0000004	John Belushi	1.949	1982	actor,soundtrack,writer	tt0072562,tt0080455,tt0077975,tt0078723	dead
4	nm0000005	Ingmar Bergman	1.918	2007	writer,director,actor	tt0069467,tt0050976,tt0050986,tt0060827	dead
5	nm0000006	Ingrid Bergman	1.915	1982	actress,soundtrack,producer	tt0038787,tt0077711,tt0034583,tt0038109	dead
6	nm0000007	Humphrey Bogart	1.899	1957	actor,soundtrack,producer	tt0042593,tt0037382,tt0033870,tt0034583	dead
7	nm0000008	Marlon Brando	1.924	2004	actor,soundtrack,director	tt0047296,tt0068646,tt0078788,tt0070849	dead
8	nm0000009	Richard Burton	1.925	1984	actor,soundtrack,producer	tt0057877,tt0059749,tt0061184,tt0087803	dead
9	nm0000010	James Cagney	1.899	1986	actor,soundtrack,director	tt0031867,tt0035575,tt0042041,tt0029870	dead
10	nm0000011	Gary Cooper	1.901	1961	actor,soundtrack,producer	tt0027996,tt0044706,tt0035896,tt0034167	dead

Solution

Exercise IV:

3.1 Create table `imdb_movies_and_ratings_partitioned` partitioned by column `partition_year` using fields of table `title_basics` and `title_ratings`:

```
CREATE TABLE IF NOT EXISTS imdb_movies_and_ratings_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,
            average_rating DECIMAL(2,1),
            num_votes BIGINT
) PARTITIONED BY (partition_year int) STORED AS PARQUET LOCATION '/user/hadoop/imdb/
movies_and_ratings_partitioned';
```

Solution

Exercise IV:

3.2 Use dynamic partitioning to create and fill partition `partition_year`:

```
SET hive.exec.dynamic.partition.mode=nonstrict;
INSERT OVERWRITE TABLE imdb_movies_and_ratings_partitioned partition(partition_year)
SELECT
            tb.tconst,
            tb.title_type,
            tb.primary_title,
            tb.original_title,
            tb.is_adult,
            tb.start_year,
            tb.end_year,
            tb.runtime_minutes,
            tb.genres,
            tr.average_rating,
            tr.num_votes,
            tb.start_year
FROM title_basics tb JOIN title_ratings tr ON (tb.tconst = tr.tconst)
```

Solution

Exercise IV:

3.3 Check Results:

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

[…]
drwxr-xr-x   - hadoop supergroup          0 2021-02-27 17:24 /user/hadoop/imdb/movies_and_ratings_partitioned/partition_year=1874
drwxr-xr-x   - hadoop supergroup          0 2021-02-27 17:24 /user/hadoop/imdb/movies_and_ratings_partitioned/partition_year=1878
drwxr-xr-x   - hadoop supergroup          0 2021-02-27 17:24 /user/hadoop/imdb/movies_and_ratings_partitioned/partition_year=1881
drwxr-xr-x   - hadoop supergroup          0 2021-02-27 17:24 /user/hadoop/imdb/movies_and_ratings_partitioned/partition_year=1883
drwxr-xr-x   - hadoop supergroup          0 2021-02-27 17:24 /user/hadoop/imdb/movies_and_ratings_partitioned/partition_year=1885
drwxr-xr-x   - hadoop supergroup          0 2021-02-27 17:24 /user/hadoop/imdb/movies_and_ratings_partitioned/partition_year=1887
drwxr-xr-x   - hadoop supergroup          0 2021-02-27 17:24 /user/hadoop/imdb/movies_and_ratings_partitioned/partition_year=1888
drwxr-xr-x   - hadoop supergroup          0 2021-02-27 17:24 /user/hadoop/imdb/movies_and_ratings_partitioned/partition_year=1889
drwxr-xr-x   - hadoop supergroup          0 2021-02-27 17:24 /user/hadoop/imdb/movies_and_ratings_partitioned/partition_year=1890
drwxr-xr-x   - hadoop supergroup          0 2021-02-27 17:24 /user/hadoop/imdb/movies_and_ratings_partitioned/partition_year=1891
drwxr-xr-x   - hadoop supergroup          0 2021-02-27 17:24 /user/hadoop/imdb/movies_and_ratings_partitioned/partition_year=1892
[…]
```

Solution

Exercise IV:

3.3 Check Results:

```
SELECT tconst, primary_title, start_year, genres, average_rating, num_votes
FROM imdb_movies_and_ratings_partitioned
WHERE partition_year = 2020 AND title_type = 'movie' AND num_votes > 25000 ORDER BY average_rating DESC LIMIT 100
```

Result

SELECT tconst, primary_title, start_year, genres, average_ | Geben Sie einen SQL-Ausdruck ein, um die Ergebnisse zu filtern (verwenden Sie Strg+ Leertaste).

	tconst	primary_title	start_year	genres	average_rating	num_votes
1	tt10189514	Soorarai Pottru	2.020	Drama	8,6	55.837
2	tt8503618	Hamilton	2.020	Biography,Drama,History	8,5	57.348
3	tt2948372	Soul	2.020	Adventure,Animation,Comedy	8,1	179.805
4	tt8110330	Dil Bechara	2.020	Comedy,Drama,Romance	7,9	111.891
5	tt1070874	The Trial of the Chicago 7	2.020	Drama,History,Thriller	7,8	98.179
6	tt10288566	Another Round	2.020	Comedy,Drama	7,8	40.297
7	tt11464826	The Social Dilemma	2.020	Documentary,Drama	7,7	65.207
8	tt7212754	Ludo	2.020	Action,Comedy,Crime	7,6	28.350
9	tt6723592	Tenet	2.020	Action,Sci-Fi,Thriller	7,5	296.730
10	tt9620292	Promising Young Woman	2.020	Crime,Drama,Thriller	7,5	34.475
11	tt7146812	Onward	2.020	Adventure,Animation,Comedy	7,4	102.711
12	tt9484998	Palm Springs	2.020	Comedy,Fantasy,Mystery	7,4	81.009
13	tt10618286	Mank	2.020	Biography,Comedy,Drama	7,1	38.884
14	tt9686708	The King of Staten Island	2.020	Comedy,Drama	7,1	39.349
15	tt7395114	The Devil All the Time	2.020	Crime,Drama,Thriller	7,1	93.645
16	tt1051906	The Invisible Man	2.020	Drama,Horror,Mystery	7,1	169.172